Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2015-08-20
Sal (1)	TER3
Tid	8-12
Kurskod	TDDD48
Provkod	TEN1
Kursnamn/benämning **Provnamn/benämning**	Automatisk planering Skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	5
Jour/Kursansvarig Ange vem som besöker salen	Jonas Kvarnström
Telefon under skrivtiden	0704-737579
Besöker salen ca klockan	ca kl. 09
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 2015-08-20

Important Instructions: Read before you begin!

- Though the questions are in English, feel free to **answer in Swedish** if you prefer!
- *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.
- Make an effort to write explanations that can be understood by someone 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, then 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.
- 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*!
- 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 Fundamental Concepts

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*, but since the technique itself is general, it could also be applied to for example backward planning or HTN planning.

Explain what lifted planning is in general.

Then give a concrete example of why it 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 informative fragment of one), **contrast** it against a non-lifted alternative, and **explain** what is better about the lifted version (including *how* one can make use of lifting in practice). **(3 points)**

2 Partial-Order Planning

Partial-order causal link (POCL) planning differs from forward state space search by generating partially ordered plans. This requires a different search space as well as a different search method.

Name one important strength that these differences give POCL planning over forward state space planning, as well as one important strength that the differences give forward state space planning over POCL planning, when it comes to finding a feasible plan in a reasonable amount of time.

Make sure that you explain why these strengths exist and why they are not shared by the other planning paradigm. For example, techniques such as lifted planning can be applied to both of the planning methods mentioned above and are therefore not strengths of one method *over* the other. Features that are inherent in one method and not in the other, or techniques that can be applied to one method but not to the other, are of greater interest. **(3 points)**

3 Heuristics and Search Guidance

a) 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)**

b) Describe the general *relaxation principle*. In particular: Given a problem, what characterizes a *relaxed* problem? How, exactly, can a relaxed problem be used to calculate an admissible heuristic value for a particular state s in the original problem? (That is, describe how $h(s)$ could be calculated given a relaxed problem and a specific state s.) Why does this method (calculation) guarantee admissibility? **(2 points)**

c) Is a *pattern database heuristic* a form of *relaxation heuristic*? Is it a form of *delete relaxation heuristic*? In each case, motivate why / why not.

Example of how you might motivate and what kind of answer we are looking for: "*A relaxation heuristic is any heuristic doing X. A PDB heuristic is calculated by doing Y1, Y2, Y3. Clearly, as a consequence, X must also be true, so PDBs heuristics are relaxation heuristics /-/ As shown in the example below, this can violate X, so PDB heuristics are not relaxation heuristics*". **(2 points)**

d) What is a *fact landmark*? Explain the general definition of a fact landmark. Then give a (formal) example of a concrete fact landmark from a problem instance in a simple standard planning domain such as for example Logistics, Blocks World or Towers of Hanoi. Explain why this is a fact landmark and make sure you provide sufficient information about the problem instance so that this can be verified. **(2 points)**

4 SATisfiability-based Planning

We will now consider planning based on propositional satisfiability ("SAT planning").

a) SAT planners translate planning problems into a set of boolean propositions and a set of formulas over these propositions. Typically, there are two specific types of information encoded as boolean propositions. In other words, once a SAT solver has found a solution, two specific types of information can be inferred from the values of the boolean propositions. Which ones? **(2 points)**

b) SAT *planners* must be able to find solution plans of arbitrary length, where the length is not known in advance. In contrast, the SAT *solvers* that they are based on require an input containing a fixed and finite number of boolean propositions. How is this discrepancy handled by a SAT planner? **(2 points)**

c) SAT planners require *frame axioms* (for example, *explanatory* frame axioms) to ensure that the SAT solver gives correct results that correspond to valid plans.

   Explain how planning could go wrong if frame axioms were not included in the SAT translation of a planning problem. Also demonstrate using a concrete example. The example does not have to involve an entire SAT assignment and SAT formula processed from beginning to end, as this would take considerable time to generate by hand. It is sufficient to illustrate the most central aspects of how the planning process could return invalid plans given a lack of frame axioms. **(2 points)**

5 Domain-Configurable Planning

Some domain-configurable planners allow the specification of *control rules*.

What *is* a control rule? In other words, what additional information can be expressed using a control rule compared to what can be expressed in an "ordinary" classical planning problem?

Give a concrete example of a control rule that could help a domain-configurable planner in a domain of your choice. You do not have to use any specific syntax; a clear description of the rule is sufficient.

Explain how a forward-chaining state space planner can concretely make use of such a rule: When during the search process does it use/apply the rule, what can the result be, and what does it do depending on the result? **(3 points)**