Einleitung Stand Vorher

Änderungen

Live-Demo

Fazit

Gainsif Gain

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Isabel Wingen

Institut für Informatik Heinrich-Heine-Universität Düsseldorf

18.05.2017

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

> Institut für Indomesia Perinsiah Perine Universität Düsselder 18.05.2017

137 / P. · · · P. Misser

Gliederung



- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- **5** Fazit

p. phain

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Einleitung

Gliederung

Einleitung

Stand Vorher

Änderungen

Live-Demo

Fazit



Einleitung

· Weiterführung der Bachelorarbeit von Fabian Ruhland



- 1. Weiterführung der Arbeit von Fabian Ruhland
- 2. in Java geschrieben

Stand Vorher Änderungen Live-Demo Fazit

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Einleitung

Einleitung

- · Weiterführung der Bachelorarbeit von Fabian Ruhland
- Toolbox zur Visualisierung von Automaten und Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Einleitung

Waterforung or Bacherswelet von Falsen Afrande
- Tateba zur Vaulischen von Afranden und Gemender

- 1. Weiterführung der Arbeit von Fabian Ruhland
- 2. in Java geschrieben

-Einleitung

3. Toolbox zur Visualisierung von Automaten und Grammatiken

HEINRICH HEINE

Einleitung

3 / 37

- · Weiterführung der Bachelorarbeit von Fabian Ruhland
- Toolbox zur Visualisierung von Automaten und Grammatiken
- Fokus auf Automaten und Compilerbau



- 1. Weiterführung der Arbeit von Fabian Ruhland
- 2. in Java geschrieben

-Einleitung

- 3. Toolbox zur Visualisierung von Automaten und Grammatiken
- 4. Fokus: Automaten und Compilerbau

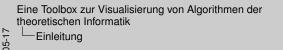
leitung

Weiterführung der Bachelonarbeit von Fabian Ruhland
Toolbox zur Visualisienung von Automaten und Grammatiker
Frans eri Automaten und Compilenten
Frans eri Automaten

HEINRICH HEINE

Einleitung

- · Weiterführung der Bachelorarbeit von Fabian Ruhland
- Toolbox zur Visualisierung von Automaten und Grammatiken
- Fokus auf Automaten und Compilerbau
- Weiterentwicklung des Programmes





- 1. Weiterführung der Arbeit von Fabian Ruhland
- 2. in Java geschrieben

-Einleitung

- 3. Toolbox zur Visualisierung von Automaten und Grammatiken
- 4. Fokus: Automaten und Compilerbau
- 5. Weiterentwicklung

HEINRICH HEINE

Einleitung

- · Weiterführung der Bachelorarbeit von Fabian Ruhland
- Toolbox zur Visualisierung von Automaten und Grammatiken
- Fokus auf Automaten und Compilerbau
- Weiterentwicklung des Programmes
- sowohl GUI als auch Konsole

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
Einleitung

Einleitung

Weiterführung der Bachelonarbeit von Fabian Ruhland
Toolbox zur Visualisierung von Automaten und Grammatiker
Fokus self Automaten und Complettau
Weiterentwicklung des Programmes
sowohl GUI als auch Konsole

- 1. Weiterführung der Arbeit von Fabian Ruhland
- 2. in Java geschrieben
- 3. Toolbox zur Visualisierung von Automaten und Grammatiken
- 4. Fokus: Automaten und Compilerbau
- 5. Weiterentwicklung
- 6. GUI und Konsole
- 7. ENDE

Einleitung Stand Vorher Änder

Änderungen

Live-Demo

Fazit

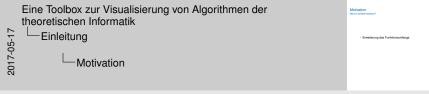
limited of



Motivation

Warum weiterentwicklen?

• Erweiterung des Funktionsumfangs



Warum weiterentwickeln?

1. Funktionsumfang erweiteren

Änderungen Fazit Einleitung Stand Vorher Live-Demo

Motivation

Warum weiterentwicklen?

- Erweiterung des Funktionsumfangs
 - · Algorithmen für Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Einleitung -Motivation



Warum weiterentwickeln?

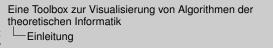
- Funktionsumfang erweiteren
- Mehr Algorithmen für Grammatiken
 Chomsky-Normalform oder CYK

Fazit Einleitung Stand Vorher Änderungen Live-Demo

Motivation

Warum weiterentwicklen?

- Erweiterung des Funktionsumfangs
 - · Algorithmen für Grammatiken
 - Kommentierte Ausführung & LATEX-Generierung





Warum weiterentwickeln?

Funktionsumfang erweiteren

-Motivation

- Mehr Algorithmen für Grammatiken
 Chomsky-Normalform oder CYK
- 4. Kommentierte Ausführung der Algorithmen
- 5. Latex

Fazit Einleitung Stand Vorher Änderungen Live-Demo

Motivation

Warum weiterentwicklen?

- Erweiterung des Funktionsumfangs
 - Algorithmen f
 ür Grammatiken
 - Kommentierte Ausführung & LATEX-Generierung
- Verbesserung der Benutzerfreundlichkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Einleitung

-Motivation

 Algorithmen für Grammatiken
 Kommentierte Ausführung & Iffe-X Generierung Verbesserung der Benutzerfreundlichkeit

Warum weiterentwickeln?

- Funktionsumfang erweiteren
- 2. Mehr Algorithmen für Grammatiken
- 3. Chomsky-Normalform oder CYK
- 4. Kommentierte Ausführung der Algorithmen
- 5. Latex
- 6. Verbesserung Benutzerfreundlichkeit

Motivation

Warum weiterentwicklen?

- Erweiterung des Funktionsumfangs
 - Algorithmen f
 ür Grammatiken
 - Kommentierte Ausführung & LATEX-Generierung
- Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
Einleitung

-Motivation

Enweiterung des Funktionsumfungs
 Algorithmen für Gammaliken
 Kommerliche Austihnung & Egyk Generierung
 Verbesserung der Beimutzerfesundlichkeit
 Andersung föckglenig macket.

Warum weiterentwickeln?

- 1. Funktionsumfang erweiteren
- 2. Mehr Algorithmen für Grammatiken
- 3. Chomsky-Normalform oder CYK
- 4. Kommentierte Ausführung der Algorithmen
- 5. Latex
- 6. Verbesserung Benutzerfreundlichkeit
- 7. Änderungen rückgängig





Fazit

Motivation

Warum weiterentwicklen?

