Theory of Hybrid Systems Informatik 2
RWTH AACHEN UNIVERSITY

The present work was submitted to the LuFG Theory of Hybrid Systems

BACHELOR OF SCIENCE THESIS

CONFLICT GENERALIZATION FOR QUADRATIC REAL-ARITHMETIC CONSTRAINTS IN SMT-RAT

André Leon Poncelet

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. Jürgen Giesl

Additional Advisor:
Jasper Nalbach, M.Sc.

Aachen, 28.12.2021

Abstract

A big advantage of Satisfiability Modulo Theories (SMT) solvers over traditional automated reasoning frameworks such as SAT solvers is its broad applicability to a wide range of *background theories* [BT18]. A SMT solver can answer mathematical questions about logic, arithmetic, bitvectors and other data structures which are combined by Boolean operators [DdM06][BB09]. Most modern SMT solvers are based on the *DPLL(T)* algorithm which in turn is based on the DPLL algorithm for SAT problems [GHN$^+$04][DP60][DLL62]. DPLL(T) essentially operates by guessing a solution to the given problem, guided by some lookahead. If the guess of a variable assignment is not a solution, it then tries to generalize this *conflict* to refine the next guess. The SMT-solver SMT-RAT which is being developed at the Theory of Hybrid Systems Research Group at RWTH Aachen University is also based on this premise [CLJÁ12]. In this thesis we study the background theory of *quadratic real-arithmetic* formulas and, more precisely, the step of generalizing a conflict.

For this we define a variation of the single-source single-destination shortest path problem for node-colored graphs in which the distance or cost of a path is given by the total cost of all edges and the cost of all *colors* of the traversed nodes. This cost is non-local because nodes may share a color whose cost is only accounted for once. Because well-known graph algorithms such as Dijkstra's algorithm [GF13] work on the premise that local and global optima coincide, which is not the case here, we show that this problem is computationally hard, but also present algorithms to mitigate some of its complexity and algorithms to solve an easier variation of this problem efficiently.

Zentrales Prüfungsamt/Central Examination Office

RWTH AACHEN UNIVERSITY

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Poncelet, André Leon
Name, Vorname/Last Name, First Name

377871
Matrikelnummer (freiwillige Angabe)
Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/Bachelorarbeit/ ~~Masterarbeit~~* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present ~~paper~~/Bachelor thesis/~~Master thesis~~* entitled

Conflict Generalization for Quadratic Real-Arithmetic Constraints in SMT-RAT

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 28. Dez. 2021
Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen
*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 28. Dez. 2021
Ort, Datum/City, Date

Unterschrift/Signature

Acknowledgements

I would first like to thank my supervisor Prof. Dr. Erika Ábrahám who always made herself available for long and very insightful conversations, who was more understanding then I could have hoped for when I struggled during the process of writing this thesis and who encouraged and supported me throughout.

I would also like to extend my gratitude to my advisor Jasper Nalbach without whom this thesis would not have been possible, who was always quick to answer any question I might have had, who more than once lead me into the right direction and whose support and patience I greatly appreciate.

I am also grateful to Miriam Greidanus Romaneli who took it on herself to proofread an early draft of this thesis and offered some great advice.

Last, but certainly not least, I would like to express my deepest gratitude toward my family who always supported me in whatever capacity, not only while writing this thesis, but throughout my entire life. Words cannot express how grateful I am to have you by my side so I will just say: *Thank you!*

Contents

Chapter 1

Introduction

One of the best known problems in computer science is *Boolean satisfiability*, better known as SAT. It is a decision problem which asks for a given Boolean formula whether it has a solution, i.e. an assignment to its variables under which the formula evaluates to *true*. *Satisfiability Modulo Theories* (SMT) is a generalization of this problem [BT18]. The atoms of the examined formulas may not only consist of *literals*, i.e. Boolean variables and their negation, as it is the case for SAT, but may also consist of expressions of (arbitrary) *background theories*. Consider the formula $xy^2 = 2 \wedge x \geq 1$. It consists of two atoms $xy^2 = 2$ and $x \geq 1$ which are combined using the Boolean operator $\wedge$. This formula is not representable as an instance of SAT, however it is as an instance of SMT. The interpretation and thus the possible solutions of this formula depend on and change for different choices of a background theory. The expressions $xy^2 = 2$ and $x \geq 1$ cannot be interpreted in isolation. For instance, we may choose *integer-arithmetic* as the background theory in which each of the variables x and y may only take on integer values. In this case, the formula does not have any solutions. If we however interpret it as a *real-arithmetic* formula, where x and y may take on real values, then the assignment 1 for x and $\sqrt{2}$ for y is an assignment that satisfies this formula.
There are different algorithms which can solve SAT instances, one of them being the *Davis–Putnam–Logemann–Loveland* (DPLL) algorithm, which was first proposed in 1960 by Martin Davis and Hilary Putnam [DP60] and later improved upon by Davis as well as George Logemann and Donald W. Loveland in 1961 [DLL62]. DPLL(T), a later adaption of DPLL, was proposed in 2004, which used the overall architectural design of DPLL and expanded it so that SMT problems could be solved with it by invoking *theory backends* for each of the supported background theories [GHN$^+$04]. One key advantage of DPLL(T) is its incremental design: DPLL(T) analyzes the Boolean skeleton of an SMT formula. The corresponding DPLL-SAT solver then creates a Boolean variable assignment for the skeleton formula. This results in a set of corresponding *constraints*, i.e. atoms of a formula of SMT, which are then handed over to the *theory backend*. The backend tries to find a solution to these constraints and informs the DPLL(T) framework if it was able to. Depending on the answer of the backend, the DPLL(T) framework then adds and/or removes constraints from the aforementioned set and invokes again the theory backend. It should be able to use information gathered in earlier invocations to compute an answer, thus making use of the incrementality of DPLL(T).

Most modern SMT solver work based on a similar premise of incrementality. The SMT solver SMT-RAT which is being developed at the Theory of Hybrid Systems Research Group at RWTH Aachen had its first release in 2012 [CLJÁ12]. One of the motivations for the work on the SMT-RAT project was that, at the time, although some of the existing SMT solver supported non-linear arithmetic background theories, they often lacked the option of combining different theory solvers. By using this approach it is, for example, possible to reduce a formula which contains quadratic polynomials to a linear formula. This linear formula may then be analyzed by theory solver which are optimized for that specific task. This thesis aims to improve the above described incrementality of SMT-RAT. We archive this by only considering a specialized case for which we then can optimize our approach. We study the background theory of real-arithmetic constraints (as Defined in Section 2.2) under the restriction that the polynomials of these constraints are at most quadratic in a variable of interest. As mentioned above, SMT-RAT, like DPLL(T), calls a theory backend with a set of constraints and a partial variable assignment to those constraints. The backend is then tasked with finding a solution for these constraints or to generate an explanation as to why it has not come up with one. This explanation is then used to refine the next guess for a (partial) variable assignment/solution to the original given problem.

The algorithms developed in this thesis take on one part of this process and are specifically meant to be implemented into the SMT-RAT project. We study and develop algorithms for so called *conflict generalizations*. In the context of this thesis, a *conflict* is a set of real-arithmetic constraints whose polynomials are at most quadratic in a variable y, together with an assignment α to all variables of the constraints (except y), such that α cannot be extended to a full assignment of the constraints (by assigning y as well) that is a solution to the constraints. A *conflict generalization* is a formula that described *why* this is the case, i.e. why the conflict is a conflict, and thus allows to rule out other (partial) assignments as (partial) solutions to the constraints which were given. A formal definition of conflicts and their generalization can be found in Section 2.6.

We will find algorithms for the generalizations of conflicts by following these steps:

- For a given conflict we consider $\mathbb{R}$ to be the assignment space to the unassigned variable. Each full assignment which extends the given partial assignment then corresponds to a unique real number and vice versa.
- We realize that for each of the given constraints the set of assignments under which said constraint is *not* satisfied is equivalent to a union of intervals in $\mathbb{R}$. The reason *why* a set of constraints is not satisfiable under a partial assignment lies in the fact that these intervals (corresponding to all constraints) cover every point on the real number line.
- We introduce *Virtual Substitution* (*VS*) and *square root expressions* (see Section 2.5, [CÁ11][Cor10][Cor17]) with which the bounds of these intervals can be expressed as formulas which depend on the given partial assignment.
- We find a sufficient and necessary condition under which an (ordered) collection of intervals cover $\mathbb{R}$. This condition can easily be translated into a SMT-compliant formula using VS and square root expressions.
- We rephrase the problem of finding a subset of intervals which cover $\mathbb{R}$ by checking said condition into a graph problem where nodes correspond to intervals and certain paths correspond to covers of $\mathbb{R}$ by the traversed intervals/nodes.

- We find heuristics to weigh the "goodness" of conflict generalizing formulas. We use this heuristic to assign weights to both edges and to colors which are used to color in the nodes of the graph.
- We then present an efficient algorithm that can eliminate nodes which are not part of any path of interest and thus reduce the runtime of path-finding algorithms. We present two such path-finding algorithms. One of which finds a "close to optimal" path efficiently and one of which finds an optimal path in exponential time. We further argue that there is no polynomial-time algorithm that can archive the latter (unless P = NP) based on the fact that the cost of a color is only accounted for once, even if a path traversed multiple nodes of the same color.

In Chapter 2 we introduce the mathematical foundation underlying this thesis. In particular, we introduce Virtual Substitution and square root expressions in Section 2.5. Chapter 3 then presents the contributions of this thesis, specifically the algorithms outlined above. Chapter 4 summarizes the contributions of this thesis.

Chapter 2

Foundations

In this chapter we introduce mathematical concepts underlying this thesis. The definitions are largely adopted from [Cor17], but may differ in some regards to better suit this thesis.

2.1 Notation

In this thesis $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$ denotes the set of Boolean constants. The set $\mathbb{N}$ denotes the set of all positive integers and $\mathbb{N}_0$ denotes the set of all non-negative integers. $\mathbb{R}$ and $\mathbb{Z}$ denote the real and whole numbers respectively. We define $sgn : \mathbb{R} \to \{-1,0,1\}$ to assign 0 to 0, 1 to all positive and -1 to all negative real values. $\mathcal{P}(M)$ denotes the power set of a set M. The set of all intervals is denoted by $\mathbb{I}$. The closure of $\mathbb{I}$ under $\cup$ is denoted by $\overline{\mathbb{I}} = \bigcup_{n=1}^{\infty}\{I_1 \cup \ldots \cup I_n : I_1, \ldots, I_n \in \mathbb{I}\}$. Furthermore, in the following, we use the notation $f : A \rightharpoonup B$ for a partially defined function f, i.e. for $f : A' \to B, A' \subseteq A$ and write $f(a) = \perp$ for $a \in A \setminus A'$ where we assume that $\perp$ is a fresh symbol in B, i.e. $\perp \notin B$.

2.2 Syntax of propositional and arithmetic formulas

We define *(arithmetic) formulas* in this thesis as words in a formal language. Their semantics are defined in Section 2.3. Formulas are built with *variables*. We differentiate variables by their associated *domain*, which can be $\mathbb{B}$, $\mathbb{R}$ or $\mathbb{Z}$. A variable may only take on values from their associated domains. We define $VAR_{\mathbb{B}}$, $VAR_{\mathbb{R}}$, $VAR_{\mathbb{Z}}$ as the *disjoint* sets of all variables with the domain $\mathbb{B}$, $\mathbb{R}$ or $\mathbb{Z}$ respectively. Further, we define $VAR_{\mathbb{R},\mathbb{Z}} = VAR_{\mathbb{R}} \cup VAR_{\mathbb{Z}}$ as the set of *arithmetic* variables and $VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} = VAR_{\mathbb{B}} \cup VAR_{\mathbb{R}} \cup VAR_{\mathbb{Z}}$ as the set of all variables. The *domain* associated with a variable is given by $\mathrm{Dom} : VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} \to \{\mathbb{B}, \mathbb{R}, \mathbb{Z}\}$.

Formulas consist of *propositions* given by Boolean variables and *constraints* combined using Boolean operators. In this thesis we restrict constraints to inequalities between polynomials.

Definition 2.2.1 (Polynomial[a])**.** *The set POL of polynomials is defined inductively as the smallest set satisfying the following:*

- 0,1 ∈ *POL*
- $VAR_{\mathbb{R},\mathbb{Z}} \subseteq POL$
- *if* $p,q \in POL$, *then* $(p+q),(p-q),(p \cdot q) \in POL$

[a] see Def. 4 on page 19 of [Cor17]

To define (in-)equalities over these polynomials, which in turn are used to define formulas, we first define the set $REL = \{<, >, \neq, \leq, \geq, =\}$ as the set of all relation symbols. With that, we can define formulas as follows:

Definition 2.2.2 (Arithmetic Formula[a])**.** *The set FO of (arithmetic) formulas is defined inductively as the smallest set satisfying the following:*

- *true*, *false* ∈ *FO*
- $VAR_{\mathbb{B}} \subseteq FO$
- *if* $p,q \in POL$ *and* $\sim \in REL$, *then* $p \sim q \in FO$
- *if* $\varphi \in FO$, *then* $(\neg\varphi) \in FO$
- *if* $\varphi, \psi \in FO$, *then* $(\varphi \vee \psi) \in FO$ *and* $(\varphi \wedge \psi) \in FO$
- *if* $\varphi \in FO$ *and* $x \in VAR_{\mathbb{R},\mathbb{Z}}$, *then* $(\exists x.\varphi) \in FO$ *and* $(\forall x.\varphi) \in FO$

[a] see Def. 4 on page 19 of [Cor17]

Parentheses and the multiplication symbol are often omitted to improve readability. The following order of operation is used:

$$\cdot,\ -,\ +,\ \sim,\ \neg,\ \wedge,\ \vee,\ \forall,\ \exists$$

for any $\sim \in REL$ ($\cdot$ is first and $\exists$ last).
Further, we use some syntactic sugar:

$$\begin{aligned} n &\equiv \underbrace{1 + \ldots + 1}_{\text{n times}}, \text{ for some } n \in \mathbb{N} \\ x^0 &\equiv 1, \text{ for some } x \in \mathrm{VAR}_{\mathbb{R},\mathbb{Z}} \\ x^n &\equiv \underbrace{x \cdot \ldots \cdot x}_{\text{n times}}, \text{ for some } n \in \mathbb{N} \text{ and } x \in \mathrm{VAR}_{\mathbb{R},\mathbb{Z}} \end{aligned}$$

We say for formulas which contain a sub-formula of the type $(\mathcal{Q}x.\ \varphi)$ for a *quantifier* $\mathcal{Q} \in \{\exists, \forall\}$, a variable x and a formula φ, that the variable x is *bound* to the quantifier $\mathcal{Q}$. If a variable is bound to a quantifier, it is assumed that it only occurs within the scope of its quantifier. A variable which is not bound is called *free*. We define $Vars, FreeVars : (POL \cup FO) \to VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}}$ to obtain the set of all variables and the set of all free variables in a polynomial or formula.

We denote $POL[x_1,\ldots,x_n] = \{p \in POL : Vars(p) \subseteq \{x_1,\ldots,x_n\}\}$ as the set of all polynomials in variables $x_1,\ldots,x_n$.

An arithmetic formula $\varphi \in FO$ is called *real-arithmetic*, *integer arithmetic* or *mixed integer-real-arithmetic* if $Vars(\varphi) \subseteq VAR_{\mathbb{R}}$, $Vars(\varphi) \subseteq VAR_{\mathbb{Z}}$ or $Vars(\varphi) \subseteq VAR_{\mathbb{R},\mathbb{Z}}$

respectively. It is called *propositional* or *Boolean* if $Vars(\varphi) \subseteq VAR_{\mathbb{B}}$.
A formula $\varphi \in FO$ is called *real-arithmetic with Boolean values*, *integer arithmetic with Boolean values* or *mixed integer-real-arithmetic with Boolean values* if $Vars(\varphi) \subseteq VAR_{\mathbb{R}} \cup VAR_{\mathbb{B}}$, $Vars(\varphi) \subseteq VAR_{\mathbb{Z}} \cup VAR_{\mathbb{B}}$ or $Vars(\varphi) \subseteq VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}}$ respectively.

We call relations $<$, $>$ and $\neq$ *strict* and relations $\leq$, $\geq$ and $=$ *weak*. A formula in FO is called an *arithmetic constraint*, if it is of the form $p \sim q$ for some polynomial p and q and for $\sim \in REL$. It is called a *strict constraint* (*weak constraint*) for $\sim$ strict (weak). The set of all constraints is denoted by CS. The set of all constraints in a formula is obtained by $CS_{\sim} : FO \to \mathcal{P}(CS)$. We define $rel\colon CS \to REL$, $(p \sim q) \mapsto \sim$ to obtain the relation symbol of a constraint.

We define the following function for substituting arithmetic variables in polynomials and arithmetic formulas:

Definition 2.2.3 (Substitution[a])**.** *We define*

$$\cdot[\cdot/\cdot] : (POL \cup FO) \times POL \times VAR_{\mathbb{R},\mathbb{Z}} \to POL \cup FO$$

such that it holds for polynomials p, p_1, p_2, variables $x, y \in VAR_{\mathbb{R},\mathbb{Z}}$, $x \neq y$, $b \in VAR_{\mathbb{B}}$, formulas φ_1, φ_2, $\diamond \in \{+, \cdot, -\} \cup REL$, $\circ \in \{\vee, \wedge\}$ and $\mathcal{Q} \in \{\exists, \forall\}$:

$$\begin{aligned}
0[p/x] &= 0 \\
1[p/x] &= 1 \\
x[p/x] &= p \\
y[p/x] &= y \\
(p_1 \diamond p_2)[p/x] &= (p_1[p/x] \diamond p_2[p/x]) \\
b[p/x] &= b \\
(\neg\varphi_1)[p/x] &= (\neg\varphi_1[p/x]) \\
(\varphi_1 \circ \varphi_2)[p/x] &= (\varphi_1[p/x] \circ \varphi_2[p/x]) \\
(\mathcal{Q}x.\varphi_1)[p/x] &= (\mathcal{Q}x.\varphi_1) \\
(\mathcal{Q}y.\varphi_1)[p/x] &= (\mathcal{Q}y.\varphi_1[p/x])
\end{aligned}$$

[a] see Def. 5 on page 21 of [Cor17]

2.3 Semantics of propositional and arithmetic formulas

Up until now we have only defined the *syntax* of polynomials and arithmetic formulas. The symbols in the languages FO and POL don't have an inherit meaning. To emphasize the difference between the symbols and their assigned semantics, in this section we annotate operators and constants with the set they're in or operate on. For example we use $1_{\mathbb{R}}$ to denote the standard identity element of multiplication of the real numbers (otherwise widely also known as the number "one") and $\wedge_{\mathbb{B}}$ to denote the standard Boolean *and*-operator.

To assign an arithmetic formula or a polynomial a truth- or number-value we have to assign free variables to specific elements in $\mathbb{R}$ or $\mathbb{B}$.

Definition 2.3.1 (Assignment[a])**.** *An assignment is a partial function*

$$\alpha : VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} \rightharpoonup \mathbb{B} \cup \mathbb{R}$$

where $\alpha(v) \in Dom(v)$. ASS denotes the set of all assignments. For a polynomial or formula φ an assignment is called a (full) assignment for φ *if*

$$FreeVars(\varphi) \subseteq \{x \in VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} : \alpha(x) \neq \bot\}$$

and a partial assignment for φ *otherwise. The set of full assignments for a polynomial or formula φ is denoted by $Assigns(\varphi)$. Furthermore we define*

$$\cdot[\cdot/\cdot] : ASS \times (\mathbb{B} \cup \mathbb{R}) \times VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} \to ASS$$

such that

$$\alpha[d/v](v') = \begin{cases} d & \text{if } v = v', \\ \alpha(v') & \text{otherwise} \end{cases}$$

for all $\alpha \in ASS$, $d \in \mathbb{B} \cup \mathbb{R}$, v,$v' \in VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}}$.

[a] see Def. 6 on page 22 of [Cor17]

Given a polynomial or formula φ and a full assignment α for φ, we can now evaluate φ under α.

Definition 2.3.2 (Evaluation of Polynomials and Formulas[a])**.** *Let φ be a polynomial or a formula and α be a assignment for φ. We call $[\![\varphi]\!]^\alpha$ the* evaluation of φ under α *where*

$$[\![\cdot]\!]^{\cdot} : (FO \cup POL) \times ASS \rightharpoonup (\mathbb{B} \cup \mathbb{R})$$

such that

$$\begin{aligned}
[\![0]\!]^\alpha &= 0_{\mathbb{R}} \\
[\![1]\!]^\alpha &= 1_{\mathbb{R}} \\
[\![x]\!]^\alpha &= \alpha(x) \\
[\![b]\!]^\alpha &= \alpha(b) \\
[\![p_1 \diamond p_2]\!]^\alpha &= [\![p_1]\!]^\alpha \diamond_{\mathbb{R}} [\![p_2]\!]^\alpha \\
[\![p_1 \sim p_2]\!]^\alpha &= [\![p_1]\!]^\alpha \sim_{\mathbb{R}} [\![p_2]\!]^\alpha \\
[\![\neg\varphi_1]\!]^\alpha &= \neg_{\mathbb{B}} [\![\varphi_1]\!]^\alpha \\
[\![\varphi_1 \circ \varphi_2]\!]^\alpha &= [\![\varphi_1]\!]^\alpha \circ_{\mathbb{B}} [\![\varphi_2]\!]^\alpha \\
[\![\mathcal{Q}z.\varphi_1]\!]^\alpha &= \mathcal{Q}v \in Dom(z).[\![\varphi_1]\!]^{\alpha[v/z]}
\end{aligned}$$

for $x \in VAR_{\mathbb{R},\mathbb{Z}}$, $b \in VAR_{\mathbb{B}}$, $p_1, p_2 \in POL$, $\varphi_1, \varphi_2 \in FO$, $\diamond \in \{+, \cdot, -\}$, $\sim \in REL$, $\circ \in \{\wedge, \vee\}$ and $\mathcal{Q} \in \{\exists, \forall\}$ and assignment α which is a full assignment for the respective inputs. For $\xi \in FO \cup POL$ and an assignment $\alpha \in ASS$ which is not a

full assignment of ξ, the evaluation $[\![\xi]\!]^\alpha$ is undefined, i.e. $[\![\xi]\!]^\alpha = \bot$.

[a] see Def. 7 on page 23 of [Cor17]

For a polynomial p we call an assignment α for p a *zero* of p if $[\![p]\!]^\alpha = 0$.

Convention 1. *To increase readability we often omit a formal definition of an assignment as a (partially defined) function and use for an assignment*

$$\alpha : VAR_{\mathbb{B},\mathbb{R},\mathbb{Z}} \rightharpoonup \mathbb{B} \cup \mathbb{R},\ \chi \mapsto \begin{cases} a \text{ if } \chi = x \\ b \text{ if } \chi = y \\ c \text{ if } \chi = z \\ \dots \end{cases}$$

the shorthand notation $\alpha = (x \mapsto a, y \mapsto b, z \mapsto c, \dots) \in ASS$. Furthermore, if an assignment assigns variables $x = (x_1, \dots, x_m) \in VAR^m_{\mathbb{B},\mathbb{R},\mathbb{Z}}$ to values $s = (s_1, \dots, s_m) \in (\mathbb{R} \cup \mathbb{B})^m$ and if those variables and their order are obvious from the context we simply write $x \mapsto s$. Thus, we may, for example, write $[\![x_1 x_2 + x_2^2]\!]^{x \mapsto (-3,4)} = -3 \cdot 4 + 4^2 = 4$ or $[\![x_1 x_2 y]\!]^{x \mapsto (1,2), y \mapsto 2} = 1 \cdot 2 \cdot 2 = 4$.

