Bachelorarbeit

Entwicklung einer Anwendung zum Vermitteln von Lerninhalten über E-Graphs und Equality Saturation für die Lehre

Ben Steinhauer

Lehrstuhl für Softwaretechnik und Programmiersprachen
Heinrich-Heine-Universität Düsseldorf

25.04.2025

Überblick

Einleitung

Wie kommen wir von links nach rechts?

$$\left(\frac{x \cdot 2}{2} + x \cdot \frac{y - 3 + 3 + z \cdot 0}{x}\right)^2 \longrightarrow (x + y)^2$$

Grundlagen

Rewrite Rules

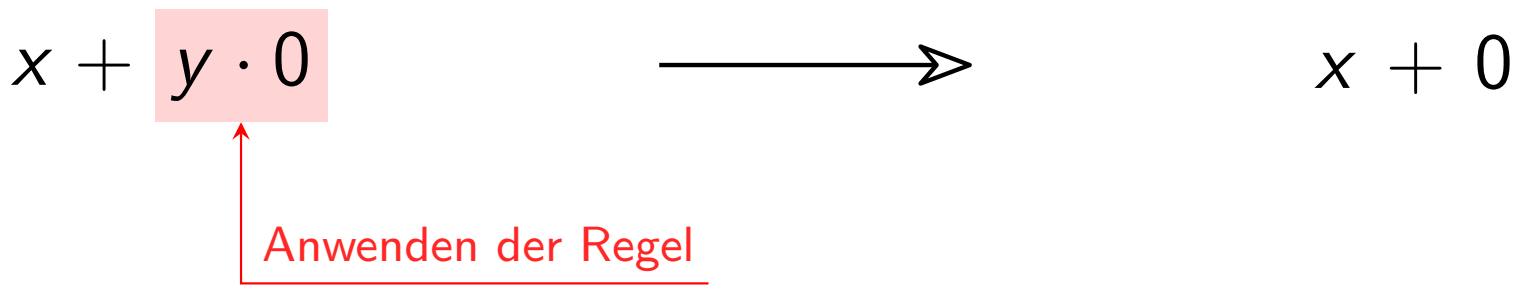
Benutzen einer Rewrite Rule:

$$a \cdot 0 \Leftrightarrow 0$$

Rewrite Rules

Benutzen einer Rewrite Rule:

$$a \cdot 0 \Leftrightarrow 0$$



Naiver Algorithmus

Algorithm 1 Naiver Algorithmus zur Optimierung von Ausdrücken

```
Funktion OPTIMIZE_EXPRESSION(expression)
    rules ← [...]
    while old_expression ≠ expression do
        old_expression ← expression
        for rule in rules do
            if match(expression, rule) then
                apply(expression, rule)
    return expression
```

Phase Ordering Problem

Welche Regel soll zuerst angewendet werden?

$$\left(\frac{x \cdot 2}{2} + x \cdot \frac{y - 3 + 3 + z \cdot 0}{x} \right)^2$$

Mögliche Resultate:

$$\frac{x \ll 1}{2}$$

$$x$$

Lösung

Speichern aller möglichen Ausdrücke in Liste:

ExprList

$$\left(\frac{x\cdot 2}{2} + x\cdot\frac{y-3+3+z\cdot 0}{x}\right)^2,\qquad \left(\frac{x\ll 1}{2} + x\cdot\frac{y-3+3+z\cdot 0}{x}\right)^2,$$
$$\left(x + x\cdot\frac{y-3+3+z\cdot 0}{x}\right)^2$$

Verbesserung

Sharing und Klassen:

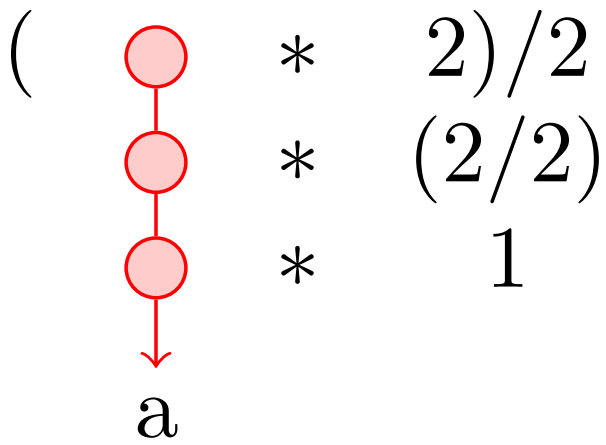


Abbildung 1: Sharing bei einer Liste von Ausdrücken

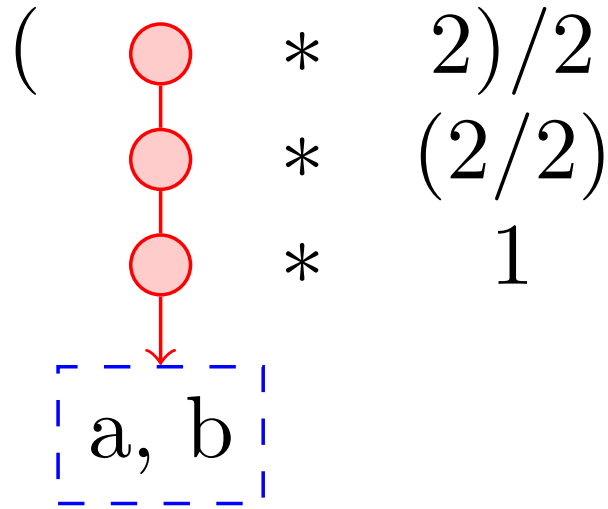


Abbildung 2: Kombination aus Sharing und Klassen bei einer Liste von Ausdrücken

E-Graphs (1)