- Erweiterung des Funktionsumfangs
 - Algorithmen f
 ür Grammatiken
 - Kommentierte Ausführung & LATEX-Generierung
- · Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen
 - komfortablere Benutzeroberfläche

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Einleitung

 Algorithmen für Grammatiken
 Kommentierte Ausführung & Iffe-X-Generierun Verbesserung der Benutzerfreundlichkeit

Warum weiterentwickeln?

Funktionsumfang erweiteren

-Motivation

- 2. Mehr Algorithmen für Grammatiken
- 3. Chomsky-Normalform oder CYK
- 4. Kommentierte Ausführung der Algorithmen
- 5. Latex
- 6. Verbesserung Benutzerfreundlichkeit
- 7. Änderungen rückgängig
- 8. GUI Benutzer komfortabler

Gliederung



- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- **5** Fazit



Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

—Stand Vorher

—Gliederung

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit

Datentypen

• (kontextfreie) Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen

Datentypen der Toolbox

- 1. kontextfreie Grammatiken
- 2. endliche Automaten, sowohl deterministisch als auch nicht-deterministisch
- 3. Aufzählung aller Symbole, dann Regeln, Grund für kontextfrei
- 4. SableCC ist ein Parser-Generator

Fazit

Datentypen

- (kontextfreie) Grammatiken
- (endliche) Automaten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

HHEINE DOSSELDORF

Stand Vorher



Datentypen der Toolbox

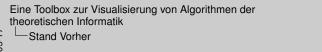
1. kontextfreie Grammatiken

- 2. endliche Automaten, sowohl deterministisch als auch nicht-deterministisch
- 3. Aufzählung aller Symbole, dann Regeln, Grund für kontextfrei
- 4. SableCC ist ein Parser-Generator

Datentypen

- · (kontextfreie) Grammatiken
- (endliche) Automaten
- Eigenes Speicherformat

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



Datentypen

- (kontaetfreie) Crammatike
- (endichs) Automatie
- Eigenes Speicherformat

Datentypen der Toolbox

1. kontextfreie Grammatiken

- 2. endliche Automaten, sowohl deterministisch als auch nicht-deterministisch
- 3. Aufzählung aller Symbole, dann Regeln, Grund für kontextfrei
- 4. SableCC ist ein Parser-Generator

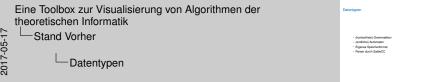
Datentypen

6 / 37

- · (kontextfreie) Grammatiken
- (endliche) Automaten
- Eigenes Speicherformat
- Parser durch SableCC

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Fazit



Datentypen der Toolbox

- 1. kontextfreie Grammatiken
- 2. endliche Automaten, sowohl deterministisch als auch nicht-deterministisch
- 3. Aufzählung aller Symbole, dann Regeln, Grund für kontextfrei
- 4. SableCC ist ein Parser-Generator

Änderungen Einleitung Stand Vorher Live-Demo

Fazit

Datentypen

Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Stand Vorher

-Datentypen

1. Was konnte die Toolbox mit den Daten machen?

Datentypen

Grammatiken

- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Änderungen Einleitung Stand Vorher Live-Demo

Fazit

Datentypen

- Grammatiken
 - Anzeigen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Datentypen

Grammatiken

Stand Vorher

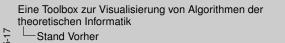
- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen

- Grammatiken
 - Anzeigen
 - Bearbeiten

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit





- Datentypen
- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen



- Anzeigen
- Bearbeiten
- LL-Parsing und First- und Follow-Set

HEINRICH HEINE

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Fazit Einleitung Stand Vorher Änderungen Live-Demo

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - · LL-Parsing und First- und Follow-Set
- Automaten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

-Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - LL-Parsing und First- und Follow-Set
- Automaten
 - Anzeigen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - · LL-Parsing und First- und Follow-Set
- Automaten
 - Anzeigen
 - Bearbeiten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - · LL-Parsing und First- und Follow-Set
- Automaten
 - Anzeigen
 - Bearbeiten
 - Minimalisieren

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Fazit Einleitung Stand Vorher Änderungen Live-Demo

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - · LL-Parsing und First- und Follow-Set
- Automaten
 - Anzeigen
 - Bearbeiten
 - Minimalisieren
 - ϵ -Übergänge entfernen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Stand Vorher



Datentypen

- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Datentypen



- Grammatiken
 - Anzeigen
 - Bearbeiten
 - · LL-Parsing und First- und Follow-Set
- Automaten
 - Anzeigen
 - Bearbeiten
 - Minimalisieren
 - ϵ -Übergänge entfernen
 - Vervollständigen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Datentypen



- 1. Was konnte die Toolbox mit den Daten machen?
- 2. Grammatiken
- 3. anzeigen
- 4. und bearbeiten
- 5. LL-Parsing-Tabelle, First- und Follow-Set
- 6. Automaten
- 7. anzeigen
- 8. und bearbeiten
- 9. minimalisieren
- 10. epsilon-Übergänge entfernen
- 11. Vervollständigen

Einleitung Stand Vorher

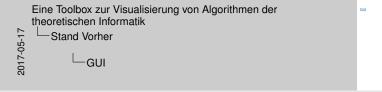
Änderungen

Live-Demo

Fazit



GUI



GUI Grammar Modus, Menü wechseln

Einleitung Stand Vorher

Änderungen

Live-Demo

Fazit

System of Special

GUI

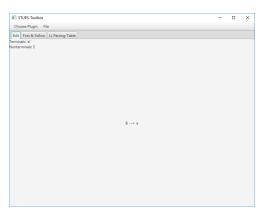
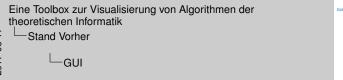


Figure: Die GUI hatte zwei Modi, Grammar und Automaton. Über die Bedienleiste lässt sich zwischen den Modi wechseln.