Also note that we may use expressions of the form $[\![\cdot]\!]$ as atoms in Boolean compositions if it is obvious from the context that the evaluation maps to a Boolean value. For example we may write $[\![x \geq 0]\!]^{x \mapsto 0} \wedge [\![x = 0]\!]^{x \mapsto 0}$ instead of $\left([\![x \geq 0]\!]^{x \mapsto 0} = true_{\mathbb{B}}\right) \wedge \left([\![x = 0]\!]^{x \mapsto 0} = true_{\mathbb{B}}\right)$.

Definition 2.3.3 (Satisfiability and Solution Space[a])**.** *We call a full assignment α for an arithmetic formula φ a* solution *of φ if φ evaluates to $true_{\mathbb{B}}$ under α, i.e. if $[\![\varphi]\!]^\alpha = true_{\mathbb{B}}$. We define $\Theta(\varphi)$ as the set of all solutions of φ. If $\Theta(\varphi) \neq \emptyset$ we call φ* satisfiable*, otherwise* unsatisfiable*. If φ evaluates to $true_{\mathbb{B}}$ under every full assignment of φ we say φ is a* tautology*.*

We call formulas φ and ψ equisatisfiable *if $\Theta(\varphi) \neq \emptyset$ if and only if $\Theta(\psi) \neq \emptyset$. We call them* equivalent *if $\Theta(\varphi) = \Theta(\psi)$.*

[a] see Def. 8 on page 24 of [Cor17]

2.4 Normalization and Degree of a Variable in a Polynomial

In oder to simplify some definitions, lemmas, theorems and proofs, we define a *normal form* for formulas, constraints and polynomials. Consider for example the formulas $\varphi_1 \equiv xyx \geq 0 \wedge yx < 0$ and $\varphi_2 \equiv x^2 y \geq 0 \wedge xy < 0$. Obviously φ_1 and φ_2 differ in syntax, however they are semantically equivalent, i.e. $[\![\varphi_1]\!]^\alpha = [\![\varphi_2]\!]^\alpha$ for every variable assignment α. Our goal then is to assign each formula/constraint/polynomial a formula/constraint/polynomial which is equivalent and has a "nicer" form. Note that the formula $\varphi_3 \equiv x < 0 \wedge y > 0$ is equivalent to the formulas φ_1 (and φ_2). It also more or less solves the SMT-instance "Does there exist a satisfying variable assignment for φ_1". This however is not the goal of normalizing formulas as we want to define normal forms to ease up computation to solve such instances. We want a normal form to be

easily computable on top of easy to compute with. Hence, we define normalized forms by applying rules similar to the associative, commutative and distributive properties.

We adopt the normalized form of polynomials from Section 2.4 of [Cor17]. We also define the normalized form of polynomials *in* one of their variables. For this we first define the degree of variables in polynomials:

A variable x of a polynomial p has degree $\deg(p,x) := k$ if $k > 0$ is the biggest exponent of x in the normal form of p in x and $\deg(p,x) := 0$ if the variable x is not contained in the normal form of p in x. Because normal forms are unique, the degrees of (their) variables is well-defined. We have for example $\deg(p, x_1) = 2$ and $\deg(p, x_4) = 0$ for $p = (x_2x_3 + 2x_3)x_1^2 + (3x_2)x_1 + 1$. We say that a polynomial p is *linear* or *quadratic* in a variable x if $\deg(p,x) \leq 1$ or $\deg(p,x) \leq 2$ respectively. [1]

Then for each polynomial p and real-valued variable x there are unique polynomials $q_0, \ldots, q_{\deg(p,x)}$ in normalized forms as defined in [Cor17] such that

$$r = q_{\deg(p,x)} x^{\deg(p,x)} + \cdots + q_1 x^1 + q_0$$

is equivalent to p, i.e. $[\![r]\!]^\alpha = [\![p]\!]^\alpha$ for every full assignment α of p. We call r the normalized form of p *in the variable* x (or *in* x for short).

We define $n \sim 0$ as the normal form (in a variable) of a constraint $p \sim q$ where n is the normal form of the polynomial $p - q$ (in a variable). We define

$$Pol\colon CS \to POL,\ p \sim 0 \mapsto p$$

to obtain the (left-hand) polynomial of a normalized constraint. As every constraint has a unique and equivalent normalized form, this function is well-defined. Further we define

$$Pols\colon FO \to \mathcal{P}(POL),\ \varphi \mapsto \{Pol(c) : c \in CS_\sim(\varphi)\}$$

to obtain all polynomials in a formula.

The normal form of formulas (in a variable) can also be constructed by normalizing their respectively included constraints (in a variable) while preserving the Boolean structure.

2.5 Virtual Substitution

Virtual Substitution (VS) is a procedure for the elimination of variables and quantifiers from quadratic real-arithmetic formulas [Cor10]. A normalized real-arithmetic constraint as obtained in Section 2.4 is a polynomial compared to 0 by a relation in *REL*. VS utilizes the fact, that two variable assignments α_1 and α_2 for a polynomial p, for which the sign of the evaluated polynomial is equal, i.e. $sgn([\![p]\!]^{\alpha_1}) = sgn([\![p]\!]^{\alpha_2})$, are either both a solution or both not a solution for a normalized constraint $p \sim 0$. The assignment space of a real-arithmetic polynomial p with n variables, which can equivalently be thought of as $\mathbb{R}^n$, can always be split into finitely many disjoint connected[2] subsets, in each of which the sign of the polynomial is constant under evaluation by assignments in those respective subsets. It is then sufficient to check only one *test candidate* of each of those *sign-invariant regions* to determine wether a given normalized constraint $p \sim 0$ is satisfiable.

[1] Note that this definition of the degrees of a polynomial differs from the definition of *the* degree of a polynomial in [Cor17] (Def. 9, p. 25)

[2] For every pair of points of the set there is a continuos path between them that lies completely within the set.

Example 2.5.1. *To illustrate this, let for example* $p_1 \equiv 3x+1$ *and* $p_2 \equiv x^2-2x$ *be two polynomials. We consider* $\mathbb{R}$ *to be the set of all possible assignments of* p_1 *and/or* p_2*. Then, because* p_1 *and* p_2 *are univariate polynomials, they have sign-invariant intervals of their assignment-spaces for the variable* x *exactly at and in between their (finitely many) zeros, since polynomials are continuous and thus the sign of the function can only change at their zeros. See Figure 2.1 for an illustration of this example.*

Consider the formula $\varphi \equiv p_1 \geq 0 \wedge p_2 \neq 0$*. To determine whether* φ *is satisfiable, we can now obtain the zeros of both constraints (see Figure 2.1) and check for each interval* I *(of maximum length) in which* both *polynomials are sign-invariant one representative, i.e. some arbitrary point* $s \in I$*, whether the formula is satisfied under assignment* $x \mapsto s$*, i.e. whether* $[\![\varphi]\!]^{x \mapsto s} = true$*.*

All zeros of both p_1 *and* p_2 *are* $-\frac{1}{3}$*,* 0 *and* 2*. Thus,* p_1 *and* p_2 *are both sign-invariant in the intervals* $(-\infty,-\frac{1}{3})$,$[-\frac{1}{3},-\frac{1}{3}]$,$(-\frac{1}{3},0)$,[0,0],(0,2),[2,2],$(2,\infty)$*. We then take for each of these intervals a representative, e.g.* -1*,* $-\frac{1}{3}$*,* $-\frac{1}{6}$*,* 0*,* 1*,* 2 *and* 3*.*

Therefore, the formula φ *is satisfiable iff it is satisfied under at least one of those representatives. We therefore get:*

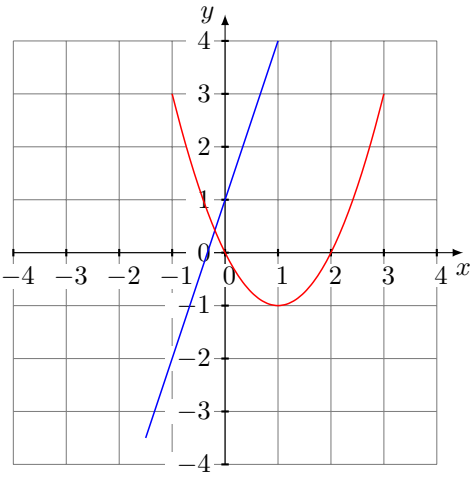
$$\exists \alpha \in ASS. [\![\varphi]\!]^{\alpha} \Leftrightarrow \bigvee_{s \in \{-1,-\frac{1}{3},-\frac{1}{6},0,1,2,3\}} [\![\varphi]\!]^{x \mapsto s}$$

Because of

$$\begin{aligned}
&[\![\varphi]\!]^{x \mapsto \frac{1}{3}} \\
\Leftrightarrow &[\![3x+1 \geq 0 \wedge x^2 - 2x \neq 0]\!]^{x \mapsto \frac{1}{3}} \\
\Leftrightarrow &3 \cdot \frac{1}{3} + 1 \geq 0 \wedge \left(\frac{1}{3}\right)^2 - 2 \cdot \frac{1}{3} \neq 0 \\
\Leftrightarrow &2 \geq 0 \wedge -\frac{5}{9} \neq 0 \\
\Leftrightarrow &true
\end{aligned}$$

we can thus derive that φ *is satisfiable (with one solution being* $x \mapsto \frac{1}{3}$*).*

As we allow for *multivariate* polynomials, the steps above are not directly transferable to the general case. Every zero of a multivariate polynomial over n variables is a n-dimensional point in its assignment-space. Thus, those zeros cannot divide the assignment space into sign-invariant *intervals*. To circumvent this issue we may *treat* all but one variable as having a fixed assignment. Note that for a multivariate polynomial $p \in POL[x_1,\ldots,x_n]$ and any assignment α for $x_2,\ldots,x_n$, the function $p_\alpha : \mathbb{R} \to \mathbb{R}$, $r \mapsto [\![p]\!]^{\alpha[r/x_1]}$ defines a standardly defined univariate polynomial. We have to be careful not to mix up the notion of polynomials as words in a formal language and the notion of polynomials as functions as they are normally understood. However, in the context of this explanation, we forgo this distinction. Note that, as it was the case above, it is sufficient to check one representative from each sign-invariant interval of p_α to determine whether a constraint $p_\alpha \sim 0$ is satisfiable. We want to find a (parameterized) description of the zeros of p_α which *symbolically* depends on α, and exist under certain symbolic side conditions. We also want to describe symbolically defined intervals whose bounds are given by those zeros. Furthermore, just as in the steps taken above, we want to describe a *test candidate* of those (symbolic) intervals. We find that we only have a small number of said symbolic side conditions



x	$sgn(3x+1)$	$sgn(x^2-2x)$
$(-\infty,-\frac{1}{3})$	-1	$+1$
$[-\frac{1}{3},-\frac{1}{3}]$	0	$+1$
$(-\frac{1}{3},0)$	$+1$	$+1$
$[0,0]$	$+1$	0
$(0,2)$	$+1$	-1
$[2,2]$	$+1$	0
$(2,+\infty)$	$+1$	$+1$

Figure 2.1: Illustration of Example 2.5.1

and intervals. Therefore a whole range of different assignments α can be checked "simultaneously" by checking one of the said side conditions and its associated test candidates. If we do this for every possible side condition, we have, in effect, checked the unrestricted constraint $p \sim 0$ as well.

For quadratic univariate polynomials the solution formula describing their zeros contains square roots. Thus, to represent a parameterized description of the zeros of p_α we define *square root expressions*:

Definition 2.5.1 (Square Root Expression[a]). *For polynomial $p,q,r,s \in POL$ we call*

$$\frac{p+q\sqrt{r}}{s}$$

a square root expression. *The set of all square root expressions is denoted by SqrtEx. The set of all square root expressions in the variables $x_1,\ldots,x_n$ is given by*

$$SqrtEx[x_1,\ldots,x_n] = \left\{ \frac{p+q\sqrt{r}}{s} : p,q,r,s \in POL[x_1,\ldots,x_n] \right\}$$

We may write $\frac{p}{s}$ as a short-hand form for $\frac{p+0\sqrt{0}}{s}$.

[a] see Def. 20 on page 54 of [Cor17]

Definition 2.5.1 allows us to define square root expressions of the form $\frac{p+q\sqrt{r}}{0}$ where we "divide" by 0. More generally, an assignment α may evaluate the "denominator" of a square root expression to 0. Therefore we cannot sensibly define evaluations of these terms (under those assignments). In practice we only evaluate square root expressions (explicitly as in Definition 3.2.2 or implicitly using Definition 2.5.3) under specific assignments when certain side conditions are met.

Square root expressions can be used to describe *symbolic zeros* of a multivariate

polynomial viewed as an univariate polynomial with symbolic (polynomial) coefficients. As mentioned above, we also need a description for test candidates of the symbolic intervals bound by these zeros. In the non-symbolic example from above we simply chose a convenient value for each interval bound by the zeros of the examined polynomial. For the symbolic case, two symbolic zeros z_1,$z_2 \in SqrtEx$ may lie arbitrarily close to each other given arbitrary assignments. Thus, we simply define ϵ as a placeholder for a sufficiently small *positive* value such that for any two symbolic zeros z_1,$z_2 \in SqrtEx$ the expression $z_1 + \epsilon$ (or $z_2 + \epsilon$) represents a value in between the zeros (again given an assignment). Furthermore we define $-\infty$ as an arbitrarily small value which is smaller than any zero in a given context. We do not need to define $+\infty$ equivalently because we may create a representative which is greater than all zeros by using ϵ.

Definition 2.5.2 (Test Candidate[a])**.** *The set of all (possible) test candidates is given by:*

$$TCS = SqrtEx \cup \{t + \epsilon : t \in SqrtEx\} \cup \{-\infty\}$$

For a real-valued variable x and a real-arithmetic constraint c, in which x is quadratic, the set of all test candidates for the variable x in the constraint c *is obtained by:*

$$tcs : VAR_{\mathbb{R}} \times CS \rightharpoonup \mathcal{P}(TCS),\ \ (x, p_1x^2 + p_2x + p_3 \sim 0)$$
$$\mapsto \begin{cases} \{-\infty, \frac{-p_3}{p_2}, \frac{-p_2-\sqrt{p_2^2-4p_1p_3}}{2p_1}, \frac{-p_2+\sqrt{p_2^2-4p_1p_3}}{2p_1}\} & \text{if } \sim \text{ is weak} \\ \{-\infty, \frac{-p_3}{p_2} + \epsilon, \frac{-p_2-\sqrt{p_2^2-4p_1p_3}}{2p_1} + \epsilon, \frac{-p_2+\sqrt{p_2^2-4p_1p_3}}{2p_1} + \epsilon\} & \text{if } \sim \text{ is strict} \end{cases}$$

The set of all test candidates for the variable x in a formula φ *in which a $x \in VAR_{\mathbb{R}}$ only occurs at most quadratic is given by:*

$$tcs : VAR_{\mathbb{R}} \times FO \rightharpoonup \mathcal{P}(TCS), (x,\varphi) \mapsto \bigcup_{c \in CS_{\sim}(\varphi)} tcs(x,c)$$

The side condition of a test candidate *is given by:*

$$sc : TCS \to FO, t \mapsto \begin{cases} sc(t') & \text{if } t \text{ is of the form } t' + \epsilon, t' \in TCS \\ s \neq 0 \wedge r \geq 0 & \text{if } t \text{ is of the form } \frac{p+q\sqrt{r}}{s}, \\ & q \text{ or } r \text{ are not the polynomial } 0 \\ s \neq 0 & \text{if } t \text{ is of the form } \frac{-p}{s} \\ true & \text{if } t = -\infty \end{cases}$$

[a] see Def. 21 on page 55 of [Cor17]

We have now defined test candidates and their side conditions for eliminating variables in real-arithmetic formulas. Variables cannot be substituted directly by the constructed test candidates. This would result in expressions that are not element of *FO*, therefore, amongst other things, VS could not be applied again for the remaining variables. *Virtual* Substitution defines transformation rules for a real-arithmetic formula φ with a test candidate t for the variable x, such that it maps φ to ψ, $Vars(\psi) = Vars(\varphi) \setminus \{x\}$ and φ and ψ are equisatisfiable when substituting x by t.

Definition 2.5.3 (Virtual Substitution[a])**.** Virtual Substitution *of a variable in a real-arithmetic formula by a test candidate is defined by a function*

$$\cdot[\cdot \mathbin{/\!\!/} \cdot] : FO \times TCS \times VAR_{\mathbb{R}} \to FO$$

that is specified by Substitution rules as shown in [Cor10].

[a] see Def. 22 on page 57 of [Cor17]; see [Cor10]

[Cor17] also states in Theorem 2 on page 58 that:

Theorem 2.5.1.

$$[\![\exists x.\ \varphi]\!]^\alpha = \left[\!\!\left[\bigvee_{t \in tcs(x,\varphi)} (sc(t) \wedge \varphi[t \mathbin{/\!\!/} x]) \right]\!\!\right]^\alpha$$

$$[\![\forall x.\ \varphi]\!]^\alpha = \left[\!\!\left[\bigwedge_{t \in tcs(x,\varphi)} (sc(t) \to \varphi[t \mathbin{/\!\!/} x]) \right]\!\!\right]^\alpha$$

for any variable assignment α of $\exists x.\varphi$ (or of $\forall x.\varphi$).

Example 2.5.2. *Consider again Example 2.5.1 and formula*

$$\varphi \equiv p_1 \geq 0 \wedge p_2 \neq 0 \equiv 3x + 1 \geq 0 \wedge x^2 - 2x \neq 0$$

We have already shown φ to be satisfiable and want to recheck this result by using VS and Theorem 2.5.1.
As x is the only variable contained in φ, we get that φ is satisfiable iff $[\![\exists x.\varphi]\!]^\alpha$ for any *variable assignment α. It holds:*

$$tcs(x,3x+1 \geq 0) = \left\{ -\infty, \frac{-1}{3}, \frac{-3+\sqrt{3^2-4\cdot 0\cdot 1}}{2\cdot 0}, \frac{-3-\sqrt{3^2-4\cdot 0\cdot 1}}{2\cdot 0} \right\}$$

And thus $\frac{-1}{3} \in tcs(x,\varphi) = tcs(x,3x+1 \geq 0) \cup tcs(x,x^2-2x \neq 0)$. Note that even tough we "divide by zero", the expressions above are well-defined as we are not considering them as real-valued terms but as words in the formal language TCS.
We get $sc(\frac{-1}{3}) \equiv (3 \neq 0)$ and according to [Cor10]:

$$(3x+1 \geq 0)\left[\frac{-1}{3} \mathbin{/\!\!/} x\right] \equiv (3 > 0 \wedge 3\cdot(-1) + 1\cdot 3 \geq 0) \vee (3 < 0 \wedge 3\cdot(-1) + 1\cdot 3 \leq 0)$$

$$(x^2 - 2x \neq 0)\left[\frac{-1}{3} \mathbin{/\!\!/} x\right] \equiv ((-1)^2 - 2\cdot(-1)\cdot 3 \neq 0)$$

Let α be an arbitrary assignment. It holds:

$$\left[\!\!\left[sc\left(\frac{-1}{3}\right) \right]\!\!\right]^\alpha \Leftrightarrow 3 \neq 0 \qquad \Leftrightarrow \mathit{true}$$

$$\left[\!\!\left[(3x+1 \geq 0)\left[\frac{-1}{3} \mathbin{/\!\!/} x\right] \right]\!\!\right]^\alpha \Leftrightarrow (3 > 0 \wedge 0 \geq 0) \vee (3 < 0 \wedge 0 \leq 0) \qquad \Leftrightarrow \mathit{true}$$

$$\left[\!\!\left[(x^2 - 2x \neq 0)\left[\frac{-1}{3} \mathbin{/\!\!/} x\right] \right]\!\!\right]^\alpha \Leftrightarrow 7 \neq 0 \qquad \Leftrightarrow \mathit{true}$$

And because of

$$\begin{aligned}
&\left[\!\left[sc\left(\frac{-1}{3}\right) \wedge \varphi\left[\frac{-1}{3} /\!\!/ x\right]\right]\!\right]^{\alpha} \\
\Leftrightarrow &\left[\!\left[sc\left(\frac{-1}{3}\right) \wedge (3x+1 \geq 0)\left[\frac{-1}{3} /\!\!/ x\right] \wedge (x^2 - 2x \neq 0)\left[\frac{-1}{3} /\!\!/ x\right]\right]\!\right]^{\alpha} \\
\Leftrightarrow &\left[\!\left[sc\left(\frac{-1}{3}\right)\right]\!\right]^{\alpha} \wedge \left[\!\left[(3x+1 \geq 0)\left[\frac{-1}{3} /\!\!/ x\right]\right]\!\right]^{\alpha} \wedge \left[\!\left[(x^2 - 2x \neq 0)\left[\frac{-1}{3} /\!\!/ x\right]\right]\!\right]^{\alpha} \\
\Leftrightarrow &\mathit{true} \wedge \mathit{true} \wedge \mathit{true} \Leftrightarrow \mathit{true}
\end{aligned}$$

we get:

$$\mathit{true} \Leftrightarrow \left[\!\left[sc\left(\frac{-1}{3}\right) \wedge \varphi\left[\frac{-1}{3} /\!\!/ x\right]\right]\!\right]^{\alpha} \Rightarrow \left[\!\left[\bigvee_{t \in tcs(x,\varphi)} (sc(t) \wedge \varphi[t /\!\!/ x])\right]\!\right]^{\alpha} \Leftrightarrow [\![\exists x.\varphi]\!]^{\alpha}$$

Thus, φ is satisfiable.

Theorem 2.5.1 allows us to eliminate quantifiers in a formula to construct an equisatisfiable formula. Furthermore, because in the context of satisfiability checking every formula φ with a free variable x can be thought of as the formula $\exists x.\ \varphi$, Theorem 2.5.1 also allows us to reduce an real-arithmetic formula to a constant real-arithmetic formula which can be checked for satisfiability relatively easily. However, note that the number of terms of the reduced formula grows exponentially, thus this approach is not feasible in many cases.

In the next chapter, we build upon the main idea underlying VS, namely the idea of considering each variable separately (at first) and splitting the real number-line into intervals in which the analyzed formula has some invariant property under changes of a variable assignment of the respective variable. We also use the Virtual Substitution 2.5.3 directly to encode expressions involving square root terms into syntactically allowed formulas.

2.6 SMT solving

SAT solvers check satisfiability or even generate a solution of propositional (Boolean) formulas. Due to their efficiency gains, modern SAT solvers have found a wide range of applicability in academia and industry alike. Even though the number of possible assignments for propositional formulas grows exponentially with the number of their contained variables, propositional formulas have a limited expressiveness compared to many naturally arising mathematical problems. Satisfiability Modulo Theories (SMT) formulas extend the expressive power of SAT formulas by incorporating the use of *background theories*. In the context of this thesis, we consider formulas as defined in 2.2.2 as instances of SMT. The expressiveness of SMT formulas and in particular formulas as defined in 2.2.2 is much stronger and thus offers an even wider range of applicability, compared to SAT formulas. SMT solvers most often are based on an underlying SAT solver, which is applied to Boolean abstractions of SMT formulas in which theory expressions are substituted by propositional variables. Decades of research into SAT solvers, which improved them significantly, therefore also benefit SMT solver directly. In the early 2000s *lazy SMT solvers* such as DPLL(T) began

to emerge [BDS02][MR02]. Such solvers improve their efficiency even further by improving the interplay between the incorporated SAT solver and *theory solvers* for background theories. Whereas in previous designs, SAT solvers were treated as black boxes, lazy SMT solvers enable communication of theory solvers and the SAT solver. If a theory solver determines a set of predicates/constraints to be unsatisfiable it can offer an *explanation*, also called a *conflict generalization*. These conflict generalizations can then be used by the underlying SAT solver to refine a next guess of a possible solution. Figure 2.2 which can be found in [CÁ11] illustrates the basic composition of lazy SMT solvers.