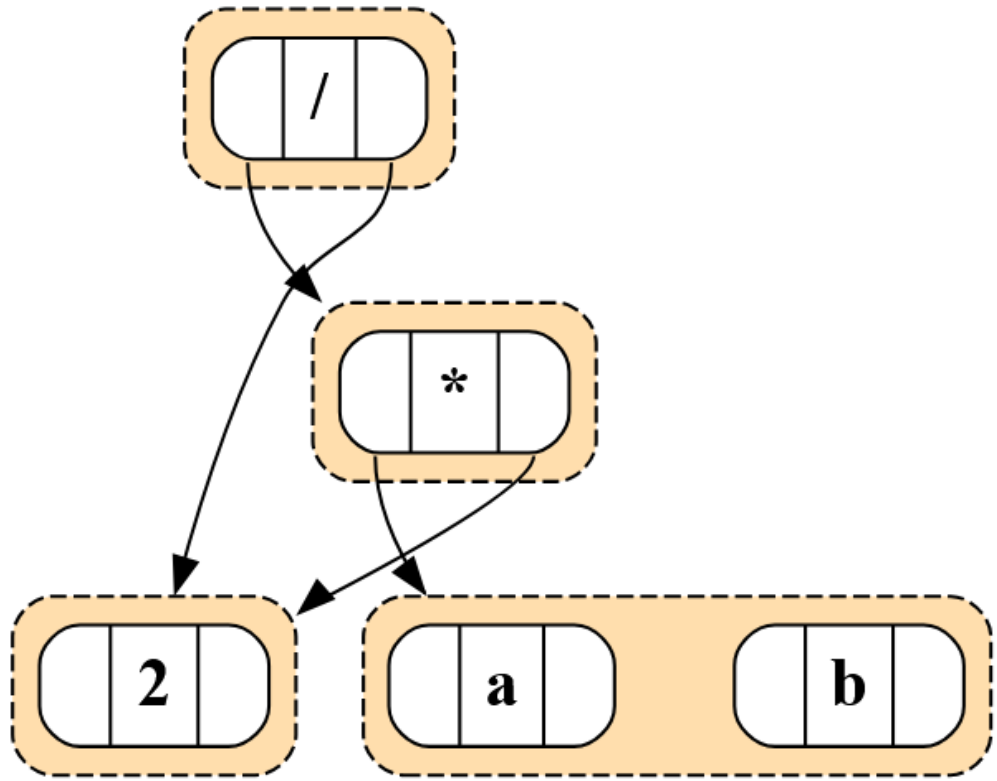


Abbildung 3: Beispiel eines E-Graphs, der die Ausdrücke $(a \cdot 2)/2$ und $(b \cdot 2)/2$ enthält

E-Graphs (2)

Der *E-Graph* als Datenstruktur für Abbildung 3:

- **U**: $\{\text{ID1}\}, \quad \{\text{ID2}\}, \quad \{\text{ID3}\}, \quad \{\text{ID4, ID5}\}$
- **M**:
  $ID1 \rightarrow EClass(\ldots)$,
  $ID2 \rightarrow EClass(\ldots)$,
  $ID3 \rightarrow EClass(\ldots)$, $\ldots$
- **H**:
  $/ \rightarrow ID1$,
  $* \rightarrow ID2$,
  $2 \rightarrow ID3$,
  $a \rightarrow ID4$,
  $b \rightarrow ID5$

Equality Saturation

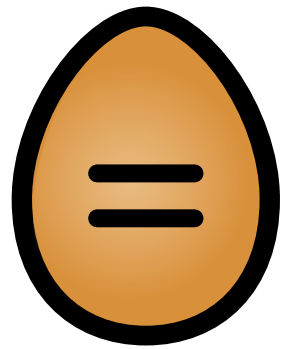
Algorithm 2 Traditioneller Equality Saturation Workflow nach [Wil+21]

```
Funktion EQSAT(expr, rewrites)
    egraph ← initial_egraph(expr)
    while not egraph.is_saturated_or_timeout() do
        for rw in rewrites do
            for (subst, eclass) in egraph.ematch(rw.lhs) do
                eclass2 ← egraph.add(rw.rhs.subst(subst))
                egraph.merge(eclass, eclass2)
    return egraph.extract_best()
```

Aufbau & Entwicklung der Anwendung

egg

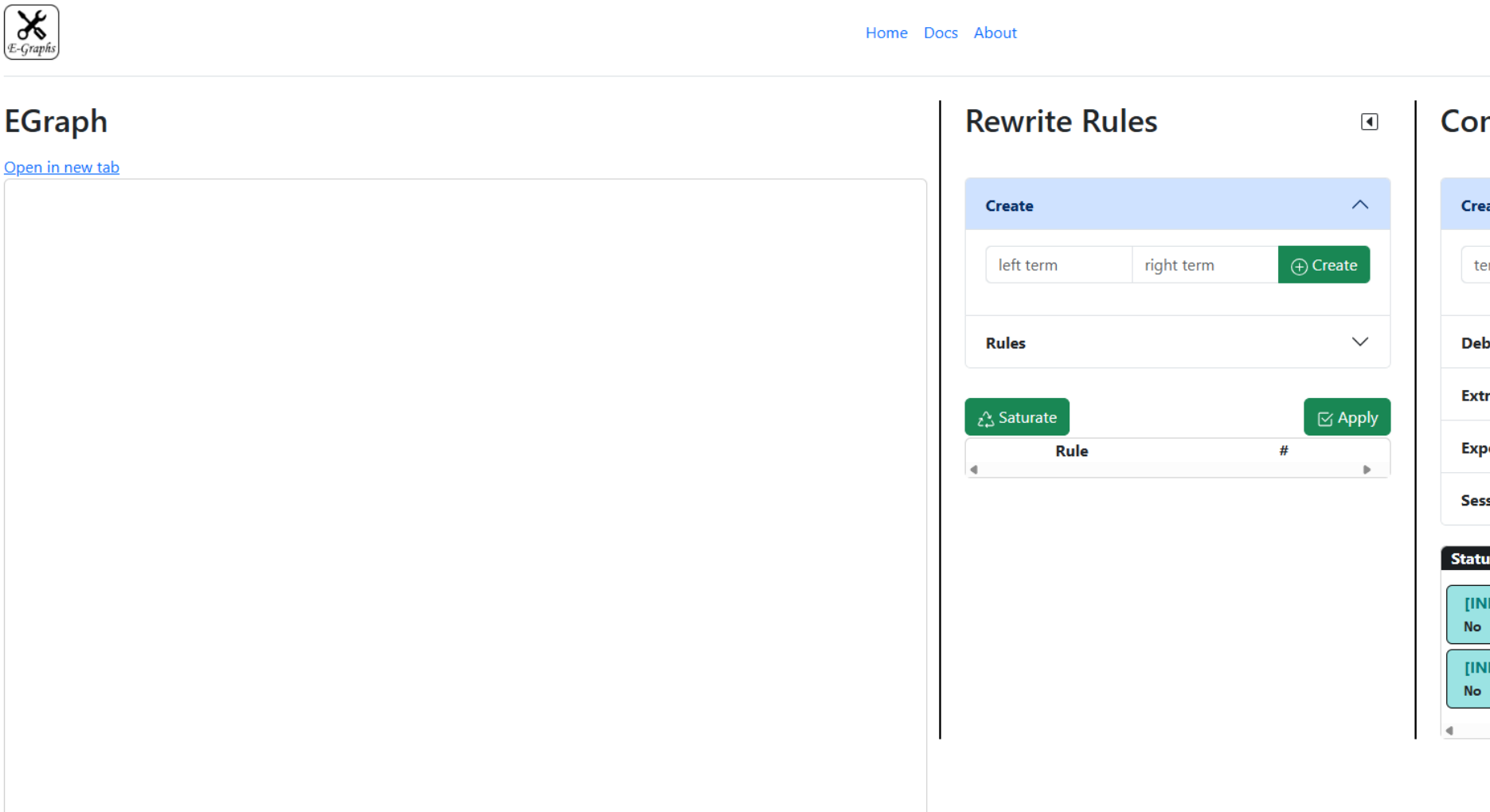
- **egg**: **e**-**g**raphs **g**ood (`https://egraphs-good.github.io/`)
- Bibliothek in Rust zur Erstellung von E-Graphs
- **Paper**: Willsey u.a. 2021 [Wil+21] (`https://doi.org/10.1145/3434304`)



