

Collection of Exam Questions and Things to Remember

The document contains a brief collection of almost all Question-Types that have been asked in previous exams with solutions. Additionally, it provides a small chapter on important things that should be remembered for the exam. Note however, that this document doesn't make the old exams obey, it should rather build up upon them. :D

Contents

1. Things to Remember	2
2. SAT-Checking	5
3. Equality Logic and Uninterpreted Functions	8
4.1. Eager SMT-Solving	8
4.2. Less-Lazy SMT Solving	10
4. Bitvector Arithmetic	12
5. Fourier-Motzkin Variable Elimination	13
6. Simplex	14
7. Integer Arithmetic	15
8. Interval Constraint Propagation	16
9. Subtropical Satisfiability	17
10. Virtual Substitution	19
11. Cylindrical Algebraic Decomposition.....	21

1. Things to Remember

Some general things to look out for in the exam

- Be precise with the wording – i.e., use “univariate polynomial” instead of “polynomial” and “open interval” instead of “interval”, “positive number” instead of “number”, and “real root” instead of “root”!
- Watch out the variable ordering! As it can vary, like $x < y$ and pick lowest/highest first, or something like xy^2 vs y^2x (i.e., when constructing newton polytope!)

Decision Heuristics

Naive Decision: Static

- Stick to the given variable and sign ordering.

Dynamic Largest Individual Sum (DLIS): Dynamic

- For each literal l , let C_l be the number of unresolved clauses (= not already SAT)
- Decide for the literal with highest C_l

Jeroslow-Wang: Static

- For each literal l compute $J(l) = \sum_{\text{clause } c \text{ in CNF containing } l} 2^{|-c|}$
This gives **exponentially higher weights** to literals in **shorter clauses**
- Choose a literal l that maximizes $J(l)$

Variable State Independent Decaying Sum (VSIDS): Quasi-Static

- Each literal has a counter initialized to 0.
- When resolution gets applied to a clause, the counters of the involved literals are increased.
This gives **priority** to variables **involved in recent conflicts**.
- Decide for the literal with the highest counter.
- Periodically, all the counters are divided by a constant.

Terminology

- Variable = “Atomic” unit
- Literal positive/negated variable
- **Term** = **Conjunction** of Literals ($a \wedge b$)
- **Clause** = **Disjunction** of Literals ($a \vee b$)
- **Valid** (= Tautology), that is, all assignments satisfy φ
- **Satisfiable** = There is at least one assignment where φ becomes true
- **Unsatisfiable** = No assignment makes φ true

Eager vs. Lazy SMT-Solving

Eager (Theory first):

- Transform into satisfiability-equivalent formula, then apply SAT-Solving.

Lazy (Theory later):

- Find satisfying assignment to Boolean skeleton using SAT-Solving, afterwards check whether the assignment is consistent with our theory.
 - o Full Lazy: Only check full assignments.
 - o Less Lazy: Also check partial assignments (typically after each decision level).

Making an E-Graph Chordal

Pick one single node and connect it to all others.

Intuitively, a graph is chordal if it consists of only “triangles” (= smallest circles)

Determining Minimal Infeasible Subsets

Full-Lazy SMT-Solving:

- Resolution

Less-Lazy SMT Solving (Theory Solver):

- Create the equality graph for all received equalities (ignore disequalities!) and find a shortest path from one conflicting edge to the other (i.e., if $\neg(x_1 = x_2)$ is conflicting, then find a shortest path from x_1 to x_2)
- Note that the following SAT-Solver receives this subset, abstracts it into a clause and makes it asserting:
 $\{x_1 \neq x_2, x_1 = x_4, x_2 = x_4\}$ means NOT all of them can be true at the same time. Thus, our abstracted clause is:

$$\neg(\neg(x_1 = x_2) \wedge (x_1 = x_3) \wedge (x_2 = x_4)) \equiv \underbrace{(x_1 = x_2)}_{a_1} \vee \neg \underbrace{(x_1 = x_3)}_{a_2} \vee \neg \underbrace{(x_2 = x_4)}_{a_3}$$

Fourier-Motzkin:

- Each conflicting constraint (i.e., $1 \leq -1/3$) generates an infeasible subset.
- We can also look up which terms were involved in generating this conflicting constraint (i.e., $x \geq 1$ and $x \leq -1/3$).
- Doing this repeatedly finally gives an infeasible subset of the original constraints.

Simplex:

- For each basic variable that violates its bound (conflicting) and that cannot be pivoted anymore: Select the basic variable itself, and all non-basic variables in this row with non-zero coefficients to get an infeasible subset.
- Additionally, we can look up the original constraints that the auxiliary variables s_i encode.
- Subsequently, there might be more than one infeasible subset.

CAD:

- We can compute an infeasible subset based on the sample points. For every sample point we collect one constraint that conflicts with this sample point. The set of all these constraints is an infeasible subset.

Watched Literal

A pair is suited to be watched if:

- at least one variable is true, or
- both variables are unassigned.

Valid Σ -formulas

- Functions and predicates must have the correct number of arguments.
- Variables and constants cannot have arguments.

Signed / Unsigned

For signed, we only consider the FIRST bit.

That is, $\langle 010 \rangle_S = 2$, whereas $\langle 110 \rangle_S = -4 + 2 = -2$

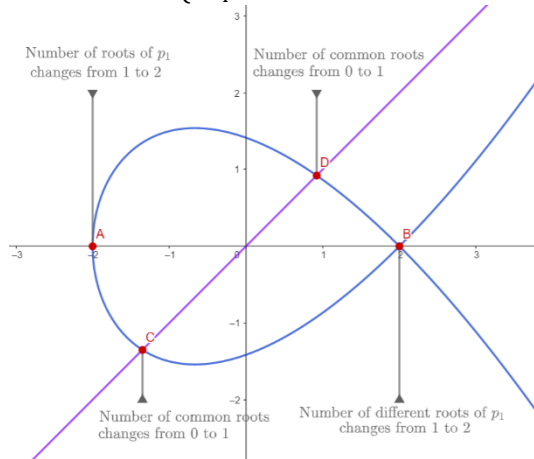
Delineability

A set of polynomials $P = \{p_1, \dots, p_n\}$ is delineable on R if for each $i, j \leq n$ ($i \neq j$):

- The number of roots of p_i is constant,
- The number of different roots of p_i is constant, and
- The number of common roots of p_i and p_j is constant.