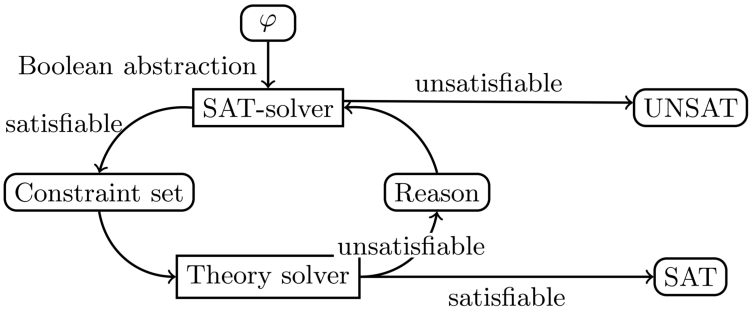


Figure 2.2: Lazy SMT solver

In this thesis, we are particulary interested in the step of generalization of a conflict for a set of real-arithmetic constraints. For this we formalize such a conflict (generalizations) as follows:

Definition 2.6.1 (Conflict Generalization)**.** *We say that real-arithmetic constraints $c_1,\ldots,c_m$ in variables $x_1,\ldots,x_n$,y which are at most quadratic in y have a* conflict *in a* sample $s \in \mathbb{R}^n$*, if*

$$\neg\exists t \in \mathbb{R}. \left[\!\!\left[\bigwedge_{i=1}^{m} c_i \right]\!\!\right]^{x\mapsto s, y\mapsto t}$$

A conflict generalization *of this conflict is a formula φ in variables $x_1,\ldots,x_n$ such that $(x \mapsto s) \in \Theta(\varphi)$ and*

$$\forall \alpha \in \Theta(\varphi). \neg\exists t \in \mathbb{R}. \left[\!\!\left[\bigwedge_{i=1}^{m} c_i \right]\!\!\right]^{\alpha[t/y]}$$

In other words: A set $c_1,\ldots,c_m$ of real-arithmetic constraints have a conflict in sample s, if the partial assignment of their variables given by the sample cannot be extended to a solution to *all* constraints. A conflict generalization is a description of s and other samples for which a conflict also exists. Note that a formula φ that only describes the sample s, i.e. $\Theta(\varphi) = \{x \mapsto s\}$, would be a conflict generalization. However, this would not be very helpful as we ideally want to eliminate all samples which are not part of a solution of $\wedge_{i=1}^{m} c_i$. We could, in theory, construct the conflict

generalization so that it describes *all* conflicts and therefore decides if the input constraints are satisfiable or not. However, in designing an algorithm which constructs a conflict generalization, we have to balance between two requirements. As mentioned, we want to describe as many conflicts as possible but we also want to do this as efficient as possible. Therefore we construct the conflict generalization such that we only describe samples for which a "similar" conflict exist to the given conflict of the input constraints and the given sample. In the following chapter we formulate how we define this similarity, describe a construction for conflict generalizations and a heuristic to differentiate "good" and "bad" conflict generalizations. We then present algorithms to find conflict generalizations under those considerations.

Convention 2. *In the rest of this thesis, if not otherwise mentioned, all formulas and variables are real-arithmetic in the variables* $x_1 \ldots, x_n, y$ *and at most quadratic in* y*.*

Chapter 3

Generalizing conflicts in real-arithmetic quadratic formulas

3.1 Ordered covering of $\mathbb{R}$

In this section we first consider univariate constraints to understand the main idea for the proposed conflict generalization of this chapter. Of course, if a conflict is found for univariate constraints (all constraints are only in the variable y), then we already know that $\wedge_{i=1}^{m} c_i$ is unsatisfiable. Therefore $\varphi \equiv \mathit{true}$ would be the only reasonable conflict generalization. Despite this, we construct φ differently so that this construction may inform how to construct the conflict generalization for the multivariate case.

Example 3.1.1. *Consider the unsatisfiable constraints* $c_1 \equiv y^2 - 2y \leq 0$ *and* $c_2 \equiv -y - 1 > 0$. *Figure 3.1 shows the plot of their respective polynomials and also indicates the intervals for assignments for* y $(-\infty,0)$,$(2,\infty)$ *in which* c_1 *is not satisfied and* $[-1,\infty)$ *in which* c_2 *is not satisfied.*

The reason why $c_1 \wedge c_2$ is not satisfiable in Example 3.1.1 is because these intervals for assignments for y, in which either c_1 or c_2 is not satisfied, cover $\mathbb{R}$. In other words: Under each assignment of y either c_1 or c_2 is not satisfied. We want to find an sufficient and necessary condition under which a set of intervals cover $\mathbb{R}$ which then can easily be encoded in a formula.

In order to do so, we first introduce a few notations and definitions. To increase readability of formulas which depend on the bounds of intervals and on the fact wether the bounds are closed or open we define:

$$\mathbb{R}_{+\epsilon} = \{r,r+ : r \in \mathbb{R}\} \cup \{-\infty\}$$

as the set of left interval bounds and

$$\mathbb{R}_{-\epsilon} = \{r,r- : r \in \mathbb{R}\} \cup \{+\infty\}$$

as the set of right interval bounds. We further define

$$\mathbb{R}_{\pm\epsilon} = \mathbb{R}_{+\epsilon} \cup \mathbb{R}_{-\epsilon} = \{r,r-,r+ : r \in \mathbb{R}\} \cup \{-\infty, +\infty\}$$

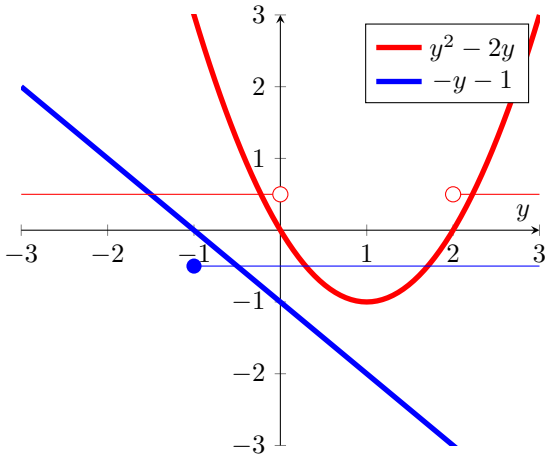


Figure 3.1: Univariate polynomials

and introduce the following notation which is used from now on instead of the standard interval notation:

$$\begin{aligned}
[-\infty, b-] &= (-\infty,b) \\
[-\infty, b] &= (-\infty,b] \\
[-\infty, +\infty] &= (-\infty, +\infty) \\
[a+, b-] &= (a,b) \\
[a+, b] &= (a,b] \\
[a+, +\infty] &= (a, +\infty) \\
[a, b-] &= [a,b) \\
[a, b] &= [a,b] \\
[a, +\infty] &= [a, +\infty)
\end{aligned}$$

for $a,b \in \mathbb{R}, a < b$, so that every interval $\emptyset \neq I \in \mathbb{I}$ can be written in the form $I = [a,b]$ for $a,b \in \mathbb{R}_{\pm\epsilon}$. We also define

$$[a\psi_a,b\psi_b] = \emptyset \text{ for } a,b \in \mathbb{R}, a > b \text{ or } a = b \wedge (\psi_a = + \vee \psi_b = -)$$

Using this notation we can define the functions *leftBound* and *rightBound* which extract the bounds of intervals:

$$\begin{aligned}
\mathit{leftBound} &: \mathbb{I} \rightharpoonup \mathbb{R}_{+\epsilon}, \quad \emptyset \mapsto \bot, [a,b] \mapsto a \\
\mathit{rightBound} &: \mathbb{I} \rightharpoonup \mathbb{R}_{-\epsilon}, \quad \emptyset \mapsto \bot, [a,b] \mapsto b
\end{aligned}$$

We call interval bounds $x \in \mathbb{R}$ *closed* and $x \in \mathbb{R}_{\pm\epsilon} \setminus \mathbb{R}$ *open*. We also define

$$\pi_{\mathbb{R}} : \mathbb{R}_{\pm\epsilon} \rightharpoonup \mathbb{R}, x \mapsto \begin{cases} \bot \text{ for } x \in \{-\infty, +\infty\} \\ r \quad \text{if } x \in \{r-, r, r+\} \text{ for some } r \in \mathbb{R} \end{cases}$$

as the real component of an interval bound. Furthermore we define an order $<$ and $\leq$ on $\mathbb{R}_{\pm\epsilon}$ as follows:

$$a\psi_a < b\psi_b :\Leftrightarrow a < b \vee (a = b \wedge (\psi_a,\psi_b) \in \{(-,\emptyset), (-,+), (\emptyset,+)\})$$
$$a\psi_a \leq b\psi_b :\Leftrightarrow a\psi_a < b\psi_b \vee (a = b \wedge \psi_a = \psi_b)$$

for $a,b \in \mathbb{R}, \psi_a,\psi_b \in \{-,\emptyset,+\}$ where $\emptyset$ stands for the omission of a symbol $+$ or $-$. Note that $\emptyset$ does *not* denote an empty set in this context. Furthermore $-\infty < a < +\infty$ and $-\infty \leq a \leq +\infty$ for all $a \in \mathbb{R}_{\pm\epsilon} \setminus \{-\infty, +\infty\}$.

Intuitively, we expand the real numbers so that $r+$ $(r-)$ is a number greater (smaller) than $r \in \mathbb{R}$ but smaller (greater) than any other real number which is greater (smaller) than r.
Using the introduced notation we define:

Definition 3.1.1 (Gap-less Intervals)**.** *For two intervals I and J we define $I \between J$ such that:*
If $I = \emptyset$ or $J = \emptyset$ or $rightBound(I) = +\infty$ or $leftBound(J) = -\infty$, then:

$$I \between J :\Leftrightarrow \mathit{false}$$

If not first case and both $rightBound(I)$ and $leftBound(J)$ are open:

$$I \between J :\Leftrightarrow \Big(\pi_{\mathbb{R}}(rightBound(I)) > \pi_{\mathbb{R}}(leftBound(J)) \Big)$$

If not first case and one or both of $rightBound(I)$ and $leftBound(J)$ are closed:

$$I \between J :\Leftrightarrow \Big(\pi_{\mathbb{R}}(rightBound(I)) \geq \pi_{\mathbb{R}}(leftBound(J)) \Big)$$

We also define the relation $\between^{>}$ on intervals I and J such that:

$$I \between^{>} J :\Leftrightarrow I \between J \wedge rightBound(J) > rightBound(I)$$

The case distinction for closed and open intervals bounds is necessary to ensure in Definition 3.1.2 below, that the real-valued components of interval bounds is even then covered by at least one of the intervals, if both of their respective bound are open and thus do not include its bounds. The relations $\between$ and $\between^{>}$ are not symmetrical. Also note that neither the condition $I \between J$ nor $I \between^{>} J$ are sufficient for two intervals I and J to *intersect*. However this definition is useful to make the following definition and observations, which we then use to formulate a conflict generalization:

Definition 3.1.2 (Ordered Covering)**.** *A list $I_1, \ldots, I_m$ is called an* ordered covering*, if*

- $leftBound(I_1) = -\infty$,
- $rightBound(I_m) = +\infty$ *and*
- $I_i \between I_{i+1}$ *holds for every* $i = 1, \ldots, m-1$.

The list is called a strong ordered covering*, if*

- $\mathit{leftBound}(I_1) = -\infty$,
- $\mathit{rightBound}(I_m) = +\infty$ *and*
- $I_i \mathrel{\text{⧏}}^{>} I_{i+1}$ *holds for every* $i = 1, \ldots, m-1$.

Lemma 3.1.1. *For a (strong) ordered covering* $I_1, \ldots, I_m$*, it holds that* $\bigcup_{i=1}^{m} I_i = \mathbb{R}$.

Proof. Let $I_1, \ldots, I_m$ be an ordered covering. We proof by induction that $I_1 \cup \ldots \cup I_n = [-\infty, \max_{i=1,\ldots,n} \mathit{rightBound}(I_i)]$ for all $n = 1, \ldots, m$. Note that for non-strong ordered coverings, the equality $\max_{i=1,\ldots,n} \mathit{rightBound}(I_i) = \mathit{rightBound}(I_n)$ does *not* necessarily hold true for all $n = 1, \ldots, m$.

Base Case

It holds $I_1 = [\mathit{leftBound}(I_1), \mathit{rightBound}(I_1)] = [-\infty, \max_{i=1} \mathit{rightBound}(I_i)]$.

Induction Step

Let $n \in \{1, \ldots, m-1\}$ such that $I_1 \cup \ldots \cup I_n = [-\infty, \max_{i=1,\ldots,n} \mathit{rightBound}(I_i)]$. Because $I_n \mathrel{\text{⧏}} I_{n+1}$ is not *false* we know that (at least) the weak relation $\pi_{\mathbb{R}}(\mathit{rightBound}(I_n)) \geq \pi_{\mathbb{R}}(\mathit{leftBound}(I_{n+1}))$ holds. If the strong relation holds, then $I_n \cap I_{n+1} \neq \emptyset$ and because $I_1 \cup \ldots \cup I_n = [-\infty, \max_{i=1,\ldots,n} \mathit{rightBound}(I_i)]$ it then also holds that $I_1 \cup \ldots \cup I_n \cup I_{n+1} = [-\infty, \max_{i=1,\ldots,n,n+1} \mathit{rightBound}(I_i)]$. Otherwise it holds that

$$b := \pi_{\mathbb{R}}(\mathit{rightBound}(I_n)) = \pi_{\mathbb{R}}(\mathit{leftBound}(I_{n+1}))$$

and at least one of $\mathit{rightBound}(I_n)$ and $\mathit{leftBound}(I_{n+1})$ is closed.
Then

$$\begin{aligned}
[-\infty, \mathit{rightBound}(I_{n+1})] &= [-\infty, b-] \cup \underbrace{[b,b]}_{b \in I_n \text{ or } b \in I_{n+1}} \cup [b+, \mathit{rightBound}(I_{n+1})] \\
&\subseteq (I_1 \cup \ldots \cup I_n) \cup (I_n \cup I_{n+1}) \cup I_{n+1} \\
&= I_1 \cup \ldots \cup I_{n+1}
\end{aligned}$$

Because already $I_1 \cup \ldots \cup I_n = [-\infty, \max_{i=1,\ldots,n} \mathit{rightBound}(I_i)]$ and $\sup(I_1 \cup \ldots \cup I_n) = \max_{i=1,\ldots,n} \mathit{rightBound}(I_i)$ it then also holds that $I_1 \cup \ldots \cup I_{n+1} = [-\infty, \max_{i=1,\ldots,n+1} \mathit{rightBound}(I_i)]$.
By induction we have thus shown, that

$$I_1 \cup \ldots \cup I_m = [-\infty, \max_{i=1,\ldots,m} \mathit{rightBound}(I_i)] = [-\infty, \mathit{rightBound}(I_m)] = [-\infty, +\infty] = \mathbb{R}$$

Because a strong ordered covering is especially also an ordered covering, we have shown both cases.

□

Theorem 3.1.2. *A finite set* U *of intervals covers* $\mathbb{R}$*, i.e.* $\bigcup_{I \in U} I = \mathbb{R}$*, if and only if there exist* $m \in \{1, \ldots, |U|\}$ *and an injective function* $I : \{1, \ldots, m\} \mapsto U$ *such that* $I(1), \ldots, I(m)$ *is an ordered covering.*
This equivalence also holds for strong *ordered coverings.*

Proof. Let U be a finite set of intervals which covers $\mathbb{R}$. Let $U' \subseteq U$ be a subset of U that still covers $\mathbb{R}$ but such that no strict subset $U'' \subsetneq U'$ covers $\mathbb{R}$. Then for every $I \in U'$ there exist a point $x(I) \in \mathbb{R}$ that is only covered by I and no other interval in U'. The function $x : U' \to \mathbb{R}$ is thus injective. Let $I : \{1,\ldots,m\} \to U$ such that $\{I(1),\ldots,I(m)\} = U'$ and such that

$$x(I(1)) < x(I(2)) < \ldots < x(I(m))$$

We show that $I(1),\ldots,I(m)$ is a strong ordered covering.
For $i > 1$ it holds that $x(I(1)) \notin I(i)$. Thus, for $\mathit{leftBound}(I(i)) = -\infty$ to hold, it must follow that $\mathit{rightBound}(I(i)) < x(I(1))$. But then $x(I(i)) < x(I(1))$ which is a contradiction. Therefore it holds for all $i > 1$ that $\mathit{leftBound}(I(i)) \neq -\infty$. But because U' is a finite set of intervals which cover $\mathbb{R}$ for at least one $J \in U'$ it holds that $\mathit{leftBound}(J) = -\infty$. Therefore it follows that $\mathit{leftBound}(I(1)) = -\infty$. Analogously it holds that $\mathit{rightBound}(I(i)) = +\infty \Leftrightarrow i = m$.
Assume there exist $i = 1,\ldots,m-1$ such that $I(i) \not\boxtimes I(i+1)$. Then $\pi_{\mathbb{R}}(\mathit{rightBound}(I(i))) \leq \pi_{\mathbb{R}}(\mathit{leftBound}(I(i+1)))$ and there exists $x \in [\pi_{\mathbb{R}}(\mathit{rightBound}(I(i))),\pi_{\mathbb{R}}(\mathit{leftBound}(I(i+1)))]$ such that $\mathit{rightBound}(I(i)) < x < \mathit{leftBound}(I(i+1))$ and thus $x \notin I(i) \cup I(i+1)$. Because however the intervals in U' cover $\mathbb{R}$, there must be an $j \in \{1,\ldots,m\} \setminus \{i,i+1\}$ such that $x \in I(j)$. W.l.o.g. assume $j < i$. Then we get

$$\mathit{leftBound}(I(j)) \leq x(j) < x(i) \leq \mathit{rightBound}(I(i)) < x \leq \mathit{rightBound}(I(j))$$

and thus $x(i) \in I(j)$ which is a contradiction. Our assumption is false and for all $i = 1,\ldots,m-1$ it holds that $I(i) \boxtimes I(i+1)$. Because for $i = 1,\ldots,m-1$ it also holds that

$$\mathit{leftBound}(I(i)) \leq x(I(i)) \leq \mathit{rightBound}(I(i)) < x(I(i+1)) \leq \mathit{leftBound}(I(i+1))$$

and especially $\mathit{rightBound}(I(i)) < \mathit{rightBound}(I(i+1))$, it then also follows $I(i) \boxtimes^{>} I(i+1)$ for all $i = 1,\ldots,m-1$. Altogether we get that $I(1),\ldots,I(m)$ is a (strong) ordered covering of $\mathbb{R}$. Together with proof for Lemma 3.1.1 we have therefore shown Theorem 3.1.2 to be true.

□

We have shown an equivalent condition under which a set of intervals covers the real numbers. In general, for a given set of (multivariate) constraints $c_1,\ldots,c_m$, we find the intervals for assignments for y in which one of $c_i,\ldots,c_m$ is not satisfied given an assignment s for x. If those intervals cover $\mathbb{R}$, shown by checking Theorem 3.1.2, we know that $x \mapsto s$ is not part of a solution for $c_1,\ldots,c_m$. In case of Example 3.1.1, which presents an univariate instance, this already proofs that it is unsatisfiable because these intervals do not depend on variable assignments for other variables than y. In general however, this is not the case and the bounds of those intervals and even their "structure" or "type" might change.

Example 3.1.2. *Consider the constraint* $c \equiv xy^2 + 3y > 0$. *If we fix the assignment* $x \mapsto s$ *we get for* $t \in \mathbb{R}$*:*

$$[\![c]\!]^{x\mapsto s, y\mapsto t} = \mathit{false} \Leftrightarrow \begin{cases} t \in [-\infty, -\frac{3}{s}-] \cup [0+, +\infty] & \text{for } s > 0 \\ t \in [0+, +\infty] & \text{for } s = 0 \\ t \in [0+, -\frac{3}{s}-] & \text{for } s < 0 \end{cases}$$

Note that the sets $T(c,s) := \{t \in \mathbb{R} : [\![c]\!]^{x\mapsto s, y\mapsto t} = \mathit{false}\}$ can take on (only) three different "types", depending on sample s, namely $[-\infty, -\frac{3}{s}-] \cup [0+, +\infty]$, $[0+, +\infty]$ or $[0+, -\frac{3}{s}-]$. This leads us to the following observation:
For a conflict in $c_1, \ldots, c_m$ and sample s, we know that

$$\begin{aligned} &\neg\exists t \in \mathbb{R}. \left[\!\!\left[\bigwedge_{i=1}^{m} c_i \right]\!\!\right]^{x\mapsto s, y\mapsto t} \\ \Leftrightarrow \mathbb{R} &= \left\{ t \in \mathbb{R} \ : \ \left[\!\!\left[\bigwedge_{i=1}^{m} c_i \right]\!\!\right]^{x\mapsto s, y\mapsto t} = \mathit{false} \right\} \\ &= \bigcup_{i=1}^{m} \left\{ t \in \mathbb{R} \ : \ [\![c_i]\!]^{x\mapsto s, y\mapsto t} = \mathit{false} \right\} \\ &= \bigcup_{i=1}^{m} T(c_i,s) \end{aligned}$$

We also observe that each set $T(c_i,s)$ can always be expressed as a union of three or less intervals. (For details see Section 3.2.) When certain side conditions are met by (other) samples s', the formulas that describe $T(c_i,s')$ or $T(c_i,s)$ do not change (see example above). We therefore want to encode samples s' for which the sets $T(c_i,s')$ covers $\mathbb{R}$ (based on the definition of an ordered covering). We also want to encode said side conditions in a formula. This formulas is a conflict generalization.
In the following section, we first formalize the notion of formulas for the sets $T(c_i,s')$ and the "type" of these sets.

3.2 Type of a constraint and a sample

Consider a normalized constraint $c \equiv p \sim 0 \equiv ay^2 + by + c \sim 0 \in CS[x_1, \ldots, x_n, y]$, $\sim \in \mathit{REL}$, $a,b,c \in \mathit{POL}[x_1, \ldots, x_n]$ where we again fix a variable assignment $x \mapsto s, s \in \mathbb{R}^n$. Let again

$$T(c,s) = \{t \in \mathbb{R} : [\![c]\!]^{x\mapsto s, y\mapsto t} = \mathit{false}\} \subseteq \mathbb{R}$$

be the set of all assignments for y for which c is not satisfied (given the fixed assignment $x \mapsto s$). Note that

$$t \in T(c,s) \Leftrightarrow \neg[\![c]\!]^{x\mapsto s, y\mapsto t} \Leftrightarrow \mathit{sgn}\left([\![p]\!]^{x\mapsto s, y\mapsto t}\right) \in \mathcal{S}$$

for some fixed set $\mathcal{S} \subset \{-1,0,+1\}$ depending on the relation $\sim$. Because the function $f : \mathbb{R} \to \mathbb{R}$, $t \mapsto [\![p]\!]^{x\mapsto s, y\mapsto t}$ is a polynomial and thus continuos, it holds that f is sign-invariant in (some) intervals whose bounds are $-\infty$, $+\infty$ or a zero $t_0 \in \mathbb{R}, f(t_0) = 0$ of f. It holds that $T(c,s)$ is a union of intervals whose bounds are $-\infty$, $+\infty$ or a zero $t_0 \in \mathbb{R}, [\![p]\!]^{x\mapsto s, y\mapsto t_0} = 0$ of p. As we have discussed in Section 3.1, we want to describe the set $T(c,s)$ using a formula that depends on the sample s. Zeros of a quadratic polynomials can naturally be described by using square root expressions we have introduced in Section 2.5. To define intervals whose bounds are parameterized and evaluated by an assignment $x \mapsto s$, we thus use elements from *SqrtEx* as defined in Definition 2.5.1 as bounds of these intervals. We defined elements from *SqrtEx*

as words in a formal language. We also define these intervals as such words. To differentiate "proper" intervals from those parametrized intervals defined below, we hereinafter call the latter *symbolic* intervals.

