Aalto University School of Science, Department of Computer Science | TJ & TJ

CS-E3220 Declarative Programming

Examination, May 31st, 2019

Use of electronic devices, such as calculators, is not allowed in the examination.

Assignment 1 DPLL and Resolution (Max. 5p)

Consider the unsatisfiable CNF formula $(\neg a \vee b \vee c) \wedge (a \vee b) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (\neg b \vee c)$. Give a DPLL search tree and a resolution proof of unsatisfiability for it.

Assignment 2 Representations and Normal Forms (Max. 10p)

Let $f(a,b,c)$ be the Boolean function defined by the formula $(a \leftrightarrow c) \vee (b \wedge c)$.

(a) Derive a ROBDD representation for the function.

(b) Compile the ROBDD into a *smooth* and *deterministic* dNNF.

(c) Compute the minimum cardinality of $f(a,b,c)$ using the resulting sd-dNNF representation and simplify the sd-dNNF on basis of the cardinality found.

Assignment 3 Conflict-Driven Clause Learning (Max. 5p)

Consider a set of clauses consisting of $\neg a \vee c$, $\neg a \vee e$, $\neg b \vee \neg e \vee g$, $\neg b \vee \neg c \vee \neg d$, $\neg c \vee d \vee \neg f$, and $d \vee f$.

(a) Simulate the CDCL algorithm by choosing decision variables in the alphabetic order $a, b, c, \ldots$, always setting the chosen variable true first, until a conflict is reached.

(b) Derive a new clause by using the 1-UIP learning scheme and indicate the level for backjumping.

Assignment 4 CSPs and Arc Consistency (Max. 5p)

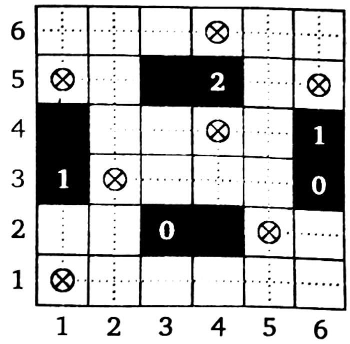
Consider the sum constraint $2x_1 + 2x_2 + x_3 + x_4 + 2x_5 = z$ when the domains are $D(x_1) = \{1,2\}$, $D(x_2) = \{1,3\}$, $D(x_3) = \{2,3\}$, $D(x_4) = \{1,3\}$, $D(x_5) = \{0,1,2\}$, and $D(z) = \{1,3,5,9\}$.

Answer the following questions by using the dynamic programming approach described in the lecture slides (illustrate the relevant parts of the graph presentation in your answer).

- Does the constraint have solutions? If it has, give one. If it does not have any, argue why this is the case.
- Is the constraint arc-consistent? If it is, explain why this is the case. If it is not, propagate the constraint to be arc-consistent and give the resulting new domains.

NB: Assignments continue on the other side of the question sheet!

Assignment 5 Answer Set Programming (Max. 10p)



Consider a grid puzzle as illustrated in the picture. The goal is to place *lights* "⊗" on white grid cells so that the following requirements are met:

1. Every white cell contains a light or is visible, in horizontal or vertical direction, from a cell with a light via a straight path of white cells.
2. No distinct cells with lights are mutually visible via straight paths of white cells.
3. Black cells with numbers determine how many of their horizontally or vertically adjacent white cells must contain lights.

A solution for the given instance of the grid puzzle is also indicated in the picture. Observe that the lights in adjacent white cells match the numbers given in some black cells, e.g., the lights at ⟨4,4⟩ and ⟨4,6⟩ are adjacent to the black cell with a 2 at ⟨4,5⟩. Moreover, every white cell has a light or is visible from a light, as indicated by dotted straight lines, while no distinct lights are mutually visible. Also note that lights may be placed freely, i.e., not having an adjacent black cell with a number, as it is the case for the lights at ⟨1,1⟩, ⟨1,5⟩, and ⟨5,2⟩. The puzzle instance given in the picture can be described by facts as follows:

White(1,6).	White(2,6).	White(3,6).	White(4,6).	White(5,6).	White(6,6).
White(1,5).	White(2,5).		Black(4,5,2).	White(5,5).	White(6,5).
	White(2,4).	White(3,4).	White(4,4).	White(5,4).	Black(6,4,1).
Black(1,3,1).	White(2,3).	White(3,3).	White(4,3).	White(5,3).	Black(6,3,0).
White(1,2).	White(2,2).	Black(3,2,0).		White(5,2).	White(6,2).
White(1,1).	White(2,1).	White(3,1).	White(4,1).	White(5,1).	White(6,1).

A solution shall be represented in terms of the output predicate Light(·,·), e.g., Light(1,1), Light(1,5), Light(2,3), Light(4,4), Light(4,6), Light(5,2), and Light(6,5) for the solution indicated in the picture.

Formalize the requirements for solutions by writing a uniform ASP encoding in the syntax of GRINGO.

Assignment 6 Satisfiability Modulo Theories (Max. 15p)

(a) Recall that T_{EUF} is the theory of uninterpreted functions and predicates (with built-in equality, as usual) and T_{LIA} is the theory of linear arithmetic over integers. Which of the following are true? Justify your answer by giving a model or a proof.

1. $a \approx c \wedge a \not\approx b \wedge g(f(a),b) \approx g(f(c),a)$ is T_{EUF}-satisfiable.
2. $(y < x+1) \wedge (x \leq 2) \wedge (y \geq 1) \rightarrow (x \geq 1)$ is T_{LIA}-valid. (5p)

(b) Use the congruence closure algorithm to decide whether the following conjunction is T_{EUF}-satisfiable (T_{EUF} is the theory of equality and uninterpreted functions with built-in equality, as usual). If it is, give a model for it. You shall show how the algorithm proceeds when you enter the conjuncts one-by-one in the order they appear in the formula.

$$x \approx h(y) \wedge g(f(h(y))) \not\approx g(z) \wedge f(x) \approx z$$

(5p)

(c) Explain the abstract Nelson–Oppen theory combination method with less than 20 sentences; what problem it solves, what does it require and how does it work. (You don't have to explain implementation methods such as "delayed theory combination" etc.) Apply the method to the $(T_{LIA} \cup T_{EUF})$-conjunction

$$y = -2x+2 \wedge y = x-1 \wedge f(x+y) \not\approx f(x)$$

to determine whether it is $(T_{LIA} \cup T_{EUF})$-satisfiable or not (T_{LIA} is the theory of linear arithmetic over integers). If it is $(T_{LIA} \cup T_{EUF})$-satisfiable, also give a model for it. (5p)

Assignment 7 When did you complete your answers (please record the time)?