Example

Consider $P = \left\{ -\frac{1}{4}(x-2)^3 = (x-2)^2 - y^2, x-y \right\}$



DPLL + CDCL Algorithm (“Default” SAT-Solving Algorithm)

```

if (!BCP()): return UNSAT;
while (true):
    // Choose the next variable & value, return false if all are assigned
    if (!decide()): return SAT;
    // Boolean Constraint Propagation, return false if conflict
    while (!BCP()):
        // Conflict resolution and Backtracking, return false if impossible
        if (!resolve_conflict()): return UNSAT;
    
```

Branch and Bound

Let S be an integer linear system

Branch-And-Bound(S) :

```

// Try to find a (possibly real = relaxed) solution to our problem
Solution = Solver(relaxed(S));
// Relaxed Problem UNSAT ⇒ Original Problem must be UNSAT
if (Solution == UNSAT): return UNSAT;
// If we've found an integer-solution: Everything fine
else if (Solution is Integer): return SAT;
// Otherwise, recursively Branch and Bound
else:
    select a variable  $v$  that is assigned a non-integer value  $r$ 
    if (Branch-And-Bound ( $S \cup \{v \leq \lfloor r \rfloor\}$ ) == SAT): return SAT;
    else if (Branch-And-Bound ( $S \cup \{v \geq \lceil r \rceil\}$ ) == SAT): return SAT;
    else: return UNSAT;
    
```

2. SAT-Checking

Enumeration and Resolution

Enumeration

- Check all assignments whether they are satisfying.
- A decision procedure would iterate over all possible assignments, substitute them into the formula, and check whether the formula evaluates to true.

Resolution

- Resolution can be used to eliminate a proposition x from a set C of clauses.
- We can iteratively resolve all possible clauses and test whether we obtain true or false.

CDCL (with Watch-List)

Consider the following propositional logic formula:

$$c_1: (b \vee c) \wedge c_2: (\neg c) \wedge c_3: (\neg b \vee c \vee d \vee \neg e) \wedge c_4: (\neg a \vee \neg b \vee e) \wedge c_5: (\neg a \vee \neg d)$$

Apply CDCL SAT-Checking algorithm to the given formula until the first conflict appears. When making a decision, take the lexicographically smallest unassigned variable and assign true to it. Show the initial watch list with the first two literals for each non-unary clause. Show the full list of assignments and updated watch lists after each decision level.

Solution

Initial Watch List

Literal	a	$\neg a$	b	$\neg b$	c	$\neg c$	d	$\neg d$	e	$\neg e$
Watched		c_4, c_5	c_1	c_3, c_4	c_1, c_3			c_5		

Assignments

	1.	2.	3.	4.	5.
DL0	$c = 0$	$b = 1$			
DL1	$a = 1$	$e = 1$	$d = 0$		
DL2					
DL3					

DL0: $\neg c$: NULL, b : c_1

Check watch list of c :

Propagate c in c_1 : b : c_1

Propagate c in c_3 : Watch d instead of c

Check watch list of $\neg b$:

Propagate $\neg b$ in c_3 : Watch $\neg e$ instead of $\neg b$

Propagate $\neg b$ in c_4 : Watch e instead of $\neg b$

Updated Watch List

Literal	a	$\neg a$	b	$\neg b$	c	$\neg c$	d	$\neg d$	e	$\neg e$
Watched		c_4, c_5	c_1		c_1		c_3	c_5	c_4	c_3

DL1: a : NULL, e : c_4 , $\neg d$: c_5

Check watch list of $\neg a$:

Propagate $\neg a$ in c_4 : e : c_4

Propagate $\neg a$ in c_5 : $\neg d$: c_5

Check watch list of $\neg e$:

Propagate $\neg e$ in c_3 : Conflict!

Watch list remains unchanged, the assignments are given in the initial table.

CDCL (without Watch-List)

Apply 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 it to true.

$$c_1: (\neg a \vee e) \wedge c_2: (\neg a \vee d \vee \neg e) \wedge c_3: (\neg b \vee \neg c) \wedge c_4: (\neg a \vee c \vee f) \wedge c_5: (\neg a \vee \neg b \vee \neg f)$$

Solution

DL0: –

Nothing to do as there are no unary clauses.

DL1: $a: \text{NULL}$, $e: c_1$, $d: c_2$

Decide for a

Propagate $\neg a: e: c_1$

Propagate $\neg e: d: c_2$

Nothing more to propagate.

DL2: $b: \text{NULL}$, $c: c_3$, $f: c_4$

Decide for b

Propagate $\neg b: \neg c: c_3$

Propagate $\neg c: f: c_4$

c_5 is conflicting because $\alpha(a) = \alpha(b) = \alpha(f) = 1$

Example of Conflict Resolution:

$$c_5: (\neg a \vee \neg b \vee \neg f) \quad c_4: (\neg a \vee c \vee f)$$

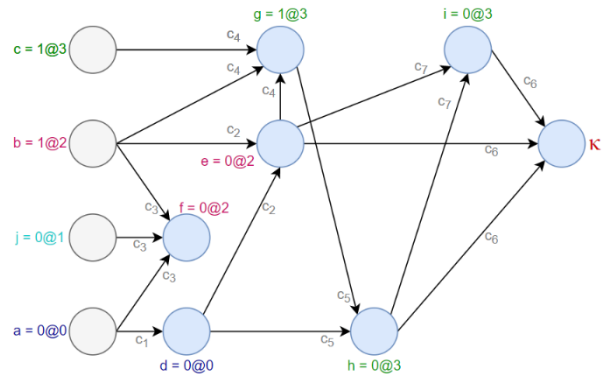
$$\frac{c_6: (\neg a \vee \neg b \vee c) \quad c_3: (\neg b \vee \neg c)}{c_7: (\neg a \vee \neg b)}$$

Note that we must resolve twice, because tc_6 isn't asserting yet, since b and c are both from DL2.

Implication Graph

Given the following implication graph, answer

- What is the conflicting clause?
- Which clause is generated as a result of conflict resolution in CDCL? Which is the first UIP?
- Give the node labels of all further UIPs
- Which decisions from the implication graph were sufficient to cause this conflict?



Solution

- $c_6: (i \vee e \vee h)$ is the conflicting clause. How do we know it? Well, they lead into a conflict with the given assignment $\alpha(e) = \alpha(h) = \alpha(i) = 0$. Subsequently, at least one of the must be true in order to resolve the conflict. Thus, the clause must be $c_6: (i \vee e \vee h)$
- The first UIP is h (= nodes that all paths from current decision c to conflict κ must cross) For conflict resolution, we would first resolve $c_6: (i \vee e \vee h)$ with $c_7: (e \vee h \vee \neg i)$. This would give us $(e \vee h)$ which is asserting, as only g is from current decision level.
- The other UIPs are $g = 1@3$ and $c = 1@3$
- We observe that f isn't involved in any "useful" implications. Subsequently, $\{a = 0, b = 1, c = 1\}$ would be sufficient to cause this conflict.

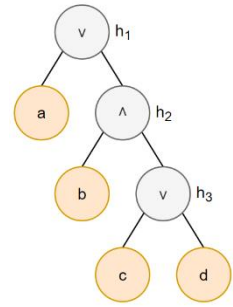
Tseitin Encoding

Encode $\varphi := a \vee (b \wedge (c \vee d))$ using Tseitin's Encoding.
Additionally, transform it into CNF.

