Satisfiability Checking - WS 2019/2020

Written Exam I

Saturday, February 22, 2020

Forename and surname:	**Matriculation number:**
Sign here:	

- Do not open the exam until we give the start signal.
- Please place your student identity card on your desk for identification purposes.
- The duration of the exam is 120 minutes.
- Use a blue or black (permanent) pen only.
- Please write your name and matriculation number on each page of this exam.
- Please write clear and legible answers.
- Please use the space below each task to solve it. If you need more sheets, indicate this by a hand signal.
- 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. In particular, switch off your electronic devices! Cheating disqualifies from the exam.

Task:	1.)	2.)	3.)	4.)	5.)	6.)	7.)	8.)	9.)	**Total**
Maximum score:	22	23	14	15	9	9	8	9	11	120
Reached score:										

Good luck!

1.) SAT Checking **14+8 Points**

i) Consider the following four variants of the CDCL algorithm. Which of these are both sound and complete (for checking the satisfiability of propositional logic formulas in CNF)? For each variant, please either give a short argument why it is sound and complete, or provide a counterexample formula showing that it is unsound or incomplete. In your arguments you can rely on soundness and completeness of the CDCL algorithm presented in the lecture.

```
if !BCP() then
    return UNSAT
while true do
    if !decide() then
        return SAT
    while !BCP() do
        if !resolve_conflict() then
            return UNSAT
```

(a) Variant A

```
if !BCP() then
    return UNSAT
while true do
    if !decide() then
        return SAT
    if !BCP() then
        if !resolve_conflict() then
            return UNSAT
```

(b) Variant B

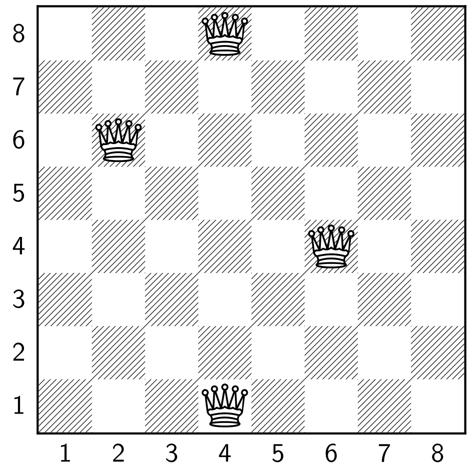
```
while true do
    while !BCP() do
        if !resolve_conflict() then
            return UNSAT
    if !decide() then
        return SAT
```

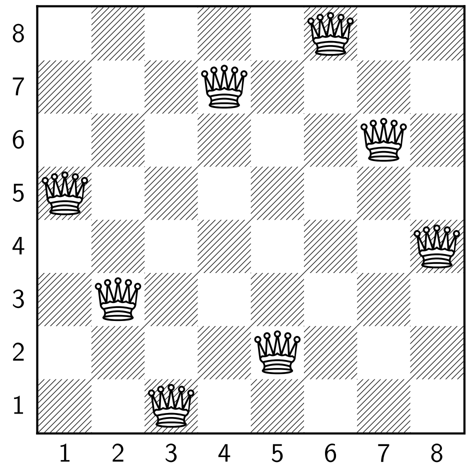
(c) Variant C

```
while true do
    if !decide() then
        return SAT
    while !BCP() do
        if !resolve_conflict() then
            return UNSAT
```

(d) Variant D

ii) A position $(c, r) \in \{1, \ldots, n\}^2$ on an $n \times n$ chess board is identified by a column index c and a row index r. We say that a queen on a chess board threatens another one (and vice versa) if they are in the same row, column or diagonal. The n *queens problem* poses the question how to place n (chess) queens on an $n \times n$ chess board in such a way that no two queens threaten each other. In the left example below, the queens at positions `(4,8)` and `(2,6)` as well as at `(4,8)` and `(4,1)` threaten each other. On the right, you can see a satisfying solution for the *eight queens problem*.





Encode for the n queens problem the following statements as formulae, using the Boolean variables x_{ij} to encode whether there is a queen at position (i, j):

(a) At least one queen is in column i.
$\varphi_{col}(i) =$

(b) At least one queen is in row j.
$\varphi_{row}(j) =$

(c) A queen at position (i, j) is threatened.
$\varphi_{threat}(i, j) =$

Using the above formulae, please build a formula whose models encode the solutions to the n queens problem.