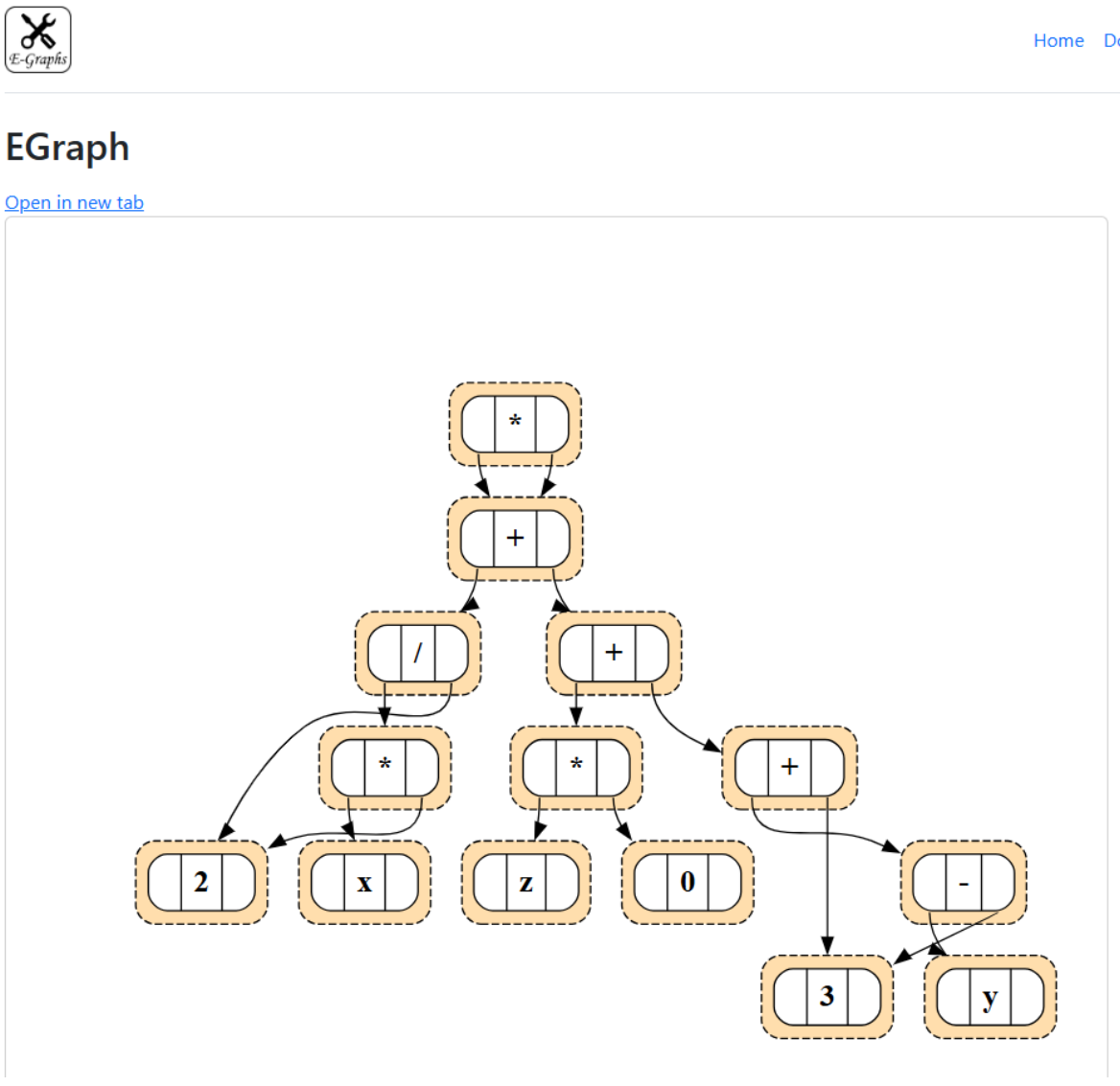
Google Colab Notebook

- **Google Colab Notebook**
- Prototyp in Python basierend auf egg
- **Zachary DeVito** [DeV] (`https://colab.research.google.com/drive/1tNOQijJqe5tw-Pk9iqd6HHb2abC5aRid?usp=sharing`)

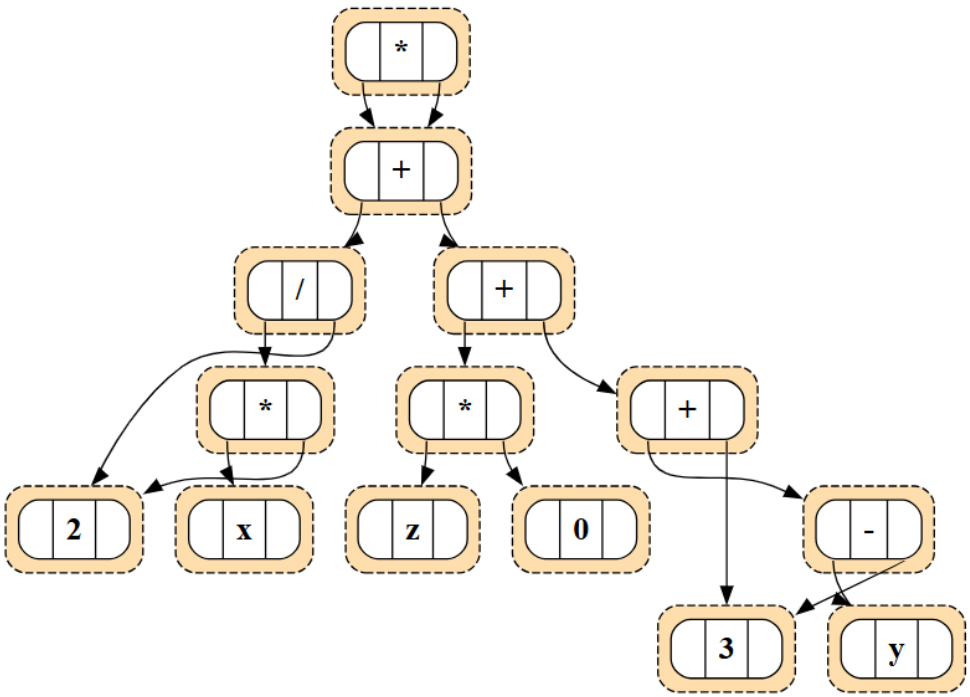
Anwendung (1)


E-Graphs
Home
Docs
About
EGraph
Open in new tab
Rewrite Rules
Create
left term
right term
Create
Rules
Saturate
Apply
Rule
#

Anwendung (2)


E-Graphs
Home Docs About
EGraph
Open in new tab
Rewrite Rules
Create
left term
right term
Create
Rules
Upload
Download
Saturate
Apply
Rule
#
(* x 2) => (<< x 1) 0
(<< x 1) => (* x 2) 1
(/ x x) => (1) 2
(1) => (/ x x) 3
(/ (* x y) z) => (* x (/ y z)) 4
(* x (/ y z)) => (/ (* x y) z) 5
(* (+ x y) (+ x y)) => (+ (* x x) (+ (* 2 (* x y)) (* y y))) 6
(+ (* x x) (+ (* 2 (* x y)) (* y y))) => (* (+ x y) (+ x y)) 7
(* x 0) => (0) 8
(0) => (* x 0) 9
Cont
Creat
Debu
Extra
Expor
Sessi
Status
[INFO EGra
[INFO Load
[INFO Adde
[INFO Load

Anwendung (3)