Solution

Construct the tree as a helper (see on the left).

$$h_1 \wedge (h_1 \leftrightarrow (a \vee h_2)) \wedge (h_2 \leftrightarrow (b \wedge h_3)) \wedge (h_3 \leftrightarrow (c \vee d))$$



General Forms of Converting Tseitin to CNF:

$$\begin{aligned} a \leftrightarrow (b \vee c) &\equiv (a \rightarrow (b \vee c)) \wedge ((b \vee c) \rightarrow a) \equiv (\neg a \vee (b \vee c)) \wedge (\neg(b \vee c) \vee a) \\ &\equiv (\neg a \vee b \vee c) \wedge ((\neg b \wedge \neg c) \vee a) \equiv (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c) \\ a \leftrightarrow (b \wedge c) &\equiv (a \rightarrow (b \wedge c)) \wedge ((b \wedge c) \rightarrow a) \equiv (\neg a \vee (b \wedge c)) \wedge (\neg(b \wedge c) \vee a) \\ &\equiv (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c) \end{aligned}$$

Plugging it in gives us:

$$\begin{aligned} h_1 \wedge (\neg h_1 \vee a \vee h_2) \wedge (h_1 \vee \neg a) \wedge (h_1 \vee \neg h_2) \wedge (\neg h_2 \vee b) \wedge (\neg h_2 \vee h_3) \wedge (h_2 \vee \neg b \vee \neg h_3) \\ \wedge (\neg h_3 \vee c \vee d) \wedge (h_3 \vee \neg c) \wedge (h_3 \vee \neg d) \end{aligned}$$

Encode Problems in Propositional Logic

Consider a normal 9x9 Sudoku. Encode the following properties:

1. φ_{digit} : Exactly one digit is assigned to every cell.
2. $\varphi_{row}(i)$: No digit appears twice in row i

Solution

1. In each cell must be one digit, and this digit must not be in any other cell.

$$\varphi_{digit} := \bigwedge_{i=1}^9 \bigwedge_{j=1}^9 \bigvee_{k \in \{1, \dots, 9\}} \left(x_{ijk} \wedge \bigwedge_{k' \in \{1, \dots, 9; k \neq k'\}} \neg x_{ijk'} \right)$$

2. If it is in a certain column, then it cannot be in any other column:

$$\varphi_{row}(i) := \bigwedge_{k \in \{1, \dots, 9\}} \bigwedge_{j=1}^9 \left(x_{ijk} \rightarrow \bigwedge_{j'=1, j \neq j'}^9 \neg x_{ijk'} \right)$$

Resolution

Apply resolution to eliminate the propositions in the order A, B, C, D from the following propositional logic formula in CNF:

$$(A \vee \neg B \vee C) \wedge (B \vee D) \wedge (\neg A \vee \neg B \vee \neg C \vee D) \wedge (\neg A \vee C \vee D)$$

Solution

Resolve A : $\underbrace{(\neg B \vee C \vee \neg C \vee D)}_{\text{trivially true}} \wedge (\neg B \vee C \vee D) \wedge (B \vee D)$

Resolve B : $\underbrace{(C \vee \neg C \vee D)}_{\text{trivially true}} \wedge (C \vee D)$

Resolve C : $()$

Resolve D : $()$

3. Equality Logic and Uninterpreted Functions

Eager vs. Lazy SMT Solving

Eager SMT-Solving: “Theory First”

- Transform logical formula over some theories into satisfiability-equivalent propositional logic formulas and apply SAT-Solving.

Lazy SMT: “Theory Later”

- Find a solution to the Boolean Skeleton, then check whether it is consistent with our theory.
- Full-Lazy = Only check full assignments, Less-Lazy = Also check partial assignments
- Conditions for Less-Lazy:
 - o **Incrementality**: To make computing efficient, we want to be able to add the new constraints on the foundation of previous calculations, instead of calculating everything fresh from ground up.
 - o (Possibly Minimal) **Infeasible Subset**: Generate an explanation for UNSAT.
 - o **Backtracking**: Store the steps in reverse chronological order.

4.1. Eager SMT-Solving

Ackermann’s Reduction

Use Ackermann’s Reduction to construct an equality logic formula φ^E without UFs that is satisfiability-equivalent to φ^{UF} .

$$\varphi^{UF} := x_1 = x_2 \wedge (F(x_1, x_3) \neq F(x_1, x_2) \vee F(x_1, x_3) = F(x_1, x_1))$$

Solution

Assign indices to the F -instances: $f_1 := F(x_1, x_3)$, $f_2 := F(x_1, x_2)$, $f_3 := F(x_1, x_1)$

Boolean Skeleton: $\varphi_{sk} = x_1 = x_2 \wedge (f_1 \neq f_2 \vee f_1 \neq f_3)$

Encode Congruence: $\varphi_{cong} = (x_1 = x_1 \wedge x_3 = x_2) \rightarrow f_1 = f_2$
 $\wedge (x_1 = x_1 \wedge x_3 = x_1) \rightarrow f_1 = f_3$
 $\wedge (x_1 = x_1 \wedge x_2 = x_1) \rightarrow f_2 = f_3$

Final Formula: $\varphi^E := \varphi_{sk} \wedge \varphi_{cong}$

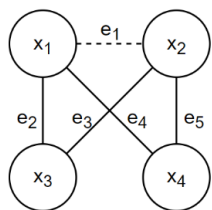
Sparse-Method (Bryant) with **Polar E-Graphs**: Reduce to propositional logic formula

Consider the equality logic formula:

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

Draw the **E-Graph with polarity** and use it to transform φ^{EQ} to a satisfiability-equivalent propositional logic formula φ as presented in the lecture for **eager** SMT solving.

Solution



$$\varphi_{sk} = \neg e_1 \vee (e_2 \wedge e_3) \vee (e_4 \wedge e_5)$$

// Encode each simple contradictory cycle

// Because we have polarity, we only need to encode the transitivity for negative edges!

$$\varphi_{cong} = (e_2 \wedge e_3) \rightarrow e_1 \wedge (e_4 \wedge e_5) \rightarrow e_1$$

$$\varphi = \varphi_{sk} \wedge \varphi_{cong}$$

Sparse-Method (Bryant) with Polar E-Graphs: Transitivity & Congruence

Consider the following conjunction of constraints in equality logic with uninterpreted functions. Check the satisfiability using the sparse method.

$$\varphi := a = b \wedge f(f(a)) = a \wedge f(f(b)) \neq b$$

Solution

Initial partitions: $\{a\}, \{b\}, \{f(a)\}, \{f(b)\}, \{f(f(a))\}, \{f(f(b))\}$

Transitivity: $\{a, b, f(f(a))\}, \{f(f(b))\}, \{f(a)\}, \{f(b)\}$

Congruence: $\{a, b, f(f(a)), f(f(b))\}, \{f(a), f(b)\}$

Because $f(f(a))$ and a are in the same EQ, the conjunction is unsatisfiable.