2.) SMT Solving **23 Points**

Consider the equality logic formula

$$\begin{array}{rcll}
\varphi^{EQ} & := & (& x_1 = x_2 \quad \vee \quad \neg(x_3 = x_4) \quad) \\
& \wedge & (& x_1 = x_2 \quad \vee \quad \neg(x_1 = x_4) \quad \vee \quad x_1 = x_3 \quad) \\
& \wedge & (& x_2 = x_3 \quad \vee \quad x_1 = x_4 \quad) \\
& \wedge & (& \neg(x_2 = x_3) \quad \vee \quad x_1 = x_3 \quad)
\end{array}$$

with the Boolean abstraction

$$\begin{array}{rcl}
\varphi^{skel} & := & c_1 : (\quad a \quad \vee \quad \neg b \quad) \\
& \wedge & c_2 : (\quad a \quad \vee \quad \neg d \quad \vee \quad e \quad) \\
& \wedge & c_3 : (\quad c \quad \vee \quad d \quad) \\
& \wedge & c_4 : (\quad \neg c \quad \vee \quad e \quad)
\end{array}$$

Show how a *less-lazy* SMT solver solves φ^{EQ} for satisfiability as presented in the lecture. Stop the computations after the second conflict has been resolved. Describe

- the initial watch lists of the SAT solver,
- all propagations and decisions of the SAT solver,
- the watch lists of the SAT solver after every decision level,
- the theory call(s) and
- the conflict resolutions in the SAT solver.

Assume that

- in each original clause initially the first two literals are watched,
- the SAT solver uses the lexicographically smallest unassigned variable for a decision and assigns *false* to it,
- all assigned constraints (the true constraints and the negations of false constraints) are passed to the theory solver (i.e. no constraints are dropped because only one polarity is present).

Name: Student number:

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Assignments:

	1.	2.	3.	4.	5.
DL0					
DL1					
DL2					

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Theory call:

Equalities

x_1

x_4 x_2

x_3

Disequalities

Conflict

☐ yes ☐ no

Conflict clause:

Conflict resolution:

Newly learnt clause:

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Assignments:

	1.	2.	3.	4.	5.
DL0					
DL1					
DL2					

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Name: Student number:

Theory call:

Equalities Disequalities Conflict

x_1

☐ yes ☐ no

x_4 x_2

Conflict clause:

x_3

Conflict resolution: Newly learnt clause:

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Assignments:

	1.	2.	3.	4.	5.
DL0					
DL1					
DL2					

Watch lists: ☐ as before

a:	b:	c:	d:	e:
$\neg a$:	$\neg b$:	$\neg c$:	$\neg d$:	$\neg e$:

Theory call:

Equalities Disequalities Conflict

x_1

☐ yes ☐ no

x_4 x_2

Conflict clause:

x_3

Conflict resolution: Newly learnt clause:

3.) Fourier-Motzkin Variable Elimination **4+6+2+2 Points**

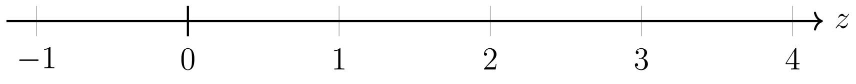
Consider the following set of linear real-arithmetic constraints:

$$\begin{array}{rrcrcrcr}
c_1: & -x & & & + & 3z & \leq & 6 \\
c_2: & & & & - & z & \leq & 5 \\
c_3: & x & - & 2y & + & 2z & \geq & 0 \\
c_4: & x & - & y & & & = & -1 \\
c_5: & -2x & + & y & + & z & \leq & 2 \\
c_6: & x & - & 2y & & & \geq & -4
\end{array}$$

i) *Eliminate* x by applying Gauß variable elimination. Bring the resulting constraint set into *matrix form* $A \cdot x \leq b$.

ii) *Eliminate* y from the resulting constraint set by applying the Fourier-Motzkin algorithm.

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



iv) Give a *satisfying assignment* for all variables.

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

i) Apply one *pivoting step* and update the current assignment, where the variable order is $x_1 < x_2 < s_1 < s_2 < s_3 < s_4$ and where the current tableau, the bounds and the current assignment are given by:

	s_1	s_3
s_2	1	2
x_2	-2	0
s_4	3	-1
x_1	2	-1

$$\begin{array}{lcr} s_1 & \geq & 3 \\ s_2 & \leq & -1 \\ s_3 & \leq & -1 \\ s_4 & \leq & 3 \end{array}$$

$$\begin{array}{ll} \alpha(s_1) & = 3 \\ \alpha(s_2) & = 1 \\ \alpha(s_3) & = -1 \\ \alpha(s_4) & = 10 \\ \alpha(x_1) & = 7 \\ \alpha(x_2) & = -6 \end{array}$$

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 infeasible subset.

