Satisfiability Checking - WS 2020/2021

Written Exam I

Saturday, March 06, 2021

Forename and surname:	**Matriculation number:**
Forename Surename	000000
Your identifier: abcd **Your room number:** 2	

- Do not start the exam until we give the start signal.
- The duration of the exam is 120 minutes.
- Write your solution on empty sheets. You may use blank, lined or squared paper.
- Use a blue or black (permanent) pen only.
- Please write your name, matriculation number and a page number on every page.
- Please use a new page for every of the nine tasks and indicate which task is solved on this page.
- Please write clear and legible answers.
- Please clearly cross out parts you do *not* wish to be evaluated.
- If you have problems understanding a task, indicate this by a hand signal.
- You are not allowed to use auxiliary material except for a pen and a ruler. Cheating disqualifies from the exam.
- In case the upload via Moodle fails, please upload your solution to `https://gigamove.rz.rwth-aachen.de/` and send us an E-Mail to exam@ths.rwth-aachen.de with `2 000000` as subject and only the GigaMove link as body.
- In case there are any troubles during the exam, call us at `0241/123456`.

Task	1.)	2.)	3.)	4.)	5.)	6.)	7.)	8.)	9.)	**Total**
Maximum score	20	13	15	15	10	13	10	12	12	120
Reached score										

Good luck!

1.) SAT Checking **(6+4)+(5+3)+2 Points**

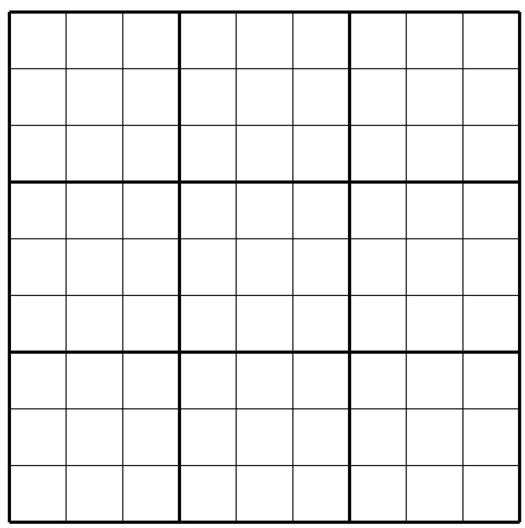
i) Consider the following propositional logic formula:

$$\begin{array}{llll} & c_1 : (b \vee \neg c \vee \neg d) & \wedge \quad c_2 : (a \vee b \vee d \vee f) & \wedge \quad c_3 : (\neg c \vee e) \\ \wedge & c_4 : (\neg a) & \wedge \quad c_5 : (a \vee \neg b) & \wedge \quad c_6 : (a \vee b \vee \neg c \vee \neg f) \end{array}$$

(a) **Apply the CDCL SAT-checking algorithm** to the given formula until the first conflict appears. When making a decision, take the lexicographically smallest unassigned variable first and assign **true** to it. Show the (partial) assignments after each decision level.

(b) Apply **conflict resolution** as presented in the lecture.

ii) The objective of a Sudoku puzzle is to fill a 9×9 grid with digits in $\{1, 2, \ldots, 9\}$ such that no digit appears twice in a row, a column, or in a 3×3 subgrid (see figure below).



Note: The set of feasible solutions to a Sudoku is usually constrained by some digits filled into the grid. For this task, no such constraints are given.

Using the Boolean variables x_{ijk}, $i, j, k \in \{1, \ldots, 9\}$ to encode that digit k is assigned to the cell in row i and column j, define the below-listed parts of the following formula to encode the set of feasible solutions for the Sudoku puzzle:

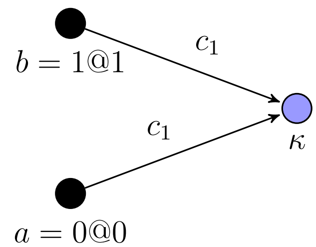
$$\varphi_{\text{digit}} \wedge \bigwedge_{i=1,\ldots,k} \varphi_{\text{row}}(i) \wedge \bigwedge_{j=1,\ldots,k} \varphi_{\text{column}}(j) \wedge \varphi_{\text{subgrid}}$$

a) φ_{digit}: Exactly one digit is assigned to every cell.

b) $\varphi_{\text{row}}(i)$: No digit appears twice in row i.

Note: The subformulas $\varphi_{\text{column}}(j)$ encoding that no digit appears twice in column j and φ_{subgrid} encoding that no digit appears twice in a subgrid do not need to be specified in this task.

iii) Can the following conflict occur in DPLL? Why? (Note: A reason for the answer is neccessary to obtain any points for this subtask.)