Create

left term | right term | Create

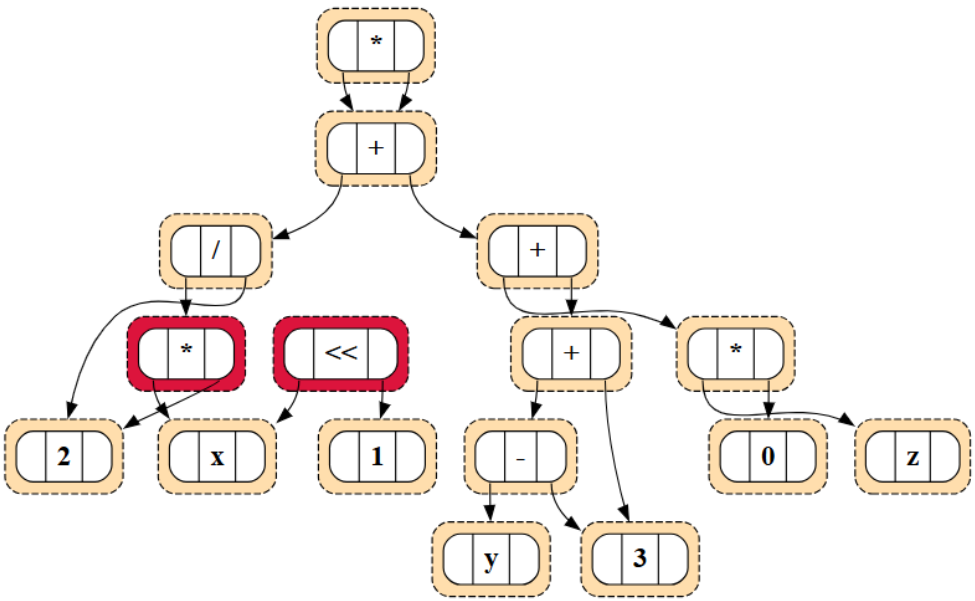
Rules

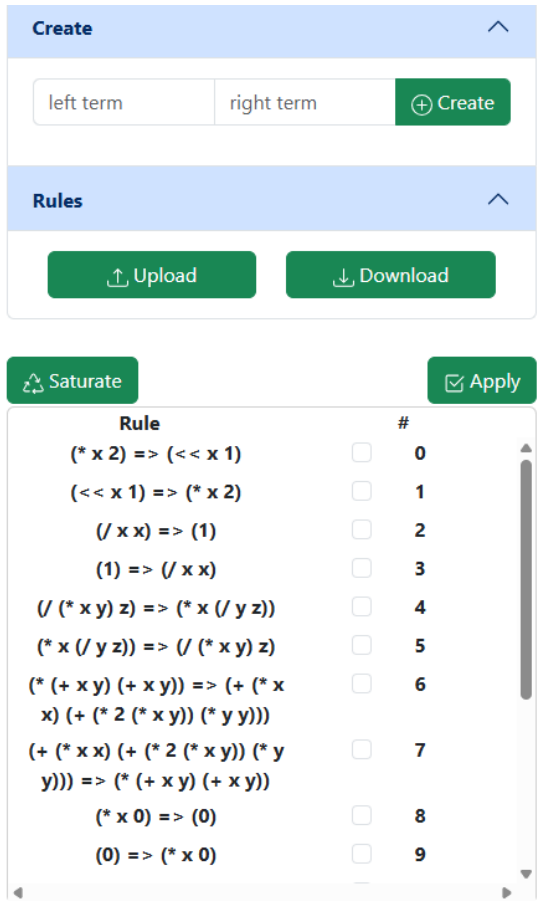
Upload | Download

Saturate | Apply

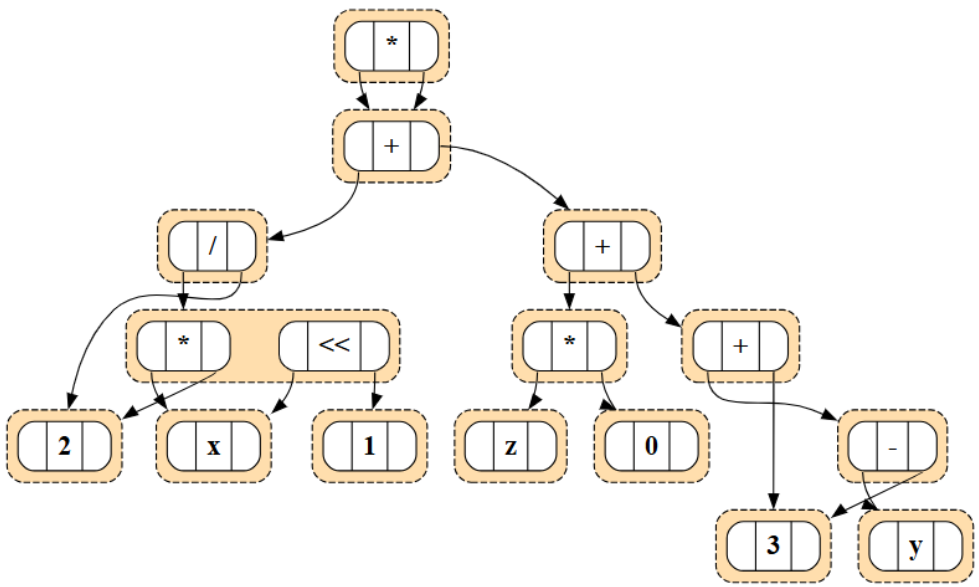
Rule	#
(* x 2) => (<< x 1)	0
(<< x 1) => (* x 2)	1
(/ x x) => (1)	2
(1) => (/ x x)	3
(/ (* x y) z) => (* x (/ y z))	4
(* x (/ y z)) => (/ (* x y) z)	5
(* (+ x y) (+ x y)) => (+ (* x x) (+ (* 2 (* x y)) (* y y)))	6
(+ (* x x) (+ (* 2 (* x y)) (* y y))) => (* (+ x y) (+ x y))	7
(* x 0) => (0)	8
(0) => (* x 0)	9

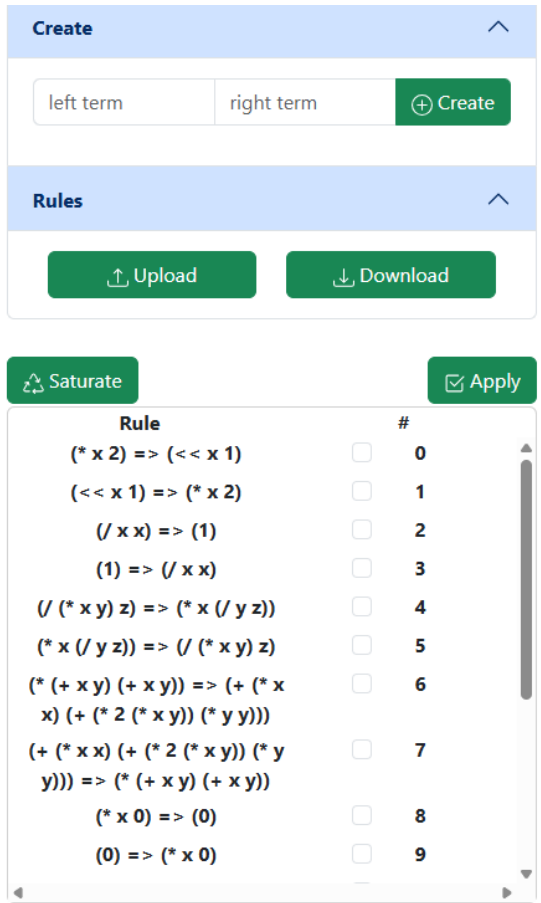
Anwendung (4)