5.) Integer Arithmetic **6+3 Points**

Branch&Bound was presented as an incomplete method for linear integer arithmetic.

i) Give the core algorithm of Branch&Bound as pseudocode.

ii) Can we apply Branch&Bound to nonlinear integer arithmetic as well?
Sketch how the pseudocode can be adapted, or argue why this can not be done.

6.) Interval Constraint Propagation **6+1+2 Points**

i) Apply *interval arithmetic* as presented in the lecture to compute the following:

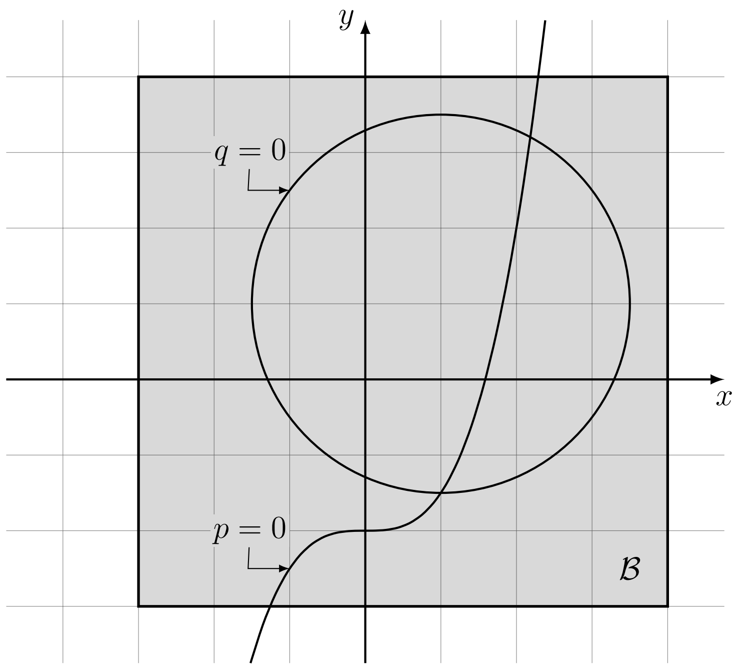
a) $[1;2] - [-3;4] + [1;3] =$

b) $[3;5] \cdot [-2;2] =$

c) $[4;5]/(-\infty;-2] =$

d) $[1;2] + [-3;4] \cdot [5;6] =$

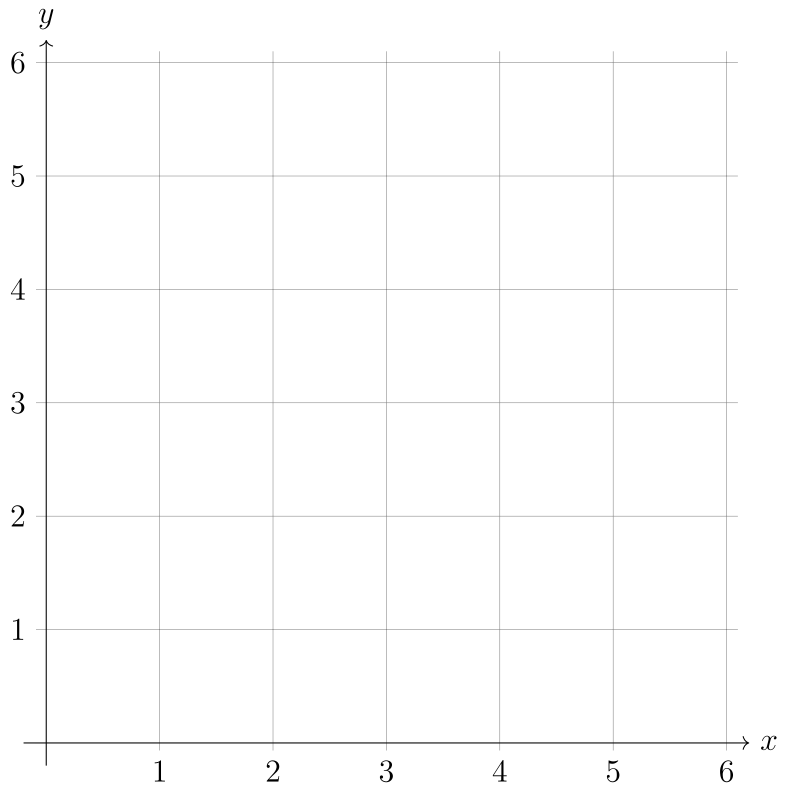
ii) Assume two constraints $p(x, y) = 0$ and $q(x, y) = 0$ in the variables x and y and an initial box $\mathcal{B} \subseteq \mathbb{R}^2$ restricting the values of the variables x and y, as shown below. Sketch *the smallest box* that contains all common solutions to both constraints inside $\mathcal{B}$.



iii) Why is the ICP algorithm as presented in the lecture designed to be able to return UNSAT but not designed to return SAT? What does the ICP algorithm do instead of returning SAT?