2.) Equality Logic and Uninterpreted Functions 6+7 Points

i) Consider the equality logic formula

$$\varphi^{EQ} \quad := \quad \neg(\underbrace{x_1 = x_2}_{e_1}) \quad \vee \quad \left(\underbrace{x_1 = x_3}_{e_2} \quad \wedge \quad \underbrace{x_1 = x_4}_{e_3}\right) \quad \vee \quad \left(\underbrace{x_2 = x_3}_{e_4} \quad \wedge \quad \underbrace{x_2 = x_4}_{e_5}\right)$$

Draw the E-graph **with polarity** and use it (without any modification) to transform φ^{EQ} to a satisfiability-equivalent propositional logic formula φ as presented in the lecture for **eager** SMT solving.

ii) Consider the following conjunction of constraints in equality logic with uninterpreted functions:

$$a = b \quad \wedge \quad f(f(a)) = a \quad \wedge \quad f(f(b)) \neq b$$

Check the satisfiability of the above conjunction using the sparse method from the lecture. Please describe the partition

- initially,
- after considering transitivity and
- after considering congruence,

and explain how we can read off the answer to the satisfiability question.

3.) Fourier-Motzkin Variable Elimination **5+6+2+2 Points**

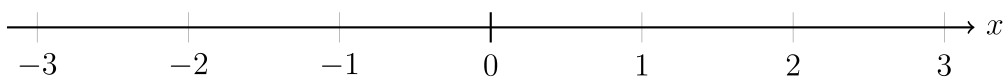
Consider the following set of linear real-arithmetic constraints:

$$
\begin{array}{rrcrcr}
c_1: & 2x & + & y & \leq & 2 \\
c_2: & x & - & y & \geq & -2 \\
c_3: & x & - & 2y & \leq & 1 \\
c_4: & -x & - & y & \leq & 2
\end{array}
$$

i) Sketch the **solution set** of this problem graphically in a coordinate system.

ii) **Eliminate** y from this constraint set by applying the Fourier-Motzkin variable elimination. Simplify the resulting constraints such that they define constant bounds on x.

iii) Sketch the **solution set** of the resulting constraints in x graphically.



iv) What is the relation between the two solution sets, that you depicted in i) and iii)?

4.) Simplex **12+3 Points**

i) Apply one **pivoting step** and update the current assignment, where the variable order is $x_1 < x_2 < s_1 < s_2 < s_3$ and where the current tableau, the bounds and the current assignment are given by:

	s_1	s_2
x_1	$1/2$	$1/2$
x_2	$1/2$	$-1/2$
s_3	$1/2$	-1

$$\begin{aligned} s_1 &\geq 2 \\ s_2 &\leq -2 \\ s_3 &\geq 5 \end{aligned}$$

$$\begin{aligned} \alpha(x_1) &= 0 \\ \alpha(x_2) &= 2 \\ \alpha(s_1) &= 2 \\ \alpha(s_2) &= -2 \\ \alpha(s_3) &= 3 \end{aligned}$$

ii) The simplex method terminates on the resulting tableau. Is the tableau satisfied or conflicting? Why? Give the satisfying assignment for the original variables or the slack variables corresponding to the infeasible subset.

5.) Integer Arithmetic **10 Points**

In the lecture, we presented a Branch and Bound algorithm in pseudocode, which used recursive calls to itself. The following pseudocode describes an alternative algorithm, which uses **iteration** instead of recursion. In the code,

- each value of type `ConstraintSet` stores a set of linear constraints over variables V,
- `n` is an integer,
- `A` is an infinite array of constraint sets and
- the method's return value as well as the variable `relaxed` are of type `Result`, storing either the value `UNSAT` or a variable assignment $f : V \to \mathbb{Q}$.

Please specify the **five missing details**, such that the recursive and the iterative algorithms always give the same result to each input. *Note: when branching, the recursive method processes the lower (i.e. upper-bounded) branch first.*

```
Result Branch&Bound(ConstraintSet P) {

  int n;
  ConstraintSet[] A;
  Result relaxed;

  n := 1; A[n] := P;
  while (n>0) do {
    P := A[n]; n := n-1;
    relaxed = LRA(Relaxed(P));
    if ([          1          ]){
      if (isInteger(relaxed)) [          2          ];
      let v be a variable such that relaxed(v) ∉ ℤ;
      n := n+1; A[n] := [          3          ];
      n := n+1; A[n] := [          4          ];
    }
  }
  return [          5          ];
}
```

6.) Interval Constraint Propagation **4+9 Points**

i) Specify the formal rule for **squaring** an arbitrary interval as presented in the lecture, without using arithmetic operators for intervals.

ii) As an advanced contraction method, the **interval Newton** method was presented in the lecture. Consider the polynomial $f(x) = {}^1\!/_3 \cdot x^3 - x^2 + 1$ and the starting interval $A = [1; 2]$ for x with $f'(A) = [-3; 2]$. Let $s = 1$ be a sample point from A. Perform one Newton contraction step.