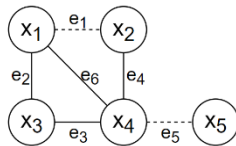
Eager SMT Solving based on Non-Polar E-Graphs: Reduce to propositional logic formula

Given: $\varphi^E := (\underbrace{\neg(x_1 = x_2)}_{e_1} \vee \underbrace{x_1 = x_3}_{e_2}) \wedge (\underbrace{x_3 = x_4}_{e_3} \vee \underbrace{x_4 = x_2}_{e_4}) \wedge (\underbrace{\neg(x_4 = x_5)}_{e_5} \vee \underbrace{x_1 = x_4}_{e_5})$

- Construct the E-Graph with Polarity for φ^E .
- Simplify the constructed E-Graph using the method presented in the lecture.
- Make the simplified graph without polarity chordal.
- Construct the SAT-equivalent propositional logic formula φ using the results from c).

Solution

- The E-Graph is the following:

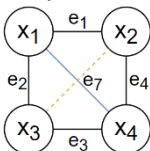


- We can replace all variables whose edges that are not part of ANY (not necessarily minimal!) contradictory cycle by their “meaning” (true/false). This gives us:

$$\begin{aligned} & (\neg(x_1 = x_2) \vee x_1 = x_3) \wedge (x_3 = x_4 \vee x_4 = x_2) \wedge (\neg(\text{false}) \vee x_1 = x_4) \\ \Leftrightarrow & (\neg(x_1 = x_2) \vee x_1 = x_3) \wedge (x_3 = x_4 \vee x_4 = x_2) \wedge (\text{true} \vee x_1 = x_4) \equiv \\ \Leftrightarrow & (\neg(x_1 = x_2) \vee x_1 = x_3) \wedge (x_3 = x_4 \vee x_4 = x_2) \end{aligned}$$

Note: We cannot, i.e., remove e_3 because it is part of the contradictory cycle $e_1 e_2 e_3 e_4$.

- The resulting (simplified) E-Graph without polarity looks like the following. We’ve added the blue edge to make it chordal (note that we could also add the yellow one instead; “Making a Graph Chordal” in general is not unique).



- Construct a SAT-equivalent propositional logic formula:

$$\varphi_{sk} = (\neg e_1 \vee e_2) \wedge (e_3 \vee e_4)$$

// Because we have NO polarity, we need to encode all transivities

$$\begin{aligned} \varphi_{trans} = & (e_1 \wedge e_4) \rightarrow e_7 \wedge (e_4 \wedge e_7) \rightarrow e_1 \wedge (e_7 \wedge e_1) \rightarrow e_4 \\ & \wedge (e_2 \wedge e_7) \rightarrow e_3 \wedge (e_7 \wedge e_3) \rightarrow e_2 \wedge (e_3 \wedge e_2) \rightarrow e_7 \end{aligned}$$

$$\varphi := \varphi_{sk} \wedge \varphi_{trans}$$

4.2. Less-Lazy SMT Solving

Consider the following equality formula:

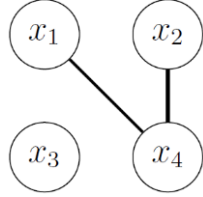
$$\varphi^{EQ} := (x_3 = x_4 \vee x_2 = x_4) \wedge (x_3 = x_4 \vee x_1 = x_2 \vee x_1 = x_4) \wedge (\neg(x_1 = x_2) \vee x_1 = x_4) \\ \wedge (\neg(x_3 = x_4) \vee \neg(x_2 = x_4) \vee \neg(x_2 = x_3))$$

Boolean abstraction:

$$\varphi_{sk} := (a \vee b) \wedge (a \vee c \vee d) \wedge (\neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg e)$$

Assign lexicographically smallest unassigned variable to false.

Solution

#0 SAT-Solver	#0 Theory Solver
DL0: –	Received (In)equalities: \emptyset No conflicts.
#1 SAT-Solver	#1 Theory Solver
DL0: – DL1: $a: 0, b: 1$	Received (In)equalities: $\neg(x_3 = x_4), (x_2 = x_4)$ Equivalence Classes: $\langle x_2 \rangle = \{x_2, x_4\}$ $\langle x_3 \rangle = \{x_3\}$ No conflicts.
#2 SAT-Solver	#2 Theory Solver
DL0: – DL1: $a: 0, b: 1$ DL2: $c: 0, d: 1$	Received (In)equalities: $\neg(x_3 = x_4), \neg(x_1 = x_2), (x_2 = x_4), (x_1 = x_4)$ Equivalence Classes: $\langle x_2 \rangle = \{x_1, x_2, x_4\}$ $\langle x_3 \rangle = \{x_3\}$ $\neg(x_1 = x_2)$ is conflicting. Draw E-Graph without polarity and find shortest path from x_1 to x_2 .  Minimal Infeasible Subset: $\{x_1 = x_4, x_2 = x_4, \neg(x_1 = x_2)\}$
#3 SAT-Solver	#3 Theory Solver
New constraint: $\neg d \vee \neg b \vee c$ Clause is not asserting, thus resolve: $\frac{(\neg d \vee \neg b \vee c) \quad (a \vee c \vee d)}{(a \vee \neg b \vee c)}$ Asserting. Add $(a \vee \neg b \vee c)$ to constraint set and backtrack to DL1 DL0: – DL1: $a: 0, b: 1, c: 1, d: 1$	Received (In)equalities: $\neg(x_3 = x_4), (x_2 = x_4), (x_1 = x_2), (x_1 = x_4)$ Equivalence Classes: $\langle x_1 \rangle = \{x_1, x_2, x_4\}$ $\langle x_3 \rangle = \{x_3\}$ No conflicts.
#4 SAT-Solver	#4 Theory Solver
DL0: –	Received (In)equalities:

$DL1: a: 0, b: 1, c: 1, d: 1$ $DL2: e: 0$	$\neg(x_3 = x_4), \neg(x_2 = x_3), (x_2 = x_4),$ $(x_1 = x_2), (x_1 = x_4)$ Equivalence Classes: $\langle x_1 \rangle = \{x_1, x_2, x_4\}$ $\langle x_3 \rangle = \{x_3\}$ No conflicts. SAT!
--	--

4. Bitvector Arithmetic

Don't use 0,1,=,≠,... ONLY use $\wedge, \vee, \neg, \leftrightarrow, \rightarrow$

Remember, that $x_{15} \dots x_0$ is the ordering!