*
+
/
+
*
<<
+
*
2
x
1
-
0
z
y
3

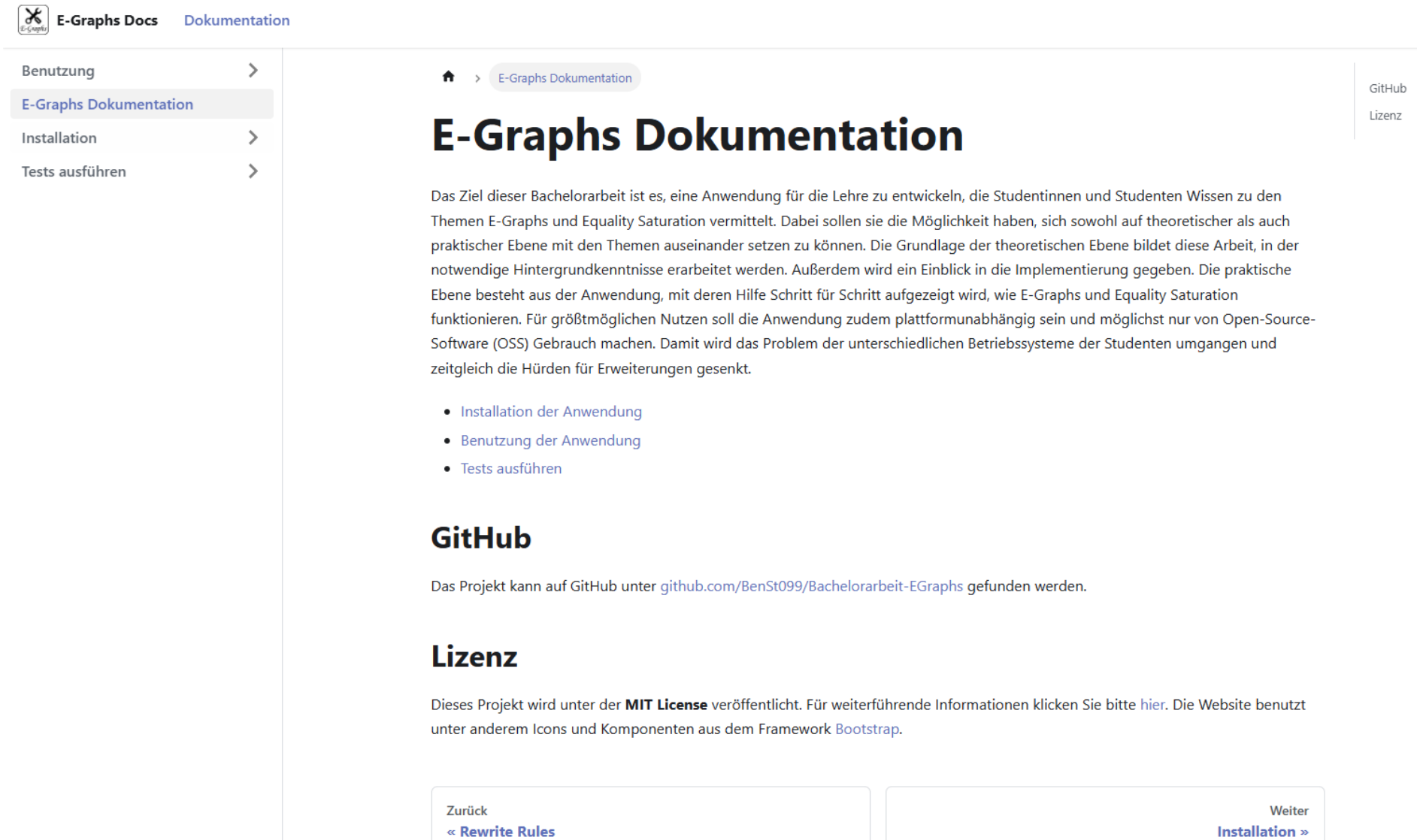

Create
left term
right term
Create
Rules
Upload
Download
Saturate
Apply
Rule
#
(* x 2) => (<< x 1) 0
(<< x 1) => (* x 2) 1
(/ x x) => (1) 2
(1) => (/ x x) 3
(/ (* x y) z) => (* x (/ y z)) 4
(* x (/ y z)) => (/ (* x y) z) 5
(* (+ x y) (+ x y)) => (+ (* x x) (+ (* 2 (* x y)) (* y y))) 6
(+ (* x x) (+ (* 2 (* x y)) (* y y))) => (* (+ x y) (+ x y)) 7
(* x 0) => (0) 8
(0) => (* x 0) 9

Anwendung (5)


*
+
/
+
*
<<
*
+
2
x
1
z
0
-
3
y


Create
left term
right term
Create
Rules
Upload
Download
Saturate
Apply
Rule
#
(* x 2) => (<< x 1) 0
(<< x 1) => (* x 2) 1
(/ x x) => (1) 2
(1) => (/ x x) 3
(/ (* x y) z) => (* x (/ y z)) 4
(* x (/ y z)) => (/ (* x y) z) 5
(* (+ x y) (+ x y)) => (+ (* x x) (+ (* 2 (* x y)) (* y y))) 6
(+ (* x x) (+ (* 2 (* x y)) (* y y))) => (* (+ x y) (+ x y)) 7
(* x 0) => (0) 8
(0) => (* x 0) 9

Docs


E-Graphs Docs
Dokumentation
Benutzung
E-Graphs Dokumentation
Installation
Tests ausführen
E-Graphs Dokumentation
GitHub
Lizenz
E-Graphs Dokumentation
Das Ziel dieser Bachelorarbeit ist es, eine Anwendung für die Lehre zu entwickeln, die Studentinnen und Studenten Wissen zu den Themen E-Graphs und Equality Saturation vermittelt. Dabei sollen sie die Möglichkeit haben, sich sowohl auf theoretischer als auch praktischer Ebene mit den Themen auseinander setzen zu können. Die Grundlage der theoretischen Ebene bildet diese Arbeit, in der notwendige Hintergrundkenntnisse erarbeitet werden. Außerdem wird ein Einblick in die Implementierung gegeben. Die praktische Ebene besteht aus der Anwendung, mit deren Hilfe Schritt für Schritt aufgezeigt wird, wie E-Graphs und Equality Saturation funktionieren. Für größtmöglichen Nutzen soll die Anwendung zudem plattformunabhängig sein und möglichst nur von Open-Source-Software (OSS) Gebrauch machen. Damit wird das Problem der unterschiedlichen Betriebssysteme der Studenten umgangen und zeitgleich die Hürden für Erweiterungen gesenkt.
Installation der Anwendung
Benutzung der Anwendung
Tests ausführen
GitHub
Das Projekt kann auf GitHub unter github.com/BenSt099/Bachelorarbeit-EGraphs gefunden werden.
Lizenz
Dieses Projekt wird unter der MIT License veröffentlicht. Für weiterführende Informationen klicken Sie bitte hier. Die Website benutzt unter anderem Icons und Komponenten aus dem Framework Bootstrap.
Zurück
« Rewrite Rules
Weiter
Installation »

