Friedrich-Alexander-Universität
Erlangen-Nürnberg

Chair for Computer Science 8
Theoretical Computer Science

SMT-basierte Analyse von Konsistenzregeln für digitale Formulare

Bachelorarbeit in der Informatik

Julian Pollinger

Betreuer:

Prof. Dr. Lutz Schröder Merlin Humml

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Erlangen, 9. Februar 2023

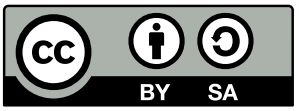
Disclaimer

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 9. Februar 2023 _____

Julian Pollinger

Lizensierung



Inhaltsverzeichnis

1 Einleitung

2 Hintergrund

2.1 Das Projekt

Dieser Arbeit liegt ein Open Source Projekt namens Rita[1] zugrunde, das ich bei meinem Arbeitgeber "Educorvi GmbH & Co KG"[2] entwickle. Grundgedanke dieses Projekts ist es, eine Lösung für regelbasierte Abarbeitung von Formularen zu bieten. Kern dieses Projekts ist eine syntaktisch auf JSON basierende Sprache, in der die Regeln formuliert werden können. Weiterhin enthält es eine TypeScript Implementierung um in dieser Sprache formulierte Regeln mit Formulardaten auswerten zu können, sowie einen Webserver, der diese Funktionen über eine HTTP API zur Verfügung stellt.

Im Rahmen dieser Arbeit wurde weiterhin ein Tool entwickelt, um Regeln dieser Sprache in SMT zu übersetzen, um sie dann anschließend in dieser Form mit einem SMT Solver analysieren zu können, beispielsweise auf Erfüllbarkeit.

2.2 Aktueller Umfang (Stand November 2022)

Aufgrund des Verwendungszweck des Projekts, muss die Sprache alle Funktionen unterstützen, die bei der Bearbeitung von Formularen notwendig sein können. Im Folgenden soll der aktuelle Umfang der Sprache zunächst grob umrissen werden, eine formale Definition der Syntax und Semantik folgen in Kapitel 3.

Konstanten In den Regeln können Konstanten definiert werden.

Atome Eine der grundlegendsten Voraussetzungen ist, Daten aus den Formularen auslesen zu können. Dies wird durch sogenannte Atome realisiert. Die Daten können entweder vom Typ String, Number oder Boolean sein. Zudem können Datumsangaben eingelesen werden. Diese müssen dafür als String im ISO 8601 Format [1] angegeben werden (z. B. $2022{-}12{-}04T09{:}38{:}02.375Z$).

Logische Verknüpfungen Um Daten vom Typ Boolean, die entweder aus Atomen oder aus Funktionen der Sprache stammen, kombinieren zu können, gibt es die logischen Verknüpfungen $\wedge$, $\vee$ und $\neg$.

Vergleiche Weiterhin können Daten aller Typen verglichen werden.Vergleiche zwischen unterschiedlichen Datentypen evaluieren immer zu $\bot$.

Berechnungen Es ist möglich, simple Berechnungen durchzuführen. Unterstützte Berechnungen sind Addition, Subtraktion, Multiplikation, Division und Modulo.

[1] `https://github.com/educorvi/rita`
[2] `https://educorvi.de`

Berechnungen mit Datumsangaben Dies umfasst im wesentlichen zwei Funktionen:

1. Subtraktion zweier Datumsangaben, um das Zeitintervall zwischen diesen zu berechnen.
2. Addition oder Subtraktion eines Zeitintervalls auf/von eine/r Datumsangabe.

Quantoren Enthalten die übergebenen Daten ein Array von Werten/Objekten, kann überprüft werden, ob eine Bedingung für alle oder mindestens ein Element dieses Arrays gilt.

Macros Aktuell gibt es zwei Macros:

1. now: Kann in einer Regel angegeben werden um zum Zeitpunkt der Evaluation durch die aktuelle Kombination aus Datum/Uhrzeit ersetzt zu werden.
2. length: Wertet zur Länge des übergebenen Strings aus.

Plugins Plugins dienen dazu, dem Nutzer die Möglichkeit zu geben, Regeln mit eigenen Funktionen zu erweitern. Das Konzept dahinter ist, dass das Plugin die Daten zu Beginn der Auswertung übergeben bekommt, um dann die Daten mit eigenen Ergebnissen anreichern zu können, die dann den weiteren Funktionen zur Verfügung stehen.

Hinweis

Da RITA stetig weiterentwickelt wird, wird für diese Arbeit die Version der Sprache auf 5.0_alpha unter Ausschluss von Quantoren, Macros und Plugins festgesetzt.

3 Ritas Sprache

In diesem Kapitel soll nun die Sprache beschrieben werden, in der Regeln für Rita formuliert werden können.

3.1 Syntax

Die Sprache basiert auf JSON und ist durch ein JSON Schema definiert [2, Einstieg ist schema.json].

Gesendete Daten werden immer gegen ein sogenanntes Ruleset ausgewertet. Das ist eine Menge von Regeln, die diese Daten erfüllen müssen. Eine solche Regel kann in Backus Naur Form wie folgt beschrieben werden:

$$
\begin{aligned}
Rule &::= Formula \\
Formula &::= a \mid Connectives \mid Comparison && (a \in \mathcal{A}) \\
Connectives &::= and(Formula, Formula) \mid or(Formula, Formula) \mid not(Formula) \\
Comparison &::= comp_{=}(CA, CA) \mid comp_{<}(CA, CA) \mid comp_{>}(CA, CA) \mid \\
&\quad comp_{<=}(CA, CA) \mid comp_{>=}(CA, CA) \\
CA &::= a \mid c \mid Calculation \mid DateCalculation && (a \in \mathcal{A}, c \in \mathcal{C}) \\
Calculation &::= calc_{+}(CA, CA) \mid calc_{-}(CA, CA) \mid \\
&\quad calc_{*}(CA, CA) \mid calc_{\div}(CA, CA) \mid calc_{\%}(CA, CA) \\
DateUnit &::= seconds \mid minutes \mid hours \mid days \mid months \mid years \\
DateCalculation &::= dateCalc_{+}(CA, CA, DateUnit) \mid dateCalc_{-}(CA, CA, DateUnit)
\end{aligned}
$$