GUI Grammar Modus, Menü wechseln

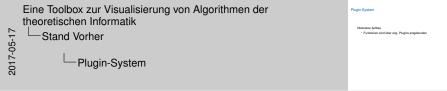
HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit

Plugin-System

Modularer Aufbau

• Funktionen sind über sog. Plugins eingebunden



1. Funktionen = Plugins

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit

Plugin-System

Modularer Aufbau

- Funktionen sind über sog. Plugins eingebunden
- ein Plugin ist eine Klasse, die bestimmtes Interface implementiert





- 1. Funktionen = Plugins
- 2. Implementiert bestimmtes Interface

-Plugin-System

3. automatisch einbinden

Plugin-System

9/37



Modularer Aufbau

- · Funktionen sind über sog. Plugins eingebunden
- ein Plugin ist eine Klasse, die bestimmtes Interface implementiert
- in dem Plugin ist ein Input- und ein Output-Typ definiert, auf dem das Plugin arbeitet

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Stand vorher

- 1. Funktionen = Plugins
- 2. Implementiert bestimmtes Interface

-Plugin-System

- 3. automatisch einbinden
- 4. hat Input-Typ und Output-Typ, bekommt Instanz Input, Instanz Output zurück

Plugin-System



Modularer Aufbau

- · Funktionen sind über sog. Plugins eingebunden
- ein Plugin ist eine Klasse, die bestimmtes Interface implementiert
- in dem Plugin ist ein Input- und ein Output-Typ definiert, auf dem das Plugin arbeitet
- · die Plugins verändern ggf. das Objekt, auf dem sie arbeiten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

--Plugin-System

- Plugin-System
- Funktionen sind über sog, Pfugins eingebunden
 ein Plugin ist eine Klasse, die bestimmtes Interlace implementiert
 ein dem Pfugin ist ein Inpst- und ein Output-Typ definiert, auf dem dies Pfugin arbeitet
 die Pfugins verändern ggl. dies Objekt, auf dem sie arbeiten

- 1. Funktionen = Plugins
- 2. Implementiert bestimmtes Interface
- 3. automatisch einbinden
- 4. hat Input-Typ und Output-Typ, bekommt Instanz Input, Instanz Output zurück
- 5. Instanz verändern

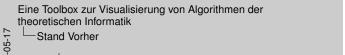
Plugin-System

9 / 37



Modularer Aufbau

- · Funktionen sind über sog. Plugins eingebunden
- · ein Plugin ist eine Klasse, die bestimmtes Interface implementiert
- in dem Plugin ist ein Input- und ein Output-Typ definiert, auf dem das Plugin arbeitet
- die Plugins verändern ggf. das Objekt, auf dem sie arbeiten
- je Typ nur eine Instanz



Modularer Aufbau

Funktionen sind über sog, Plugins eingetzunden

ein Plugin ist eine Klasse, die bestimmtes Interface implementiert

ein dem Plugin ist ein hyzu-trei dei Output-Typ definiert, auf dem das
Plugin arbeitet

die Plugin verlandern ggf, das Objekt, auf dem sie arbeiten

Plugin-Systen

ie Typ nur eine Instanz

1. Funktionen = Plugins

-Plugin-System

- 2. Implementiert bestimmtes Interface
- 3. automatisch einbinden
- 4. hat Input-Typ und Output-Typ, bekommt Instanz Input, Instanz Output zurück
- 5. Instanz verändern
- 6. je typ nur eine Instanz, neue Grammatik: alte Speichern, neue Laden

Plugin-System

9 / 37



Modularer Aufbau

- · Funktionen sind über sog. Plugins eingebunden
- · ein Plugin ist eine Klasse, die bestimmtes Interface implementiert
- in dem Plugin ist ein Input- und ein Output-Typ definiert, auf dem das Plugin arbeitet
- die Plugins verändern ggf. das Objekt, auf dem sie arbeiten
- je Typ nur eine Instanz
- leichte Weiterentwicklung ohne Einarbeitung in den Legacy-Code möglich

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

— Stand Vorher

— Plugin-System

| Plugin-System | Plugin-System | Plugin standard of the Code for the Code f

- 1. Funktionen = Plugins
- 2. Implementiert bestimmtes Interface
- 3. automatisch einbinden
- 4. hat Input-Typ und Output-Typ, bekommt Instanz Input, Instanz Output zurück
- 5. Instanz verändern
- 6. je typ nur eine Instanz, neue Grammatik: alte Speichern, neue Laden
- 7. Weiterentwicklung

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Fazit

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.



HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

Modifizierbarkeit der Objekte

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Probleme

Problems

Alter: Für die Weiterminischung ergaben sich einige Problems.

- Modifizierberbeit der Clipake

Probleme Weiterentwicklung Kurz erläutern, dann weiter

1. Objekte waren modifizierbar

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Fazit

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

and vomer

-Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

Modifizierbarkeit der Objekte
Plugins verladerten das Objekt. Keine einfache Speichenung von Vorgingerversionen möglich.

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Fazit

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Aber: Für die Weiterenteicklung ergeben sich einige Probleme.

- Moditzierbankeit der Öbjekte

- Pugne weiterbene dur Clijkt. Keine einfache Speichenung von

- keine Vergleichbankeit möglich.

Probleme Weiterentwicklung Kurz erläutern, dann weiter

-Probleme

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit

HEINRICH HEINE

Fazit

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit
 - · Zwei Nichtterminale mit gleichem Namen waren nicht gleich

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Probleme

Aber: Für die Weiterenheicklung ergaben sich einige Probleme.

Modfüserbankeit der Übijskle

Plugins weldsderen des Digisk. Keine eintsche Speichenung vor Vorglagerwestionen möglich.

kains Wergleichbankeit

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit
- 4. genau diesselbe Referenz

HEINRICH HEINE

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit
 - Zwei Nichtterminale mit gleichem Namen waren nicht gleich
 - bei Algorithmen mussten immer genau dieselben Referenzen verwendet werden

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Probleme

Aber: Für die Weiterenheicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Chijekte
- Pulpris verländerin das Chijekt. Keine einbache Speichenung von Verglingerverstinnen möglich.

- keine Vergleichbarkeit
- Zwei Nichterwinste mit gleichen Namen waren nicht gleich

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit
- 4. gleicher Name, nicht gleich
- 5. genau diesselbe Referenz



Probleme

10 / 37



Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit
 - · Zwei Nichtterminale mit gleichem Namen waren nicht gleich
 - bei Algorithmen mussten immer genau dieselben Referenzen verwendet werden
- Build-Prozess mit Makefile

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Probleme

Plugins verländerlen das Chijekt. Naine einbache Speicherung von begüngenweistenen nöglich.
 keinns Vergleichberkost
 zweis Nichtseminale mit gleichen Namen waren nicht gleich bei Algotifnen massen immer geraus dieselben Referenzen verwen werden.

