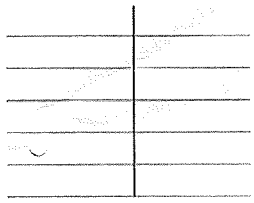
Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2017-05-31
Sal (1)	TER3(29)
Tid	14-18
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	´mellan kl. 15-16
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 2017-05-31

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

a) *Lifted* planning is a general technique applicable to a variety of search spaces and planning algorithms. During the lectures, we specifically discussed *lifted partial-order planning* as well as *lifted backward goal-space planning*, but since the technique itself is general, it could also be applied to for example HTN planning.

Explain what lifted planning is in general.

Then give a concrete example of why lifting can be useful when generating partial-order plans in particular. In this example you should:

- **Show** a small lifted partial-order plan structure (or a sufficiently complete and informative fragment of one),
- **Contrast** it against a non-lifted alternative structure that you also include in your answer, and
- **Explain** what is better about the lifted version. This should be be done through contrasting how planning would proceed in the two alternative (lifted and non-lifted) structures and showing how the lifted structure has an advantage.

Note again: The plan fragment must be *sufficiently complete and informative*, which implies that you should include all details that help explain the usefulness of lifting.

(3 points)

b) There are many different types of plans. Two types that we have used in this course are *sequential plans* and *policies*.

What *is* a policy? How do you *execute* a policy?

(1 point)

2 Search, Search Guidance and Heuristics

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) Suppose that you are given a large number of planning problem instances, and that your objective is to produce a planner that solves these instances as quickly as possible.

- As you are well aware from the lectures, a large number of planning algorithms already exist. Explain how you could *combine* several of these algorithms – at a high level, without needing to integrate them at the deeper search level – in order to solve the average problem instance more quickly than using a single algorithm in isolation. The explanation should describe approximately how you choose which planners to run and how you decide when/how to run them.
- Also explain *why* this technique can work – why it isn't necessarily better to run a single algorithm until it finds a plan.

Clarifications:

- The method used must also solve any other instances from the same problem domains as quickly as possible, so implementing a simple table lookup is out of the question.
- We assume that you are given a single CPU core, so applying multiple algorithms must mean dividing the available time between the algorithms in some way.

(3 points)

3 Planning with Incomplete Information

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).

a) What *is* the Stochastic Shortest Path Problem?

   You might want to explain this by contrasting it to the classical planning problem and explaining the differences in the respective state transition systems of these two problems. For example: What knowledge do we have before we start? What can we observe? How do actions work? What is the objective? What kind of solution structure do we get?

   (2 points)

b) The Stochastic Shortest Path Problem is different from the general MDP (Markov Decision Process) problem. Nevertheless, an SSPP problem instance can 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? Why is this transformation required?

   (2 points)

4 Motion Planning

a) In motion planning, we must distinguish between the *workspace* (WS) and the *configuration space* (CS) of a vehicle or an object.

   - Give an example of a workspace and a configuration space for a typical *car* and a typical *fixed wing aircraft*.

     In case there are several possible options, choose one specific WS/CS combination that seems reasonable and that provides the necessary degrees of freedom.

   - Is the *goal* of a motion planner specified in the WS or in the CS? Why?

   (3 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.

     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.

   - When the PRM planner does receive a starting point and goal, how does it make use of the result of the pre-processing phase to create a motion plan?

   (4 points)