As we want to be able to describe open and closed intervals we define $SqrtEx_{-\epsilon}$, $SqrtEx_{+\epsilon}$, $SqrtEx_{\pm\epsilon}$ similarly to $\mathbb{R}_{-\epsilon}$, $\mathbb{R}_{+\epsilon}$, $\mathbb{R}_{\pm\epsilon}$:

$$SqrtEx_{+\epsilon} = \{s{+} : s \in SqrtEx\} \cup \{-\infty\}$$

$$SqrtEx_{-\epsilon} = \{s{-} : s \in SqrtEx\} \cup \{+\infty\}$$

$$SqrtEx_{\pm\epsilon} = SqrtEx_{+\epsilon} \cup SqrtEx_{-\epsilon}$$

and with that:

Definition 3.2.1 (Symbolic Intervals)**.** *Let $\mathbb{I}_{sym}$ denote the set of all* symbolic intervals*:*

$$\mathbb{I}_{sym} = \{[a,b] : a \in SqrtEx_{+\epsilon}, b \in SqrtEx_{-\epsilon}\} \cup \{\emptyset\}$$

We may notate $\mathbb{R} \in \mathbb{I}_{sym}$ instead of $[-\infty, +\infty] \in \mathbb{I}_{sym}$.

We also define

$$\overline{\mathbb{I}_{sym}} = \bigcup_{n=1}^{\infty} \{I_1 \cup \ldots \cup I_n : \ I_1, \ldots, I_n \in \mathbb{I}_{sym}\}$$

as the closure of symbolic intervals under the (symbolic) operator $\cup$. Note that $\mathbb{I}_{sym} \subset \overline{\mathbb{I}_{sym}}$.

We still need a way to evaluate square root expressions and symbolic interval bounds and symbolic intervals under a given assignment. For that we extend Definition 2.3.2 which defines the evaluation of polynomials and formulas:

Definition 3.2.2 (Evaluation of Square Root Expressions and Symbolic Intervals)**.** *We call $[\![\xi]\!]^\alpha$ the evaluation of $\xi \in (FO \cup POL \cup SqrtEx_{\pm\epsilon} \cup \overline{\mathbb{I}_{sym}})$ under assignment $\alpha \in ASS$ where*

$$[\![\cdot]\!]^{\cdot} : (FO \cup POL \cup SqrtEx_{\pm\epsilon} \cup \overline{\mathbb{I}_{sym}}) \times ASS \rightharpoonup (\mathbb{B} \cup \mathbb{R}_{\pm\epsilon} \cup \bar{\mathbb{I}})$$

such that for every assignment α:

- *$[\![\xi]\!]^\alpha$ is defined as in Definition 2.3.2 for $\xi \in (FO \cup POL)$*
- *$[\![-\infty]\!]^\alpha = -\infty$*
- *$[\![+\infty]\!]^\alpha = +\infty$*
- *$\left[\!\left[\frac{p+q\sqrt{r}}{s}\right]\!\right]^\alpha = \frac{[\![p]\!]^\alpha + [\![q]\!]^\alpha \sqrt{[\![r]\!]^\alpha}}{[\![s]\!]^\alpha} \in \mathbb{R} \cup \{\bot\}$,*
  where $\left[\!\left[\frac{p+q\sqrt{r}}{s}\right]\!\right]^\alpha = \bot$ when $[\![s]\!]^\alpha = 0$ or $[\![r]\!]^\alpha < 0$.
- *$[\![a]\!]^\alpha = [\![s]\!]^\alpha \psi$ for $a = s\psi \in SqrtEx_{\pm\epsilon}, a \in SqrtEx, \psi \in \{-,\emptyset,+\}$,*
  where $[\![a]\!]^\alpha = \bot$ when $[\![s]\!]^\alpha = \bot$

- $[\![[a,b]]\!]^\alpha = [[\![a]\!]^\alpha,[\![b]\!]^\alpha]$ *for* $[a,b] \in \mathbb{I}_{sym} \setminus \{\emptyset\}$,
  where $[\![[a,b]]\!]^\alpha = \bot$ *when* $[\![a]\!]^\alpha = \bot$ *or* $[\![b]\!]^\alpha = \bot$
- $[\![\emptyset]\!]^\alpha = \emptyset$
- $[\![I_1 \cup \ldots \cup I_n]\!]^\alpha = [\![I_1]\!]^\alpha \cup \ldots \cup [\![I_n]\!]^\alpha$ *for* $(I_1 \cup \ldots \cup I_n) \in \overline{\mathbb{I}_{sym}}$,
  where $[\![I_1 \cup \ldots \cup I_n]\!]^\alpha = \bot$ *when* $[\![I_i]\!]^\alpha = \bot$ *for a* $i = 1,\ldots,n$

Similarly to non-symbolic intervals we define:

$$
\begin{aligned}
\mathit{leftBound}: &\quad \mathbb{I}_{sym} \rightharpoonup \mathit{SqrtEx}_{+\epsilon}, && \emptyset \mapsto \bot, [a,b] \mapsto a \\
\mathit{rightBound}: &\quad \mathbb{I}_{sym} \rightharpoonup \mathit{SqrtEx}_{-\epsilon}, && \emptyset \mapsto \bot, [a,b] \mapsto b \\
\pi_{\mathit{SqrtEx}}: &\quad \mathit{SqrtEx}_{\pm\epsilon} \rightharpoonup \mathit{SqrtEx}, && x \mapsto \begin{cases} \bot \text{ for } x \in \{-\infty, +\infty\} \\ e \text{ if } x \in \{e-, e, e+\} \text{ for some } e \in \mathit{SqrtEx} \end{cases}
\end{aligned}
$$

With that we can formalize the sets denoted by T from above as unions of symbolic intervals.

Definition 3.2.3 (Type of a Constraint)**.** *Let* $c \equiv ay^2+by+c \sim 0 \in CS[x_1,\ldots,x_n,y]$ *be a normalized constraint. Let* $z = -\frac{c}{b}$, $z'_1 = \frac{-b-\sqrt{b^2-4ac}}{2a}$ *and* $z'_2 = \frac{-b+\sqrt{b^2-4ac}}{2a}$. *We then define:*

- $T_\emptyset(c) = \emptyset$
- $T_\mathbb{R}(c) = [-\infty, +\infty]$
- $T_{(1,1)}(c) = [z,z]$
- $T_{(1,2)}(c) = [-\infty,z-] \cup [z+, +\infty]$
- $T_{(1,3)}(c) = [-\infty,z-]$
- $T_{(1,4)}(c) = [-\infty,z]$
- $T_{(1,5)}(c) = [z+, +\infty]$
- $T_{(1,6)}(c) = [z, +\infty]$
- $T_{(2,1)}(c) = [z'_1,z'_1] \cup [z'_2,z'_2]$
- $T_{(2,2)}(c) = [-\infty,z'_1-] \cup [z'_1+,z'_2-] \cup [z'_2+, +\infty]$
- $T_{(2,3)}(c) = [-\infty,z'_1-] \cup [z'_2+, +\infty]$
- $T_{(2,4)}(c) = [-\infty,z'_1] \cup [z'_2, +\infty]$
- $T_{(2,5)}(c) = [z'_1+,z'_2-]$
- $T_{(2,6)}(c) = [z'_1,z'_2]$

We say that a pair (c,s), where c is a constraint and s a sample, has type T_X *if* $\{t \in \mathbb{R} : [\![c]\!]^{x \mapsto s, y \mapsto t} = false\} = [\![T_X(c)]\!]^{x \mapsto s}$. *Note that the above indices may indicate in some cases whether the polynomial is linear or quadratic but otherwise have no inherit function or meaning other than to differentiate between different types.*

Lemma 3.2.1. *Every constraint/sample-pair* (c,s) *has at a* type *as defined in Definition 3.2.3.*

The proof of Lemma 3.2.1 is a direct result of Lemma 3.2.2 (see below).

Given a conflict in constraints $c_1, \ldots, c_m$ and sample s, we want to assign each pair (c_i,s) a type $T(c_i,s) \in \{T_\emptyset, \ldots, T_{(2,6)}\}$. This type shall be a description of assignments $y \mapsto t$ for which $[\![c_i]\!]^{x \mapsto s, y \mapsto t} = false$. This collection of types is then used to construct a conflict generalization. As mentioned previously we also want to find a side condition $sc(c_i,s)$ for which this description $T(c_i,s)$ does not change, i.e.

$$(\forall s' \in \Theta(sc(c_i,s)).(c_i,s') \text{ has type } T(c_i,s)) \text{ and } s \in \Theta(sc(c_i,s))$$

Table 3.1: Assignment of a type and a side condition

$\sim$	$T_{\emptyset}$	$T_{\mathbb{R}}$	$T_{(1,1)}$	$T_{(1,2)}$	$T_{(1,3)}$	$T_{(1,4)}$	$T_{(1,5)}$	$T_{(1,6)}$	$T_{(2,1)}$	$T_{(2,2)}$	$T_{(2,3)}$	$T_{(2,4)}$	$T_{(2,5)}$	$T_{(2,6)}$
$=$	$a = 0 \wedge b = 0 \wedge c = 0$	$(a = 0 \wedge b = 0 \wedge c \neq 0)$ $\vee \quad (a \neq 0 \wedge D < 0)$		$a = 0 \wedge b \neq 0$						$a \neq 0 \wedge D \geq 0$				
$\neq$	$(a = 0 \wedge b = 0 \wedge c \neq 0)$ $\vee \quad (a \neq 0 \wedge D < 0)$	$a = 0 \wedge b = 0 \wedge c = 0$	$a = 0 \wedge b \neq 0$						$a \neq 0 \wedge D \geq 0$					
$<$	$(a = 0 \wedge b = 0 \wedge c < 0)$ $\vee \quad (a < 0 \wedge D < 0)$	$(a = 0 \wedge b = 0 \wedge c \geq 0)$ $\vee \quad (a > 0 \wedge D < 0)$				$a = 0 \wedge b < 0$		$a = 0 \wedge b > 0$				$a > 0 \wedge D \geq 0$		$a < 0 \wedge D \geq 0$
$>$	$(a = 0 \wedge b = 0 \wedge c > 0)$ $\vee \quad (a > 0 \wedge D < 0)$	$(a = 0 \wedge b = 0 \wedge c \leq 0)$ $\vee \quad (a < 0 \wedge D < 0)$				$a = 0 \wedge b > 0$		$a = 0 \wedge b < 0$				$a < 0 \wedge D \geq 0$		$a > 0 \wedge D \geq 0$
$\leq$	$(a = 0 \wedge b = 0 \wedge c \leq 0)$ $\vee \quad (a < 0 \wedge D < 0)$	$(a = 0 \wedge b = 0 \wedge c > 0)$ $\vee \quad (a > 0 \wedge D < 0)$			$a = 0 \wedge b < 0$		$a = 0 \wedge b > 0$				$a > 0 \wedge D \geq 0$		$a < 0 \wedge D \geq 0$	
$\geq$	$(a = 0 \wedge b = 0 \wedge c \geq 0)$ $\vee \quad (a > 0 \wedge D < 0)$	$(a = 0 \wedge b = 0 \wedge c < 0)$ $\vee \quad (a < 0 \wedge D < 0)$			$a = 0 \wedge b > 0$		$a = 0 \wedge b < 0$				$a < 0 \wedge D \geq 0$		$a > 0 \wedge D \geq 0$	

$ay^2 + by + c \sim 0$, $\sim \in REL$, a,b,$c \in POL$, $D = b^2 - 4ac$

Table 3.1 shows for a normalized constraint $c \equiv p \sim 0 \equiv ay^2 + by + c \sim 0 \in CS[x_1, \ldots, x_n, y]$ and a sample $s \in \mathbb{R}^n$ an assignment of a type and a side condition to (c,s) which fulfills the above stated requirement. Assign the unique type t and side condition sc from the row corresponding to the relation symbol $\sim \in REL$ to (c,s) such that $[\![sc]\!]^{x \mapsto s} = true$. Then (c,s) has type t. Here D is defined as the discriminant $b^2 - 4ac$.

Lemma 3.2.2. *Let $c \equiv p \sim 0 \equiv ay^2 + by + c \sim 0$, $a,b,c \in Pol[x_1, \ldots, x_n]$ be a (normalized) constraint and $s \in \mathbb{R}^n$ a sample.*

(i) Exactly one side condition sc in the row of Table 3.1 corresponding to $\sim$ is satisfied by assignment $x \mapsto s$.

(ii) For every $(x \mapsto s') \in \Theta(sc)$ (especially $s = s'$) it holds, that (c,s') has the type T_X corresponding to the column of $\sim$ and sc

For any $p \in \{a,b,c,D\}$ (see Table 3.1) obviously exactly one of $[\![p \sim 0]\!]^{x \mapsto s}$ for $\sim \in \{<\ ,\ =\ ,\ >\}$ (or $\sim \in \{\leq\ ,\ >\}$ etc.) holds. Thus, (i) holds true. Informally speaking, because (univariate) polynomials are continuous, their graph must cross the x-Axis for them to change their sign. Thus, in studying the zeros of univariate polynomials of degree at most two by applying simple algebraic rearrangements and formulas for solving linear and quadratic equations as they are taught in highschool, one can derive the entries of Table 3.1 easally (albeit somewhat tedious). We omit a formal proof of lemma 3.2.2 here.

Note that in some cases the constraint takes on the same type for very different "reasons". For example if $\sim$ is $\neq$ it takes on type $T_\emptyset$ if either p is constant and non-zero in y or if p is quadratic and has no zeros ($D < 0$). As discussed in the introduction of this chapter we ultimately want to construct a conflict generalization that generalizes to different samples for which a "similar" conflict exists. As mentioned above, the restriction to descriptions of "similar" conflicts is useful for finding a balance between the size of the set of the described samples and the algorithmic complexity of the construction of the generalization. Moreover, by reducing the complexity of these side conditions we also reduce the complexity of conflict generalizations which depend on them. Any algorithm analyzing them thus gains efficiency doing so. Therefore, while we could use the whole formula in the cell corresponding to $\neq$ and $T_\emptyset$, we choose only one of the disjuncts, namely the one that is satisfied by the given pair (c,s). Therefore we get for $c \equiv p \sim 0 \equiv ay^2 + by + c \neq 0$ and a sample $s \in \mathbb{R}^n$ the following assignment to a type t and a side condition:

$$(T(p \neq 0,s), sc(p \neq 0,s)) := \begin{cases} (T_\emptyset, a = 0 \wedge b = 0 \wedge c \neq 0) & \text{if } [\![a = 0 \wedge b = 0 \wedge c \neq 0]\!]^{x \mapsto s} = true \\ (T_\emptyset, a \neq 0 \wedge D < 0) & \text{if } [\![a \neq 0 \wedge D < 0]\!]^{x \mapsto s} = true \\ (T_{\mathbb{R}}, a = 0 \wedge b = 0 \wedge c = 0) & \text{if } [\![a = 0 \wedge b = 0 \wedge c = 0]\!]^{x \mapsto s} = true \\ (T_{(1,1)}, a = 0 \wedge b \neq 0) & \text{if } [\![a = 0 \wedge b \neq 0]\!]^{x \mapsto s} = true \\ (T_{(2,1)}, a \neq 0 \wedge D \geq 0) & \text{if } [\![a \neq 0 \wedge D \geq 0]\!]^{x \mapsto s} = true \end{cases}$$

where $D = b^2 - 4ac \in POL[x_1, \ldots, x_n]$. We define $T(c,s)$ and $sc(c,s)$ analogously for all other relation symbols $\sim$. $T(c,s)$ and $sc(c,s)$ are always well-defined and

$$(\forall s' \in \Theta(sc(c_i,s)).(c_i,s') \text{ has type } T(c_i,s)) \text{ and } s \in \Theta(sc(c_i,s))$$

Convention 3. *For a constraint c and sample s, $T(c,s) \in \{T_\emptyset, \ldots, T_{(2,6)}\}$ defines a type of (c,s). Therefore $(T(c,s))(c)$ defines the (symbolic) union of the corresponding symbolic intervals as they are "generated" by the type and the constraint. To increase readability we hereinafter often shorten $(T(c,s))(c)$ to simply $T(c,s)$ if it is obvious from the context if either the type or the symbolic intervals are meant to be represented.*

3.3 Conflict generalization

In Section 3.1 we defined relations $\lessdot$ (and $\lessdot^>$) to define (strong) ordered coverings of $\mathbb{R}$ by intervals. In this section we define for symbolic intervals I and J the formula $I \lessdot_{\text{sym}} J$ with which we construct a formula whose solution encodes a ordered covering by the evaluated symbolic intervals used to construct it.

Definition 3.3.1 (Gap-less Symbolic Intervals)**.** *For two symbolic intervals I and J we define the formula $I \lessdot_{sym} J$ such that:*
If $I = \emptyset$ or $J = \emptyset$ or $rightBound(I) = +\infty$ or $leftBound(J) = -\infty$, then:

$$I \lessdot_{sym} J :\equiv false$$

If not first case and both $rightBound(I)$ and $leftBound(J)$ are open:

$$I \lessdot_{sym} J :\equiv (r > l)[\pi_{SqrtEx}(rightBound(I)) /\!\!/ r][\pi_{SqrtEx}(leftBound(J)) /\!\!/ l]$$

If not first case and one or both of $rightBound(I)$ and $leftBound(J)$ are closed:

$$I \lessdot_{sym} J :\equiv (r \geq l)[\pi_{SqrtEx}(rightBound(I)) /\!\!/ r][\pi_{SqrtEx}(leftBound(J)) /\!\!/ l]$$

Lemma 3.3.1. *Let $I_1, \ldots, I_m$ be symbolic intervals such that $leftBound(I_1) = -\infty$ and $rightBound(I_m) = +\infty$. It then holds for any full assignment α of $I_1, \ldots, I_m$ that*

$$\left[\!\!\left[\bigwedge_{i=1}^{m-1} (I_i \lessdot_{sym} I_{i+1}) \right]\!\!\right]^\alpha \Leftrightarrow [\![I_1]\!]^\alpha, \ldots, [\![I_m]\!]^\alpha \text{ is an ordered covering}$$

For the proof of this lemma we use the fact that for square root expressions s and t and an assignment α for s and t it holds that

$$[\![(r \geq l)[s /\!\!/ r][t /\!\!/ l]]\!]^\alpha \Leftrightarrow [\![s]\!]^\alpha \geq [\![t]\!]^\alpha \text{ and}$$
$$[\![(r > l)[s /\!\!/ r][t /\!\!/ l]]\!]^\alpha \Leftrightarrow [\![s]\!]^\alpha > [\![t]\!]^\alpha$$

We omit a formal proof of this fact here, however this result can be shown by applying substitution rules found in [Cor10], some algebraic rearrangements, simple transformations by equivalences and the fact that the Virtual Substitution, although not formally but effectively, allows us to change the order of (i) evaluation of square root expressions and (ii) combination of those with relations.
From this equivalence it also follows for symbolic intervals I and J and an assignment α for I and J that

$$[\![I \lessdot_{\text{sym}} J]\!]^\alpha \Leftrightarrow [\![I]\!]^\alpha \lessdot [\![J]\!]^\alpha$$

Utilizing this fact, we proof Lemma 3.3.1:

Proof. Let $I_1,\ldots,I_m$ be symbolic intervals such that $\mathit{leftBound}(I_1) = -\infty$ and $\mathit{rightBound}(I_m) = +\infty$ and an assignment α such that $\left[\!\left[\bigwedge_{i=1}^{m-1}(I_i \mathbin{\mathfrak{O}_{\text{sym}}} I_{i+1})\right]\!\right]^{\alpha}$.
It then follows:

$$\left[\!\left[\bigwedge_{i=1}^{m-1}(I_i \mathbin{\mathfrak{O}_{\text{sym}}} I_{i+1})\right]\!\right]^{\alpha} \Leftrightarrow \bigwedge_{i=1}^{m-1}[\![I_i \mathbin{\mathfrak{O}_{\text{sym}}} I_{i+1}]\!]^{\alpha} \Leftrightarrow \bigwedge_{i=1}^{m-1}[\![I_i]\!]^{\alpha} \mathbin{\mathfrak{O}} [\![I_{i+1}]\!]^{\alpha}$$

Because $\mathit{leftBound}([\![I_1]\!]^{\alpha}) = [\![\mathit{leftBound}(I_1)]\!]^{\alpha} = [\![-\infty]\!]^{\alpha} = -\infty$ and $\mathit{rightBound}([\![I_m]\!]^{\alpha}) = [\![\mathit{rightBound}(I_m)]\!]^{\alpha} = [\![+\infty]\!]^{\alpha} = +\infty$, it then follows that $[\![I_1]\!]^{\alpha},\ldots,[\![I_m]\!]^{\alpha}$ is an ordered covering iff $\left[\!\left[\bigwedge_{i=1}^{m-1}(I_i \mathbin{\mathfrak{O}_{\text{sym}}} I_{i+1})\right]\!\right]^{\alpha}$ (see Definition 3.1.2).

□

With the theoretical foundations laid out, we can now construct for a conflict in constraints $c_1,\ldots,c_m$ and a sample s a conflict generalization φ as follows:
Let $I_{(i,1)},\ldots,I_{(i,k_i)} \in \mathbb{I}_{sym}, k_i \leq 3$ so that $T(c_i,s) = I_{(i,1)} \cup \ldots \cup I_{(i,k_i)}$ for $i = 1,\ldots,m$. First we show that $\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}[\![I_{(i,j)}]\!]^{x\mapsto s} = \mathbb{R}$. By definition of a conflict, it holds that $\neg\exists t \in \mathbb{R}.\, [\![\bigwedge_{i=1}^{m} c_i]\!]^{x\mapsto s}$. Let $t \in \mathbb{R}$ be arbitrary. Then there is a $i = 1,\ldots,m$ such that $[\![c_i]\!]^{x\mapsto s,y\mapsto t} = \mathit{false}$. Thus, $t \in [\![T(c_i,s)]\!]^{x\mapsto s} = \left[\!\left[\bigcup_{j=1}^{k_i} I_{(i,j)}\right]\!\right]^{x\mapsto s}$ and especially $t \in \bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}[\![I_{(i,j)}]\!]^{x\mapsto s}$ (see Section 3.2). Because choice of $t \in \mathbb{R}$ was arbitrary, we get $\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}[\![I_{(i,j)}]\!]^{x\mapsto s} = \mathbb{R}$. From Theorem 3.1.2 we then know, that there exist an injective function $\pi : \{1,\ldots,o\} \mapsto \{(1,1),\ldots,(1,k_1),(2,1),\ldots,(2,k_2),\ldots,(m,k_m)\}$ such that $[\![I_{\pi(1)}]\!]^{x\mapsto s},\ldots,[\![I_{\pi(o)}]\!]^{x\mapsto s}$ is an *strong* ordered covering.
Then we construct a conflict generalization as follows:

$$\varphi \equiv \left(\bigwedge_{\substack{i=1 \\ \exists t.\exists r.(i,r)=\pi(t)}}^{m} sc(c_i,s)\right) \wedge \left(\bigwedge_{i=1}^{o-1} I_{\pi(i)} \mathbin{\mathfrak{O}_{\text{sym}}} I_{\pi(i+1)}\right)$$

The formula φ only includes those side conditions associated with one or more of its included symbolic intervals via their respective constraint.