Aber: Für die Weiterentwicklung ergaben sich einige Problem

- Modifizierbarkeit der Obiekte

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit
- 4. gleicher Name, nicht gleich
- 5. genau diesselbe Referenz
- 6. MakeFile, Linux-Befehlen

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit
 - · Zwei Nichtterminale mit gleichem Namen waren nicht gleich
 - bei Algorithmen mussten immer genau dieselben Referenzen verwendet werden
- Build-Prozess mit Makefile
 - kein plattformunabhängiger Build-Prozess möglich

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Stand Vorher

-Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Problem Plugins veränderten das Obiekt. Keine einfache Speicherung von Zwei Nichtterminale mit gleichem Namen waren nicht gleich bei Algorithmen mussten immer genau dieselben Referen. kein plattormunabhängiger Build-Prozess möglich

Build-Prozess mit Makefile

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit
- 4. gleicher Name, nicht gleich
- 5. genau diesselbe Referenz
- 6. MakeFile, Linux-Befehlen
- 7. nicht plattformunabhängig, blöd



Probleme

10 / 37



Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte
 - · Plugins veränderten das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
- keine Vergleichbarkeit
 - · Zwei Nichtterminale mit gleichem Namen waren nicht gleich
 - bei Algorithmen mussten immer genau dieselben Referenzen verwendet werden
- Build-Prozess mit Makefile
 - kein plattformunabhängiger Build-Prozess möglich
 - Erstellung einer JAR nicht möglich

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik Stand Vorher

> Zwei Nichtterminale mit gleichem Namen waren nicht gleich bei Algorithmen mussten immer genau dieselben Referen. kein platformunabhängiger Build-Prozess möglich
> Entiellung einer JAR nicht möglich

Probleme Weiterentwicklung Kurz erläutern, dann weiter

-Probleme

- 1. Objekte waren modifizierbar
- 2. keine einfache Speicherung, Deep-Copy
- 3. keine Vergleichbarkeit
- 4. gleicher Name, nicht gleich
- 5. genau diesselbe Referenz
- 6. MakeFile, Linux-Befehlen
- 7. nicht plattformunabhängig, blöd
- 8. keine Jar, externe Bibliotheken

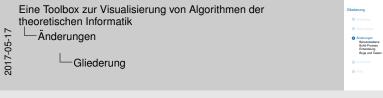
Aber: Für die Weiterentwicklung ergaben sich einige Problem Plugins veränderten das Obiekt. Keine einfache Speicherung vo

Gliederung

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- 5 Fazit

p. phrim



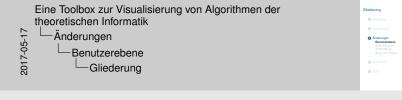
Änderungen

Gliederung

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Rugs und Testen
- 4 Live-Demo
- 5 Fazit





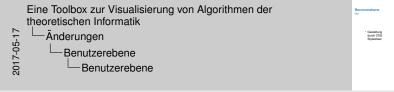
Benutzerebene

GUI



Fazit

 Gestaltung durch CSS-Stylesheet



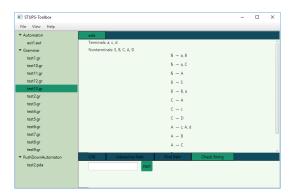
zunächst zur GUI

1. css-Stylesheet

Einleitung Stand Vorher Änderungen

Benutzerebene GUI

- Gestaltung durch CSS-Stylesheet
- · Design an bekannten **IDEs** orientiert.



Live-Demo

Fazit

Figure: die GUI in der Toolbox 2.0

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik 2017-05-17 -Änderungen Gestaltung durch CSS-Stylesheet Design an bekannten IDEs orientiert. -Benutzerebene -Benutzerebene

zunächst zur GUI

- 1. css-Stylesheet
- 2. GUI m Standard-Farbschema



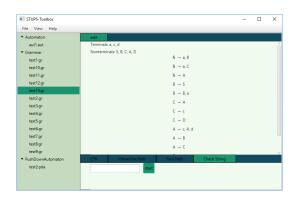


Einleitung Stand Vorher

Änderungen

Benutzerebene

- Gestaltung durch CSS-Stylesheet
- Design an bekannten IDEs orientiert.



Live-Demo

Fazit

Figure: die GUI in der Toolbox 2.0

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen

Benutzerebene

Benutzerebene

Fact dit not beken 21

zunächst zur GUI

2017-05-17

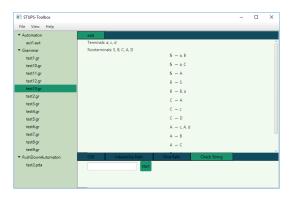
- 1. css-Stylesheet
- 2. GUI m Standard-Farbschema
- 3. IDES. Ei-Di-Is, Links Baum mit Objekten, rechts Bearbeiten, unten Funktionalität

Einleitung Stand Vorher

Änderungen

Benutzerebene

- Gestaltung durch CSS-Stylesheet
- Design an bekannten IDEs orientiert.
- Unicode-Zeichen
 Darstellung



Live-Demo

Fazit

Figure: die GUI in der Toolbox 2.0

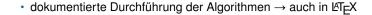


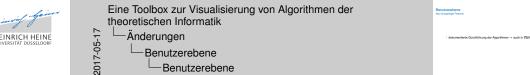
zunächst zur GUI

- 1. css-Stylesheet
- 2. GUI m Standard-Farbschema
- 3. IDES. Ei-Di-Is, Links Baum mit Objekten, rechts Bearbeiten, unten Funktionalität
- 4. Sonderzeichen Unicode

Benutzerebene

Neu hinzugefügte Features



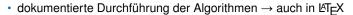


neu hingefügte Features

1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte

Benutzerebene

Neu hinzugefügte Features



paralleles Arbeiten auf mehreren Objekten

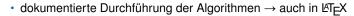


neu hingefügte Features

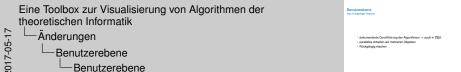
- 1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte
- 2. parallel, mehrere Objekte. Vorher: eins

Benutzerebene

Neu hinzugefügte Features



- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen



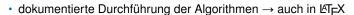
neu hingefügte Features

- 1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte
- parallel, mehrere Objekte. Vorher: eins
 Änderungen rückgängig

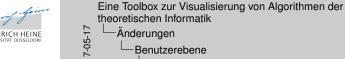
Benutzerebene

14 / 37

Neu hinzugefügte Features



- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen
- Automatisches Speichern der Änderung beim Beenden