Sei R ein Ruleset, und r eine Regel. Sei weiterhin

$$
\begin{aligned}
S &= \{Bool, String, Date, Number, DurationType\} \\
S_0 &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
F = \{ & \\
& comp_{Bool,=} : Bool \times Bool \to Bool, \\
& comp_{Bool,<} : Bool \times Bool \to Bool, \\
& comp_{Bool,>} : Bool \times Bool \to Bool, \\
& comp_{Bool,<=} : Bool \times Bool \to Bool, \\
& comp_{Bool,>=} : Bool \times Bool \to Bool, \\
& comp_{String,=} : String \times String \to Bool, \\
& comp_{String,<} : String \times String \to Bool, \\
& comp_{String,>} : String \times String \to Bool, \\
& comp_{String,<=} : String \times String \to Bool, \\
& comp_{String,>=} : String \times String \to Bool, \\
& comp_{Date,=} : Date \times Date \to Bool, \\
& comp_{Date,<} : Date \times Date \to Bool, \\
& comp_{Date,>} : Date \times Date \to Bool, \\
& comp_{Date,<=} : Date \times Date \to Bool, \\
& comp_{Date,>=} : Date \times Date \to Bool, \\
& comp_{Number,=} : Number \times Number \to Bool, \\
& comp_{Number,<} : Number \times Number \to Bool, \\
& comp_{Number,>} : Number \times Number \to Bool, \\
& comp_{Number,<=} : Number \times Number \to Bool, \\
& comp_{Number,>=} : Number \times Number \to Bool, \\
& and : Bool \times Bool \to Bool, \\
& or : Bool \times Bool \to Bool, \\
& not : Bool \to Bool \\
& calc_{+} : Number \times Number \to Number, \\
& calc_{-} : Number \times Number \to Number, \\
& calc_{\div} : Number \times Number \to Number + Error, \\
& calc_{\%} : Number \times Number \to Number + Error, \\
& dateCalc_{(Date \times Date),-} : Date \times Date \times DurationType \to Number, \\
& dateCalc_{(Date \times Number),+} : Date \times Number \times DurationType \to Date, \\
& dateCalc_{(Date \times Number),-} : Date \times Number \times DurationType \to Date, \\
& dateCalc_{(Number \times Date),+} : Number \times Date \times DurationType \to Date, \\
& getTime : Date \to Number, \\
& getTime^{-1} : Number \to Date, \\
& convertInterval : Number \times DurationType \times DurationType \to Number \\
\} &
\end{aligned}
$$

Dann hat Rita die Signatur $\Sigma = (S_0, S, F)$

3.2 Semantik

Die Interpretation eines Rulesets geschieht rekursiv nach folgendem Ablauf:

$$
\begin{aligned}
\eta &= \text{Umgebung aus den gesendeten Daten in Form von Key-Value Paaren} \\
\eta(x) &= \text{Der Wert für das Datum mit dem Key } x \text{ in den Daten} \\
\varepsilon &= \text{Error} \\
C &= \text{Menge der Konstanten} \\
\mathfrak{M}[\![R]\!]\eta &\in \{\top, \bot, \varepsilon\} \\
\mathfrak{M}[\![r]\!]\eta &\in \{\top, \bot, \varepsilon\} \\
\mathfrak{M}[\![R]\!]\eta &= \begin{cases} \varepsilon & \text{wenn } \exists r \in R.\mathfrak{M}[\![r]\!]\eta = \varepsilon \\ \top & \text{sonst wenn } \forall r \in R.\mathfrak{M}[\![r]\!]\eta = \top \\ \bot & \text{sonst} \end{cases} \\
\mathfrak{M}[\![f(arg_1, ..., arg_n)]\!]\eta &= \begin{cases} \varepsilon & \text{wenn } \mathfrak{M}[\![arg_i]\!]\eta = \varepsilon \\ \mathfrak{M}[\![f]\!](\mathfrak{M}[\![arg_1]\!]\eta, ..., \mathfrak{M}[\![arg_n]\!]\eta) & \text{sonst} \end{cases} \\
\mathfrak{M}[\![x]\!]\eta &= \begin{cases} \eta(x) & \text{wenn } x \text{ in } \eta \text{ definiert} \\ x & \text{sonst wenn } x \in C \\ \varepsilon & \text{sonst} \end{cases}
\end{aligned}
$$

Wobei für
S die Menge aller Strings
D die Menge aller Zeitstempel
um eine Umrechnungsmatrix für Zeitintervalle (siehe Appendix A.1)

die Symbole der Signatur durch $\mathfrak{M}$ wie folgt interpretiert werden:

$$
\begin{aligned}
\text{Domain } M &= \{\top, \bot, S, D, \mathbb{R}\} \\
\mathfrak{M}[\![Bool]\!] &= \{\top, \bot\} \\
\mathfrak{M}[\![String]\!] &= S \\
\mathfrak{M}[\![Date]\!] &= D \\
\mathfrak{M}[\![Number]\!] &= \mathbb{R} \\
\mathfrak{M}[\![DurationType]\!] &= \{seconds, minutes, hours, days, months, years\}
\end{aligned}
$$

Die Funktionen *and* und *or* werden faul ausgewertet, bei Vergleichen von Booleans gilt $\top > \bot$.

$$\mathfrak{M}[\![and]\!](a,b) = \begin{cases} b & \text{wenn } a = \top \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![or]\!](a,b) = \begin{cases} \top & \text{wenn } a = \top \\ b & \text{sonst} \end{cases}$$

$$\begin{aligned}
\mathfrak{M}[\![not]\!](a) &= \neg a \\
\mathfrak{M}[\![comp_{Bool,=}]\!](a,b) &= (a \wedge b) \vee (\neg a \wedge \neg b) \\
\mathfrak{M}[\![comp_{Bool,<}]\!](a,b) &= \neg a \wedge b \\
\mathfrak{M}[\![comp_{Bool,>}]\!](a,b) &= a \wedge \neg b \\
\mathfrak{M}[\![comp_{Bool,<=}]\!](a,b) &= \mathfrak{M}[\![comp_{Bool,<}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Bool,=}(a,b)]\!] \\
\mathfrak{M}[\![comp_{Bool,>=}]\!](a,b) &= \mathfrak{M}[\![comp_{Bool,>}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Bool,=}(a,b)]\!]
\end{aligned}$$

Bei Vergleichen zwischen Strings wird die lexikalische Ordnung verwendet.

$$\mathfrak{M}[\![comp_{String,=}]\!](a,b) = \begin{cases} \top & \text{wenn } a \text{ zeichenweise identisch zu } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{String,<}]\!](a,b) = \begin{cases} \top & \text{wenn } a \text{ lexikalisch kleiner als } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{String,>}]\!](a,b) = \begin{cases} \top & \text{wenn } a \text{ lexikalisch größer als } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\begin{aligned}
\mathfrak{M}[\![comp_{String,<=}]\!](a,b) &= \mathfrak{M}[\![comp_{String,<}(a,b)]\!] \vee \mathfrak{M}[\![comp_{String,=}(a,b)]\!] \\
\mathfrak{M}[\![comp_{String,>=}]\!](a,b) &= \mathfrak{M}[\![comp_{String,>}(a,b)]\!] \vee \mathfrak{M}[\![comp_{String,=}(a,b)]\!]
\end{aligned}$$

Berechnungen und Vergleiche mit Zahlen folgen der für reelle Zahlen üblichen Interpretation.