Theorem 3.3.2. *φ is a conflict generalization, i.e. $(x \mapsto s) \in \Theta(\varphi)$ and*

$$\forall \alpha \in \Theta(\varphi).\neg\exists t \in \mathbb{R}.\left[\!\left[\bigwedge_{i=1}^{m} c_i\right]\!\right]^{\alpha[t/y]}$$

Proof.

First step: show $(x \mapsto s) \in \Theta(\varphi)$

Because $[\![I_{\pi(1)}]\!]^{x\mapsto s},\ldots,[\![I_{\pi(o)}]\!]^{x\mapsto s}$ is a (strong) ordered covering, it holds according to Lemma 3.3.1, that $\left[\!\left[\bigwedge_{i=1}^{o-1} I_{\pi(i)} \mathbin{\mathfrak{O}_{\text{sym}}} I_{\pi(i+1)}\right]\!\right]^{\alpha}$. Also note that $\mathit{leftBound}(I_{\pi(1)}) = -\infty$ and $\mathit{leftBound}(I_{\pi(o)}) = +\infty$. It also holds $(x \mapsto s) \in \Theta(sc(c_i,s))$ for every $i = 1,\ldots,m$ and especially for every $i = 1,\ldots,m$ such that $\exists t.\exists r.(i,r) = \pi(t)$ (see Lemma 3.2.2 and Section 3.2).

Thus:

$$(x \mapsto s) \in \Theta\left(\left(\bigwedge_{\substack{i=1 \\ \exists t.\exists r.(i,r)=\pi(t)}}^{m} sc(c_i,s)\right) \wedge \left(\bigwedge_{i=1}^{o-1} I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}\right)\right) = \Theta(\varphi)$$

Second step: show $\forall \alpha \in \Theta(\varphi).\neg\exists t \in \mathbb{R}.[\![\bigwedge_{i=1}^{m} c_i]\!]^{\alpha[t/y]}$

Let $\alpha \in \Theta(\varphi)$ be arbitrary. Because $\mathit{Vars}(\varphi) \subseteq \{x_1,\ldots,x_n\}$ we may assume without loss of generality, that $\alpha = (x \mapsto s')$ for some $s' \in \mathbb{R}^n$. Because $\alpha \in \Theta(\varphi)$ holds, it especially holds that $\alpha = (x \mapsto s') \in \Theta(sc(c_i,s))$ for all $i = 1,\ldots,m$, $\exists t.\exists r.(i,r) = \pi(t)$. Lemma 3.2.2 then states that (c_i,s') has type $T(c_i,s)$. Also, because $\alpha \in \Theta(\varphi)$, it especially holds

$$\begin{aligned} &\left[\!\!\left[\bigwedge_{i=1}^{o-1} I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}\right]\!\!\right]^{\alpha} \\ \Rightarrow\ &[\![I_{\pi(i)}]\!]^{\alpha} \mathbin{\text{⧖}} [\![I_{\pi(i+1)}]\!]^{\alpha} \end{aligned}$$

for $i = 1,\ldots,o-1$.
As we have mentioned above, it holds that $\mathit{leftBound}(I_{\pi(1)}) = -\infty$ and $\mathit{rightBound}(I_{\pi(o)}) = +\infty$. Therefore, according to Theorem 3.1.2, $[\![I_{\pi(1)}]\!]^{\alpha},\ldots,[\![I_{\pi(o)}]\!]^{\alpha}$ is an ordered covering and $\bigcup_{i=1}^{o} [\![I_{\pi(i)}]\!]^{\alpha} = [\![\bigcup_{i=1}^{o} I_{\pi(i)}]\!]^{\alpha} = \mathbb{R}$. Then especially $[\![\bigcup_{i=1}^{m} T(c_i,s)]\!]^{\alpha} = \mathbb{R}$. Because we have shown above that (c_i,s') has type $T(c_i,s)$ for $i = 1,\ldots,m$, $\exists t.\exists r.(i,r) = \pi(t)$, i.e.

$$\begin{aligned} &\{t \in \mathbb{R} : [\![c_i]\!]^{x\mapsto s', y\mapsto t} = \mathit{false}\} \\ =&\{t \in \mathbb{R} : [\![c_i]\!]^{\alpha[t/y]} = \mathit{false}\} \end{aligned}$$

and because $[\![\bigcup_{i=1}^{m} T(c_i,s)]\!]^{\alpha} = \mathbb{R}$, it then follows that $\neg\exists t \in \mathbb{R}.[\![\bigwedge_{i=1}^{m} c_i]\!]^{\alpha[t/y]}$.
Because the choice of $\alpha \in \Theta(\varphi)$ was arbitrary, we get $\forall \alpha \in \Theta(\varphi).\neg\exists t \in \mathbb{R}.[\![\bigwedge_{i=1}^{m} c_i]\!]^{\alpha[t/y]}$ thus proving Theorem 3.3.2.

□

Because some of the terms $sc(c_i,s)$ and $I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}$ of φ may be constant and because $\Theta(\varphi) \neq \emptyset$, we can refine the construction of the proposed conflict generalization as follows:

$$\varphi \equiv \left(\bigwedge_{\substack{i=1 \\ \exists t.\exists r.(i,r)=\pi(t) \\ \mathit{Vars}(sc(c_i,s))\neq\emptyset}}^{m} sc(c_i,s)\right) \wedge \left(\bigwedge_{\substack{i=1 \\ \mathit{Vars}\left(I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}\right)\neq\emptyset}}^{o-1} I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}\right)$$

Also note that the terms $I_{\pi(i)} \mathbin{\text{⧖}_{\text{sym}}} I_{\pi(i+1)}$ are comprised of multiple constraints (see substitution rules in [Cor10]) and thus we may refine φ even further by eliminating constant constraints.

Theorem 3.3.2 only ensures that there exist a conflict generalization of the form from above. In the next sections we discuss an heuristic for the complexity of (those) formulas and how to find one conflict generalization with optimal, or at least "close" to optimal, complexity as given by our heuristic.

Convention 4. *In the following sections and chapters, if not otherwise mentioned, all referrals to conflict generalizations refer to conflict generalizations of the (refined) form as given by the construction above.*

3.4 Heuristics on complexity

In the previous section we have shown a construction of a conflict generalization. This construction depends on a selection of some constraints and some symbolic intervals corresponding to those constraints and is therefore not (necessarily) a unique conflict of this form. Because this conflict generalization is further analyzed, there is an incentive to choose a conflict generalization which minimizes the time spend on analyzing it.

The complexity of the framework, in which our conflict generalization algorithm operate in, necessitates such a selection to be based on a model for the complexity or the *cost* of a conflict generalization as it would be practically infeasible, not to mention computationally expensive by itself, to generate a conflict generalization which is guaranteed to minimize this computing time. Therefore we have to make some assumptions and simplifications, which may be to some degree inaccurate or at least approximative, in order to derive a sensible model:

Let F be the closure of constraints under $\wedge$ and $\vee$. Note that a conflict generalization lies in F. We will only define costs for formulas in F and not for all formulas in *FO*.

A model or a heuristic for the complexity of formulas in F is then a function *cost* that maps from formulas in F to some set of possible costs C on which a total order is defined as to ascertain a conflict generalization φ^* such that

$$cost(\varphi^*) = \min_{\text{confl.gen.}\varphi} cost(\varphi)$$

Because there is only a finite number of conflict generalizations, $\min_{\text{confl.gen.}\varphi} cost(\varphi)$ is then well defined.

In computer science, the "costs" of mathematical objects are traditionally represented or modelled as numbers, more specifically often as (non-negative) integers. This has some advantages:

- Computers and programming languages are optimized for the representation of and calculations with numbers.
- It represents in many cases the most obvious and most easily understandable model of costs.
- For most problems, there is a sensible way to combine sub-solutions or candidates for parts of a solution such that the assigned cost of those parts may be added to get the costs of the combined solution. The order axiom $a \leq b \implies a + c \leq b + c$ holds, which enables said partition.

We show below that the assumptions we make for our heuristic are not compatible with a cost model based on integers or, even more generally, numbers in $\mathbb{R}$. However, because preserving the last of the aforementioned advantages is especially reasonable in order to partition the resulting problem, for now we "relax" the condition that *cost* maps to $C = \mathbb{R}$ and only require the set C of costs to be a totally ordered commutative monoid (see "tomonoid" in [Whi99]) of which $\mathbb{R}$ is a special case.

Definition 3.4.1 (Totally Ordered Commutative Monoid)**.** *A totally ordered commutative monoid is a set M together with a binary operation $+$ on M and a binary relation $\leq$ on M such that:*

- *(*associativity*)* $(a+b)+c = a+(b+c)$ *for all* $a,b,c \in M$
- *(*identity element*) There is an element* $0 \in M$ *such that* $0+a = a+0 = a$ *for all* $a \in M$*.*
- *(*commutativity*)* $a+b = b+a$ *for all* $a,b \in M$
- *(*reflexivity*)* $a \leq a$ *for all* $a \in M$
- *(*transitivity*)* $a \leq b \wedge b \leq c \implies a \leq c$ *for all* $a,b,c \in M$
- *(*anti-symmetry*)* $a \leq b \wedge b \leq a \implies a = b$ *for all* $a,b \in M$
- *(*strong connectivity*)* $a \leq b \vee b \leq a$ *for all* $a,b \in M$
- *(*order axiom/translational invariance*)* $a \leq b \implies a+c \leq b+c$ *for all* $a,b \in M$

Note that the corresponding strict order $<$

$$a < b :\Leftrightarrow a \leq b \wedge a \neq b$$

is implicitly defined.

We also define for a monoid element m and $n \in \mathbb{N}$:

$$\begin{aligned} 0 \cdot m &= 0 \\ n \cdot m &= \underbrace{m + \ldots + m}_{n \text{ times}} \end{aligned}$$

3.4.1 Assumptions on the heuristic

It is self-evident that a formula $f \in F$ should not be associated with negative costs. Moreover, it should be assigned no costs, i.e. $cost(f) = 0$, when the truth value of f is constant for any given assignment and if this can be ascertained easily:

Assumption 1. *For* $f \in F$ *it holds* $cost(f) \geq 0$ *and* $cost(f) = 0$ *if* f *is a constraint with normalized form* $p \sim 0$ *where* p *is a constant polynomial.*

Since a constraint is eventually converted to its normal form and because the relation of a constraint does have an insignificant impact on its costs compared to the importance of its polynomial, we state this assumption:

Assumption 2. *For two constraints* c_1 *and* c_2 *which have the same polynomial in their normalized form, i.e.* $p \sim_i 0$ *is the normalized form of* c_i*,* $i = 1,2$ *for a (fixed) polynomial* p*, it holds* $cost(c_1) = cost(c_2)$*.*

Consider a conflict generalization φ. Note that φ is a conjunction of side conditions and comparisons of symbolic interval bounds both of which may be considered

"atomic" parts of a any and especially of the optimal conflict generalization of this form. Given the advantages of the use of numbers as the base of a cost model, as stated above, it is therefore reasonable to make the following assumption on our heuristic:

Assumption 3. *For* $f_1,f_2 \in F$ *it holds* $cost(f_1 \wedge f_2) = cost(f_1) + cost(f_2)$.

It is not obvious as to how $cost(f_1 \vee f_2)$ should be evaluated when already requiring Assumption 3. There are certain properties both $cost(f_1 \wedge f_2)$ and $cost(f_1 \vee f_2)$ should have that might inform a choice, them being:

- $cost(f_1 \oplus f_2) = cost(f_2 \oplus f_1)$ for all $f_1,f_2 \in F$, $\oplus \in \{\wedge,\vee\}$
- For $\oplus \in \{\wedge,\vee\}$ there should be a function $h_\oplus : C \times C \to C$ such that $cost(f_1 \oplus f_2) = h_\oplus(cost(f_1), cost(f_2))$ for all $f_1,f_2 \in F$

Also, informally, $cost(f_1 \vee f_2)$ should be computationally easy to compute, just as $cost(f_1 \wedge f_2)$ is computationally easy to compute. Note that we assume here the addition to be easily computable. Moreover, for any $f_1,f_2,g \in F$ it should hold that $cost(g)$ is *much greater* than $cost(f_1 \vee f_2)$ if and only if $cost(g)$ is *much greater* than $cost(f_1 \wedge f_2)$. Below we show what "much greater" means in this context. In conclusion it is therefore not unreasonable to simplify our model with this following assumption:

Assumption 4. *For* $f_1,f_2 \in F$ *it holds* $cost(f_1 \vee f_2) = cost(f_1) + cost(f_2)$.

As mentioned above, the most determining factor for the complexity of formulas are the (normalized) polynomials they contain, mainly their degree. This has such an impact on their complexity that we require for $f_1,f_2 \in F$ that if f_1 contains a polynomial of higher degree than *any* polynomial contained in f_2, that $cost(f_1) > cost(f_2)$ no matter what other properties f_1 and f_2 might have, e.g. number of constraints or Boolean structure. However, a degree of a (multivariate) polynomial depends on a choice of a variable (see Section 2.4). Moreover, the maximum degree(s) of a polynomial is not the only factor determining its cost ($cost(p) := cost(p \sim 0)$ for any relation $\sim$). Also the number of variables in each degree and also the order of elimination of variables (in the context of this chapter assumed to be $y,x_n,\ldots,x_1$) are relevant, if only to a lesser significance. We codify the above considerations in the notion of *ranks*:

Definition 3.4.2 (Rank)**.** *For a polynomial* p *and* $k \in \mathbb{N}_0$ *define*

$$\#deg(p,k) := |\{i : \deg(p,x_i) = k\}|$$

We say that a polynomial p *is of higher* rank *than a polynomial* q *(written* $p >^{rnk} q$*) if*

- $\exists k.\ \#deg(p,k) > \#deg(q,k) \wedge \forall \hat{k} > k.\ \#deg\left(p,\hat{k}\right) = \#deg\left(q,\hat{k}\right)$ *or*
- $\forall k.\ \#deg(p,k) = \#deg(q,k) \wedge$
  $(\exists j.\ \deg(p,x_j) > \deg(q,x_j) \wedge \forall \hat{j} > j.\ \deg(p,x_{\hat{j}}) = \deg(q,x_{\hat{j}}))$

Furthermore we say that a polynomial p *is of higher* rank-class *than a polynomial* q *(written* $p \gg^{rnk} q$*) if*

- $\exists k.\ \#deg\,(p,k) > \#deg\,(q,k) = 0 \wedge \forall \hat{k} > k.\ \#deg\left(p,\hat{k}\right) = \#deg\left(q,\hat{k}\right) = 0$ *or*
- $\forall k.\ \#deg\,(p,k) = \#deg\,(q,k)\ \wedge$
  $(\exists j.\ \deg(p,x_j) > 0 = \deg(q,x_j) \wedge \forall \hat{j} > j.\ \deg(p,x_{\hat{j}}) = \deg(q,x_{\hat{j}}) = 0)$

Note that $p \gg^{rnk} q$ *implies* $p >^{rnk} q$. *For a formula* f *define* $Poly(f)$ *as the set of all polynomials contained in* f, *i.e.:*

- $Poly(p \sim q) = \{r : r$ *is the normalized form of* $p - q\}$ *for polynomials* p,q *and relation* $\sim$
- $Poly(g_1 \oplus g_2) = Poly(g_1) \cup Poly(g_2)$ *for* g_1,$g_2 \in F$, $\oplus \in \{\wedge,\vee\}$

Then define $>^{rnk}$ *and* $\gg^{rnk}$ *for formulas* f_1,$f_2 \in F$ *as follows:*

$$f_1 >^{rnk} f_2 :\Leftrightarrow \exists p \in Poly(f_1).\forall q \in Poly(f_2).(p >^{rnk} q)$$
$$f_1 \gg^{rnk} f_2 :\Leftrightarrow \exists p \in Poly(f_1).\forall q \in Poly(f_2).(p \gg^{rnk} q)$$

We then require the following:

Assumption 5. *For polynomials* p_1, p_2 *with* $p_1 <^{rnk} p_2$ *it holds* $cost(p_1 \sim_1 0) < cost(p_2 \sim_2 0)$ *for every relation* $\sim_1$ *and* $\sim_2$.
For f_1,$f_2 \in F$ *with* $f_1 \ll^{rnk} f_2$ *it holds* $cost(f_1) < cost(f_2)$.

Lemma 3.4.1. *Let* f_1,$f_2 \in F$. *If* f_1 *contains a polynomial which has one variable whose degree is higher than the degree of every variable in every polynomial contained in* f_2, *then* $f_1 \gg^{rnk} f_2$:

$$\left(\exists p \in Poly(f_1).\exists i.\forall q \in Poly(f_2).\forall j.\deg(p,x_i) > \deg(q,x_j)\right) \Rightarrow f_1 \gg^{rnk} f_2$$

Then especially for polynomials p_1 *and* p_2 *with* $\max_i \deg(p_1,x_i) > \max_i \deg(p_2,x_i)$, *it follows* $p_1 \gg^{rnk} p_2$.

Proof. Let f_1,$f_2 \in F$, $p \in Poly(f_1)$ and i such that for all $q \in Poly(f_2)$ and j it holds that $k := \deg(p,x_i) > \deg(q,x_j)$. Then $\#\deg\,(p,k) \geq 1 > 0 = \max_{q \in Poly(f_2)} \#\deg\left(q,\hat{k}\right)$ for every $\hat{k} \geq k$. Thus, $p \gg^{\text{rnk}} q$ for every $q \in Poly(f_2)$. Therefore $f_1 \gg^{\text{rnk}} f_2$. □

Assumptions 1 to 5 all seem to be reasonable restrictions and simplifications to inform a choice of a cost model. As mentioned above, these assumptions are not compatible with $C \subseteq \mathbb{R}$:

Lemma 3.4.2. *Let* $cost : F \to C$ *be a cost-model.* C *is the set (the totally ordered commutative monoid) of cost-values. Then* $C \nsubseteq \mathbb{R}$.

Proof. Assume $C \subseteq \mathbb{R}$. Let $f_1 \equiv (x_1 > 0)$, $f_2 \equiv (x_1^2 > 0)$. By Assumption 1 it holds $m_i := cost(f_i) > 0$ for $i = 1,2$. By Assumption 3 it holds for $n \in \mathbb{N}$:

$$cost\left(\bigwedge_{i=1}^{n} f_1\right) = \sum_{i=1}^{n} cost(f_1) = \sum_{i=1}^{n} m_1 = nm_1$$

By Assumption 5 and Lemma 3.4.1 it then holds for *all* $n \in \mathbb{N}$:

$$nm_1 = cost\left(\bigwedge_{i=1}^{n} f_1\right) < cost(f_2) = m_2$$

But because m_1,$m_2 > 0$ this is a contradiction. Thus, our assumption stated above is false and $C \nsubseteq \mathbb{R}$. □

Informally speaking, this proof works analogously to rule out any commutative monoid C which doesn't allow for some sort of "infinities" in respect to its ordering, i.e. it rules out every totally ordered monoid C for which no two elements m_1,$m_2 > 0$ exist such that $\sum_{i=1}^{n} m_1 < m_2$ for every $n \in \mathbb{N}$. To be more precise: For every cost model which satisfies Assumption 1 to 5, the underlying totally ordered commutative monoid must satisfy the following definition:

Definition 3.4.3 (Infinity-Class-System)**.** *Let $(M, + , \leq)$ be a totally ordered commutative monoid for which we define $[\cdot] : M \to \mathcal{P}(M)$ such that for $m \in M$ it holds that $[m]$ is the smallest set (intersection of all sets) T such that:*

- $m \in T$
- $\forall a \in M.\, \forall b \in T.\, \left(\wedge \begin{array}{l} a \leq b \wedge (\exists n \in \mathbb{N}.\, n \cdot a \geq b \vee a \geq n \cdot b) \Rightarrow a \in T \\ a \geq b \wedge (\exists n \in \mathbb{N}.\, n \cdot a \leq b \vee a \leq n \cdot b) \Rightarrow a \in T \end{array}\right)$

We call $[m]$ the infinity-class *for $m \in M$. We call $(M, + , \leq)$ an* infinity-class-system *if for every $m \in M$ there exist $m' \in M$ such that $m' \geq m$ and $[m'] \neq [m]$.*

Let in this section $(M, + , \leq)$ be an arbitrary infinity-class-system.

Lemma 3.4.3. *$[\cdot]$ partitions M in equivalence classes, i.e. For m',$m \in M$:*

$$m' \in [m] \Leftrightarrow [m'] = [m]$$

Proof. Let $m' \in [m]$ for $m \in M$. Then there exist $\sim_1, \sim_2 \in \{\leq, \geq\}$, $\sim_1 \neq \sim_2$ and $n \in \mathbb{N}$ such that $m' \sim_1 m$ and $n \cdot m' \sim_2 m$ or $m' \sim_2 n \cdot m$. But exactly then it also holds $m \in [m']$. Then *both* $[m]$ and $[m']$ are the smallest sets T such that $m \in T$ and $m' \in T$ and such that the second condition from Definition 3.4.3 holds. Thus, $[m] = [m']$.

Now let m',$m \in M$ such that $[m'] = [m]$. By definition $m' \in [m']$ and thus $m' \in [m] = [m']$ holds. □

Lemma 3.4.4. *infinity-classes are sign-invariant.*

Proof. Let $m \in M$ and assume there exist a,$b \in [m]$ such that $a \leq 0 \leq b$. Then there exist $n \in \mathbb{N}$ such that $n \cdot a \geq b$ or $a \geq n \cdot b$. But from the order axiom and it follows $a \leq 0 \leq b$ that $n \cdot a \leq b$ and $a \leq n \cdot b$. Thus, this is a contradiction. □

Lemma 3.4.5. *infinity-classes are intervals, i.e. for a,$b \in [m]$ and $a \leq m' \leq b$ it holds that $m' \in [m]$.*

Proof. Let m,$m' \in M$, a,$b \in [m]$ such that $a \leq m' \leq b$. There exist $n \in \mathbb{N}$ such that $n \cdot a \geq b$ or $a \geq n \cdot b$. From the order axiom and from the transitivity of $\leq$ it then also follows $n \cdot m' \geq b$ or $m' \geq n \cdot b$ respectively. Thus, $m' \in [m]$ by second condition from Definition 3.4.3. □

Lemma 3.4.6. *infinity-classes are closed under addition.*

Proof. Let $m \in M$, a,$b \in [m]$. Because $\leq$ is strongly connected we may assume w.l.o.g. that $a \leq b$. From the order axiom we know $c := a + b \leq b + b =: b'$ and $c := a + b \geq a + a =: a'$.
case 1: $a \geq 0$
Then also $b \geq 0$ by Lemma 3.4.4. it holds $2 \cdot c = a + a + b + b \geq 0 + 0 + b + b = b'$ by the order axiom. Thus, by the second condition in Definition 3.4.3 it holds $c \in [b']$. Because $0 \leq b \leq 2 \cdot b =: b'$ and $3 \cdot b \geq 2 \cdot b = b'$ it also holds $b \in [b']$. Thus, by Lemma 3.4.3 it holds $[b'] = [b] = [m]$ and thus $a + b = c \in [m]$.
case 1: $a \leq 0$
Then also $b \leq 0$ by Lemma 3.4.4. it holds $2 \cdot c = a + a + b + b \leq a + a + 0 + 0 =: a'$ by the order axiom. Thus, by the second condition in Definition 3.4.3 it holds $c \in [a']$. Because $0 \geq a \geq 2 \cdot a = a'$ and $3 \cdot a \leq 2 \cdot a = a'$ it also holds $a \in [a']$. Thus, by Lemma 3.4.3 it holds $[a'] = [a] = [m]$ and thus $a + b = c \in [m]$.
□

Lemma 3.4.7. *Let m_1,$m_2 \in M$, $[m_1] \neq [m_2]$. It either holds for all $a_1 \in [m_1]$, $a_2 \in [m_2]$ that $a_1 < a_2$ or it holds for all $a_1 \in [m_1]$, $a_2 \in [m_2]$ that $a_1 > a_2$. We notate this with $[m_1] < [m_2]$ or $[m_1] > [m_2]$ respectively.*

Proof. Let m_1,$m_2 \in M$. Let $a_1, a_2 \in [m_1]$, $a_2, b_2 \in [m_2]$. Assume $a_1 \leq a_2$ and $b_1 \geq b_2$.
case 1: $a_2 \leq b_1$
Then $a_1 \leq a_2 \leq b_1$ and thus by Lemma 3.4.5 it holds $a_2 \in [m_1]$. By Lemma 3.4.3 it then holds $[m_1] = [m_2]$.
case 2: $a_2 \geq b_1$
Then $b_2 \leq b_1 \leq a_2$ and thus by Lemma 3.4.5 it holds $b_1 \in [m_2]$. By Lemma 3.4.3 it then holds $[m_1] = [m_2]$. □

Definition 3.4.4 (Valid Cost Model)**.** *A cost-model which satisfies all Assumptions 1 to 5 is called* valid.