7.) Subtropical Satisfiability **2+4+4 Points**

i) Please specify the **frame** *frame*(p) of the polynomial

$$p(a, b, c) = -5a^4 + 3ab^4 - abc - 3 \ .$$

ii) Specify a formula using the variables n_x, n_y, n_u and b that encodes the **separating hyperplanes** for the only positive frame element of

$$p(x, y, u) = -x^5 + 2x^2y^4 - 5u - xyu \ .$$

iii) Specify conditions on the coefficients to describe the exact set of all *univariate* polynomials $p \in \mathbb{Z}[x]$, for which the subtropical satisfiability method as presented in the lecture *cannot* find a solution for $p(x) > 0$.

8.) Virtual Substitution **(6+4)+2 Points**

i) Assume we want to virtually substitute the test candidate $t = y^2 + \epsilon$ **for the variable** $\boldsymbol{x}$ in the constraint $x \cdot y - y^3 + 2 \cdot x \cdot y^2 \leq 0$ using one of the following rules:

- Substitute $e + \varepsilon$ for x in $b \cdot x + c \leq 0$:

$$
\begin{array}{lllll}
 & (& (bx + c < 0)[e//x] & &) \\
\vee & (& (bx + c = 0)[e//x] & \wedge \quad b < 0 &) \\
\vee & (& b = 0 & \wedge \quad c = 0 &)
\end{array}
$$

- Substitute $e + \varepsilon$ for x in $a \cdot x^2 + b \cdot x + c \leq 0$:

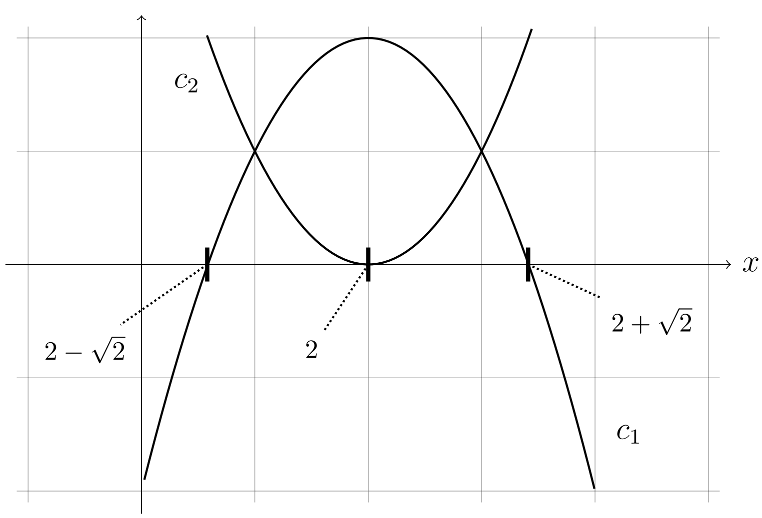
$$
\begin{array}{lllllll}
 & (& (ax^2 + bx + c < 0)[e//x] & & & &) \\
\vee & (& (ax^2 + bx + c = 0)[e//x] & \wedge & (2ax + b < 0)[e//x] & &) \\
\vee & (& (ax^2 + bx + c = 0)[e//x] & \wedge & (2ax + b = 0)[e//x] & \wedge \quad 2a < 0 &) \\
\vee & (& a = 0 & \wedge & b = 0 & \wedge \quad c = 0 &
\end{array}
$$

a) Identify the appropriate substitution rule from above and apply virtual substitution. Normalize all polynomials by transforming them to sums of products.

b) Further simplify the result as far as possible. Use the result to state whether the formula is satisfiable.

ii) Consider the following constraints and their graphical depiction below. Give the test candidates for x resulting from these constraints with **true** side conditions as symbolic expressions.

$$c_1 : -(x-2)^2 + 2 > 0 \qquad\qquad c_2 : x^2 - 4x + 4 \leq 0$$



9.) Cylindrical Algebraic Decomposition **5+7 Points**

Consider the polynomial $p = x^3 + 2 \cdot x^2 - 3 \cdot x$.

i) Use the Cauchy bound $C = 4$ and the Sturm sequence of p:

$$\begin{aligned} p_0 &= x^3 + 2 \cdot x^2 - 3 \cdot x \\ p_1 &= 3 \cdot x^2 + 4 \cdot x - 3 \\ p_2 &= {}^{26}\!/_{9} \cdot x - {}^{2}\!/_{3} \\ p_3 &= {}^{324}\!/_{169} \end{aligned}$$

to determine the number of real roots of p as presented in the lecture.

ii) Isolate all real roots of p with the method presented in the lecture, using interval midpoints for splitting. Give the resulting interval representations of the real roots.