$$\mathfrak{M}[\![comp_{Number,=}]\!](a,b) = \begin{cases} \top & \text{wenn } (a = b) \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Number,<}]\!](a,b) = \begin{cases} \top & \text{wenn } (a < b) \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Number,>}]\!](a,b) = \begin{cases} \top & \text{wenn } (a > b) \\ \bot & \text{sonst} \end{cases}$$

$$\begin{aligned}
\mathfrak{M}[\![comp_{Number,<=}]\!](a,b) &= \mathfrak{M}[\![comp_{Number,<}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Number,=}(a,b)]\!] \\
\mathfrak{M}[\![comp_{Number,>=}]\!](a,b) &= \mathfrak{M}[\![comp_{Number,>}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Number,=}(a,b)]\!] \\
\mathfrak{M}[\![calc_{+}]\!](a,b) &= a + b \\
\mathfrak{M}[\![calc_{-}]\!](a,b) &= a - b \\
\mathfrak{M}[\![calc_{*}]\!](a,b) &= a * b
\end{aligned}$$

$$\mathfrak{M}[\![calc_{\div}]\!](a,b) = \begin{cases} a \div b & \text{wenn } b \text{ nicht 0 ist} \\ \varepsilon & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![calc_{\%}]\!](a,b) = \begin{cases} a - (\lfloor \frac{a}{b} \rfloor * b) & \text{wenn } b \text{ nicht 0 ist} \\ \varepsilon & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Date,=}]\!](a,b) = \begin{cases} \top & \text{wenn der Zeitpunkt } a \text{ zeitgleich zum Zeitpunkt } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Date,<}]\!](a,b) = \begin{cases} \top & \text{wenn der Zeitpunkt } a \text{ vor dem Zeitpunkt } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Date,>}]\!](a,b) = \begin{cases} \top & \text{wenn der Zeitpunkt } a \text{ nach dem Zeitpunkt } b \text{ ist} \\ \bot & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![comp_{Date,<=}]\!](a,b) = \mathfrak{M}[\![comp_{Date,<}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Date,=}(a,b)]\!]$$

$$\mathfrak{M}[\![comp_{Date,>=}]\!](a,b) = \mathfrak{M}[\![comp_{Date,>}(a,b)]\!] \vee \mathfrak{M}[\![comp_{Date,=}(a,b)]\!]$$

Bei Berechnungen mit Zeitangaben werden Datumsangaben zunächst in die seit dem 1. Januar 1970, 00:00:00 Uhr GMT vergangene Zeit in Millisekunden umgerechnet. Mit der resultierenden Zahl kann dann der Abstand zwischen zwei Zeitpunkten durch normale Subtraktion der beiden Zahlen berechnet werden, die Addition und Subtraktion eines Zeitintervalls auf eine Datumsangabe erfolgt durch Addition oder Subtraktion des Intervalls in Millisekunden und anschließendes Anwenden der Funktion $getTime^{-1}$ um aus der Zahl wieder eine Datumsangabe zu erhalten.

$$\mathfrak{M}[\![getTime]\!](a) = \text{Von Jan 1, 1970, 00:00:00.000 GMT bis a verstrichene Millisekunden}$$

$$\mathfrak{M}[\![getTime^{-1}]\!](a) = \text{Inverse Funktion zu getTime}$$

$$\mathfrak{M}[\![convertInterval]\!](n,t_1,t_2) = \begin{cases} n * um[t_1][t_2] & \text{wenn } t_1 \text{ größer als } t_2 \text{ ist} \\ n \div um[t_2][t_1] & \text{sonst} \end{cases}$$

$$\mathfrak{M}[\![calc_{(Date\times Date),-}]\!](a,b,t) = \mathfrak{M}[\![convertInterval(calc_-(getTime(b), getTime(a)), \text{"}milliseconds\text{"}, t)]\!]$$

$$\mathfrak{M}[\![calc_{(Date\times Number),+}]\!](a,b,t) = \mathfrak{M}[\![getTime^{-1}(calc_+(getTime(a), convertInterval(b,t,\text{"}milliseconds\text{"})))]\!]$$

$$\mathfrak{M}[\![calc_{(Date\times Number),-}]\!](a,b,t) = \mathfrak{M}[\![getTime^{-1}(calc_-(getTime(a), convertInterval(b,t,\text{"}milliseconds\text{"})))]\!]$$

$$\mathfrak{M}[\![calc_{(Number\times Date),+}]\!](a,b,t) = \mathfrak{M}[\![calc_{(Date\times Number),+}]\!](b,a,t)$$

3.3 Beispiel

Nun folgt ein simples Beispiel, in welchem das Ruleset nur eine Regel enthält, die überprüfen soll, ob jemand zum Zeitpunkt eines Kaufs mindestens 18 Jahre alt war und nicht angestellt ist. Bei der Evaluation sollen folgende Daten übergeben werden:
geburtsdatum (Datum), **kaufdatum** (Datum), **istAngestellter** (Boolean)

Verwendet man für diese Regel die in Abschnitt 3.2 beschriebene Notation, sieht die Formel wie folgt aus:
$and(not(istAngestellter), comp_{Date,>=}(dateCalc_{(Date\times Date),-}(kaufdatum, geburtsdatum, years), 18))$
Eine Visualisierung der Regel findet sich in Abbildung 3.1. Die hierarchische Struktur einer Regel wird dort gut ersichtlich.