Aufbau

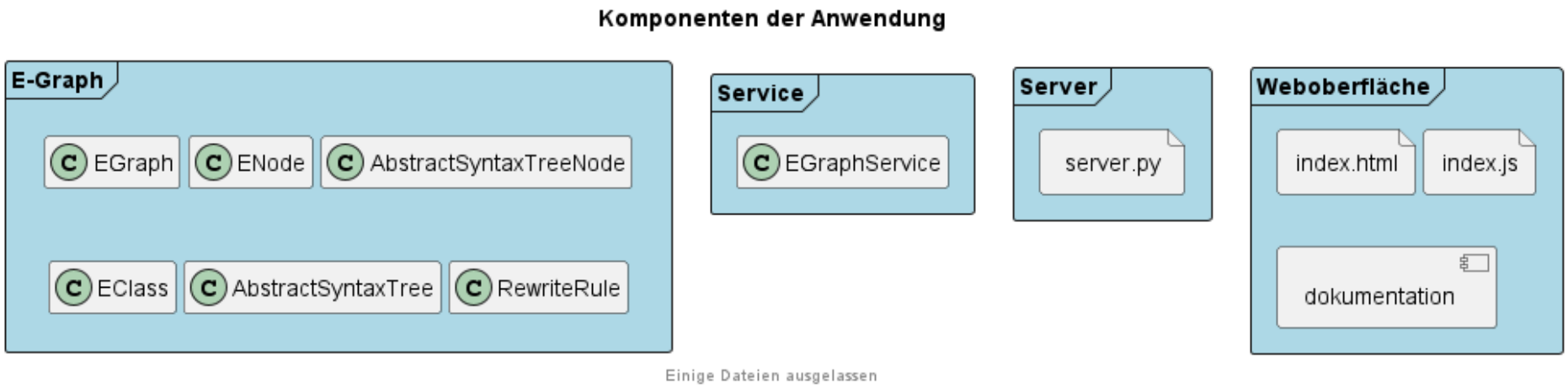


Abbildung 4: Architekturdiagramm der Anwendung

Ablauf (1)

$$\left(\frac{x \cdot 2}{2} + x \cdot \frac{y - 3 + 3 + z \cdot 0}{x}\right)^2 \longrightarrow (x + y)^2$$

Ablauf (2): AST

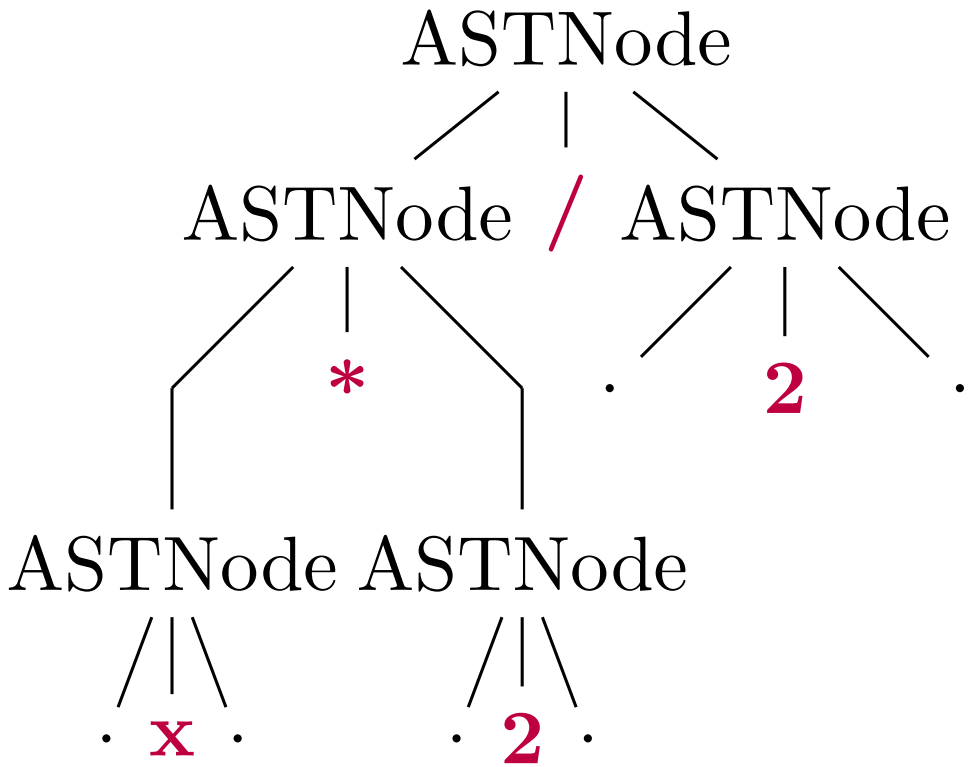


Abbildung 5: **A**bstract **S**yntax **T**ree des Teilausdrucks $(x \cdot 2)/2$

Ablauf (3): Add

```
def add_node(self, ast_root_node):
    # rekursiv Knoten des AST mit _add() zum E-Graph hinzufuegen

def _add(self, enode):
    enode = self._canonicalize(enode)
    if enode in self.h.keys():
        return self.h[enode]
    # weitere Faelle
    else:
        eclass_id = self._new_singleton_eclass(enode)
        for child in enode.arguments:
            self.m[child].parents.add((enode, eclass_id))
        self.h[enode] = eclass_id
        return eclass_id
```

Ablauf (4): Apply

```
def apply_rules(rules, egraph):
    eclasses = egraph.get_eclasses()
    list_of_matches = []

    for rule in rules:
        for eclass_id, environment in egraph._ematch(eclasses,
        rule.expr_lhs.root_node):
            list_of_matches.append((rule, eclass_id, environment))
    for rule, eclass_id, environment in list_of_matches:
        new_eclass_id = egraph._substitute(
            rule.expr_rhs.root_node, environment
            )
        egraph.merge(eclass_id, new_eclass_id)

    egraph.rebuild()
```