Benutzerebone

- Identifyriger Frame

- datumenfarre Dundführung der Algorithmen – auch in Elgi
- paralleta Arbeine auf mehrene Cipiden

- Rüdgligige mehre
- Rüdgligige mehre
- Ausmerhalten Syndrem der Anderung beim Benden

neu hingefügte Features

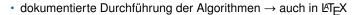
- 1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte
- 2. parallel, mehrere Objekte. Vorher: eins

-Benutzerebene

- 3. Änderungen rückgängig
- 4. automatisch Speichern Beenden. vorher: manuell

Benutzerebene

Neu hinzugefügte Features



- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen
- Automatisches Speichern der Änderung beim Beenden
- Kellerautomaten als neuer Datentyp

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

ICH HEINE Anderungen

Anderungen

Benutzerebene

Benutzerebene

Bendundsbere
tes Insegnign Fauren

- dakumentieris Dunfüllinung der Alpprötnen – auch in 193

- parallelas Andelm auf denheum Coljabere

- Müchgleige michale – Anderung beim Benden

- Autrestücken Spechen ein Anderung beim Benden

- Autrestücken Spechen ein Anderung beim Benden

- Valenstellichen sin von Disning

neu hingefügte Features

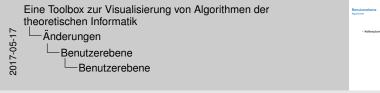
- 1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte
- 2. parallel, mehrere Objekte. Vorher: eins
- 3. Änderungen rückgängig
- 4. automatisch Speichern Beenden. vorher: manuell
- 5. Kelleratuomaten neu

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Fazit

Benutzerebene Algorithmen

Kellerautomaten



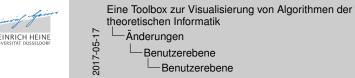
welche Algorithmen wurden implementiert?

1. Zu Kellerautomaten:

Benutzerebene

Algorithmen

- Kellerautomaten
 - Umwandeln zu Grammatik





- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik

Benutzerebene

Algorithmen





- Umwandeln zu Grammatik
- · Schritt-für-Schritt-Durchlauf

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Länderungen
Benutzerebene
Benutzerebene

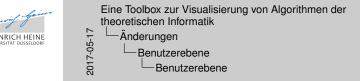


- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf

Benutzerebene

Algorithmen

- Kellerautomaten
 - Umwandeln zu Grammatik
 - Schritt-für-Schritt-Durchlauf
- Grammatik





- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik

Benutzerebene

Algorithmen



Fazit



- Umwandeln zu Grammatik
- · Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Benutzerebene
Benutzerebene



- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch

Benutzerebene

Algorithmen



- Kellerautomaten
 - Umwandeln zu Grammatik
 - · Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK
 - Umwandeln in Chomsky-Normal-Form

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

- Änderungen
Benutzerebene
Benutzerebene



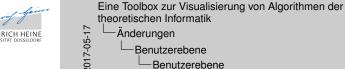
- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form

Benutzerebene

Algorithmen



- Umwandeln zu Grammatik
- · Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK
 - Umwandeln in Chomsky-Normal-Form
 - Entfernen von λ -Regeln





- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form
- 7. Entfernen von lambda-Regeln

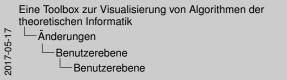
Benutzerebene

Algorithmen



- · Umwandeln zu Grammatik
- Uniwandein zu Grammatik
- Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK
 - Umwandeln in Chomsky-Normal-Form
 - Entfernen von λ -Regeln
 - Entfernen von einfachen Regeln

HEINRICH HEINE UNIVERSITÄT DÜSSELDORI





- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form
- 7. Entfernen von lambda-Regeln
- 8. Entfernen von einfachen Regeln

Benutzerebene

Algorithmen

- Kellerautomaten
 - Umwandeln zu Grammatik
 - · Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK
 - Umwandeln in Chomsky-Normal-Form
 - Entfernen von λ -Regeln
 - Entfernen von einfachen Regeln
 - Umwandeln in Kellerautomat

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik 7-05-17 -Änderungen -Benutzerebene -Benutzerebene



- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form
- 7. Entfernen von lambda-Regeln
- 8. Entfernen von einfachen Regeln
- 9. Umwandeln zu Kellerautomat

Benutzerebene

Algorithmen



- Umwandeln zu Grammatik
- Schritt-für-Schritt-Durchlauf
- Grammatik
 - CYK
 - Umwandeln in Chomsky-Normal-Form
 - Entfernen von λ -Regeln
 - Entfernen von einfachen Regeln
 - · Umwandeln in Kellerautomat
 - · interaktives Ableiten



Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen

Benutzerebene

Benutzerebene

Kellerauterraten

- Umwadeln zu Granmalik

- Schrie Si-Schrie Durhauf

Granmalik

- Umwadeln in Chonsky-Normal-Forn

- Erdense von 3-Ragele

- Erdense von 3-Ragele

- Umwadeln in Chonsky-Normal-Forn

- Erdense von 3-Ragele

- Umwadeln in Melerauteral

- Interaktives Ablellen

- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form
- 7. Entfernen von lambda-Regeln
- 8. Entfernen von einfachen Regeln
- 9. Umwandeln zu Kellerautomat
- 10. interaktives Ableiten

Benutzerebene

Algorithmen

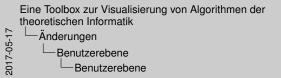


Kellerautomaten

- Umwandeln zu Grammatik
- · Schritt-für-Schritt-Durchlauf

Grammatik

- CYK
- Umwandeln in Chomsky-Normal-Form
- Entfernen von λ -Regeln
- Entfernen von einfachen Regeln
- Umwandeln in Kellerautomat
- · interaktives Ableiten
- · automatisches Ableiten



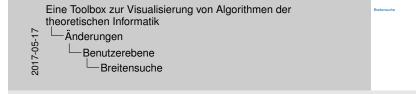


- 1. Zu Kellerautomaten:
- 2. Kellerautomat zu Grammatik
- 3. Schritt-für-Schritt Durchlauf
- 4. Zusätzlich Grammatik
- 5. CYK. Wortproblem, kubisch
- 6. Chomsky-Normal-Form
- 7. Entfernen von lambda-Regeln
- 8. Entfernen von einfachen Regeln
- Umwandeln zu Kellerautomat
- 10. interaktives Ableiten
- 11. automatisches Suchen