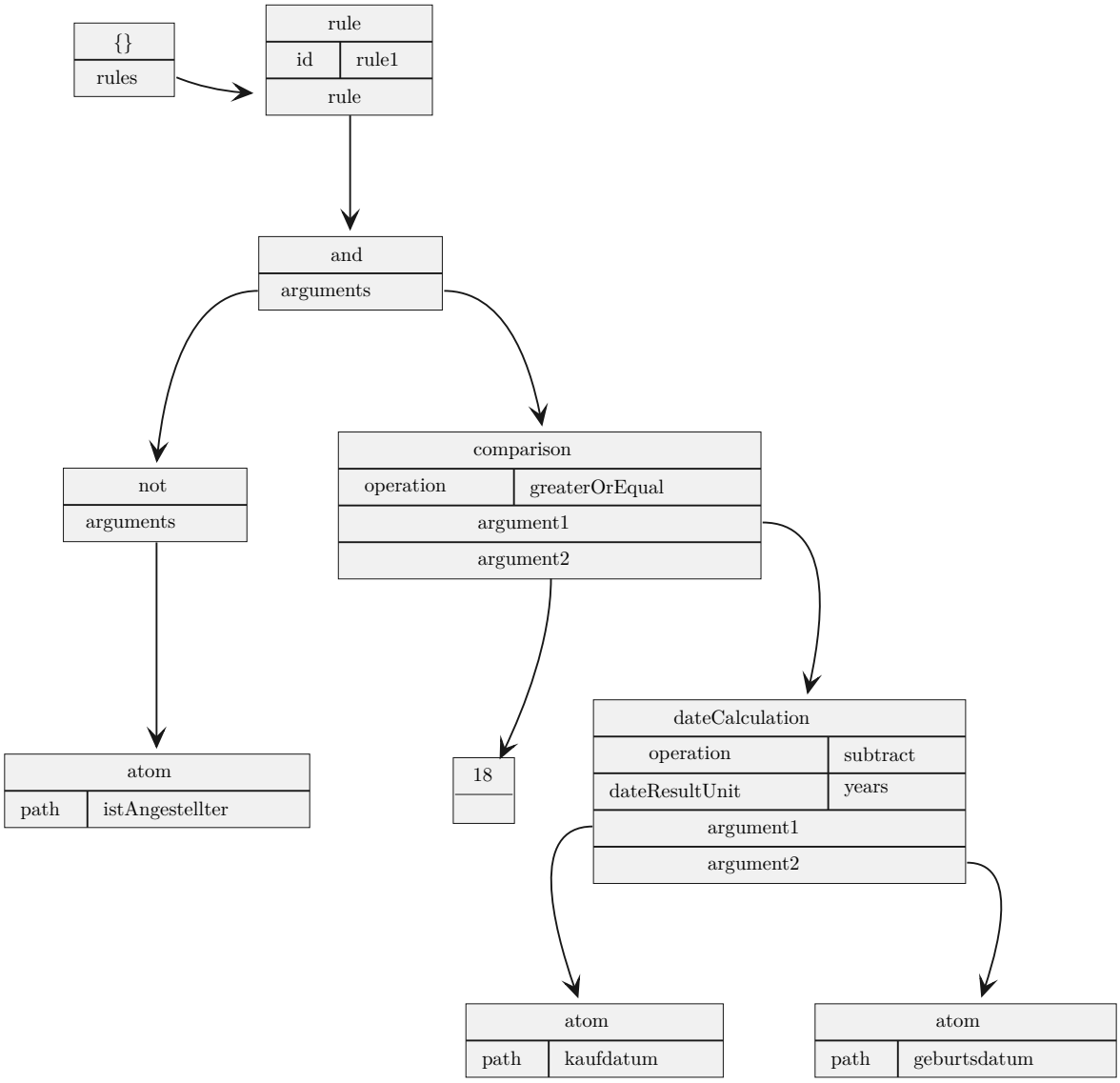


Abbildung 3.1: Visualisierung des Rulesets

Die tatsächliche Umsetzung dieser Regel sähe dann wie folgt aus:

```
{
    "$schema": "https://raw.githubusercontent.com/educorvi/rita/alpha/rita-core/src/s
↪   chema/schema.json",
    "rules": [
        {
            "id": "rule1",
            "rule": {
                "type": "and",
                "arguments": [
                    {
                        "type": "not",
                        "arguments": [
                            {
                                "path": "istAngestellter"
                            }
                        ]
                    },
                    {
                        "type": "comparison",
                        "operation": "greaterOrEqual",
                        "arguments": [
                            {
                                "type": "dateCalculation",
                                "dateResultUnit": "years",
                                "operation": "subtract",
                                "arguments": [
                                    {
                                        "type": "atom",
                                        "path": "kaufdatum",
                                        "isDate": true
                                    },
                                    {
                                        "type": "atom",
                                        "path": "geburtsdatum",
                                        "isDate": true
                                    }
                                ]
                            },
                            18
                        ]
                    }
                ]
            }
        }
    ]
}
```

Zunächst mag diese Schreibweise etwas sperrig wirken, sie ist jedoch für den Laien, dem wir mit RITA das schreiben dieser Regeln erlauben möchten, vermutlich intuitiver zu lesen. Außerdem kann JSON für unsere Zwecke ein guter Kompromiss zwischen menschlicher und maschineller Lesbarkeit.

4 SMT basierte Analyse von Rulesets

Wenn man nun Regeln in dieser Sprache formuliert, bietet es sich an eine Möglichkeit zu schaffen, zu überprüfen ob eine Menge von Regeln überhaupt erfüllbar ist, oder ob mache Regeln andere implizieren und damit obsolet machen.

4.1 Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT) bezeichnet das Problem der Erfüllbarkeit einer Formel in Logik erster Stufe vor dem Hintergrund bestimmter Theorien [3]. Um die Entscheidbarkeit dieses Problems gewährleisten zu können, werden nur Fragmente der Logik erster Ordnung zugelassen, beschränkt durch besagte Theorien, die die Interpretation von Symbolen einschränken [3]. Ein Beispiel dafür ist die "Theory of Reals" nach dem SMT-LIB Standard [4], die unter anderem die Interpretation des Funktionensymbols $*$ auf die Multiplikation der beiden übergebenen reellen Zahlen beschränkt.

4.2 SMT-LIB

Standardisierung In-/Output, Theorien Prefix Notation

4.3 Übersetzung

4.3.1 Erfüllbarkeit

Die Übersetzung eine Rulesets nach SMT-LIB soll nun an der folgenden, durch das Tool Rita-SMT erstellten Übersetzung des Beispiels aus Abschnitt 3.3 erläutert werden:

```
1  (set-logic QF_SNIRA)
2  (set-option :produce-assignments true)
3  (set-option :produce-models true)
4  (define-fun % ((a Real) (b Real)) Real (- a (* (to_int (/ a b)) b)))
5  (define-fun <=s ((a String) (b String)) Bool (str.<= a b))
6  (define-fun <s ((a String) (b String)) Bool (and (str.<= a b) (not (= a b))))
7  (define-fun >s ((a String) (b String)) Bool (not (and (str.<= a b) (not (= a b)))))
8  (define-fun >=s ((a String) (b String)) Bool (or (>s a b) (= a b)))
9  (declare-const istAngestellter Bool)
10 (declare-const kaufdatum Real)
11 (declare-const geburtsdatum Real)
12 (assert (and (not istAngestellter) (>= (/ (- (* kaufdatum 1000) (* geburtsdatum 1000))
   ↪  31536000000) 18.0)))
```

Logik In Zeile 1 wird zunächst die vom SMT Solver zu verwendende Logik festgesetzt. Für Rita Rulesets ist das QF_SNIRA. QF schließt die Verwendung von Quantoren aus und ermöglicht

so eine effizientere Lösung der SMT Problematik. S ist ein für cvc5 spezifischer Syntax und inkludiert die Theory of Strings in die Logik, um String Vergleiche durchführen zu können. NIRA erlaubt die Verwendung von nichtlinearer Arithmetik mit reellen und ganzen Zahlen.

Optionen TODO

Funktionen In den Zeilen 5 bis 9 werden mehrere benötigte Funktionen deklariert. Dies beginnt mit der Funktion %, die die Berechnung von Modulo für reelle Zahlen implementiert. Dies ist notwendig, da SMT-LIB standardmäßig Modulo Berechnungen nur für ganzzahlige Werte spezifiziert. Umgesetzt wird die in Abschnitt 3.2 spezifizierte Modulo Funktion $mod(a, b) = a - (\lfloor \frac{a}{b} \rfloor * b)$. Der Floor Operator wird hierbei durch die to_int Funktion repräsentiert, da die Theory of Reals in SMT-LIB keinen Floor Operator unterstützt und ihre Definitionen identisch sind.
In Zeilen 6 bis 9 finden sich Funktionen zum lexikalischen Vergleichen von Zeichenketten. SMT-LIB kennt hier nur die Funktion str.<=, die ein lexikalische kleiner-gleich überprüft sowie = zur Überprüfung von Gleichheit. In Zeile 6 wird für eine konsistente Benennung zunächst ein Alias für die str.<= Funktion festgelegt, in den Zeilen danach werden die anderen Funktionen zum Vergleichen von Zeichenketten aus den beiden gegebenen abgeleitet.

Konstanten Konstante können, wenn sie kein Datum sind (mehr dazu im Absatz Datum), einfach als Literale übernommen werden.

Atome Im Ruleset vorkommende Atome werden als Konstanten deklariert. So kann, sollte ein Ruleset erfüllbar sein, vom SMT Solver eine Belegung für die Atome ermittelt werden, die das Ruleset erfüllt. Der Typ dieser Konstanten wird aus dem Kontext des Atoms abgeleitet: Befindet es sich in einem logischen Konnektiv, ist es vom Typ Bool, ist es in einem Vergleich, ist es vom Typ Real oder String, je nachdem, ob es mit einer Zahl oder Zeichenfolge verglichen wird. Atome, deren Wert ein Datum ist, wird als Real deklariert (mehr dazu im Absatz Datum). Beispiele hierfür sind in den Zeilen 10 bis 12 zu finden.