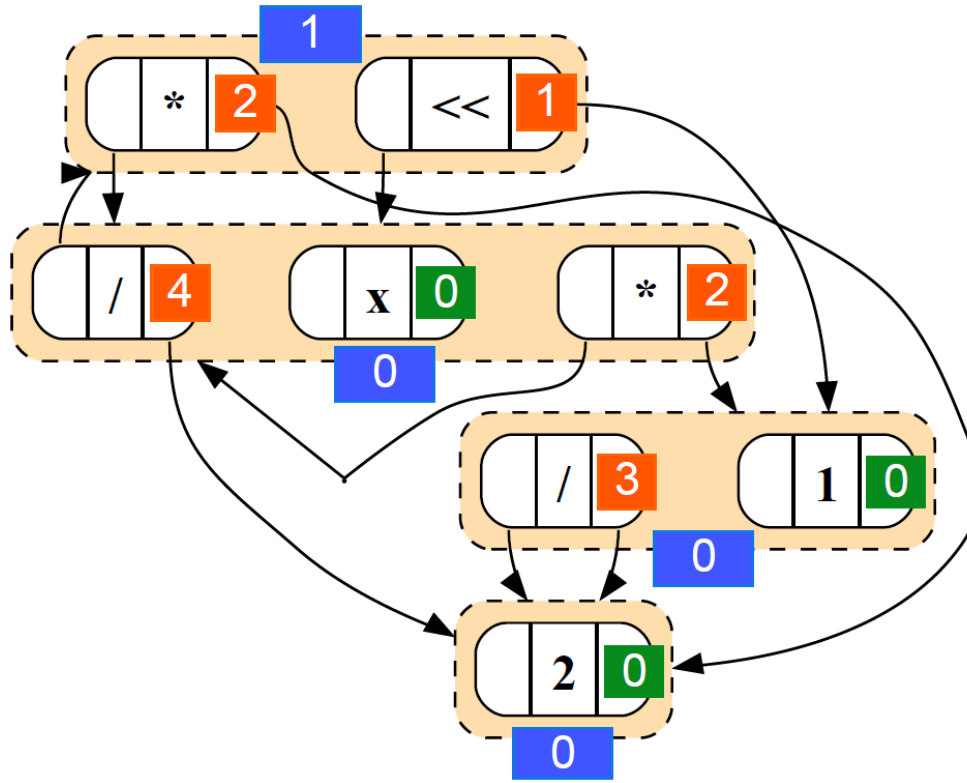
Ablauf (5): Saturate

```
def equality_saturation(rules, eterm_id, egraph):
    best_term = ""
    old_term = best_term
    if not egraph.is_saturated:
        while True:
            best_term = _extract_term(eterm_id, egraph)
            if old_term == best_term:
                break
            old_term = best_term
            egraph = apply_rules(rules, egraph)
```

Ablauf (5): Extract

Kostenfunktion:

- Operatoren:
  - $+, -, \ll, \gg$: **1**
  - $*$: **2**
  - $/$: **3**
- E-Node ohne Operatoren: 0
- E-Node mit Operatoren:
  Operator + Kosten der Kinder
- E-Class:
  Kind mit Minimum der Kosten