Lemma 3.4.8. *Let $cost : F \to C$ be a valid cost model. Then C is an infinity-class-system.*

Proof. Let $cost : F \to C$ be a valid cost model and assume C not to be an infinity-class-system. Then there exist finitely many pairwise different $M_1, \ldots, M_n \in \bigcup_{m \in C}\{[m]\}$ such that $[0] = M_1 < \ldots < M_n$ (see Lemma 3.4.7). There thus exist $T \in \{1, \ldots, n\}$ such that $\max_{f \in F}[cost(f)] = M_T$. Let $f \in F$ such that $cost(f) \in M_T$. W.l.o.g. we assume f not to be constant and define polynomial p and variable x such that $\deg(p,x) = \max_{p' \in Poly(f)} \max_{x' \in Vars(f)} \deg(p',x')$. Let $p' := p + x^{deg(p,x)+1}$ and $f' \equiv (p' > 0)$. Because of our assumption $[cost(f')] \not> [cost(f)]$ holds. However, because $p' \gg^{\text{rnk}} p$ (see Lemma 3.4.1) and thus $cost(f') > cost(f)$ (by Assumption 5), it follows from Lemma 3.4.7 that $[cost(f')] \not< [cost(f)]$. Thus, $[cost(f')] = [cost(f)]$.
Because all costs in a valid cost model are positive and because $cost(f') > cost(f)$, it then follows from $[cost(f')] = [cost(f)]$, from the order axiom and from the second condition in Definition 3.4.3 that there exist $k \in \mathbb{N}$ such that $k \cdot cost(f) \geq cost(f')$.

Let $g = \bigwedge_{i=1}^{k} f$. By Assumption 3 it holds $cost(g) = n \cdot cost(f) \geq cost(f')$. But because $f \ll^{\text{rnk}} f'$ (see Lemma 3.4.1), also $g \ll^{\text{rnk}} f'$ thus implying $cost(g) < cost(f')$ by Assumption 5. This is a contradiction. Therefore C has to be an infinity-class-system. □

3.4.2 The cost model

Building upon the rank of polynomials as defined by $<^{\text{rnk}}$ in the previous subsection, we define in this subsection a cost model which is compliant with Assumptions 1 to 5, i.e. valid.

For any two polynomials p and q in variables $x_1, \ldots, x_n$ their relative order with respect to $<^{\text{rnk}}$ is equivalent to the relative order of the tuples

$$t_r = (\#\deg(r,K),\ \#\deg(r,K-1),\ \ldots,\ \#\deg(r,0),\ \deg(r,x_n),\ \deg(r,x_{n-1}),\ \ldots,\ \deg(r,x_1))$$

for $r \in \{p,q\}$ where $K = \min\{k \in \mathbb{N}_0 : \forall l > k.\#\deg(p,l) = \#\deg(q,l) = 0\}$ with respect to a lexicographic ordering.

Note that the length of the tuples, which are constructed as above corresponding to polynomials, varies. Also note that $\#\deg(p,k) = 0$ for all polynomials p and sufficiently big $k \in \mathbb{N}$. Thus, we can modify this representation by using sequences $(a_i)_{i\in\mathbb{N}} \subset \mathbb{N}_0$ with finitely many non-zero entries as shown below. Using tuples or sequences to represent costs also naturally gives rise to an addition of them, namely element-wise addition:

Definition 3.4.5 (The Cost Model)**.** *Let*

$$C := \{\textit{sequence } (a_i)_{i\in\mathbb{N}} \subset \mathbb{N}_0 : (a_i)_{i\in\mathbb{N}} \textit{ has finitely many non-zero entries}\}$$

Let the length *of* $(a_i)_{i\in\mathbb{N}} \in C$ *be defined as*

$$len((a_i)_{i\in\mathbb{N}}) = \max(\{0\} \cup \{i \in \mathbb{N} : a_i \neq 0\})$$

Define addition on C *as element-wise addition. We also define an order on* C *such that*

$$(a_i)_{i\in\mathbb{N}} < (b_i)_{i\in\mathbb{N}} :\Leftrightarrow \exists i \in \mathbb{N}.\ a_i < b_i \wedge \forall j > i.\ a_i = b_i$$
$$(a_i)_{i\in\mathbb{N}} \leq (b_i)_{i\in\mathbb{N}} :\Leftrightarrow (a_i)_{i\in\mathbb{N}} < (b_i)_{i\in\mathbb{N}} \vee (a_i)_{i\in\mathbb{N}} = (b_i)_{i\in\mathbb{N}}$$

The cost of a polynomial p *(over variables* $x_1, \ldots, x_n$*) is then defined as* $cost(p) := (a_i)_{i\in\mathbb{N}} \in C$ *such that*

- $a_i = \deg(p,x_i)$ *for* $i \in \{1, \ldots, n\}$ *and*
- $a_{n+i+1} = \#deg(p,i)$ *for* $i \in \mathbb{N}_0$.

The cost of a formula in F *is then inductively defined as follows:*

- $cost(p \sim q) = cost(r)$ *where* r *is the normalized form of* $p - q$*, for all polynomials* p *and* q
- $cost(f_1 \wedge f_2) = cost(f_1 \vee f_2) = cost(f_1) + cost(f_2)$

Theorem 3.4.9. *Definition 3.4.5 defines a valid cost model.*

Proof.

In the context of this proof we use $a \in C$ as short-hand notation for $(a_i)_{i\in\mathbb{N}} \in C$.

C is a totally ordered commutative monoid

The properties of element-wise addition, namely associativity, commutativity and the existence of a zero element ($0 = (0)_{i\in\mathbb{N}}$), can be derived from the respective counterpart of the standard addition on $\mathbb{N}_0$. Obviously the relation $\leq$ as defined above is reflexive. Let $a, b, c \in C$ such that $a \leq b \wedge b \leq c$. If $a = b$ or $b = c$ then clearly $a \leq b$. Otherwise $a < b \wedge b < c$. Then there exist i,$j \in \mathbb{N}$ such that

$$a_i < b_i \wedge \forall k > i.\ a_k = b_k$$

and

$$b_j < c_j \wedge \forall k > j.\ b_k = c_k$$

Thus

$$a_{\max\{i,j\}} < c_{\max\{i,j\}} \wedge \forall k > \max\{i,j\}.\ a_k = c_k$$

and therefore $a < c$. Thus, $<$ and $\leq$ are transitive.
Let a,$b \in C$ such that $a \leq b \wedge b \leq a$. Assume $a \neq b$. Then $a < b \wedge b < a$. Thus, there exist i,$j \in \mathbb{N}$ such that

$$a_i < b_i \wedge \forall k > i.\ a_k = b_k$$

and

$$b_j < a_j \wedge \forall k > j.\ b_k = a_k$$

which is a contradiction. Thus, $a = b$. Therefore $\leq$ is anti-symmetric.
Let a,$b \in C$, $a \neq b$. let $I = \{i \in \mathbb{N} : a_i \neq b_i\}$. Because both a and b only have finitely many non-zero entries, I must have a maximum $i_{\max}$. Thus,

$$\begin{array}{rcc} & & a_{i_{\max}} \neq b_{i_{\max}} \wedge \forall k > i_{\max}.\ a_k = b_k \\ \Rightarrow & & a_{i_{\max}} < b_{i_{\max}} \wedge \forall k > i_{\max}.\ a_k = b_k \\ & \wedge & a_{i_{\max}} > b_{i_{\max}} \wedge \forall k > i_{\max}.\ a_k = b_k \\ \Rightarrow & & a < b \vee a > b \end{array}$$

Thus, $\leq$ is strongly connected.
$\leq$ inherits the order axiom from the standard relation $\leq$ on $\mathbb{N}_0$: Let a,b,$c \in C$. Because $a_i \sim b_i$ if and only if $a_i + c_i \sim b_i + c_i$ for $\sim \in \{<, =\}$ and every $i \in \mathbb{N}$ and because addition on C is defined element-wise, the order axiom holds for the relation $\leq$ on C.
Thus, C is a totally ordered commutative monoid.

Assumptions 1 to 4

The proofs for Assumptions 1 to 4 are either a direct result from the definition of the cost function or are fairly obvious and will not expanded upon here.

Assumption 5

Let p, q be polynomials such that $p <^{\text{rnk}} q$. Then let $K = \min\{k \in \mathbb{N}_0 : \forall l > k.\ \#\deg(p,k) = \#\deg(q,l) = 0\}$. Let

$$t_r = (\#\deg(r,K),\ \ldots,\ \#\deg(r,0),\ \deg(r,x_n),\ \ldots,\ \deg(r,x_1))$$

for $r \in \{p,q\}$. Then t_p is smaller than t_q w.r.t. a lexicographic order. Thus, by construction of *cost* and relation $<$ on C, it also holds that $\mathit{cost}(p) < \mathit{cost}(q)$.
Let $f_1,f_2 \in F$ such that $f_1 \ll^{\text{rnk}} f_2$. Let $p_2 \in \mathit{Poly}(f_2)$ such that $q \ll^{\text{rnk}} p$ for all $q \in \mathit{Poly}(f_1)$. Let $c^{(1)} := \max_{q \in \mathit{Poly}(f_1)} \mathit{cost}(q)$ and some $p_1 \in \mathit{Poly}(f_1)$ such that $c^{(1)} = \mathit{cost}(p_1)$. Because C defines a totally ordered abelian monoid, $c^{(1)}$ is well-defined. Let n be the number of constraints in the formula f_1. Because f_1 might contain two constraints whose normalized forms have the same polynomial, it holds that $n \geq |\mathit{Poly}(f_1)|$. Then, by the additive property of costs and the order axiom, we get $\mathit{cost}(f_1) \leq n \cdot \max_{q \in \mathit{Poly}(f_1)} \mathit{cost}(q) = n \cdot c^{(1)}$. We also get because $p_2 \in \mathit{Poly}(f_2)$, that $\mathit{cost}(f_2) \geq \mathit{cost}(p_2) =: c^{(2)}$.

Because $p_1 \ll^{\text{rnk}} p_2$ there exist, by construction of *cost*, a $j \in \mathbb{N}$ such that

$$c^{(2)}_j > 0 = c^{(1)}_j \wedge \forall l > j.\ c^{(2)}_l = c^{(1)}_l = 0$$

and thus

$$c^{(2)}_j > 0 = \left(n \cdot c^{(1)}\right)_j \wedge \forall l > j.\ c^{(2)}_l = \left(n \cdot c^{(1)}\right)_l = 0$$

Therefore we have

$$\mathit{cost}(f_1) \leq n \cdot c^{(1)} < c^{(2)} \leq \mathit{cost}(f_2)$$

thus proving Assumption 5 to hold for this cost model.

Altogether it then holds that this cost model is valid. From Lemma 3.4.8 we can derive further that C is an infinity-class-system.

□

Note that we have so far only discussed the cost of formulas in the context of the computational complexity of analyzing it further. A good heuristic for the overall cost of a conflict generalization should however also include considerations about the size, e.g. the volume as defined by the *Lebesgue-measure*, of the space of assignments for the variables $x_1, \ldots, x_n$ which are excluded from any possible solution to the conjunction of the given constraints by the conflict generalization. What constitutes a "good" conflict generalization should consider both its complexity as done by our model and this space of assignments. Even tough we did not mention it explicitly, our cost model takes the latter consideration into account. A polynomial with lower degrees in its variables also has less zeros, thus (in most cases) bigger sign-invariant regions. As we are optimized for low degrees, we, in effect, optimize (heuristically) also for bigger such regions. Any other more accurate method of estimation for the size of this (high-dimensional) space would in practice be very difficult, that is to say computationally expensive. Thus, the use of the cost model defined in this section may, in extreme cases, result in conflict generalizations which only cover a single point, the sample which was given. This however should be fairly rare.

3.5 Conflict generalization as an optimization problem

In the following sections we want to mathematically describe the algorithmic problem of finding a conflict generalization given a valid cost model (over an infinity-class-system) and a conflict in constraints $c_1, \ldots, c_m$ in sample s and propose algorithms to do so. We consider three kinds of problems: *decision problems*, *search problems* and *optimization problems*. A decision problem is a problem in which a yes/no question is stated about specific input instances. A search problem's goal is to find a substructure of a given structure that satisfies certain properties. An optimization problem is a search problem in which each solution is assigned a cost or a weight. The goal of an optimization problem is then to find an *optimal* solution, meaning a solution to its underlying search problem which has minimal costs. In Section 3.4 we have defined an infinity-class-system as the base for a cost model. We also use infinity-class-systems as the base for the optimization problems in the following sections. We therefore further define for an optimization problem a *relaxed* version whose goal it is to find a solution to its underlying search problem such that the cost of the solution is *class-optimal*, meaning in the same infinity-class as the cost obtained by an optimal solution.

Complexity classes such as P and NP, which represent the notion of the "hardness" of problems, are normally only applied to decision problems. In the context of this thesis we define an optimization problem to be *hard*, if its corresponding decision problem "Given an instance and a cost c, does a solution exist with costs $c^* \leq c$?" is NP-hard. Furthermore we define an optimization problem to be *efficiently computable* or *easy* if a polynomial time algorithm exist that generates an optimal solution to a given instance. Likewise we define the relaxed optimization problem to be hard if the corresponding decision problem "Given an instance and a cost c, does a solution exist with costs $\bar{c}$ such that $[\bar{c}] \leq [c]$?" is NP-HARD and efficiently computable or easy if a polynomial time algorithm exist that generates a class-optimal solution to a given instance. Unless P = NP, easy and hard are mutually exclusive terms. We can proof a (relaxed) optimization problem to be hard, if we can show that there is a reduction from a know NP-HARD decision problem to the decision version of the (relaxed) optimization problem.

With those terms introduced we want to describe the problem of conflict generalization.

In Section 3.3 we found a description of possible conflict generalizations for which we defined a cost function in Section 3.4. Hence, our goal is to find a conflict generalization of this form which minimizes said cost. We want to define this problem more precisely:

Definition 3.5.1 ((RELAXED)-CONFLICTGENERALIZATION)**.** CONFLICTGEN-

ERALIZATION *is an optimization problem. Given an instance*

$$c_1,\ldots,c_m; s: \text{ conflict in constraints } c_1,\ldots,c_m \text{ and sample } s$$
$$\text{with elimination variable } y$$
$$(C, +, \leq): \text{ infinity-class-system}$$
$$cost: F \to C \text{ a valid cost model}$$

a solution to CONFLICTGENERALIZATION *is a conflict generalization of the form as described in Section 3.3. An optimal solution is a conflict generalization* φ^* *such that*

$$cost(\varphi^*) = \min_{confl.gen.\varphi} cost(\varphi)$$

A class-optimal solution is a conflict generalization $\bar{\varphi}$ *such that*

$$cost(\bar{\varphi}) \in \left[\min_{confl.gen.\varphi} cost(\varphi)\right]$$

The problem of finding a class-optimal solution to CONFLICTGENERALIZATION *is called* RELAXED-CONFLICTGENERALIZATION.

Theorem 3.5.1. CONFLICTGENERALIZATION *is hard.* CONFLICTGENERALIZATION *is even then hard, if we fix a valid cost-model or only allow a subset of all valid cost-models as parts of instances.*

Theorem 3.5.1 shows us that even a clever choice of a cost-model does not make the problem easy (unless P = NP).

In the remainder of this section, Theorem 3.5.1 is proven. In Section 3.6 algorithms to both the relaxed and the non-relaxed version of CONFLICTGENERALIZATION are presented, the second of which operates in polynomial time, thus proving RELAXED-CONFLICTGENERALIZATION to be easy.

We proof Theorem 3.5.1 by reducing a known NP-HARD problem to the decision problem of CONFLICTGENERALIZATION (with arbitrary cost-model). For this reduction we use a variant of SAT:

Definition 3.5.2 (AT-MOST-3SAT(2L))**.** AT-MOST-3SAT(2L) *is a decision problem. Given a Boolean formula* f *in conjunctive normal form such that each clause has at most three literals and each literal appears at most twice in* f*, it asks whether* f *is satisfiable, i.e. whether an assignment to its variables exist such that* f *evaluates to true under this assignment.*

By applying [LO1] on page 259 of [MRG79] we know AT-MOST-3SAT(2L) to be NP-HARD. For an instance f of AT-MOST-3SAT(2L) and an arbitrary valid cost-model we want to construct in polynomial time an instance I of CONFLICTGENERALIZATION and a cost w such that

$$f \text{ is satisfiable } \Leftrightarrow \text{CONFLICTGENERALIZATION has a solution to } I \text{ with costs} \leq w$$

thus proving the reduction.

Before we present the proposed reduction, we first present some proofs and definitions that simplify the proof for the proposed reduction:

Let *cost* be an arbitrary (allowed) cost-model over an infinity-class-system $(W, +, \leq)$. Let x and y be real-arithmetic variables. Let

$$
\begin{aligned}
&L : \mathbb{Z} \times \{\leq, \geq\} \to CS, (\alpha, \sim) \mapsto (y - \alpha \sim 0)\\
&Q : \mathbb{Z} \times \mathbb{Z} \times \mathbb{N}_0 \times \{\leq, \geq\} \to CS, (\alpha,\beta,k, \sim) \mapsto (x^k(y - \alpha)(y - \beta) \sim 0)\\
&\mathcal{L} = L(\mathbb{Z} \times \{\leq, \geq\}) \subset CS\\
&\mathcal{Q} = Q(\mathbb{Z} \times \mathbb{Z} \times \mathbb{N}_0 \times \{\leq, \geq\}) \subset CS\\
&\mathcal{C} = \mathcal{L} \cup \mathcal{Q}
\end{aligned}
$$

Let sample $s = 1 \in \mathbb{R}^1$. Because we only use the sample $s = 1$ in this section, we omit it in formulas (e.g. we write $[\![\cdot]\!]$ instead of $[\![\cdot]\!]^{x\mapsto s}$ or $T(c)$ instead of $T(c,s)$ to get the type of a constraint). We construct an instance of ConflictGeneralization only using constraints in $\mathcal{C}$ and sample $s = 1$. Let

$$
\begin{aligned}
\mathcal{T}_{\mathcal{L}} &= \bigcup_{c\in\mathcal{L}} \{I_1, \ldots, I_k \in \mathbb{I}_{sym} : T(c) = I_1 \cup \ldots \cup I_k\}\\
\mathcal{T}_{\mathcal{Q}} &= \bigcup_{c\in\mathcal{Q}} \{I_1, \ldots, I_k \in \mathbb{I}_{sym} : T(c) = I_1 \cup \ldots \cup I_k\}\\
\mathcal{T} &= \mathcal{T}_{\mathcal{L}} \cup \mathcal{T}_{\mathcal{Q}}
\end{aligned}
$$

Lemma 3.5.2. *For all $I,J \in \mathcal{T}$ it holds cost$(I \mathbin{\text{⨱}}_{sym} J) = 0$*

Proof. Let $L := L(\alpha, \sim) \in \mathcal{L}$ be arbitrary. Obviously $T(L) = [-\infty,\alpha]$ or $T(L) = [\alpha, +\infty]$. Thus, all symbolic interval bounds in $\mathcal{T}_{\mathcal{L}}$ are constant.

Let $Q := Q(\alpha,\beta,k, \sim) \in \mathcal{Q}$ be arbitrary. As mentioned in Section 3.2 the bounds of the symbolic intervals of $T(Q)$ must be an (on x dependant) assignment for y such that the polynomial $x^k(y - \alpha)(y - \beta)$ of Q evaluates to 0. The normalized forms of the zeros of this polynomial must be α and β (since we have sample $s = 1 \neq 0$ for x). Thus, all symbolic interval bounds in $\mathcal{T}_{\mathcal{Q}}$ must have a constant normalized form. Therefore $I \mathbin{\text{⨱}}_{\text{sym}} J = \pi_{SqrtEx}(rightBound(I)) \mathbin{{}_{(}\!\geq_{)}} \pi_{SqrtEx}(leftBound(J))$ must have a constant normalized form for $I,J \in \mathcal{T} = \mathcal{T}_{\mathcal{L}} \cup \mathcal{T}_{\mathcal{Q}}$. By applying Assumption 1 to 4 we then get $cost(I \mathbin{\text{⨱}}_{\text{sym}} J) = 0$ for all $I,J \in \mathcal{T}$. □

Therefore if we construct an instance of ConflictGeneralization only using constraints in $\mathcal{C}$, we don't have to account for the cost of the terms $\cdot \mathbin{\text{⨱}}_{\text{sym}} \cdot$ in any solution. The cost of a solution is thus fully given by the cost of the side condition terms. Note that the cost of side conditions of constraints in $\mathcal{L}$ must be constant and thus have 0 cost. Lets define $cost(p) := cost(p \sim 0)$ for every polynomial p and an arbitrary relation $\sim$ (see Assumption 2). Let $\omega_k = cost(x^k)$ for $k \in \mathbb{N}_0$. Note that by Assumption 1 $w_0 = 0$ holds. By Assumption 2 it holds $cost(\alpha x^k) = cost(x^k) = \omega_k$ for all $k \in \mathbb{N}_0$ and all constant $\alpha \in \mathbb{Z}$. Note that by Assumption 5 $[\omega_0] < [\omega_1] < \ldots$ must hold.