Datum Ist ein Atom oder eine Konstante ein Datum, wird es durch einen Real repräsentiert, wobei ein jeder Zeitpunkt durch die Anzahl der seit dem Epoch (1. Januar 1970, 00:00.000 Uhr UTC) [5] bis zu diesem Zeitpunkt verstrichenen Millisekunden beschrieben wird. Dies bietet sich aus mehreren Gründen an: Zum einen entspricht die dem weiter oben beschriebenen Modell der Sprache, zum anderen können so mit wenig Aufwand Probleme mit Zeitzonen vermieden werden, mehr dazu in den folgenden Absätzen.

Regeln Jede Regel wird durch eine assert Anweisung in SMT-LIB dargestellt. Ein Beispiel dafür ist in Zeile 13 zu sehen. Dem Solver wird somit mitgeteilt, dass diese Regel erfüllt sein muss, damit das gesamte Ruleset erfüllbar ist.

Logische Konnektive Die Übersetzung von logischen Konnektiven ist trivial, da diese in der Core Theory von SMT-LIB enthalten sind.
$var_1 \wedge var_2$ beispielsweise wäre in SMT-LIB (and var_1 var_2).

Vergleiche Auch Vergleiche sind in SMT-LIB für alle von uns benötigten Datentypen außer Zeitstempeln standardmäßig vorhanden. Da wir Zeitstempel aber als Zahlen darstellen, können

wir diese einfach ohne weiteren Aufwand vergleichen, da sich FORMULIERUNG. Insbesondere liegt hier auch der Grund, warum Zeitstempel nicht als String repräsentiert werden. So beschreiben
2023−01−22T11:07:36.878Z und
2023−01−22T06:07:36.878−05:00
den selben Zeitpunkt, ihre Repräsentationen als Zeichenketten sind aber nicht identisch.
Ein Vergleich, der

Berechnungen mit Zeitstempeln

Error Wie in Abschnitt 3.2 beschrieben, können bei Division und Modulo Fehler auftreten, wenn der Divisor gleich 0 ist. Dieser Fehler wird anschließend entsprechend der Semantik bis zum Level des Rulesets eskaliert. Um dies in SMT-LIB abzubilden, werden die Umstände, unter denen der Fehler auftreten könnte von vornherein ausgeschlossen. Dies ist für unsere Zwecke eine äquivalente Übersetzung, da ein Fehler niemals ein Ruleset erfüllt. Um also Fehler auszuschließen, wird sowohl für Modulo als auch für Division eine zusätzliche Bedingung eingefügt, die festlegt, dass der Divisor nicht 0 sein darf.
Man nehme zum Beispiel die folgende Regel:

```
{
    "$schema": "https://raw.githubusercontent.com/educorvi/rita/alpha/rita-core/src/s
↪   chema/schema.json",
    "rules": [
        {
            "id": "division",
            "rule": {
                "type": "comparison",
                "operation": "smaller",
                "arguments": [
                    5,
                    {
                        "type": "calculation",
                        "operation": "divide",
                        "arguments": [
                            2,
                            {
                                "type": "calculation",
                                "operation": "subtract",
                                "arguments": [
                                    {
                                        "type": "atom",
                                        "path": "number"
                                    },
                                    3
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Dann muss in SMT-LIB sichergestellt werden, dass $\neg(number - 3 = 0)$, wie in Zeile 3 zu sehen:

```
...
(declare-const number Real)
(assert (not (= (- number 3.0) 0)))
(assert (< 5.0 (/ 2.0 (- number 3.0))))
```

5 Evaluation

Zur Evaluation des Tools sollen nun fiktive Regelwerke unter Zuhilfenahme des für diese Arbeit entwickelten Tools in Rita umgesetzt werden.

5.1 Verein

In diesem Szenario soll die gesamte Bandbreite der von Rita unterstützten Funktionen getestet werden.

Szenario

Ein Verein der zur Förderung von wissenschaftlichen Experimenten gegründet wurde, möchte die Ausschüttung seiner Mittel automatisieren. Dazu soll mit Hilfe von Rita anhand von übergebenen Formulardaten überprüft werden, ob eine Auszahlung nach bestimmten Kriterien gerechtfertigt ist.

Übergebene Daten

- stiftung.finanzvolumen: Bool
- projekt.genehmigt: Bool
- projekt.genehmigtAm: Date
- projekt.intialeAusschuettung: Real
- auszahlung.beantragungsdatum: Date
- auszahlung.betrag: Real
- auszahlung.vorangegangene: Real
- auszahlung.empfaenger.istMitglied: Bool

Kriterien

- Es soll nur an Personen ausgezahlt werden, die Mitglied im Verein sind.
- Die Auszahlung muss zu einem der vom Verein genehmigten Projekte erfolgen. Genehmigungen, die älter als 365 Tage sind, sind ungültig.
- Weil der Verein viele Mathematiker als Mitglieder hat, muss der auszuschüttende Betrag stets ein Vielfaches von 3.14 sein.
- Eine Auszahlung darf nicht größer als die initiale Ausschüttung sein, muss aber größer als 0€ sein.

- Um zu verhindern, dass Projekte häufig Geld nachfordern, wird die Anzahl der Nachzahlungen auf 4 beschränkt und die erlaubte Höhe einer automatischen Auszahlung durch die Funktion f limitiert. Für $n \in \mathbb{N} \cup \{0\}$ als Nummer der bereits zuvor erfolgten Auszahlungen und $a \in \mathbb{R}^+$ als initiale Ausschüttungsmenge für dieses Projekt sei $f(n) = a * \left(\frac{1}{n-5} + \frac{6}{5}\right)$. Um Manipulation zu vermeiden, werden die Bedingungen $n \in \mathbb{N} \cup \{0\}$ und $a \in \mathbb{R}^+$ explizit überprüft.

Beobachtungen

5.2 Skalierung

Wichtig ist auch, wie gut die in den vorhergehenden Kapiteln beschriebenen Tools für große Regelmengen skalieren. Um das zu untersuchen, soll die Zeit gemessen werden, die benötigt wird, um das Überprüfen auf Erfüllbarkeit und auf überflüssige Regeln eines Rulesets durchzuführen.

Szenario

Geradengleichungen

Für n Geradengleichungen der Form $y = mx + t$ werden pro Geradengleichung zwei Regeln erstellt, die jeweils einen zufälligen Punkt

$p = \{(x, y) | x, y \in \mathbb{N}, 0 <= x, y <= 299)\}$

in die Gleichung einsetzen. Da die Regelmengen für diesen Testfall erfüllbar sein sollen, wird sichergestellt, dass die zwei Punkte einer Geradengleichung nicht den selben x-Wert haben.

Für die Anzahl der Regeln des Rulesets gilt also: $a(n) = 2n$.

Polynome

Für ein Polynom mit Grad n und der Form $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ werden pro Koeffizienten a_n eine Regel erstellt, bei dem der Punkt $(n, 0)$ in die Funktion eingesetzt wird. Außerdem wird eine Regel erstellt, die bestimmt, dass alle Koeffizienten außer a_0 ungleich 0 sind sowie eine weitere Regel, dass für alle Koeffizienten gilt $-1000 < a_n < 1000$.

Für die Anzahl der Regeln des Rulesets gilt also: $a(n) = n^2 + 2$. Außerdem steigt mit Grad n natürlich auch die Komplexität der einzelnen Regeln.

Beobachtungen

Geradengleichungen

Polynome

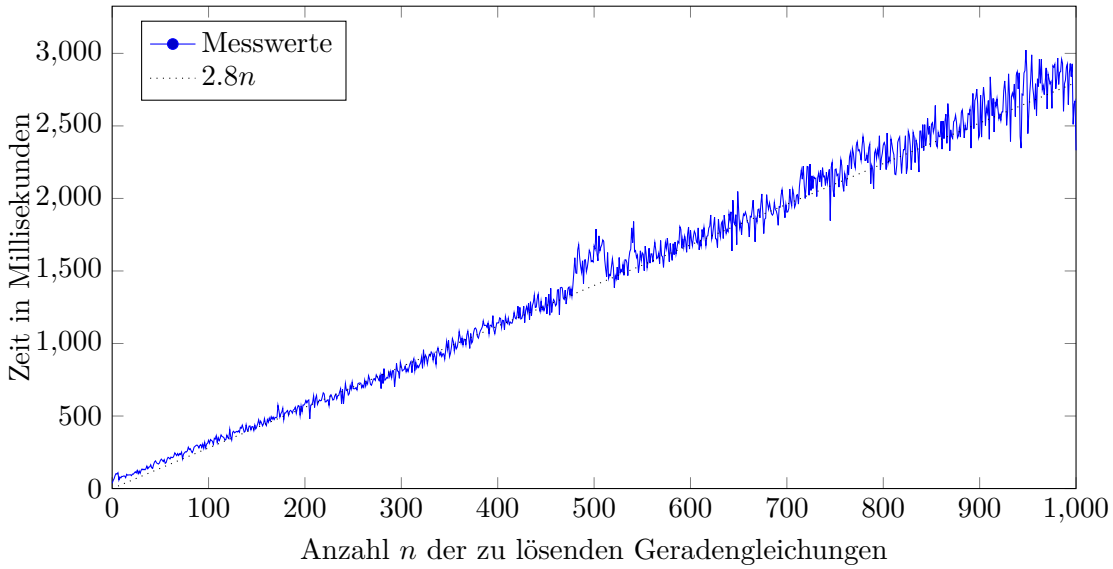


Abbildung 5.1: Zur Überprüfung der Erfüllbarkeit benötigte Zeit in Abhängigkeit von n

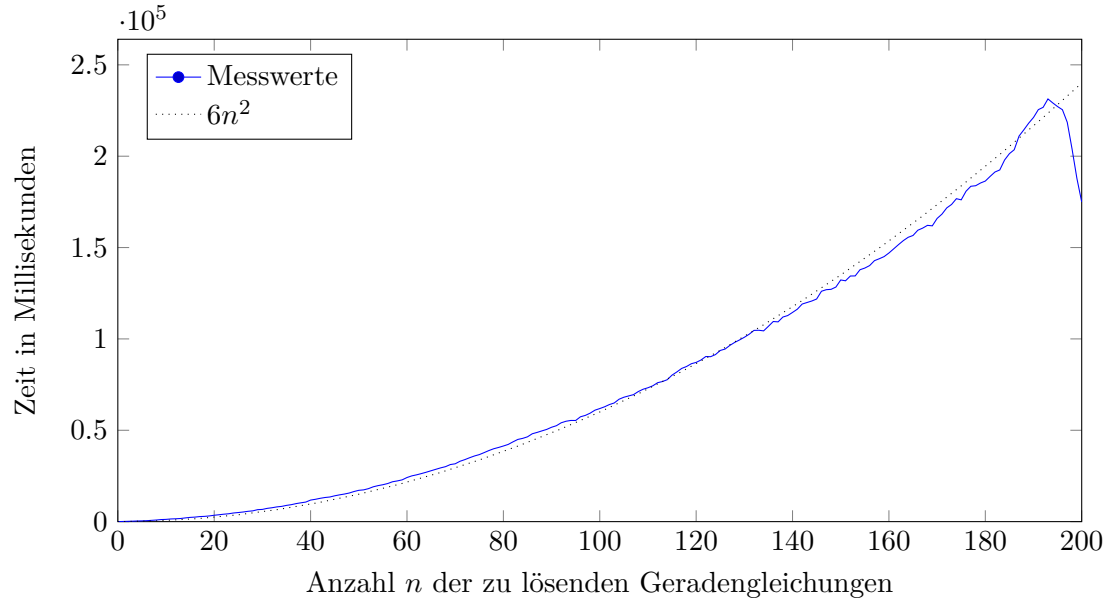


Abbildung 5.2: Zur Überprüfung auf nicht notwendige benötigte Zeit in Abhängigkeit von n

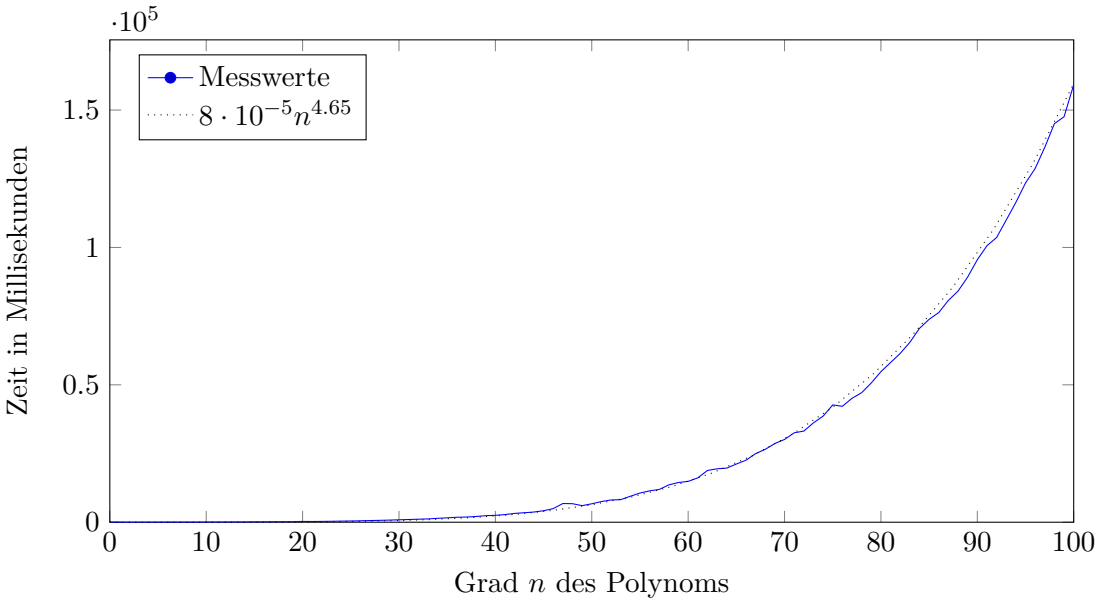


Abbildung 5.3: Zur Überprüfung der Erfüllbarkeit benötigte Zeit in Abhängigkeit von n

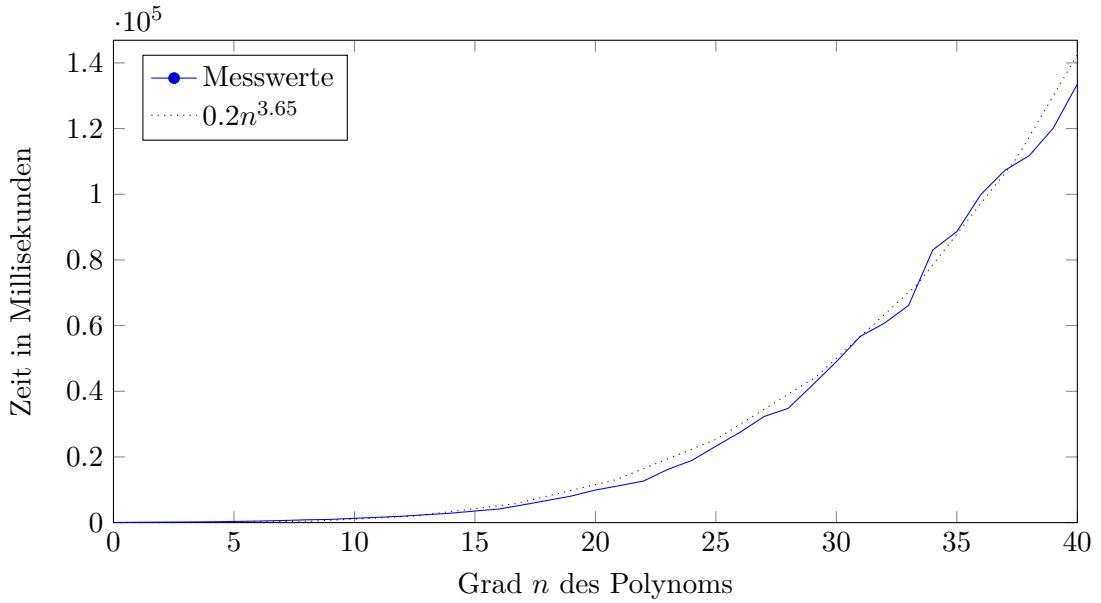


Abbildung 5.4: Zur Überprüfung auf nicht notwendige Regeln benötigte Zeit in Abhängigkeit von n

Literaturverzeichnis

[1] ISO. Iso 8601-1:2019 standard, 2019. https://www.iso.org/obp/ui/#iso:std:iso:8601:-1:ed-1:v1:en.

[2] Julian Pollinger. Rita schema, November 2022. https://github.com/educorvi/rita/tree/alpha/rita-core/src/schema.

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

[4] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2021. Available at https://www.SMT-LIB.org.

[5] IEEE. Ieee standard for information technology–portable operating system interface (posix(tm)) base specifications, issue 7. *IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)*, pages 1–3951, 2018.

A Allgemeines

A.1 Umrechnungsmatrix