Encoding of Bitvector Expressions: Examples

Encode the bitvector formula $(x \ll 3) = y$ with \ll being left-shift into a satisfiability-equivalent propositional logic formula, assuming x and y each have 16 bits.

$$\bigwedge_{i=0}^2 \neg y_i \wedge \bigwedge_{i=3}^{15} y_i \leftrightarrow x_{i-3}$$

Encode the bitvector formula $z = (x \circ y)$, with \circ being the concatenation operator, into a satisfiability-equivalent propositional logic formula, assuming x and y have 16 bits each.

$$\bigwedge_{i=0}^{15} z_i \leftrightarrow y_i \wedge \bigwedge_{i=0}^{15} z_{i+16} \leftrightarrow x_i \equiv \bigwedge_{i=0}^{15} ((z_i \leftrightarrow y_i) \wedge (z_{i+16} \leftrightarrow x_i))$$

Additional Examples

- 1) $1 \dots 1 \quad \equiv \bigwedge_{i=0}^{n-1} c_i$
- 2) $0 \dots 0 \quad \equiv \bigwedge_{i=0}^{n-1} \neg c_i$
- 3) $0 \dots 01 \quad \equiv c_0 \wedge \bigwedge_{i=1}^{n-1} \neg c_i$
- 4) $10 \dots 0 \quad \equiv c_{n-1} \wedge \bigwedge_{i=0}^{n-2} \neg c_i$
- 5) $\neg a[1] \quad \equiv \neg a_1$
- 6) $a[2:5] \quad \equiv \bigwedge_{i=0}^3 (c_i \leftrightarrow a_{i+2})$
- 7) $\sim a \quad \equiv \bigwedge_{i=0}^{n-1} (c_i \leftrightarrow \neg a_i)$
- 8) $a \& b \quad \equiv \bigwedge_{i=0}^{n-1} (c_i \leftrightarrow (a_i \wedge b_i))$
- 9) $a | b \quad \equiv \bigwedge_{i=0}^{n-1} (c_i \leftrightarrow (a_i \vee b_i))$
- 10) $a \circ b \quad \equiv \bigwedge_{i=0}^{n_b-1} c_i \leftrightarrow b_i \wedge \bigwedge_{i=0}^{n_a-1} (c_{i+n_b} \leftrightarrow a_i)$
- 11) $a \ll 1 \quad \equiv \neg c_0 \wedge \bigwedge_{i=1}^{n-1} (c_i \leftrightarrow a_{i-1})$

5. Fourier-Motzkin Variable Elimination

Generating Satisfying Assignments when using Fourier-Motzkin

Assume we've applied Fourier-Motzkin to eliminate all variables x_1, \dots, x_n from a set of non-strict inequalities using the order x_n, \dots, x_1 . Let the constraint turned out to be SAT. Explain how we can generate a satisfying assignment.

Solution

Fourier-Motzkin generates a sequence of constraint sets C_n, \dots, C_0 . The last set C_0 contains no variables and all constraints are trivially true (because the constraints are SAT according to task). We start with the empty assignment α_0 , which trivially satisfies C_0 .

Given a partial assignment α_k that satisfies C_k , we now construct α_{k+1} which shall satisfy C_{k+1} . We substitute α_k into all constraints from C_{k+1} and obtain a list of constant lower and upper bounds for x_{k+1} . We select the largest lower and smallest upper bound to obtain a solution interval $[l, u]$ (with $l = -\infty$ and $u = +\infty$ if no respective bound exists). Because the constraint set was found to be SAT, we know that $l \leq u$. Finally, we obtain α_{k+1} by extending α_k by $x_{k+1} = \beta$ for any $\beta \in [l, u]$.

So basically, C_0 doesn't tell us anything about our assignments (only that the constraint set is SAT). C_1 gives us an interval where we can choose x_1 from arbitrarily. Then, we can plug x_1 into C_2 , which then gives us an interval where we can choose x_2 arbitrarily, ...

Fourier-Motzkin Variable Elimination

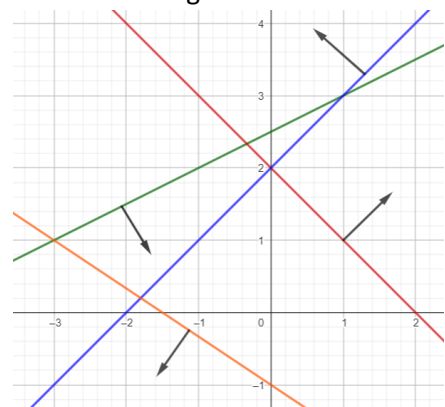
Eliminate y by applying the Fourier-Motzkin Variable Elimination to the following set of constraints:

$$\begin{array}{ll} c_1: x - 2y \leq -5 & y \geq \frac{x}{2} + \frac{5}{2} \\ c_2: x + y \leq 2 & y \leq -x + 2 \\ c_3: -x + y \leq 2 & y \leq x + 2 \\ c_4: -2x - 3y \leq 3 & y \geq -\frac{2}{3}x - 1 \end{array}$$

Solution

Pair all lower with all upper bounds:

$$\begin{array}{ll} \frac{x}{2} + \frac{5}{2} \leq -x + 2 & c_5: x \leq -\frac{1}{3} \\ \frac{x}{2} + \frac{5}{2} \leq x + 2 & c_6: x \geq 1 \\ -\frac{2}{3}x - 1 \leq -x + 2 & c_7: x \leq 9 \\ -\frac{2}{3}x - 1 \leq x + 2 & c_8: x \geq -\frac{9}{5} \end{array}$$



The set of constraints is NOT satisfiable, because i.e., $\{c_5, c_6\}$ are conflicting. We can construct the minimal infeasible subset by looking up the formulas that yielded c_5, c_6 . We used $\{c_1, c_2, c_3\}$ for this. Subsequently, this is our minimal infeasible subset.

To do such tasks graphically, i.e., for $y \leq x + 2$ we would draw $y = x + 2$ and since y needs to be smaller or equal, the arrow must point upwards (away from the solution space).

When eliminating one variable, we can consider the solution set as the **projection of the two-dimensional set to the x-/y-axis**.

6. Simplex

Pivoting

Apply one pivoting step and update the current assignment, where the variable order is $x_1 < x_2 < s_1 < \dots < s_4$ and where the current tableau, the bounds, and the current assignment are given by:

$$s_1 \geq 3 \qquad s_2 \leq -1 \qquad s_3 \geq 2 \qquad s_4 \leq 3/2$$

	$s_1[3]$	$x_2[0]$
$x_1[3]$	1	-1
$s_2[3]$	1	-2
$s_3[3]$	1	1
$s_4[0]$	0	1

Pivoting B_i with NB_j

...	NB_j	...	NB_i	...
...				
B_i	a_{ij}		a_{il}	
...				
B_k	a_{kj}		a_{kl}	
...				

 \rightarrow

...	NB_j	...	NB_i	...
...				
B_i	$\frac{1}{a_{ij}}$		$-\frac{a_{il}}{a_{ij}}$	
...				
B_k	$\frac{a_{kj}}{a_{ij}}$		$a_{kl} - \frac{a_{kj}a_{il}}{a_{ij}}$	
...				

Solution

As only s_2 violates its bounds, we must pivot it. We want to decrease it. For this, we can either decrease s_1 (not possible – already at its lower bound) or increase x_2 . Thus, we pivot s_2, x_2

	$s_1[3]$	$s_2[-1]$
$x_1[1]$	$\frac{1}{2}$	$\frac{1}{2}$
$x_2[2]$	$\frac{1}{2}$	$-\frac{1}{2}$
$s_3[5]$	$\frac{3}{2}$	$-\frac{1}{2}$
$s_4[2]$	$\frac{1}{2}$	$-\frac{1}{2}$

The resulting pivot table is conflicting and the constraints UNSAT. s_4 violates its bounds and needs to be decreased. For this, we would need to decrease s_1 (already at its lower bound), or increase s_2 (already at its upper bound). Thus, we have a conflict.