Lemma 3.5.3. *For $Q := Q(\alpha,\beta,k, \sim) \in \mathcal{Q}$ it holds that $cost(sc(Q)) = \omega_{2k} + \omega_k$.*

Proof. Let $Q := Q(\alpha,\beta,k, \sim) \in \mathcal{Q}$. We use notation $p \leftrightarrow q$ for polynomials p and q to indicate that they have the same normalized form. For formulas f and g we use notation $f \leftrightarrow g$ to indicate that g can be constructed out of f by replacing polynomials

and constraints with other polynomials or constraints which have the same normalizd form. E.g.: $xyx - 1 > 0 \wedge yx < 0 \leftrightarrow x^2y > 1 \wedge xy < 0$. Then it holds:

$$\begin{aligned} Q &= Q(\alpha,\beta,k,\sim) \\ &= x^k(y-\alpha)(y-\beta) \sim 0 \\ &\leftrightarrow \left(x^k\right) y^2 + \left(-x^k(\alpha+\beta)\right) y + \left(x^k\alpha\beta\right) \sim 0 \end{aligned}$$

Let $a(x)y^2 + b(x)y + c(x) \sim 0$ be the normalized form of Q. Then

$$a(x) \leftrightarrow x^k \wedge b(x) \leftrightarrow -x^k(\alpha+\beta) \wedge c(x) \leftrightarrow x^k\alpha\beta$$

For $D(x) := b(x)^2 - 4a(x)c(x)$ it holds

$$\begin{aligned} D(x) &= b(x)^2 - 4a(x)c(x) \\ &\leftrightarrow \left(-x^k(\alpha+\beta)\right)^2 - 4x^kx^k\alpha\beta \\ &\leftrightarrow x^{2k}\left(\alpha^2 + 2\alpha\beta + \beta^2\right) - 4x^{2k}\alpha\beta \\ &\leftrightarrow x^{2k}\left(\alpha^2 - 2\alpha\beta + \beta^2\right) \\ &\leftrightarrow x^{2k}(\alpha-\beta)^2 \end{aligned}$$

Hence

$$cost(D(x)) = cost\left(x^{2k}(\alpha-\beta)^2\right) = cost\left(x^{2k}\right) = \omega_{2k}$$

and

$$cost(a(x)) = cost\left(x^k\right) = \omega_k$$

Depending on $\sim \{\neq, >, \geq\}$, Q has either type $T_{(2,1)}$, $T_{(2,5)}$ or $T_{(2,6)}$. Therefore we get for some relation $\sim'$ (see Table 3.1, Assumption 3):

$$\begin{aligned} cost(sc(Q)) &= cost(a(x) \sim' 0 \wedge D(x) \geq 0) \\ &= cost(a(x) \sim' 0) + cost(D(x) \geq 0) \\ &= cost(a(x)) + cost(D(x)) \\ &= \omega_k + \omega_{2k} \end{aligned}$$

□

Now we present the proposed reduction:
Let f be an instance of AT-MOST-3SAT(2L) over Boolean variables $z_1,\ldots,z_n$. Let $C_1,\ldots,C_m$ be the clauses of f represented as sets of literals $z_1,\ldots,z_n,\bar{z_1},\ldots,\bar{z_n}$, i.e. $f = \bigwedge_{i=1}^{m} \bigvee_{l \in C_i} l$.

For each clause C_i define

$$p_{C_i} = -i \in \mathbb{Z}$$

For each variable z_i define

$$\begin{aligned} p_{z_i} &= 7i \\ p_{(z_i,1)} &= 7i-6 \\ p_{(z_i,2)} &= 7i-5 \\ p_{(z_i,3)} &= 7i-4 \\ p_{(\bar{z_i},1)} &= 7i-3 \\ p_{(\bar{z_i},2)} &= 7i-2 \\ p_{(\bar{z_i},3)} &= 7i-1 \end{aligned}$$

$$q_{(z_i,1)} = \begin{cases} 0 & \text{if literal } z_i \text{ appears in no clause} \\ p_{C_j} & C_j \text{ is the clause with the lowest index } j \text{ in which literal } z_i \text{ appears} \end{cases}$$

$$q_{(z_i,2)} = \begin{cases} 0 & \text{if literal } z_i \text{ appears at most once in a clause} \\ p_{C_j} & C_j \text{ is the clause with the highest index } j \text{ in which literal } z_i \text{ appears} \end{cases}$$

$$q_{(z_i,3)} = p_{z_i}$$

$$q_{(\bar{z_i},1)} = \begin{cases} 0 & \text{if literal } \bar{z_i} \text{ appears in no clause} \\ p_{C_j} & C_j \text{ is the clause with the lowest index } j \text{ in which literal } \bar{z_i} \text{ appears} \end{cases}$$

$$q_{(\bar{z_i},2)} = \begin{cases} 0 & \text{if literal } \bar{z_i} \text{ appears at most once in a clause} \\ p_{C_j} & C_j \text{ is the clause with the highest index } j \text{ in which literal } \bar{z_i} \text{ appears} \end{cases}$$

$$q_{(\bar{z_i},3)} = p_{z_i}$$

Also define for each literal l constraints

$$\begin{aligned} c_{(l,1)} &= Q(p_{(l,1)}, q_{(l,1)}, 1, \neq) \\ c_{(l,2)} &= Q(p_{(l,2)}, q_{(l,2)}, 1, \neq) \\ c_{(l,3)} &= Q(p_{(l,3)}, q_{(l,3)}, 1, \neq) \\ c_{(l,4)} &= Q(p_{(l,1)}, p_{(l,3)}, 2, >) \end{aligned}$$

Let

$$M = \bigcup_{i=1}^{n} \left(\{p_{z_i}\} \cup \bigcup_{j=1}^{3} \{p_{(z_i,j)}, p_{(\bar{z_i},j)}, q_{(z_i,j)}, q_{(\bar{z_i},j)}\} \right) \setminus \{0\}$$

Note that $\{p_{C_1}, \ldots, p_{C_m}\} \subset M$. Let $\gamma_{\min} = \min M$, $\gamma_{\max} = \max M$. Define $c_{\min} = L(\gamma_{\min}, \geq)$ and $c_{\max} = L(\gamma_{\max}, \leq)$. Let $\{v_1, \ldots, v_{|M|}\} = M$ be an enumeration of M such that $\gamma_{\min} = v_1 < v_2 < \ldots < v_{|M|} = \gamma_{\max}$. Then define for $i = 1, \ldots, |M| - 1$: $c_i = Q(v_i, v_{i+1}, 0, \geq)$. Let I be the instance of ConflictGeneralization with all constraints as described above and a given (valid) cost-model. this instance is constructable in polynomial time.

Lemma 3.5.4. *f is satisfiable iff, given instance I, there exist a solution to* ConflictGeneralization *with cost $n((w_3 + w_6) + 2(w_1 + w_2)) + n(w_2 + w_4)$ (or less).*

Proof. Each interval $[v_i + , v_{i+1} -]$ is covered by (only) by constraint c_i. Furthermore intervals $[-\infty, v_1] = [-\infty, \gamma_{\min}]$ and $[v_{|M|}, +\infty] = [\gamma_{\max}, +\infty]$ are covered (only)

by constraints $c_{\min}$ and $c_{\max}$ respectively. Note that point 0 may be covered by other constraints as well. Moreover the cost of constraints $c_{\min}$,$c_1,\ldots,v_{|M|-1}$,$c_{\max}$ are constructed to be zero. Therefore any solution to ConflictGeneralization to the given instance I and its cost differs only to any other solution in the way how points in $M = \{v_1,\ldots,v_{|M|}\}$ are covered. It holds for all literals l ($l = z$ or $l = \bar{z}$ for variable z) by construction:

$$\begin{aligned}
T(c_{(l,1)}) &= [p_{(l,1)},p_{(l,1)}] \cup [p_{(l,1)},p_{(l,1)}] \\
cost(sc(c_{(l,1)})) &= \omega_2 + \omega_1 \in [\omega_2] \\
T(c_{(l,2)}) &= [p_{(l,2)},p_{(l,2)}] \cup [p_{(l,2)},p_{(l,2)}] \\
cost(sc(c_{(l,2)})) &= \omega_2 + \omega_1 \in [\omega_2] \\
T(c_{(l,3)}) &= [p_{(l,3)},p_{(l,3)}] \cup [p_{(l,3)},p_{(l,3)}] \\
&= [p_{(l,3)},p_{(l,3)}] \cup [p_z,p_z] \\
cost(sc(c_{(l,3)})) &= \omega_6 + \omega_3 \in [\omega_6] \\
T(c_{(l,4)}) &= [p_{(l,1)},p_{(l,3)}] \\
cost(sc(c_{(l,4)})) &= \omega_4 + \omega_2 \in [\omega_4]
\end{aligned}$$

We show that every optimal solution to ConflictGeneralization encodes an assignment of variables $z_1,\ldots,z_n$ of f and has costs of at least $w^* := n((w_3 + w_6) + 2(w_1+w_2))+n(w_2+w_4)$ and that a solution of ConflictGeneralization has costs of *exactly* w^* iff its corresponding assignment of variables $z_1,\ldots,z_n$ is a solution to f.

Consider any variable z_i and the point p_{z_i}. p_{z_i} is only covered by constraints $c_{(z_i,3)}$ and $c_{(\bar{z_i},3)}$, therefore at least one of these constraints must be used to cover point p_{z_i}. All other points which are covered by either $c_{(z_i,3)}$ or $c_{(\bar{z_i},3)}$, namely $p_{(z_i,3)}$ and $p_{(\bar{z_i},3)}$, are also covered by other constraints, namely $c_{(z_i,4)}$ and $c_{(\bar{z_i},4)}$ respectively, which have lower cost associated with them. Thus, in an optimal solution to ConflictGeneralization exactly one of the constraints $c_{(z_i,3)}$ or $c_{(\bar{z_i},3)}$ is used to cover point p_{z_i} at cost $\omega_6 + \omega_3$, thus encoding a variable assignment for variable z_i. Hence every optimal solution corresponds to an unique variable assignment of variables $z_1,\ldots,z_n$. All points $p_{z_1},\ldots,p_{z_n}$ are covered at cost $n(\omega_6 + \omega_3)$ in an optimal solution.

Now consider a literal l of a variable z_i. If $c_(l,3)$ is used to cover point p_{z_i}, then it also covers point $p_{(l,3)}$ (at no "additional" cost).
Then it is always cheaper to cover points $p_{(l,1)}$ and $p_{(l,2)}$ by constraints $c_{(l,1)}$ and $c_{(l,2)}$ respectively at cost $2(\omega_2 + \omega_1)$ instead of covering them by constraint $c_{(l,4)}$ at cost $\omega_4 + \omega_2$. If however $c_(l,3)$ is *not* used to cover point p_{z_i}, then constraint $c_{(l,4)}$ must be used to cover point $p_{(l,3)}$ which then also covers points $p_{(l,1)}$ and $p_{(l,2)}$.
Thus, any optimal solution has cost of at least $n((w_3+w_6)+2(w_1+w_2))+n(w_2+w_4) = w^*$.
Let for an assignment $Z \in \mathbb{B}^n$ for variables $z_1,\ldots,z_n$:

$$\begin{aligned}
S(Z) = &\{c_{(l,1)},c_{(l,2)},c_{(l,3)} : \text{literal } l \text{ is true under assignment } Z\} \\
&\cup \{c_{(l,4)} : \text{literal } l \text{ is false under assignment } Z\} \\
&\cup \{c_{\min},c_1,\ldots,c_{|M|-1},c_{\max}\}
\end{aligned}$$

As shown above, any solution to ConflictGeneralization which uses all constraints in $S(Z)$ (for an assignment $Z \in \mathbb{B}^n$) already covers all points $\mathbb{R} \setminus (\bigcup_{i=1}^{m} \{p_{C_i}\})$ at cost $w^* = n((w_3 + w_6) + 2(w_1 + w_2)) + n(w_2 + w_4)$.

Let f be satisfiable. Let $Z \in \mathbb{B}^n$ be a solution to f. Then for each clause C_i at least one literal $l \in C_i$ must evaluate to *true* under assignemnt Z, thus

$$\forall i = 1,\ldots,m.\ \exists l \in C_i.\ \exists j = 1{,}2.\ \left(c_{(l,j)} \in S \wedge c_{(l,j)} \text{ covers point } p_{C_i}\right)$$

Thus, a solution to ConflictGeneralization which uses all constraints in $S(Z)$ for a solution $Z \in \mathbb{B}^n$ of f already covers every point on the number-line and because we assume f to be satisfiable, there exist a solution to the constructed instance of ConflictGeneralization with costs of exactly

$$\begin{aligned}
&\sum_{c \in S(Z)} cost(sc(c)) \\
&= \sum_{\text{literal } l} \left\{ \begin{array}{ll} 0 & \text{if literal } l \text{ is false under } Z \\ cost(sc(c_{(l,1)})) + cost(sc(c_{(l,2)})) + cost(sc(c_{(l,3)})) & \text{otherwise} \end{array} \right\} \\
&\quad + \sum_{\text{literal } l} \left\{ \begin{array}{ll} 0 & \text{if literal } l \text{ is true under } Z \\ cost(sc(c_{(l,4)})) & \text{otherwise} \end{array} \right\} \\
&\quad \underbrace{+ cost(sc(c_{\min})) + \sum_{i=1}^{|M|-1} cost(sc(c_i)) + cost(sc(c_{\max}))}_{=0} \\
&= n((w_3 + w_6) + 2(w_1 + w_2)) + n(w_2 + w_4) = w^*
\end{aligned}$$

Note that the cost of terms of the form $I \mathbin{\text{⨟}}_{sym} J$ is zero as we have shown above, thus the cost of these terms is not included in the calculation above! Thus, we have shown that, if f is satisfiable, then there exist a solution to our constructed instance of ConflictGeneralization with cost of at most (exactly) w^*.
Now let there exist a solution to ConflictGeneralization with cost of at most (exactly) w^*. We derive that f is then satisfiable: Let φ be a solution to ConflictGeneralization with cost of at most (exactly) w^*. Note that the cost of any solution to ConflictGeneralization is*at leat* w^* as we have shown above, thus φ must be an optimal solution. As such it encodes a variable assignment $Z \in \mathbb{B}^n$ which we also have demonstrated above. As previously mentioned, any optimal solution which encodes $Z \in \mathbb{B}^n$ must use at least all constraints in $S(Z)$ to cover all points in $\mathbb{R} \setminus (\bigcup_{i=1}^{m} \{p_{C_i}\})$ at cost of at least w^*. But because the cost of φ is exactly w^* and because there is no contraint we have constructed which can cover points $\bigcup_{i=1}^{m} \{p_{C_i}\}$ at no cost, we already know, that the constraints in $S(Z)$ cover *all* points on the real-number line, especially points $p_{C_1},\ldots,p_{C_m}$. For that to be the case, each clause C_i must have one literal $l \in C_i$ which evaluates to *true* under assignment Z, so that point p_{C_i} is already covered by a constraint. In other words: f is satisfiable and has solution Z.
Altogether we have shown that f is satisfiable iff there is a solution to instance I of cost of at most $n((w_3 + w_6) + 2(w_1 + w_2)) + n(w_2 + w_4)$. □

With this proof we have shown that an instance of At-most-3SAT(2L) can be reduced to an instance of the decision problem of ConflictGeneralization. To reiterate: Because At-most-3SAT(2L) is known to be NP-hard [MRG79], we therefore have proven ConflictGeneralization to be hard (see Theorem 3.5.1).

3.6 Non-local shortest path

In this section we want to rephrase, i.e. reduce, ConflictGeneralization to a shortest path problem in a graph. However because ConflictGeneralization is hard, we cannot reduce it to a "standard" shortest path problem that can be solved, for example by *Dijkstra's algorithm*[Dij59], in polynomial time. We therefore define the following variation of the shortest path problem:

Definition 3.6.1 ((relaxed-)NonLocalIcsShortestPath)**.** NonLocalIcsShortestPath *is an optimization problem. Given an instance*

$$P = (V,E,s,t,C,c,W,+,\leq,w_E,w_C)$$

where

- (V,E) *is a finite directed acyclic graph (DAG) with vertices* V *and edges* $E \subseteq V \times V$ *[BJG09]*
- $\varsigma,\tau \in V, \varsigma \neq \tau$ *such that there is a path from* ς *to* τ
- C *is a set of colors*
- $c : V \to C$ *is a color-label function for vertices*
- $(W,+,\leq)$ *is an infinity-class-system (ICS)*
- $w_E : E \to W$ *assigns costs to edges*
- $w_C : C \to W$ *assigns costs to colors*

A solution to NonLocalIcsShortestPath *is a path from* ς *to* τ *in* (V,E)*. A path* $p = (v_1,\ldots,v_k)$ *in* (V,E) *is assigned the combined cost of its edges and of the set of colors of its nodes:*

$$w(p) = \left(\sum_{i=1}^{k-1} w_E(v_i,v_{i+1})\right) + \left(\sum_{\kappa \in \{c(v_1),\ldots,c(v_k)\}} w_C(\kappa)\right)$$

An optimal solution p^* *is an optimal path from* ς *to* τ *with respect to its cost, i.e.*

$$w(p^*) = \min_{path\ p\ from\ \varsigma\ to\ \tau} w(p)$$

A class-optimal solution $\bar{p}$ *is a solution such that*

$$w(\bar{p}) \in \left[\min_{path\ p\ from\ \varsigma\ to\ \tau} w(p)\right]$$

The problem of finding a class-optimal solution to NonLocalIcsShortestPath *is called* relaxed-NonLocalIcsShortestPath*.*

Theorem 3.6.1. ConflictGeneralization *can be reduced to* NonLocalIcsShortestPath*. A solution* p *to* NonLocalIcsShortestPath *corresponds to a solution* φ *to* ConflictGeneralization*.* p *is (class-)optimal iff* φ *is (class-)optimal in the*

reduction proposed below.

Proof. First we show how to find for a set of intervals an ordered covering by translating the problem into an instance of NonLocalIcsShortestPath. This approach then can be used similarly for the purpose of reducing ConflictGeneralization: Let $U \subset \mathbb{I}$ be a finite set of intervals which cover $\mathbb{R}$. Let $V = U \cup \{\varsigma,\tau\}$ be nodes of a graph where we define ς and τ to be some fresh symbols, i.e. $\varsigma,\tau \notin U$. Now we create a set of edges such that the resulting node encodes all ordered coverings which are possible using intervals from U. Thus, let

$$\begin{aligned} E =& \{(I,J) \in U \times U : I \mathbin{\text{⟛}^{>}} J\} \\ & \cup \{(\varsigma,I) : I \in U,\ \mathit{leftBound}(I) = -\infty\} \\ & \cup \{(I,\tau) : I \in U,\ \mathit{leftBound}(I) = +\infty\} \end{aligned}$$

Note that every path from ς to τ in the Graph $G = (V,E)$ corresponds to a unique strong ordered covering and vice versa. Also note that because $I \mathbin{\text{⟛}^{>}} J$ implies $\mathit{rightBound}(J) > \mathit{rightBound}(I)$ for intervals I and J, that the graph $G = (V,E)$ is a directed acyclic graph. Thus, if we choose an arbitrary infinity-class-system, assign arbitrary colors to all nodes and set all color- and edge-weights arbitrarily, we get an instance of *NonLocalIcsShortestPath* where every strong ordered covering of $\mathbb{R}$ by intervals in U corresponds to a unique solution of the problem and vice versa.
This technique is essentially the same for the reduction of an instance of ConflictGeneralization to an instance of NonLocalIcsShortestPath. Let

$c_1,\ldots,c_m; s:$ conflict in constraints $c_1,\ldots,c_m$ and sample s
in variables $x_1,\ldots,x_n,y$ and with elimination variable y
$(C, +, \leq):$ infinity-class-system
$\mathit{cost} : F \to C$ a valid cost model ($(C, +, \leq)$ is an infinity-class-system)

be an instance of ConflictGeneralization.

Define the set $\{c_1,\ldots,c_m\}$ of constraints as a set of colors. For each constraint c_i obtain its unsatisfied symbolic intervals $I_1,\ldots,I_k$ under sample s, i.e. $T(c_i,s) = I_1 \cup \ldots \cup I_k$, as defined in Section 3.2 (see Table 3.1). Each of these intervals defines a node in our constructed graph which we color according to their constraint. The cost of a color is set to the cost of the side condition corresponding to the constraint as obtained by Table 3.1. The nodes ς and τ are assigned a fresh color which has no cost. Two symbolic intervals I and J share an edge if $[\![I]\!]^{x\mapsto s} \mathbin{\text{⟛}^{>}} [\![J]\!]^{x\mapsto s}$ is true. Those edges can trivially be checked by an quadratic algorithm which iterates over all pairs of intervals. Below we present a more efficient way of creating these edges, but in the context of this proof this trivial method suffices. The cost of such an edge is set to the cost of the formula $I \mathbin{\text{⟛}_{\text{sym}}} J$. We also add two more nodes ς and τ and edges (ς,I) for symbolic intervals I with $\mathit{leftBound}(I) = -\infty$ and edges (I,τ) for symbolic intervals I with $\mathit{rightBound}(I) = +\infty$. These nodes and edges are assigned zero cost.
Similarly to above, every path from ς to τ (and thus every solution to NonLocalIcsShortestPath) corresponds to a unique strong ordered covering of $\mathbb{R}$ by the evaluated intervals from above (by sample s) and thus to a unique conflict generalization of the form as in Theorem 3.3.2 and vice versa. We can translate a solution to NonLocalIcsShortestPath back to a solution of ConflictGeneralization

by creating a conjunction of all side conditions to all constraints, i.e. colors, of nodes traversed in the solution and also of all $\mathbin{\text{⧖}}_{\text{sym}}$-formulas encoded in the edges of the traversed path.

The overall cost of a solution to NonLocalIcsShortestPath is by design the exact same cost of the corresponding solution to ConflictGeneralization. Thus, a solution to ConflictGeneralization is (class-)optimal iff its corresponding solution to NonLocalIcsShortestPath is (class-)optimal, thus proving our theorem. □

As mentioned in the proof above we want to find a more efficient way of finding wether two nodes, i.e. symbolic intervals, in the constructed graph used for the reduction share an edge than simply to just check every possible pair of symbolic intervals. To do so we make the following observation for (non-symbolic) intervals I and J:

$$I \mathbin{\text{⧖}}^{>} J \Rightarrow \mathit{leftBound}(J) \leq \mathit{rightBound}(I) < \mathit{rightBound}(J)$$

To find for a set M of intervals all pairs $(I,J) \in M \times M$ with $I \mathbin{\text{⧖}}^{>} J$, we thus only have to check pairs that obey this order. We therefore put all interval bounds in a list and order them such that left bounds are always ordered before right bounds if they are equal. We may then iterate though this list and by remembering over which left bounds and which right bounds was iterated we find all candidates for these pairs.
This technique has worst-case time complexity which is quadratic, same as the trivial technique described in the proof above. However its average time complexity should be significantly less.
Altogether we sumerize the proposed reduction in Algorithm 1 and Algorithm 2.