```
{
  "years": {
    "quarters": 4,
    "months": 12,
    "weeks": 52.1775,
    "days": 365.2425,
    "hours": 8765.82,
    "minutes": 525949.2,
    "seconds": 31556951.999999996,
    "milliseconds": 31556951999.999996
  },
  "quarters": {
    "months": 3,
    "weeks": 13.044375,
    "days": 91.310625,
    "hours": 2191.455,
    "minutes": 131487.3,
    "seconds": 7889237.999999999,
    "milliseconds": 7889237999.999999
  },
  "months": {
    "weeks": 4.3481250000000005,
    "days": 30.436875,
    "hours": 730.485,
    "minutes": 43829.1,
    "seconds": 2629746,
    "milliseconds": 2629746000
  },
  "weeks": {
    "days": 7,
    "hours": 168,
    "minutes": 10080,
    "seconds": 604800,
    "milliseconds": 604800000
  },
  "days": {
    "hours": 24,
    "minutes": 1440,
    "seconds": 86400,
    "milliseconds": 86400000
  },
  "hours": {
    "minutes": 60,
    "seconds": 3600,
    "milliseconds": 3600000
  },
  "minutes": {
```

```
48      "seconds": 60,
49      "milliseconds": 60000
50    },
51    "seconds": {
52      "milliseconds": 1000
53    }
54  }
```

B Evaluations-Rulesets

B.1 Verein

```
{
    "$schema": "https://raw.githubusercontent.com/educorvi/rita/alpha/rita-core/src/schema/schema.json",
    "rules": [
        {
            "id": "mitglied",
            "comment": "Es soll nur an Personen ausgezahlt werden, die Mitglied im Verein sind.",
            "rule": {
                "type": "atom",
                "path": "auszahlung.empfaenger.istMitglied"
            }
        },
        {
            "id": "genehmigt",
            "comment": "Die Auszahlung muss zu einem der vom Verein genehmigten Projekte erfolgen. Genehmigungen, die  älter als 365 Tage sind, sind ungültig.",
            "rule": {
                "type": "and",
                "arguments": [
                    {
                        "type": "atom",
                        "path": "projekt.genehmigt"
                    },
                    {
                        "type": "comparison",
                        "operation": "smallerOrEqual",
                        "arguments": [
                            {
                                "type": "dateCalculation",
                                "dateResultUnit": "days",
                                "operation": "subtract",
                                "arguments": [
                                    {
                                        "type": "atom",
                                        "isDate": true,
                                        "path": "auszahlung.beantragungsdatum"
                                    },
                                    {
                                        "type": "atom",
                                        "isDate": true,
                                        "path": "projekt.genehmigtAm"
                                    }
                                ]
                            },
                            365
                        ]
```

```
                    }
                ]
            }
        },
        {
            "id": "pi",
            "comment": "Der auszuschüttende Betrag muss stets ein Vielfaches von 3.14
↪  sein",
            "rule": {
                "type": "comparison",
                "operation": "equal",
                "arguments": [
                    {
                        "type": "calculation",
                        "operation": "modulo",
                        "arguments": [
                            {
                                "type": "atom",
                                "path": "auszahlung.betrag"
                            },
                            3.14
                        ]
                    },
                    0
                ]
            }
        },
        {
            "id": "auszahlungsrahmen",
            "comment": "Eine Auszahlung muss geringer sein als die maximal erlaubte
↪  Höhe für das Projekt,aber größer als 0€",
            "rule": {
                "type": "and",
                "arguments": [
                    {
                        "type": "comparison",
                        "operation": "smallerOrEqual",
                        "arguments": [
                            {
                                "type": "atom",
                                "path": "auszahlung.betrag"
                            },
                            {
                                "type": "atom",
                                "path": "projekt.intialeAusschuettung"
                            }
                        ]
                    },
                    {
                        "type": "comparison",
                        "operation": "greater",
                        "arguments": [
                            {
                                "type": "atom",
                                "path": "auszahlung.betrag"
                            },
                            0
                        ]
```