Invariants of Simplex

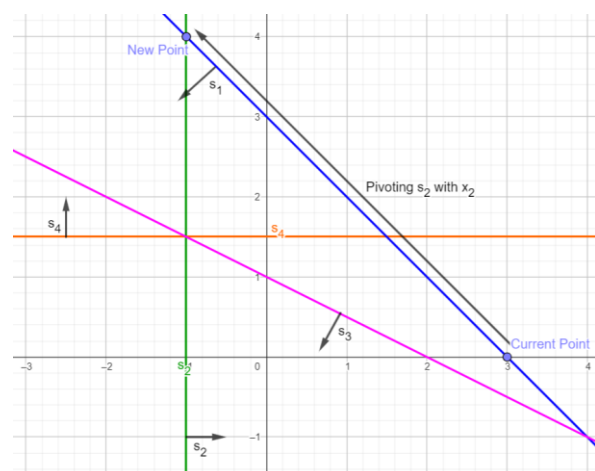
1. All Non-Basic Variables (Columns) always satisfy their bounds.
2. $A\vec{x} = 0$

Simplex Graphically

Moving the point along the lines defined by the constraints and x- and y-axis.

For example, s_1, s_2 as Non-Basic Variables mean that the point must lie on the intersection of s_1, s_2

Similarly, for s_1, y , however there we've a fixed y , so the x and y-axis are kind of swapped. In the given case, you can observe the current point for $s_1[3], y = 0$, where $y = 0$ basically defines a line that is identical to the x-axis.



7. Integer Arithmetic

Branch and Bound

Check the satisfiability of the linear integer arithmetic constraint set:

$$P = \{y \leq x + 0.5, \quad y \leq -x + 1.5, \quad y \geq x - 0.5, \quad y \geq -x + 0.5\}$$

Using the Branch and Bound Method, with the heuristics to prefer x for splitting and handle lower branches (that means upper-bounded!) first.

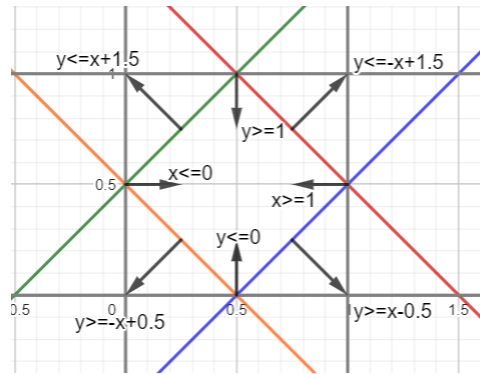
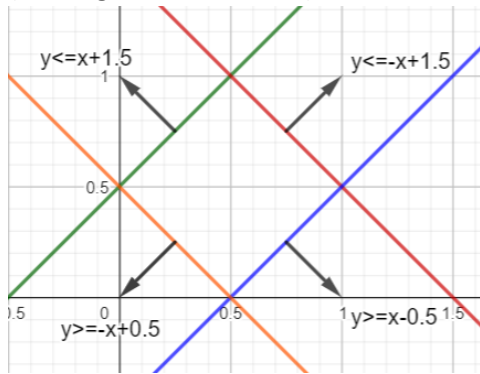
Solution

In the first iteration, simplex would either find $(0.5, 0)$ or $(0, 0.5)$ as a solution. Let's say it's the first one. Then our x -coordinate is non-integer and we must branch x into $x \leq 0$ and $x \geq 1$.

As we should use lower branches (upper bounded!) first, we continue with $x \leq 0$.

Then, simplex finds $(0, 0.5)$ as a solution. However, y is non-integer, and we branch $y \leq 0$ and $y \geq 1$.

Both subsequent tests yield no solution, thus this branch is UNSAT and we continue with $x \geq 1$ (which gives us the same).



The recursive calls could look the following:

P

$$P \cap \{x \leq 0\}$$

$$P \cap \{x \leq 0, y \leq 0\}$$

$$P \cap \{x \leq 0, y \geq 1\}$$

$$P \cap \{x \geq 1\}$$

$$P \cap \{x \geq 1, y \leq 0\}$$

$$P \cap \{x \geq 1, y \geq 1\}$$

Properties of Branch and Bound

- Incomplete, i.e., consider $1 \leq 3(x + y) \leq 2$ – the algorithm will loop forever.
- If we have a bounded solution set, it is complete since we can only have finitely many numeric points to check/branch.
- Works with linear as well as non-linear constraints (in the latter we must use something like CAD instead of Simplex to find solutions to the relaxed problem).

8. Interval Constraint Propagation

Rules for Interval Arithmetic

Give the interval division rules for $A \div B$ with $0 < \underline{A}$, $0 \in B$

Solution

Determine the different cases:

- $B = [0; 0]$
- $B = [\underline{B}; \bar{B}]$ with $\underline{B} < 0 < \bar{B}$
- $B = [\underline{B}; 0]$
- $B = [0; \bar{B}]$

Construct general formulas for those:

If $\bar{A} > \underline{A}$: $[1; 0]$ (empty)

	$B = [0; 0]$	$\underline{B} < \bar{B} = 0$	$\underline{B} < 0 < \bar{B}$	$0 = \underline{B} < \bar{B}$
$\frac{[\underline{A}, \bar{A}]}{[\underline{B}, \bar{B}]}$	$[1; 0]$ (empty)	$\left(-\infty; \frac{\underline{A}}{\underline{B}}\right]$	$\left(-\infty; \frac{\underline{A}}{\underline{B}}\right] \cup \left[\frac{\bar{A}}{\bar{B}}; +\infty\right)$	$\left[\frac{\underline{A}}{\bar{B}}; +\infty\right)$

Interval Contraction

Consider $c_1: x^2 - 2 = y$ and $c_2: 2x \geq y + 2$ and $x \in [-1; 1]$, $y \in [-2; 2]$. Perform contraction with all contraction candidates and argue which caused the largest relative contraction.

Solution

$c_{1,x}$	$[-1; 1] \cap \sqrt{[-2; 2] + 2} = [-1; 1] \cap \sqrt{0; 4} = [-1; 1] \cap [-2; 2] = [-1; 1]$	$1 - \frac{2}{2} = 0$
$c_{1,y}$	$[-2; 2] \cap ([-1; 1]^2 - 2) = [-2; 2] \cap [-2; -1] = [-2; -1]$	$1 - \frac{1}{4} = \frac{3}{4}$
$c_{2,x}$	$x \geq \frac{1}{2}y + 1 = \frac{1}{2}[-2; 2] + 1 = [0; 2]$ $x \in [-1; 1] \cap [\min(0; 2); \infty) = [0; 1]$	$1 - \frac{1}{2} = \frac{1}{2}$
$c_{2,y}$	$y \leq 2x - 2 = 2[-1; 1] - 2 = [-4; 0]$ $y \in [-2; 2] \cap [-\infty; \max(-4; 0)] = [-2; 0]$	$1 - \frac{2}{4} = \frac{1}{2}$

(Relative Contraction: $1 - \frac{D_{new}}{D_{old}}$)

Newton Interval Contraction

Given $f(x) = \frac{1}{3}x^3 - x^2 + 1$ and the starting interval $A = [1; 2]$ with $s = 1$. Perform one Newton Contraction Step.