7.) Subtropical Satisfiability **8 Points**

Consider the polynomial $f(x, y) = 2xy - 2xy^5 - 3x^2y^3 + x^3y^2 + x^3y^4 - 2x^4y^5 + 7x^5y^3 - 6x^5y$. Show the points of both *frame*$^+(f)$ and *frame*$^-(f)$ in the graphic below.



Graphically identify a hyperplane that separates some $p \in$ *frame*$^+(f)$ from the remaining frame and give its normal vector n. Use n to provide a $f^*(a)$ such that $f^*(a) > 0$ for a sufficiently large a.

8.) Virtual Substitution **6+3 Points**

i) Assume we want to virtually substitute the test candidate $t = 2 + \epsilon$ *for the variable* y in the constraint $2 \cdot x^2 \cdot y - 3 \cdot x^2 > 0$. Please identify the appropriate substitution rule from below, apply virtual substitution and simplify the result as far as possible.

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

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

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

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

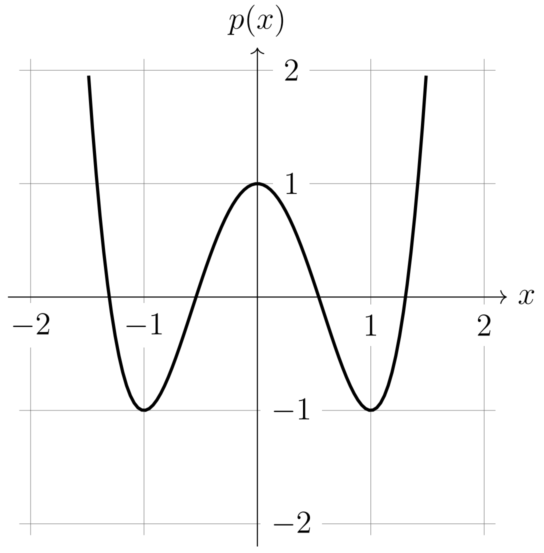
- Substitute $e + \varepsilon$ for x in $a \cdot x^2 + b \cdot x + c > 0$:

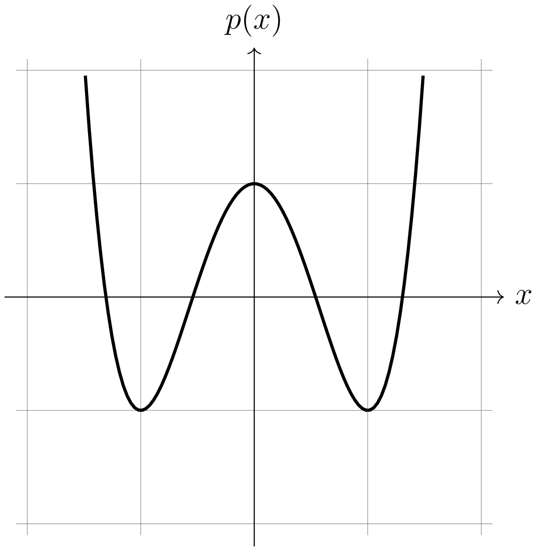
$$\begin{array}{cclclcll} & (& (ax^2 + bx + c > 0)[e//x] & & & & &) \\ \vee & (& (ax^2 + bx + c = 0)[e//x] & \wedge & (2ax + b > 0)[e//x] & & &) \\ \vee & (& (ax^2 + bx + c = 0)[e//x] & \wedge & (2ax + b = 0)[e//x] & \wedge & (2a > 0)[e//x] &) \end{array}$$