Breitensuche



Fazit



Einleitung

Stand Vorher

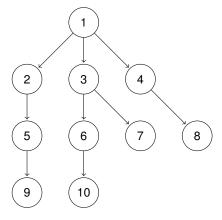
Änderungen

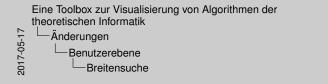
Live-Demo

Fazit



Breitensuche





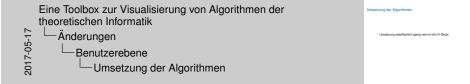


Fazit

fracional fracional

Umsetzung der Algorithmen

• Umsetzung oberflächlich genau wie im Info IV-Skript



1. oberflächlich Info 4 Skript

Umsetzung der Algorithmen



Fazit

- · Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung f
 ür Computer notwendig

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Benutzerebene
Umsetzung der Algorithmen



Umsetzung der Algorithmen

- 1. oberflächlich Info 4 Skript
- 2. Anpassung Computer

Einleitung Stand Vorher Änderungen

Umsetzung der Algorithmen



2017-05-17

Fazit

Live-Demo

- · Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung f
 ür Computer notwendig
- Beispiel: Finde alle Zykel $B_1 \to B_2, \dots, B_n \to B_1$

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Benutzerebene
Umsetzung der Algorithmen

Umsetzung oberflächlich genau wie im Info IV-Skript

 aber: Anpassung für Computer notwensig
 Belagsleit Finde alle Zylet B₁ = B₂ = ... B_n = B_n

Umsetzung der Algorithmen

- 1. oberflächlich Info 4 Skript
- 2. Anpassung Computer
- 3. Beispiel: Finden von Zykeln

otalia vollioi

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

2017-05-17

Fazit

Umsetzung der Algorithmen

17 / 37

- Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung f
 ür Computer notwendig
- Beispiel: Finde alle Zykel $B_1 \to B_2, \dots, B_n \to B_1$
- für Menschen auf einen Blick lösbar

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Benutzerebene
Umsetzung der Algorithmen

· Umsetzung oberflächlich genau wie im Info IV-Skript

für Menachen auf einen Blick lösbar

Umsetzung der Algorithmen

- 1. oberflächlich Info 4 Skript
- 2. Anpassung Computer
- 3. Beispiel: Finden von Zykeln
- 4. Für Menschen klar

CI

HEINRICH HEINE

7-05-17

Fazit

Umsetzung der Algorithmen

17 / 37

- Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung f
 ür Computer notwendig
- Beispiel: Finde alle Zykel $B_1 \to B_2, \dots, B_n \to B_1$
- für Menschen auf einen Blick lösbar
- Computer muss Tiefensuche laufen lassen und Rückwärtskanten bestimmen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Benutzerebene
Umsetzung der Algorithmen

- Umsatzung oberflächlich gerass wie im Info IV-Skript
- sber: Angessaung für Computer notwendig
- Beispelt: Finds alle Zylek 8, - - 86, --- 8, --- 8,
- IV Menschen auf einem Bild: Röber
- Computer mas Terfensuche laufen lassen und Rückwärtskarten
- besternen.

Umsetzung der Algorithmen

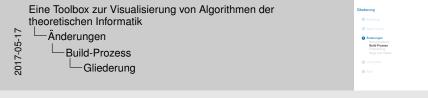
- 1. oberflächlich Info 4 Skript
- 2. Anpassung Computer
- 3. Beispiel: Finden von Zykeln
- 4. Für Menschen klar
- 5. Computer Tiefensuche

Gliederung

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

- 1 Einleitung
- Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- **5** Fazit

10. phrim

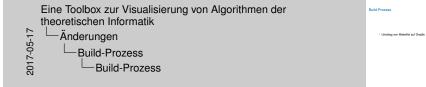


Build-Prozess



Fazit

• Umstieg von Makefile auf Gradle



- . BuildProzess
- 2. makeFile Gradle

Build-Prozess



Fazit

- · Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig



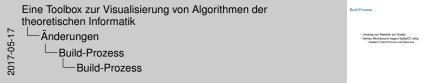
- 1. BuildProzess
- 2. makeFile Gradle
- 3. Workaround Sablecc

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Fazit

Build-Prozess

- · Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource

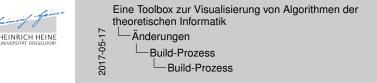


- . BuildProzess
- 2. makeFile Gradle
- 3. Workaround Sablecc
- 4. SableCC Source Resource, selber Ordner

Fazit



- · Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource
 - Probleme mit Gradle und getResource()



Umstieg von Makefile auf Gradle Meines Worksround wegen SableCC nötig

Build-Prozess

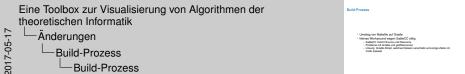
- BuildProzess
- 2. makeFile Gradle
- 3. Workaround Sablecc
- 4. SableCC Source Resource, selber Ordner
- 5. Probleme getResource()

Build-Prozess



Fazit

- · Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource
 - Probleme mit Gradle und getResource()
 - Lösung: Gradle-Skript, welches Dateien verschiebt und einige Zeilen im Code anpasst



- . BuildProzess
- 2. makeFile Gradle
- 3. Workaround Sablecc
- 4. SableCC Source Resource, selber Ordner
- 5. Probleme getResource()
- 6. Gradle-Skript, passt an

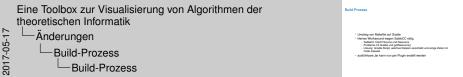
Build-Prozess

19 / 37



Fazit

- · Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource
 - Probleme mit Gradle und getResource()
 - Lösung: Gradle-Skript, welches Dateien verschiebt und einige Zeilen im Code anpasst
- ausführbare Jar kann nun per Plugin erstellt werden



- BuildProzess
- 2. makeFile Gradle
- 3. Workaround Sablecc
- 4. SableCC Source Resource, selber Ordner
- 5. Probleme getResource()
- 6. Gradle-Skript, passt an
- 7. JAR per Plugin, mit Dependencies

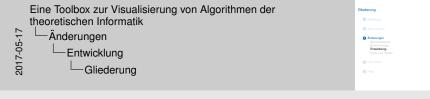
Gliederung



- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen

Entwicklung

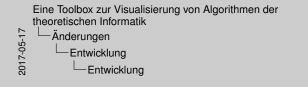
- 4 Live-Demo
- 5 Fazit



HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Entwicklung
Unmodifizierbarkeit I