Solution

$$f(s) = \frac{1}{3} - 1 + 1 = \frac{1}{3}$$

$$f'(A) = x^2 - 2x = [1; 2]^2 - 2[1; 2] = [1; 4] - [2; 4] = [-3; 2]$$

$$I = s - \frac{f(s)}{f'(A)} = 1 - \frac{\frac{1}{3}}{[-3; 2]} = 1 - \frac{1}{[-9; 6]} = 1 - \left(\left(-\infty; -\frac{1}{9}\right] \cup \left[\frac{1}{6}; +\infty\right) \right)$$

$$= \left(-\infty; \frac{5}{6}\right] \cup \left[\frac{10}{9}; +\infty\right)$$

$$A \cap I = [1; 2] \cap \left(\left(-\infty; \frac{5}{6}\right] \cup \left[\frac{10}{9}; +\infty\right) \right) = \left[\frac{10}{9}; 2\right]$$

9. Subtropical Satisfiability

Constructing the Root

Given the multivariate polynomial $p(x) = -x^2 - xy + 1$.

Which solutions can be found for the equation $p(x, y) = 0$ by the subtropical satisfiability method using the sample points $p(1,1) < 0$ and $p(0,0) > 0$?

Solution

1. Reduce two-dimensional problem to a one-dimensional problem by searching for solutions on the line segment between $(0,0)$ and $(1,1)$. Thus, we construct $l(t)$:

$$l(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + t \begin{pmatrix} 0-1 \\ 0-1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + t \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

(Note: The order doesn't matter at all, we could also swap both points!)

2. Plug the new expressions for x, y into $p(x)$, construct $p^*(x)$, and compute its roots.

$$p^*(x) = -(1-t)^2 - (1-t)^2 + 1$$

$$p^*(x) = 0 \Rightarrow -2(1^2 - 2t + t^2) + 1 = 0 \Leftrightarrow t^2 + 2t - \frac{1}{2} = 0$$

$$t_{1,2} = -1 \pm \sqrt{1 - \frac{1}{2}} = -1 \pm \frac{1}{\sqrt{2}}$$

3. Transform the problem back into two dimensions by plugging in t_0 into $l(t)$. Note, that $t_0 \in [0; 1]$ must hold:

$$l\left(-1 + \frac{1}{\sqrt{2}}\right) = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

Framepoints and Hyperplanes

Give the multivariate polynomial $p(x, y, z) = -x^5 + 2x^2y^2 - 5z - xyz$

Determine:

- a) the framepoints (pos. and neg.),
- b) the formula of a separating hyperplane using n_x, n_y, n_z and b , and
- c) explain the exact set of all univariate polynomials $p \in \mathbb{Z}[x]$ for which the subtropical SAT method as presented in the lecture cannot find a solution for $p(x) > 0$

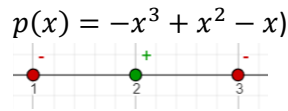
Solution

- a) $f^+ = \{(2,2,0)\}, f^- = \{(-5, 0, 0), (0, 0, -1), (1, 1, 1)\}$
- b) Separating one positive from all the others:
 $(2n_x + 2n_y > b) \wedge (-5n_x < b) \wedge (-1n_z < b) \wedge (1n_x + 1n_y + 1n_z < b)$
- c) The constraint $p(x) > 0$ can be solved iff there exists $v \in \text{frame}^+(p) \subseteq \mathbb{N}$ such that:

$$n_x v \wedge \bigwedge_{u \in \text{frame}(p) \setminus \{v\}} n_x u < b$$

The method finds no solutions if one of the following two conditions hold:

- a. The polynomial $p(x)$ has no positive framepoint (i.e., $p(x) = -x$)
- b. There are positive and negative framepoints, but we cannot separate one positive from all the others. For the one-dimensional case it means that the monomials with the largest and smallest exponents need both to have a negative coefficient (i.e.,



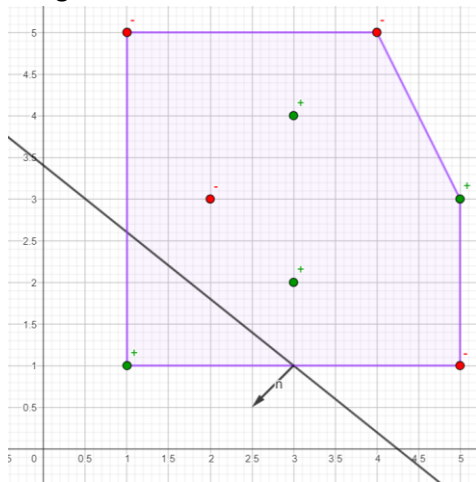
Newton Polytope, and Separating Hyperplane

Given the multivariate polynomial $p(x, y) = 2xy - 2xy^5 - 3x^2y^3 + x^3y^2 + x^3y^2 + x^3y^4 - 2x^4y^5 + 7x^5y^3 - 6x^5y$. Construct:

- the Newton Polytope, and
- a separating hyperplane. Use the resulting normal vector to derive a $f^*(a)$ such that $f^*(a) > 0$ for a sufficiently large a .

Solution

- We get:



- $\vec{n} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$

Subsequently, we can compute:

$$f^*(a) := p(a^{-1}, a^{-1}) = \frac{2}{a^2} - \frac{2}{a^6} - \frac{3}{a^5} + \frac{1}{a^5} + \frac{1}{a^5} + \frac{1}{a^7} - \frac{2}{a^9} + \frac{7}{a^8} - \frac{6}{a^7}$$

Compute a solution for Subtropical Satisfiability

Compute a solution for $p(x, y) > 0$ for the polynomial $p(x, y) = -x^2y - 3x + y$ and the separating hyperplane specified by $(n_x, n_y) = (-1, 1)$ and $b = 0$. Use the subtropical SAT method.

Solution

$$p^*(a) := p(a^{-1}, a^1) = -(a^{-1})^2 a^1 - 3 \cdot a^{-1} + a^1 = -\frac{1}{a} - \frac{3}{a} + a = -\frac{4}{a} + a$$

$$p^*(2) = -\frac{4}{2} + 2 = 0 \not> 0 \rightarrow a := a^2 \quad p^*(4) = -\frac{4}{4} + 4 = 3 > 0 \rightarrow \text{ok!}$$

(Remember that $p^*(a)$ must be STRICTLY greater than 0!)

Subsequently, for $x = 4^{-1} = \frac{1}{4}$, $y = 4^1 = 4$ we've found a solution.

10. Virtual Substitution

Identify the appropriate substitution rule and apply virtual substitution:

- Transform into normal form ($ax^2 + bx + c \sim 0$).
- Look closely which case is the right one.
- In the corresponding case, replace a, b, c by our values for a, b, c (don't make mistakes here!)
- Simplify to sums of products.
- (Optionally: Further simplify as far as possible)

Test Candidates and Side Conditions

Give all test candidates in variable x for $zx^2 + x^2y^2 + 2xy + z < 0$ and their corresponding side conditions.