Theorem 3.5.1 states that ConflictGeneralization is a hard optimization problem. Thus, NonLocalIcsShortestPath must be hard as well as shown by Theorem 3.6.1. However we show relaxed-NonLocalIcsShortestPath and therefore also relaxed-ConflictGeneralization to be easy:

Theorem 3.6.2. relaxed-NonLocalIcsShortestPath *is easy.*

Proof. To proof Theorem 3.6.2 we make the following observation: Assume we can construct a polynomial time algorithm that calculates for an instance of NonLocalIcsShortestPath a sub-graph of the given graph such that *every* path from ς to τ, i.e. every solution, is class-optimal. Then trivially we can find a solution in this sub-graph in polynomial time, e.g. by utilizing Dijkstra's algorithm[Dij59], setting all edge and vertex weights to zero. Furthermore, we know for any given path from ς to τ with summed-up cost c in the problem graph that it is either a class-optimal solution or that there exist a (class-optimal) path from s to t which traverses no edge and no vertex whose associated (color-)cost is in $[c]$ or greater than c. Therefore we may construct such a subgraph by (1) creating a list of all edges and vertices and sorting them according to their associated cost and (2) iteratively adding another edge or vertex of minimal cost until there exist at least one solution which we may check by utilizing Dijkstra's algorithm as described above. Because sorting and Dijkstra's algorithm can be done in polynomial time, we can find a sub-graph with the desired properties in polynomial time, thus proving our theorem. □

Obviously the proposed algorithm in the above proof is not optimal in terms of its runtime as it requires Dijkstra's Algorithm[Dij59] to be run in each step. Therefore

Algorithm 1: SolveConflictGeneralization

Input :

- Instance of ConflictGeneralization:

  $c_1,\dots,c_m; s:$ conflict in constraints $c_1,\dots,c_m$ and sample s with elimination variable y
  $(W, +, \leq):$ infinity-class-system
  $cost: F \to W$ a valid cost model

- An Algorithm Algo which produces a solution to NonLocalIcsShortestPath

Output: A conflict generalization

/* Define set C of colors (constraints and a "non-color") */
$C \leftarrow \{c_1,\dots,c_m,0_C\}$
/* Define set V of nodes */
$V \leftarrow \{\varsigma,\tau\}$
for *constraint* c_i **do**
  /* Retreive intervals of the type of the constraint */
  $I_{(i,1)} \cup \dots \cup I_{(i,k_i)} \leftarrow T(c_i,s)$
  $V \leftarrow V \cup \{I_{(i,1)},\dots,I_{(i,k_i)}\}$
/* Assign nodes a color */
$c[\varsigma] \leftarrow 0_C,\ c[\tau] \leftarrow 0_C$
for *symbolic interval* $I_{(i,j)}$ **do**
  $c[I_{(i,j)}] \leftarrow c_i$
/* Assign colors a cost */
$w_C[0_C] \leftarrow 0$
for *constraints* c_i **do**
  $w_C[c_i] \leftarrow cost(sc(c_i,s))$
/* Define edges from the start node and edges to the end node */
$E \leftarrow \{(\varsigma,I_{(i,j)}) : leftBound(I_{(i,j)}) = -\infty\}$
$E \leftarrow E \cup \{(I_{(i,j)},\tau) : rightBound(I_{(i,j)}) = +\infty\}$
/* Define every pair of symbolic intervals which should constitute an edge */
for $((\llbracket I_{(i_1,j_1)} \rrbracket^{x\mapsto s}, \llbracket I_{(i_1,j_1)} \rrbracket^{x\mapsto s}) \in$ FindPairs$(V \setminus \{s,t\})$ **do**
  $E \leftarrow E \cup \{(I_{(i_1,j_1)},I_{(i_2,j_2)})\}$
/* Assign each edge a cost */
for $e = (\varsigma,I_{(i,j)}) \in E$ *or* $e = (I_{(i,j)},\tau) \in E$ **do**
  $w_E[e] \leftarrow 0$
for $(I_{(i_1,j_1)},I_{(i_2,j_2)}) \in E$ **do**
  $w_E[(I_{(i_1,j_1)},I_{(i_2,j_2)})] \leftarrow cost(I_{(i_1,j_1)} \mathbin{\overline{\cap}}_{\text{sym}} I_{(i_2,j_2)})$
/* Now we can construct an instance G of NonLocalIcsShortestPath for which we produce a solution using Algo */
$G \leftarrow (V,E,\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$
$p := (\varsigma, I_{(i_1,j_1)},\dots,I_{(i_K,j_K)},\tau) \leftarrow$ Algo(G)
/* Now we translate this path back to generate a conflict generalization φ */
$$\varphi \leftarrow \left(\bigwedge_{\substack{i=1 \\ \exists j.I_{(i,j)} \text{ in path } p}}^{m} sc(c_i,s) \right) \wedge \left(\bigwedge_{l=1}^{K-1} I_{(i_l,j_l)} \mathbin{\overline{\cap}}_{\text{sym}} I_{(i_l,j_{l+1})} \right)$$
return φ

Algorithm 2: FINDPAIRS

Input : Intervals $I_1, \ldots, I_n \in \mathbb{I}$
Output: Set P of all pairs (I_{i_1}, I_{i_2}) such that $I_{i_1} \mathbin{\text{⟗}^{>}} I_{i_2}$

```
/* Create list off all interval bounds (and mark wether it is a left
   interval bound or a right interval bound)                          */
```
1 $L \leftarrow (leftBound(I_1), \ldots, leftBound(I_n), rightBound(I_1), \ldots, rightBound(I_n))$
```
/* Sort L in-place.  Then left bounds are always ordered before right
   bounds if they are otherwise equal                                 */
```
2 inPlaceSort(L)
```
/* Initialize set of all found pairs                                  */
```
3 $P \leftarrow \emptyset$
```
/* Initialize set of all "opened" and not yet "closed" intervals      */
```
4 $\mathcal{O} \leftarrow \emptyset$
```
/* iterate through bounds in ascending order                          */
```
5 **for** *bound* b *in* L *in ascending order* **do**
6 **if** b *is left bound of some interval* I_i **then**
```
/* "open" interval I_i                                                */
```
7 $\mathcal{O} \leftarrow \mathcal{O} \cup \{I_i\}$
8 **if** b *is right bound of some interval* I_i **then**
```
/* "close" interval I_i                                               */
```
9 $\mathcal{O} \leftarrow \mathcal{O} \setminus \{I_i\}$
```
/* Every pair in the set O × {I_i} is a candidate                     */
```
10 **for** $I_j \in \mathcal{O}$ **do**
11 **if** $I_j \mathbin{\text{⟗}^{>}} I_i$ **then**
12 $P \leftarrow P \cup \{(I_j, I_i)\}$
13 **return** P

we present Algorithm 3 which also reduces the graph of an instance of NonLocal-IcsShortestPath so that every solution in the reduced graph is class-optimal and it contains at least one solution.
For reasons we explore below, alltough finding an optimal solution to NonLocal-IcsShortestPath cannot be done in polynomial time, it may be advantageous to calculate an optimal solution. Therefore we can reduce the overall runtime, albeit not making it polynomial, of an algorithm which finds an optimal solution by first reducing the graph with Algorithm 3.

In this case we have another requirement for Algorithm 3, namely that it produces a reduced graph in which not only every path is class-optimal, but which contains *every* class-optimal solution of the original graph. Depending on the use-case we may choose to execute or ignore Lines 21-24. This part of the algorithm ensures that the reduced graph contains every class-optimal solution.

The main difference between the simple algorithm proposed in the proof to Theorem 3.6.2 and Algorithm 3 lies in the way how we detect that *a* solution exists. Instead of executing Dijkstra's algorithm in each step, we store and continuously update a set of all reachable nodes (starting from ς). This is done by recursively calling the subroutine updateR (see Line 4) for each (newly) reachable node. Note that updateR, when called on a node v, marks this node and that updateR is only ever called on unmarked nodes. Thus, the recursion adds only linearly many calculation steps and the runtime of Algorithm 3 is polynomial.

Algorithm 3: ShrinkGraph

Input : Instance $P = (V,E,\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$ of NonLocalIcsShortestPath
Output: Shrunk down Instace $P' = (V',E',\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$

/* Create the (empty) subgraph (V',E') */
1 $V' \leftarrow \emptyset$
2 $E' \leftarrow \emptyset$
/* Also create a set R of all reachable nodes (all nodes v for which there exist a path starting from ς to v in subgraph $(V', E' \cap (V' \cap V'))$) */
3 $R \leftarrow \emptyset$
/* We define the following subroutine to update the set of all reachable nodes */
4 **Subroutine** `updateR(`*node* v`)`**:**
5 $R \leftarrow R \cup \{v\}$
6 **for** $(v,w) \in E'$, $w \in V'$, $w \notin R$ **do**
7 updateR(w)
/* Create a list of all nodes and edges of the original graph sorted by their (color-)weight */
8 $L \leftarrow$ sortedList $(V \cup E)$
/* Now add nodes and edges to V' and E' until there is a path from ς to τ in the graph $(V', E' \cap (V' \times V'))$ */
9 **for** $l \in L$ *(in ascending order)* **do**
10 **if** l *is an edge* $l = (v_1,v_2)$ **then**
11 $E \leftarrow E \cup \{l\}$
/* Update reachable nodes */
12 **if** $v_1 \in R$, $v_2 \in V'$, $v_2 \notin R$ **then**
13 updateR(v_2)
14 **if** l *is a node* **then**
15 $V' \leftarrow V' \cup \{l\}$
/* Update reachable nodes */
16 **if** $(\exists v.\ v \in R$ *and* $(v,l) \in E')$ *or* $l = \varsigma$ **then**
17 updateR(l)
/* Is τ reachable? */
18 **if** $\tau \in R$ **then**
/* save current cost of l as the (current) maximum cost of any color or edge added */
19 $m \leftarrow w(l)$ // $w = w_C \circ c$ or $w = w_E$ depending on $l \in V$ or $l \in E$
20 **break**
/* Now add all remaining edges and nodes to V' and E' whose cost are in the same infinity-class as m (or below in the case of $m = w_C(c(s))$). This ensures that the reduced graph not only contains *only* class-optimal solutions, but *every* class-optimal solution. Depending on whether we want to find an optimal or just a class-optimal solution, we thus may choose to ignore Lines 21-24 */
21 **Resume** `iteration of` $l \in L$ `from Line 9`
22 **if** $w(l) \notin [m]$ *and* $w(l) > m$ **then** // $w = w_C \circ c$ or $w = w_E$ depending on $l \in V$ or $l \in E$
23 **break**
24 Apply steps 10 to 17 to l
/* We further restrict the generated subgraph to reachable nodes only */
25 $V' \leftarrow R$
26 $E' \leftarrow E' \cap (R \times R)$
27 **return** $(V',E',\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$

Because we have shown that a class-optimal solution of NonLocalIcsShortestPath is equivalent to a class-optimal solution to ConflictGeneralization and because we have shown that relaxed-NonLocalIcsShortestPath is easy, we have shown that relaxed-ConflictGeneralization is easy: A class-optimal solution to MinimalOrderedCover can be computed efficiently in polynomial time.

As mentioned before, once we have reduced the graph we still have to find a path from the start node to the end node. A naïve approach to this problem is simply to traverse an arbitrary edge, beginning with the start node, until we have reached the end. If we arrive at a dead-end we backtrack and try again with a different edge. However the worst-case runtime of such an algorithm is exponential, thus rendering the whole reduction step from above somewhat useless in terms of the desired (polynomial) runtime. In another approach we might use Dijkstra's algorithm to find a path as we have described above. However we may use the actual costs associated with the nodes and edges rather than setting them to zero. This has the added benefit that it searches heuristically for less than arbitrary costly paths. Of course local and global optima do not generally coincide in the given graph which is the base of greedy algorithms such as Dijkstra's algorithm, so this strategy might improve average costs while not guaranteeing minimal costs. Because we know that the given (shrunk down) graph is a DAG, we can modify Dijkstra's Algorithm to optimize for this circumstance. We use the fact that we can find a *topological sorting*[Knu97] of the nodes of any finite directed acyclic graph:

Definition 3.6.2 (Topological Sorting)**.** *Let $G = (V,E)$ be a finite directed acyclic graph with vertices V and edges $E \subseteq V \times V$. A* topological sorting *of the nodes of G is an enumeration $\sigma = (v_1, \ldots, v_{|V|})$ of its nodes such that for every edge $(u,v) \in E$ it holds that u is listed before v in σ.*

Lemma 3.6.3. *Every finite directed acyclic graph has a topological sorting.*

Proof. Let $G = (V,E)$ be a finite directed acyclic graph. W.l.o.g. let $V \neq \emptyset$. Assume G has no node without an incoming edge. Let $v_1 \in V$ be arbitrary. By iteratively traversing edge $(v_{i+1},v_i) \in E$, which is always guaranteed to exist, backwards, we can construct a path $p = (v_n, \ldots, v_1)$ of arbitrary length n. Then either G has a cycle or V is infinite, which is a contradiction. Thus, G has a node $v_1 \in V$ which has no incoming edges. Let $G' = (V',E')$ be the maximal subgraph of G which does not contain node v_1, i.e. remove node v_1 and every edge adjacent to v_1 from G to construct G'. If $\sigma' = (v_2, \ldots, v_{|V|})$ is a topological sorting of G' then $\sigma = (v_1, v_2, \ldots, v_{|V|})$ is a topological sorting of G. Because an empty graph trivially has a topological sorting and because G' has exactly one node less than G we can derive by induction that G has a topological sorting.

□

Kahn's Algorithm (Algorithm 4) computes a topological sorting of finite DAGs in polynomial time [Kah62]. We can thus modify Dijkstra's algorithm as shown in Algorithm 5 to find a path from ς to τ that is heuristically better than arbitrary paths from ς to τ.

As we have mentioned above, there are good reasons to generate an optimal solution to NonLocalIcsShortestPath rather than just a class-optimal solution, and subsequently an optimal solution to ConflictGeneralization despite the added

Algorithm 4: KahnsAlgorithm [Kah62]

Input : finite DAG $G = (V,E)$ with vertices V and edges E
Output: A topological sorting $(v_1, \ldots, v_{|V|})$ of the nodes of G

```
1 Initialize empty list L;
2 while V ≠ ∅ do
3 |   Let v be a vertex in V with no incoming edges;
4 |   Remove every edge adjacent to v from E;
5 |   Remove v from V;
6 |   Append v to end of L;
7 return L;
```

computational cost. The algorithms proposed in this thesis are not running in isolation but rather just represent one of many steps in the overall algorithm which determines wether a given formula is satisfiable or not. The conflict generalization we compute is used in turn to further advance the overall computation. The benefit of finding a class-optimal solution in less time than is needed for finding an optimal solution might be dwarfed by the resulting computational overhead if we consider the overall performance.

Thus, we also present another algorithm (Algorithm 6) which can find for a (reduced) instance of NonLocalIcsShortestPath the optimal path (in exponential time). Here, Algorithm 6 traverses each possible path from ς to τ and returns the shortest.

To sum up: We have presented two procedures by which an instance of ConflictGeneralization can be solved, one of which produces a class-optimal solution in polynomial time and one of which produces an optimal solution in exponential time (see Algorithms 7 and 8 respectively).

Algorithm 5: GreedyNlspSolver

Input : (reduced) instance $P = (V,E,\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$ of NonLocalIcsShortestPath

Output: path from ς to τ

```
   /* Calculate a topological sorting of nodes V                          */
1  T ← KahnsAlgorithm(V,E);
   /* Initialize distance map                                             */
2  dist[ς] ← 0;
3  dist[v] ← ⊥ for v ∈ V \ {ς};
   /* Initialize predecessors                                             */
4  pred[v] ← ⊥ for v ∈ V;
   /* Initialize color map (colors used in implicit predecessor path
      including respective node itself)                                   */
5  clrs[v] ← ⊥ for v ∈ V \ {ς};
6  clrs[ς] ← {c(ς)};
   /* Update distance and predecessor values in topoligical order         */
7  for v ∈ T in topological order do
       /* dist[v] = ⊥ can only occur, when ς has incoming edges           */
8      if dist[v] ≠ ⊥ then
9          for (v,w) ∈ E do
               /* Calculate distance of source to w if we traverse over v */
10             if c(w) ∈ clrs[v] then
11                 cost_(w over v) ← dist[v] + w_E(v,w);
12                 clrs_(w over v) ← clrs[v];
13             else
14                 cost_(w over v) ← dist[v] + w_E(v,w) + w_C(c(w));
15                 clrs_(w over v) ← clrs[v] ∪ {c(w)};
16             /* update predecessor/distance/color-set of node if
                  neccessary                                              */
17             if dist[w] = ⊥ or dist[w] > cost_(w over v) then
18                 dist[w] ← cost_(w over v);
19                 clrs[w] ← clrs_(w over v);
20                 pred[w] ← v;
   /* Now a path from ς to τ is encoded by pred (in reverse order)        */
   /* Extract path p                                                      */
21 p ← empty list;
22 v ← pred[τ];
23 while v ≠ ς do
24     Push v in front of p;
25 return p
```

Algorithm 6: OptimalNlspSolver

Input : (reduced) instance $P = (V,E,\varsigma,\tau,C,c,W,+,\leq,w_E,w_C)$ of NonLocalIcsShortestPath
Output: path from ς to τ of minimal cost, i.e. an optimal solution of P

```
/* Initialize current best path                                    */
opt ← ⊥;
/* Define a recursive function which traverses the graph           */
Subroutine traverse(path v_0,...,v_n from v_0 = ς to v_n):
    /* Abort this route if the current optimum is already better than
       the (prefix-)path which is given here                       */
    if opt ≠ ⊥ and w(v_0,...,v_n) ≥ w(opt) then
        return;
    /* Update optimal path                                         */
    if v_n = t then
        opt ← (v_0,...,v_n);
    else
        /* Traverse all descendants                                */
        for (v_n,v_{n+1}) ∈ E do
            traverse(v_0,...,v_n,v_{n+1});
;
/* Now we can calculate an optimal path by traversing, starting with ς
   */
traverse(ς);
/* opt now is an optimal path from ς to τ                          */
return opt
```

Algorithm 7: FindClassOptimalConflictGeneralization

Input : Instance I of ConflictGeneralization
Output: Class-optimal solution of instance I

```
Subroutine SolveNLSP(Instance P of NonLocalIcsShortestPath):
    P ← ShrinkGraph(P);
    return GreedyNlspSolver(P);
return SolveConflictGeneralization(I,SolveNLSP)
```

Algorithm 8: FindOptimalConflictGeneralization

Input : Instance I of ConflictGeneralization
Output: Class-optimal solution of instance I

```
Subroutine SolveNLSP(Instance P of NonLocalIcsShortestPath):
    P ← ShrinkGraph(P);
    return OptimalNlspSolver(P);
return SolveConflictGeneralization(I,SolveNLSP)
```

Chapter 4

Conclusion

4.1 Summary

The aim of this thesis was to develop algorithms for the generalization of conflicts for quadratic real-arithmetic constraints in the SMT Solver SMT-RAT. To do so we first identified a connection between conflicts and covers of the real number line by intervals. We found a sufficient and necessary condition under which intervals cover $\mathbb{R}$. The advantages of this condition were twofold: It could easily be encoded into a SMT compliant formula. Furthermore, we found a natural way to encode every possible atom of said formula in a graph in which certain paths correspond to said covers and thus to conflict generalizations and vice versa. Thus, we reduced the problem of finding a conflict generalization to a path-finding problem. We also developed a heuristic with which the conflict generalizations could be assigned a cost. We started by stating some assumptions the cost function should reasonably uphold and derived a naturally arising cost model based on them. Using this cost-model we weighted edges and nodes of the aforementioned graph. Each node of the graph was also assigned a color and the total weight of a path was the sum of all costs of its traversed edges and nodes so that the cost of nodes of the same color were only accounted for once. Doing so, the resulting path-finding problem could not simply make use of algorithms such as Dijkstra's algorithm which runs in polynomial time, but which only works under the assumption that local and global optima coincide. The cost of a newly added node to a path however does depend on the previous state of the path. We therefore found this path-finding algorithm not to be solvable in polynomial time (unless P = NP). We could remedy this by reducing the graph to a subgraph which contains all optimal solutions and by relaxing the optimization problem of finding an optimal path to finding a *class-optimal* path which could be interpreted as "close to optimal". Overall, we presented two algorithms to generate conflict generalizations. One of which could produce a class-optimal solution in polynomial time and one of which could produce an optimal solution in exponential time.

4.2 Future work

The most obvious and interesting question arising from this thesis is how the proposed algorithms compete with existing algorithms for conflict generalizations implemented

in SMT-RAT. As we have taken steps to optimize our proposals for a specific case, namely for conflicts in constraints in which a variable of interest is only at most quadratic, it is very likely that the proposed algorithms run much faster, especially the algorithm which finds class-optimal conflict generalizations in polynomial time. More importantly however is the question as to how well the overall runtime of SMT-RAT differs when using the standard strategy or one of the strategies proposed here. A comparison of all three possibilities is therefore of interest and could yield some interesting results.

The results of this thesis could in principle be adopted to cubic and quartic real-arithmetic conflicts. Our work heavily relied on the fact that there exists a closed formula for zeros of (at most) quadratic polynomials. This is even the case for cubic and quartic polynomials, but no such closed formulas exist for quintic polynomials or polynomials of even higher degrees[Ayo80]. Therefore, we could extend the definitions of a type of a constraint (and a partial variable assignment) and their side conditions for at most quartic constraints and thus make use of other results of this thesis to derive a conflict generalization, but with one caveat. We relied on square root expressions and Virtual Substitution as defined by [CÁ11][Cor10][Cor17] to express formulas which use the zeros of quadratic polynomials. The definition of these however cannot express nested roots or roots of higher order which are part of the formulas for zeros of cubic and quartic polynomials. To extend the results of this thesis to those cases then would fist require generalizing VS and (square) root expressions. Even then, this extension might face some difficulties by the sheer amount of different side conditions and the length of the formulas for zeros of cubic and quartic polynomials.

Bibliography

[Ayo80] Raymond G. Ayoub. Paolo ruffini's contributions to the quintic. *Archive for History of Exact Sciences*, 23(3):253–277, Sep 1980.

[BB09] Robert Brummayer and Armin Biere. Boolector: An efficient smt solver for bit-vectors and arrays. In Stefan Kowalewski and Anna Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 174–177, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[BDS02] Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to sat. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification*, pages 236–249, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[BJG09] Jørgen Bang-Jensen and Gregory Z. Gutin. *Classes of Digraphs*, pages 31–86. Springer London, London, 2009.

[BT18] Clark Barrett and Cesare Tinelli. *Satisfiability Modulo Theories*, pages 305–343. Springer International Publishing, Cham, 2018.

[CÁ11] Florian Corzilius and Erika Ábrahám. Virtual substitution for smt-solving. In Olaf Owe, Martin Steffen, and Jan Arne Telle, editors, *Fundamentals of Computation Theory*, pages 360–371, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[CLJÁ12] Florian Corzilius, Ulrich Loup, Sebastian Junges, and Erika Ábrahám. Smt-rat: An smt-compliant nonlinear real-arithmetic toolbox. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing – SAT 2012*, pages 442–448, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Cor10] Florian Corzilius. Virtual substitution in smt solving, 2010. diplomathesis. RWTH Aachen University.

[Cor17] Florian Corzilius. *Integrating Virtual Substitution into Strategic SMT Solving*. dissertation, RWTH Aachen University, Aachen, NRW, Germany, 2016 & 2017.

[DdM06] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for dpll(t). In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification*, pages 81–94, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, jul 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, jul 1960.

[GF13] Saul I. Gass and Michael C. Fu, editors. *Dijkstra's Algorithm*, pages 428–428. Springer US, Boston, MA, 2013.

[GHN+04] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Dpll(t): Fast decision procedures. In Rajeev Alur and Doron A. Peled, editors, *Computer Aided Verification*, pages 175–188, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Kah62] A. B. Kahn. Topological sorting of large networks. *Commun. ACM*, 5(11):558–562, nov 1962.

[Knu97] D.E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley Professional, 3rd edition, 1997. Section 2.2.3.

[MR02] Leonardo De Moura and Harald Rueß. Lemmas on demand for satisfiability solvers. In *In Proceedings of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT*, pages 244–251, 2002.

[MRG79] David S. Johnson Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA, 1979.

[Whi99] Gretchen Wilke Whipple. *Totally Ordered Monoids*. dissertation, Louisiana State University and Agricultural & Mechanical College, 1999.