Wunsch, vollführte Änderungen rückgängig zu machen



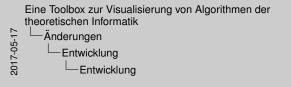
* Wursch, vollführte Änderungen rückgüngig zu machen

- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig

Entwicklung
Unmodifizierbarkeit I



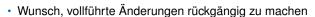
- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion



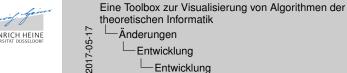


- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig
- 3. Referenz Vorgänger

Entwicklung Unmodifizierbarkeit I



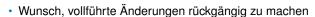
- hierfür Referenz auf Vorgängerversion
- · Problem: Plugins verändern das Objekt



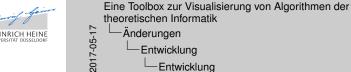


- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig
- 3. Referenz Vorgänger
- 4. Plugins verändern Objekt

Entwicklung
Unmodifizierbarkeit I



- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt
- Lösung: Unmodifizierbarkeit



wicklung diseaset i

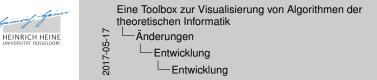
> Wursch, vollführte Änderungen rückgängig zu machen hierfür Referenz auf Vorgängerversion
> Hierfür Referenz auf Vorgängerversion
> Lösunz: Ummodifüriersterleit

- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig
- Referenz Vorgänger
 Pluging vorändern Objekt
- 4. Plugins verändern Objekt
- 5. Lösung: unmodifizierbarkeit

Entwicklung Unmodifizierbarkeit I



- hierfür Referenz auf Vorgängerversion
- · Problem: Plugins verändern das Objekt
- Lösung: Unmodifizierbarkeit
 - · Objekte nach Erstellung nicht mehr veränderbar



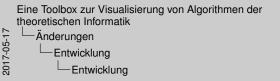
Wunsch, volführte Änderungen rückgengig zu machen hierfür Referenz auf Vorgängerversion Problem: Plugins verändern das Obiekt

- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig
- Referenz Vorgänger Plugins verändern Objekt
- 5. Lösung: unmodifizierbarkeit
- 6. nicht veränderbar

faining fain HEINRICH HEINE

Entwicklung Unmodifizierbarkeit I

- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt
- Lösung: Unmodifizierbarkeit
 - · Objekte nach Erstellung nicht mehr veränderbar
 - Zwingt dazu, neue Objekte anzulegen



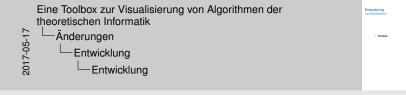
Wursich, vollübrte Änderungen rückgüngig zu machen
hierlür Reiferenz auf Vosglingseversion
Problem: Plugins werlandern das Objekt
Lössang: Umronförserhankel
Ospike noch Entsellung odet mehr werksidenbar
Zeind dass, neue Objekte noch seine

- 1. Als nächstes: Änderungen auf Entwickler-Ebene
- 2. Änderungen rückängig
- Referenz Vorgänger
 Plugins verändern Objekt
- 4. Plugins verandern Objekt
- 5. Lösung: unmodifizierbarkeit
- 6. nicht veränderbar
- 7. für Änderung neues Objekt



Vorteile





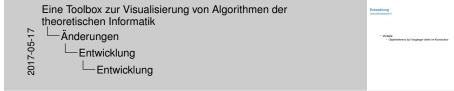
1. Umständlich? Viele Vorteile

Entwicklung Unmodifizierbarkeit II



· Objektreferenz auf Vorgänger direkt im Konstruktor





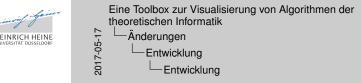
- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor

Änderungen Fazit Einleitung Stand Vorher Live-Demo

Entwicklung



- Vorteile
 - Objektreferenz auf Vorgänger direkt im Konstruktor
 - · keine Konflikte mehr mit Hashcode-Berechnung



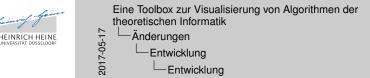


- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor
- 3. HashSet Konflikt

Entwicklung Unmodifizierbarkeit II



- Objektreferenz auf Vorgänger direkt im Konstruktor
- · keine Konflikte mehr mit Hashcode-Berechnung
- Anpassung kostete viel Zeit





- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor
- 3. HashSet Konflikt
- 4. Anpassung kostete Zeit

Entwicklung Unmodifizierbarkeit II



- Objektreferenz auf Vorgänger direkt im Konstruktor
- · keine Konflikte mehr mit Hashcode-Berechnung
- Anpassung kostete viel Zeit
 - Entfernung von set()-Methoden



```
Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

- Änderungen

Entwicklung

Entwicklung
```



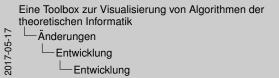
- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor
- HashSet Konflikt
- 4. Anpassung kostete Zeit
- 5. Setter entfernen

Entwicklung Unmodifizierbarkeit II



- Objektreferenz auf Vorgänger direkt im Konstruktor
- keine Konflikte mehr mit Hashcode-Berechnung
- Anpassung kostete viel Zeit
 - Entfernung von set()-Methoden
 - Anpassung der bestehenden Algorithmen + Testen







- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor
- 3. HashSet Konflikt
- 4. Anpassung kostete Zeit
- 5. Setter entfernen
- 6. Algorithmen anpassen + testen





- Objektreferenz auf Vorgänger direkt im Konstruktor
- keine Konflikte mehr mit Hashcode-Berechnung
- Anpassung kostete viel Zeit
 - Entfernung von set()-Methoden
 - Anpassung der bestehenden Algorithmen + Testen
- daher Automaton weiterhin modifizierbar



7-05-17

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Entwicklung
Entwicklung

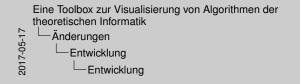


- 1. Umständlich? Viele Vorteile
- 2. Objektref Vorgänger Konstruktor
- 3. HashSet Konflikt
- 4. Anpassung kostete Zeit
- 5. Setter entfernen
- 6. Algorithmen anpassen + testen
- 7. Automaten modifizierbar

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Entwicklung Vergleichbarkeit