```
                    }
                ]
            }
        },
        {
            "id": "limiterung",
            "comment": "Funktion zur Limitierung der Auszahlungen",
            "rule": {
                "type": "and",
                "arguments": [
                    {
                        "type": "comparison",
                        "operation": "smallerOrEqual",
                        "arguments": [
                            {
                                "type": "atom",
                                "path": "auszahlung.vorangegangene"
                            },
                            4
                        ]
                    },
                    {
                        "type": "comparison",
                        "operation": "smallerOrEqual",
                        "arguments": [
                            {
                                "type": "atom",
                                "path": "auszahlung.betrag"
                            },
                            {
                                "type": "calculation",
                                "operation": "multiply",
                                "arguments": [
                                    {
                                        "type": "atom",
                                        "path": "projekt.intialeAusschuettung"
                                    },
                                    {
                                        "type": "calculation",
                                        "operation": "add",
                                        "arguments": [
                                            {
                                                "type": "calculation",
                                                "operation": "divide",
                                                "arguments": [
                                                    1,
                                                    {
                                                        "type": "calculation",
                                                        "operation": "subtract",
                                                        "arguments": [
                                                            {
                                                                "type": "atom",
                                                                "path":
    ↪  "auszahlung.vorangegangene"
                                                            },
                                                            5
                                                        ]
                                                    }
```

```
                                        ]
                                    },
                                    1.2
                                ]
                            }
                        ]
                    }
                ]
            }
        ]
    },
    {
        "id": "integritaet",
        "comment": "n ∈ Nu{0} und a ∈ R+",
        "rule": {
            "type": "and",
            "arguments": [
                {
                    "type": "comparison",
                    "operation": "greater",
                    "arguments": [
                        {
                            "type": "atom",
                            "path": "auszahlung.betrag"
                        },
                        0
                    ]
                },
                {
                    "type": "and",
                    "arguments": [
                        {
                            "type": "comparison",
                            "operation": "greaterOrEqual",
                            "arguments": [
                                {
                                    "type": "atom",
                                    "path": "auszahlung.vorangegangene"
                                },
                                0
                            ]
                        },
                        {
                            "type": "comparison",
                            "operation": "equal",
                            "arguments": [
                                {
                                    "type": "calculation",
                                    "operation": "modulo",
                                    "arguments": [
                                        {
                                            "type": "atom",
                                            "path": "auszahlung.vorangegangene"
                                        },
                                        1
                                    ]
                                },
```

```
                                    0
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

B.2 Skalierung

B.2.1 Geradengleichungen

```
[
    {
      "id": "g0_0",
      "rule": {
        "type": "comparison",
        "operation": "equal",
        "arguments": [
          52,
          {
            "type": "calculation",
            "operation": "add",
            "arguments": [
              {
                "type": "calculation",
                "operation": "multiply",
                "arguments": [
                  {
                    "type": "atom",
                    "path": "m0"
                  },
                  126
                ]
              },
              {
                "type": "atom",
                "path": "t0"
              }
            ]
          }
        ],
        "dates": false
      },
      "comment": "52=m_0*126+t"
    },
    {
      "id": "g0_1",
      "rule": {
        "type": "comparison",
        "operation": "equal",
        "arguments": [
          41,
```

```
          {
            "type": "calculation",
            "operation": "add",
            "arguments": [
              {
                "type": "calculation",
                "operation": "multiply",
                "arguments": [
                  {
                    "type": "atom",
                    "path": "m0"
                  },
                  216
                ]
              },
              {
                "type": "atom",
                "path": "t0"
              }
            ]
          }
        ],
        "dates": false
      },
      "comment": "41=m_0*216+t"
    }
  ]

```

B.2.2 Polynom

```
[
    {
      "id": "e0",
      "rule": {
        "type": "comparison",
        "operation": "equal",
        "arguments": [
          0,
          {
            "type": "calculation",
            "operation": "add",
            "arguments": [
              {
                "type": "calculation",
                "operation": "multiply",
                "arguments": [
                  {
                    "type": "atom",
                    "path": "a2"
                  },
                  0
                ]
              },
              {
                "type": "calculation",
                "operation": "add",
```

```
                "arguments": [
                  {
                    "type": "calculation",
                    "operation": "multiply",
                    "arguments": [
                      {
                        "type": "atom",
                        "path": "a1"
                      },
                      0
                    ]
                  },
                  {
                    "type": "atom",
                    "path": "a0"
                  }
                ]
              }
            ]
          }
        ],
        "dates": false
      },
      "comment": "0=a_2*0^2+a_1*0^1+a0"
    },
    {
      "id": "e1",
      "rule": {
        "type": "comparison",
        "operation": "equal",
        "arguments": [
          0,
          {
            "type": "calculation",
            "operation": "add",
            "arguments": [
              {
                "type": "calculation",
                "operation": "multiply",
                "arguments": [
                  {
                    "type": "atom",
                    "path": "a2"
                  },
                  1
                ]
              },
              {
                "type": "calculation",
                "operation": "add",
                "arguments": [
                  {
                    "type": "calculation",
                    "operation": "multiply",
                    "arguments": [
                      {
                        "type": "atom",
                        "path": "a1"
```

```
                    },
                    1
                  ]
                },
                {
                  "type": "atom",
                  "path": "a0"
                }
              ]
            }
          ]
        }
      ],
      "dates": false
    },
    "comment": "0=a_2*1^2+a_1*1^1+a0"
  },
  {
    "id": "zero_cond",
    "rule": {
      "arguments": [
        {
          "arguments": [
            {
              "type": "comparison",
              "operation": "equal",
              "arguments": [
                0,
                {
                  "type": "atom",
                  "path": "a2"
                }
              ],
              "dates": false
            }
          ],
          "type": "not"
        },
        {
          "arguments": [
            {
              "type": "comparison",
              "operation": "equal",
              "arguments": [
                0,
                {
                  "type": "atom",
                  "path": "a1"
                }
              ],
              "dates": false
            }
          ],
          "type": "not"
        }
      ],
      "type": "and"
    },
```

```
      "comment": "for n>0: a_n != 0"
    },
    {
      "id": "limits",
      "rule": {
        "arguments": [
          {
            "arguments": [
              {
                "type": "comparison",
                "operation": "greater",
                "arguments": [
                  1000,
                  {
                    "type": "atom",
                    "path": "a2"
                  }
                ],
                "dates": false
              },
              {
                "type": "comparison",
                "operation": "smaller",
                "arguments": [
                  -1000,
                  {
                    "type": "atom",
                    "path": "a2"
                  }
                ],
                "dates": false
              }
            ],
            "type": "and"
          },
          {
            "arguments": [
              {
                "arguments": [
                  {
                    "type": "comparison",
                    "operation": "greater",
                    "arguments": [
                      1000,
                      {
                        "type": "atom",
                        "path": "a1"
                      }
                    ],
                    "dates": false
                  },
                  {
                    "type": "comparison",
                    "operation": "smaller",
                    "arguments": [
                      -1000,
                      {
                        "type": "atom",
```

```
                        "path": "a1"
                      }
                    ],
                    "dates": false
                  }
                ],
                "type": "and"
              },
              {
                "arguments": [
                  {
                    "type": "comparison",
                    "operation": "greater",
                    "arguments": [
                      1000,
                      {
                        "type": "atom",
                        "path": "a0"
                      }
                    ],
                    "dates": false
                  },
                  {
                    "type": "comparison",
                    "operation": "smaller",
                    "arguments": [
                      -1000,
                      {
                        "type": "atom",
                        "path": "a0"
                      }
                    ],
                    "dates": false
                  }
                ],
                "type": "and"
              }
            ],
            "type": "and"
          }
        ],
        "type": "and"
      },
      "comment": "-1000<a_n<1000"
    }
  ]

```