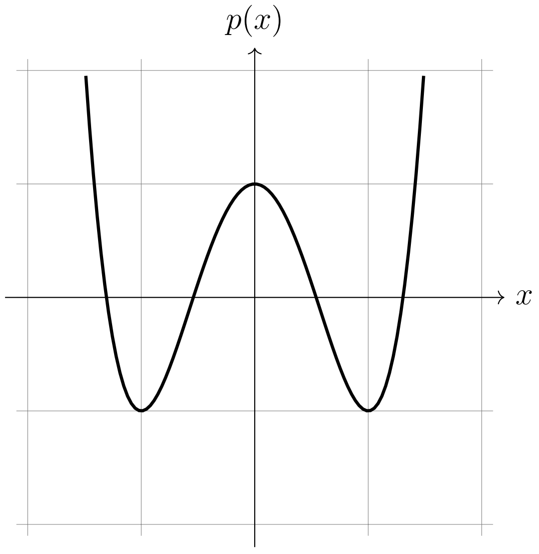
ii) Give a univariate constraint whose test candidates are $-\infty$, $-1+\varepsilon$ and $1+\varepsilon$. Show how you derive it.

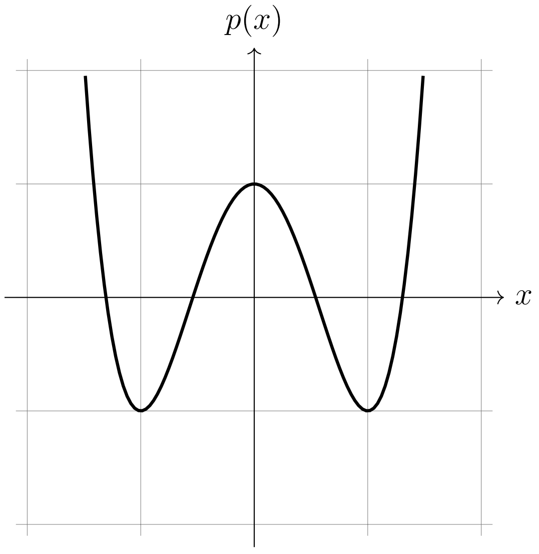
9.) Cylindrical Algebraic Decomposition **7+4 Points**

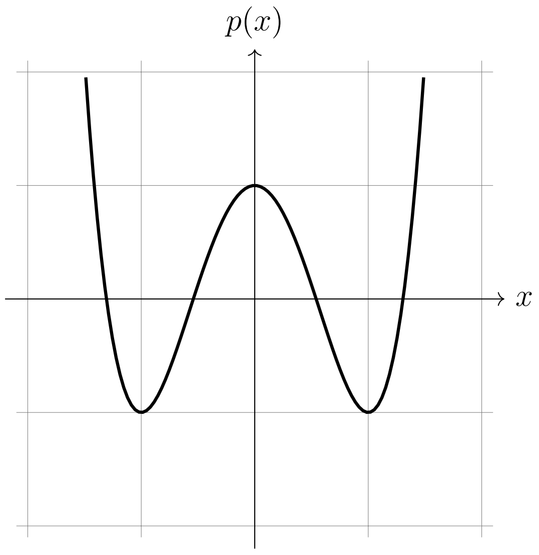
i) Isolate all real zeros of the univariate polynomial $p = 2x^4 - 4x^2 + 1$ in the interval $(-2, 2)$ with the method presented in the lecture, using interval midpoints for splitting. You do not need to compute Sturm sequences, you can read off all needed information from the plots below. After each step, please depict all pending intervals and indicate when a root has been isolated (there are more plots than necessary).

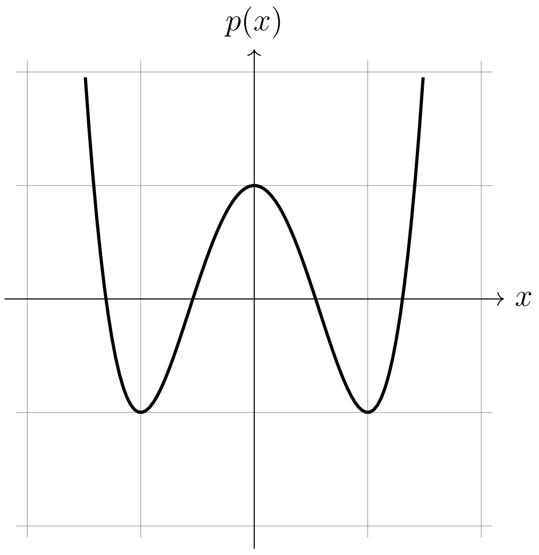












ii) One of the utility methods for the cylindrical algebraic decomposition method presented in the lecture are *Sturm sequences*. What is the *input* of this method and what does it compute?