Ablauf (5): Extract

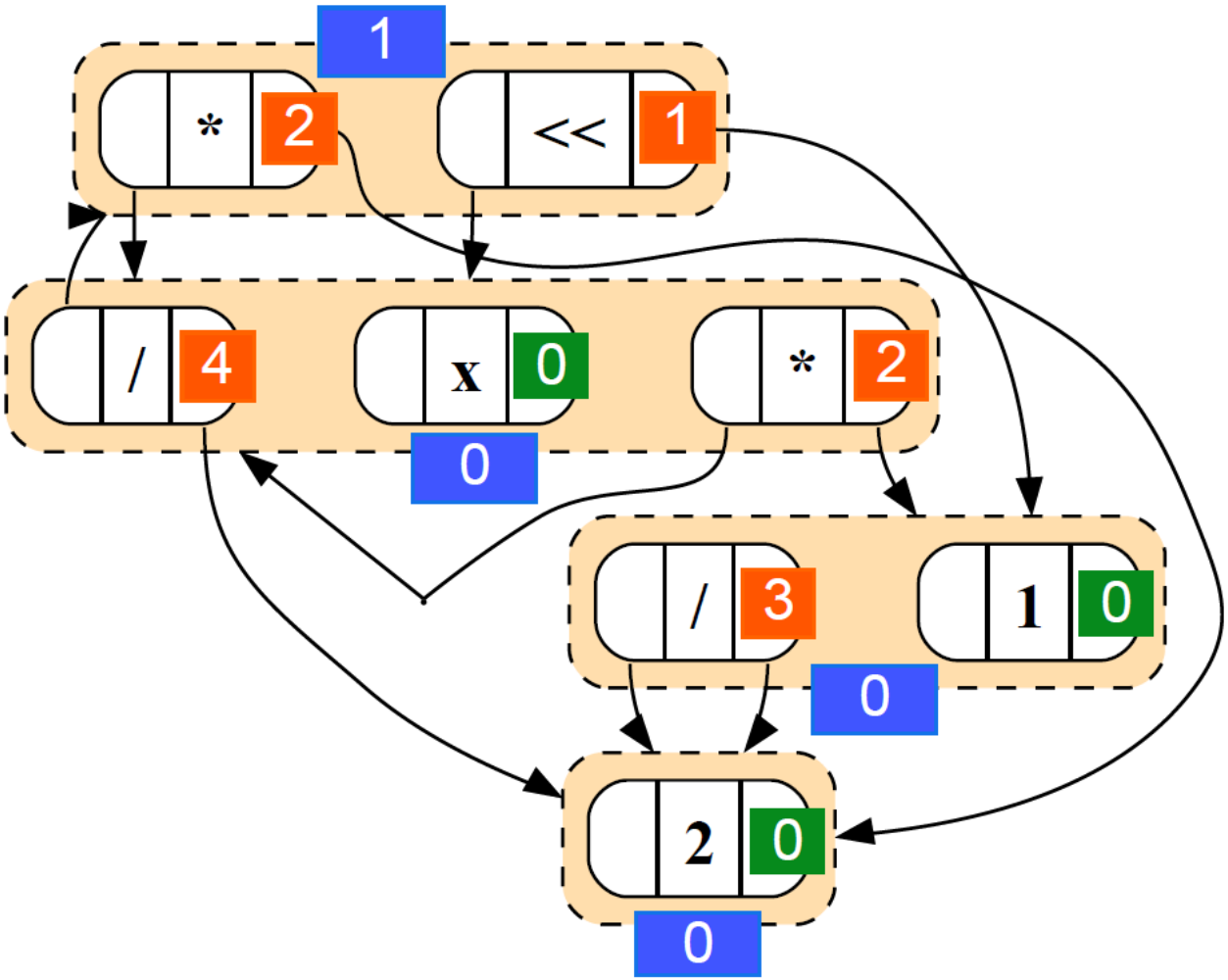


Abbildung 6: Kosten des Teilausdrucks $(x \cdot 2)/2$

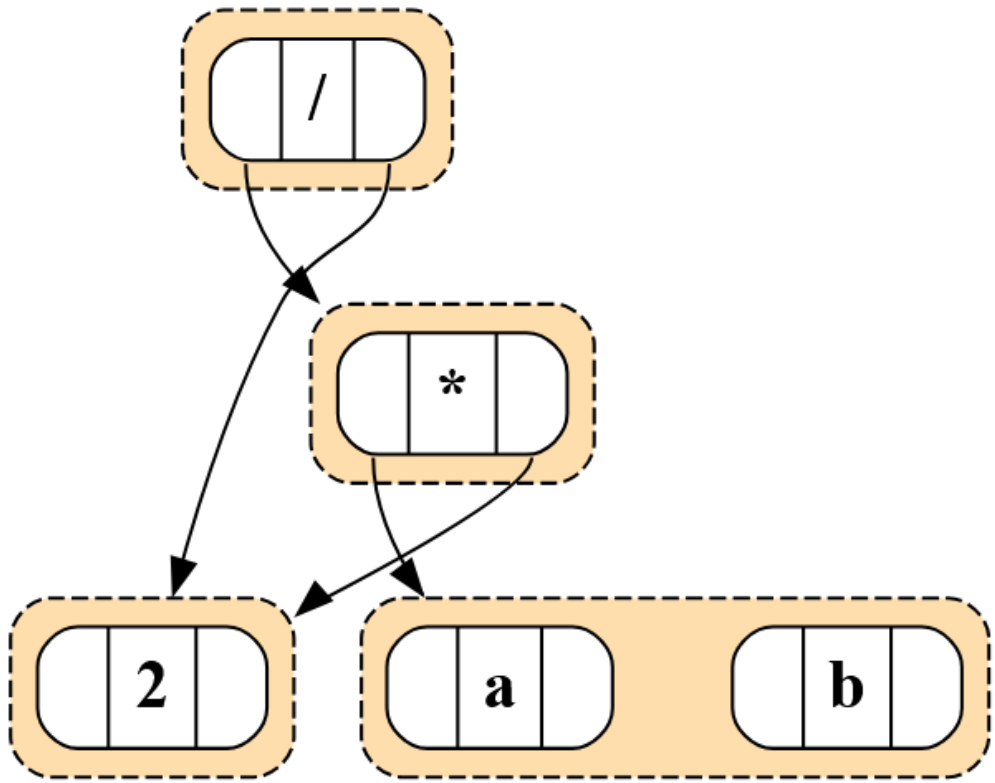
Ergebnisse & Diskussion

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**



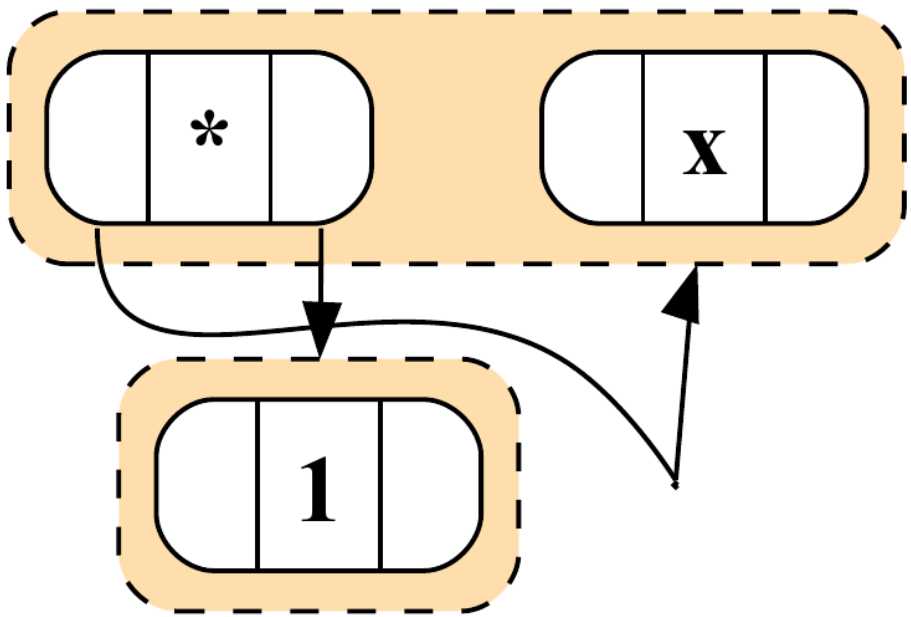
Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**

```
def egraph_to_dot(self, nodesep=0.5, ranksep=0.5,
    marked_eclasses = []):
    dot_commands = [
        "digraph parent { graph [compound=true, nodesep=" + s
        + ", ranksep=" + str(ranksep) + "]\n" + """node [fill
        fontname=\"Times-Bold\" fontsize=20 shape=record styl
        filled\"]\n"""
    ]
    # ... insgesamt 106 Zeilen lang
    return "".join(dot_commands)
```

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**
3. **Spezialfall: Kreis im E-Graph**



Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**
3. **Spezialfall: Kreis im E-Graph**

```
def equality_saturation(rules, eterm_id, egraph):
    #
    while True and timeout < 3:
        v = egraph.version
        timeout += 1
        #
        if v == egraph.version:
            break
```

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**
3. **Spezialfall: Kreis im E-Graph**

```
def equality_saturation(rules, eterm_id, egraph):
    #
    while True:
        best_term = _extract_term(eterm_id, egraph)
        if old_term == best_term:
            break
        old_term = best_term
        #
```

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**
3. **Spezialfall: Kreis im E-Graph**
4. **Pfadangabe im Server**

Komplikationen während der Entwicklung

1. **Kombination zweier Implementierungen**
2. **Darstellung der Datenstruktur**
3. **Spezialfall: Kreis im E-Graph**
4. **Pfadangabe im Server**

```
app.mount("/",
StaticFiles(directory=
    realpath(f"{realpath(__file__)}/../static"), html=True),
    name="static"
) # https://github.com/fastapi/fastapi/issues/3550
```

Ergebnisse

1. funktionstüchtige Anwendung
2. plattformunabhängig & getestet
3. basiert ausschließlich auf Open-Source-Software
4. erfüllt Anforderungen des Exposés [Ste]:
    - Benutzeroberfläche im Browser
    - Erzeugen & Visualisieren von E-Graphs
    - Erstellen & Anwenden von rewrite rules, vordefinierte rewrite rules
    - Debugging-Feature
    - Extraktion des optimalen Terms
    - Export des E-Graphs in gängige Formate
    - Eingaben als Session abspeichern
    - Dokumentation

Qualität der Software

nach ISO/OEC 20510:2011 Standard:

1. **Übertragbarkeit**
2. **Wartbarkeit**
3. **Sicherheit**
4. **Zuverlässigkeit**
5. **Funktionale Eignung**
6. **Performance**
7. **Kompatibilität**
8. **Benutzbarkeit**

Erweiterungen

1. **Erweitertes Testing**
   - Vergleichsmethode: eigene Implementierung vs. egg
2. **Erstellung von E-Graphs visualisieren**
   - schrittweises Darstellung vom Aufbauprozess (AST zu E-Graph)
3. **Abindung an egg**
   - Benutzer kann zwischen eigener Implementierung und egg als Backend wählen
   - benötigt Methoden in Rust
4. **E-Class Analysis**
   - Conditional and Dynamic Rewrites
   - Constant Folding

Referenzen

[Wil+21] Max Willsey u. a. “egg: Fast and Extensible Equality Saturation”. In: *Proc. ACM Program. Lang.* 5.POPL (Jan. 2021). DOI: 10.1145/3434304. URL: https://doi.org/10.1145/3434304.

[DeV] Zachary DeVito. *Intro to EGraphs*. besucht am 26.11.2024. URL: https://colab.research.google.com/drive/1tNOQijJqe5tw-Pk9iqd6HHb2abC5aRid?usp=sharing.

[Ste] Ben Steinhauer. *Exposé zur Bachelorarbeit*. besucht am 11.02.2025. URL: https://github.com/BenSt099/Bachelorarbeit-EGraphs/blob/main/expos/expose.pdf.