Solution

Normal Form: $(z + y^2)x^2 + 2yx + z < 0$

Recall: Roots of $ax^2 + bx + c$

Root	Side Condition
Everything	$a = 0 \wedge b = 0 \wedge c = 0$
$\xi_0 = -\frac{c}{b}$	$a = 0 \wedge b \neq 0$
$\xi_0 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$	$b \neq 0 \wedge b^2 - 4ac \geq 0$
$\xi_0 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$	$b \neq 0 \wedge b^2 - 4ac > 0$

So, in our case (simplified):

Root	Side Condition
Everything	$z + y^2 = 0 \wedge y = 0 \wedge z = 0$
$\xi_0 = -\frac{z}{2y}$	$z + y^2 = 0 \wedge y \neq 0$
$\xi_0 = \frac{-y + \sqrt{y^2 - (z + y^2)z}}{(z + y^2)}$	$y \neq 0 \wedge y^2 - z^2 - zy^2 \geq 0$
$\xi_1 = \frac{-y - \sqrt{y^2 - (z + y^2)z}}{(z + y^2)}$	$y \neq 0 \wedge y^2 - z^2 - zy^2 > 0$

Because we have a strict inequality, our test candidates are:

Root	Side Condition
$-\infty$	Always
$\xi_0 = -\frac{z}{2y} + \epsilon$	$z + y^2 = 0 \wedge y \neq 0$
$\xi_0 = \frac{-y + \sqrt{y^2 - (z + y^2)z}}{(z + y^2)} + \epsilon$	$y \neq 0 \wedge y^2 - z^2 - zy^2 \geq 0$
$\xi_1 = \frac{-y - \sqrt{y^2 - (z + y^2)z}}{(z + y^2)} + \epsilon$	$y \neq 0 \wedge y^2 - z^2 - zy^2 > 0$

Substitution Rules – Their graphical Interpretation

Consider the following substitution rule for a test candidate $e + \epsilon$ and a constraint $ax^2 + bx + c > 0$. This rule encodes three different cases. Give the general idea for every case and sketch example functions graphically.

$$((ax^2 + bx + c > 0)[e//x])$$

$$\vee ((ax^2 + bx + c = 0)[e//x] \wedge (2ax + b > 0)[e//x])$$

$$\vee ((ax^2 + bx + c = 0)[e//x] \wedge (2ax + b = 0)[e//x] \wedge (2a < 0)[e//x])$$

Intuitive meaning:

$$(ax^2 + bx + c > 0)[e//x]:$$

- At position e the value is greater 0 anyways.

$$(ax^2 + bx + c = 0)[e//x] \wedge (2ax + b > 0)[e//x]$$

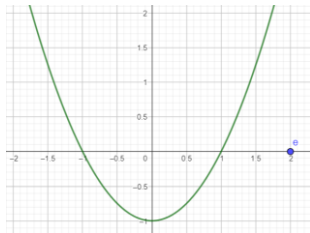
- At position e the value is exactly zero, but we have a positive slope, thus a tiny bit on the right our value is > 0

$$(ax^2 + bx + c = 0)[e//x] \wedge (2ax + b = 0)[e//x] \wedge (2a > 0)[e//x]$$

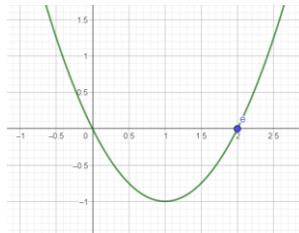
- At position e the value is exactly zero
- Our first derivative is also zero, indicating that we have an extremum.
- For the polynomial to be > 0 at some point, we must have a minimum, that is, the second derivative must be greater than 0.

Graphically, it could look like the following:

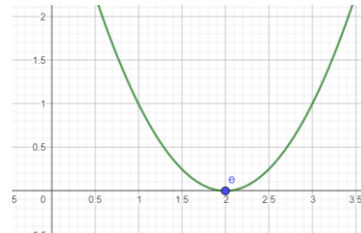
Case 1



Case 2



Case 3



11. Cylindrical Algebraic Decomposition

Compute Interval-Representation

Consider the univariate polynomial $p = x^3 - 4x^2 + 3x$ and the Sturm sequence of p :

$$\begin{aligned} p_0 &= x^3 - 4x^2 + 3x & p_2 &= \frac{14}{9}x - \frac{4}{3} \\ p_1 &= 3x^2 - 8x + 3 & p_3 &= \frac{81}{49} \end{aligned}$$

Compute the interval representation as presented in the lecture. Choose midpoints for splitting.

Solution

Cauchy Bound:

$$|\xi| \leq 1 + \max\left\{\left|\frac{4}{1}\right|, \left|\frac{3}{1}\right|\right\} = 5 \quad \text{Thus, all our intervals must be in } [-5; 5]$$

Testing interval $[-5; 5]$

Check the bounds:

$p(-5) < 0$ and $p(5) > 0 \Rightarrow$ No roots, we can safely continue with $(-5; 5)$

	-5	5	0	2.5
$p_0 = x^3 - 4x^2 + 3x$	-	+	0	-
$p_1 = 3x^2 - 8x + 3$	+	+	+	+
$p_2 = \frac{14}{9}x - \frac{4}{3}$	-	+	-	+
$p_3 = \frac{81}{49}$	+	+	+	+
$\sigma(\cdot)$	3	0	2	1

Roots in $(-5; 5)$: $\sigma(-5) - \sigma(5) = 0$. Thus, we split the interval in the middle.

Testing $[0; 0]$

$p(0) = 0 \Rightarrow$ We've found one root.

Testing $(-5; 0)$

Sturm Sequence gives us $\sigma(-5) - \sigma(0) - 1 = 0$. Thus, we have no roots in $(-5; 0)$.

Note, that we must subtract 1, because $x = 0$ is one root, and Sturm Sequences check for $(a, b]$

Testing $(0; 5)$

Sturm Sequence gives us $\sigma(0) - \sigma(5) = 2$. Thus, we must split the interval.

Testing $[2.5; 2.5]$

$p(2.5) < 0 \Rightarrow$ No root

Testing $(0; 2.5)$

Sturm Sequences gives us $\sigma(0) - \sigma(2.5) = 2 - 1 = 1 \Rightarrow$ Valid

Testing $(2.5; 5)$

Sturm-Sequence gives us $\sigma(2.5) - \sigma(5) = 1 - 0 = 1 \Rightarrow$ Valid

Our interval representation of the roots is given by:

$$0, (p, (0; 2.5)), (p, (2.5; 5))$$

Draw the Test-Candidates / Sample. First in 1D, then in 2D.

Let $\{x^2 + y^2 - 4 < 0, x^2 - y - 1 > 0\}$ be a set of constraints over the reals. The zeros of the polynomials are depicted in the graph below.

- Graphically identify a set of univariate samples in x that should be considered in the one-dimensional space.
- Then lift them to get the set of two-dimensional sample points that could be generated by the CAD method as described in the lecture.

Solution