• Überschreiben der *equals()*- und *toHash()*-Methoden

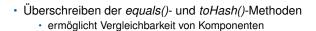


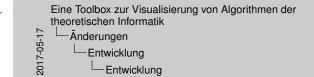


- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()

Gainvig

Entwicklung Vergleichbarkeit







- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit

Entwicklung Vergleichbarkeit

23 / 37



- Überschreiben der *equals()* und *toHash()*-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten
 - vereinfacht Algorithmen, da es nicht mehr nötig ist, genau die richtige Referenz zu haben

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

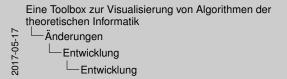


- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit
- 4. Vereinfacht Čode, nicht mehr genau Referenz



Entwicklung Vergleichbarkeit

- Überschreiben der equals()- und toHash()-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten
 - vereinfacht Algorithmen, da es nicht mehr nötig ist, genau die richtige Referenz zu haben
 - · Konsistenz in HashSets



wickdung

Observichweiten der equality- und britsahr)- Methoden

- emiglicht fregleichtwaket ein Kimpromete

- emiglicht fregleichtwaket ein Kimpromete

- werbeit die gestelleren, das nicht nich andig at, genau de nichtig

- Kinnellere in Kallicher

- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit
- 4. Vereinfacht Code, nicht mehr genau Referenz
- 5. gleiche Objekte, gleicher Hash

gainsif o

Entwicklung Vergleichbarkeit

- Überschreiben der equals()- und toHash()-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten
 - vereinfacht Algorithmen, da es nicht mehr nötig ist, genau die richtige Referenz zu haben
 - Konsistenz in HashSets
- Dafür mussten die Models umgeschrieben werden

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit
- 4. Vereinfacht Code, nicht mehr genau Referenz
- 5. gleiche Objekte, gleicher Hash
- 6. Models ändern

HEINRICH HEINE

Entwicklung Vergleichbarkeit

- Überschreiben der equals()- und toHash()-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten
 - vereinfacht Algorithmen, da es nicht mehr nötig ist, genau die richtige Referenz zu haben
 - Konsistenz in HashSets
- Dafür mussten die Models umgeschrieben werden
- natürlich immer noch keine Möglichkeit, zwei Grammatiken auf Äguivalenz zu überprüfen

Classon/Involver der agsaldig und befaalt/c Mathodolen
 emiglicht Vergeschland et van Komponismen
 verschold Ryginframm, der sollt mit mit gal, gemu die rüchige
 Komitation in Kandide
 Mathodolen in Kandide
 Daller masster die Mohale ungeschrieben ewelden
 Applichten und Schriften in Anderen
Applichten und Dereptilen

- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit
- 4. Vereinfacht Code, nicht mehr genau Referenz
- 5. gleiche Objekte, gleicher Hash
- 6. Models ändern

HEINRICH HEINE

Entwicklung Vergleichbarkeit

- Überschreiben der equals()- und toHash()-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten
 - vereinfacht Algorithmen, da es nicht mehr nötig ist, genau die richtige Referenz zu haben
 - Konsistenz in HashSets
- Dafür mussten die Models umgeschrieben werden
- natürlich immer noch keine Möglichkeit, zwei Grammatiken auf Äguivalenz zu überprüfen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

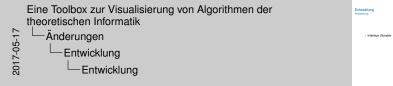
Oberschreiben der equalit/- und tof-fash/j-Methoden
 werden der equalit/- und tof-fash/j-Methoden
 werden der einer der einer der eine der einde ernerheite Angerheimen des erlott mehr relig ist, geneu die notingen Elleren ern in haben.
 Daller mussten die Modelle umgeschrieben werden, aus der einer mehr haben der einer oden keine Modelle-, zwei Chammestiken aus die nacht der einer der eine Methodel eine der einer der eine Methodel einer der einer d

- 1. Objekte nicht vergleichbar
- 2. Für Vergleichbarkeit: Überschreiben equals(), to Hash()
- 3. dadurch Vergleichbarkeit
- 4. Vereinfacht Code, nicht mehr genau Referenz
- 5. gleiche Objekte, gleicher Hash
- 6. Models ändern
- 7. Grammatiken äquivalent?

Entwicklung Persistierung

Interface Storable

HEINRICH HEINE UNIVERSITÄT DÜSSELDOR

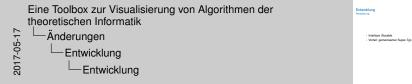


- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen



- Interface Storable
- Vorteil: gemeinsamer Super-Typ

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

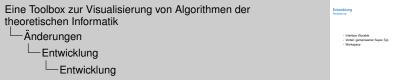


- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen



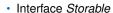
- Interface Storable
- Vorteil: gemeinsamer Super-Typ
- Workspace





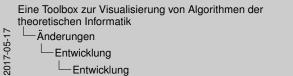
- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen

Entwicklung Persistierung



- Vorteil: gemeinsamer Super-Typ
- Workspace
- · wird beim Beenden auf der Festplatte gespeichert



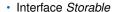




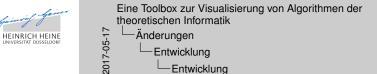
- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen



24 / 37



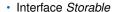
- Vorteil: gemeinsamer Super-Typ
- Workspace
- wird beim Beenden auf der Festplatte gespeichert
- · selbes Verzeichnis wie jar





- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen

Entwicklung Persistierung



- Vorteil: gemeinsamer Super-Typ
- Workspace
- · wird beim Beenden auf der Festplatte gespeichert
- selbes Verzeichnis wie jar
- · config Datei

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Light Dusselborg

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen

Entwicklung

Entwicklung

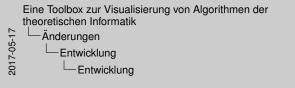
Interface Stinable
Vortale germinamer Super-Typ
Workspace
wird beim Bernden auf der Festplatte gespeichere selben Verzeichnis wie jar
config Daaie

- 1. Interface Storable
- 2. Vorteil: Super-Typ
- 3. Workspace, Objekte im Speicher, Baum, Pendant Festplatte
- 4. Beenden Festplatte
- 5. Verzeichnis jar
- 6. Config-Datei, Benutzer Änderungen

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Entwicklung LATEX-Generierung

• im LaTEX-Modus Schreiben von kompilierbaren LaTEX-Dateien



Erteicklung
Fig Sammung

- im #Egif Mobale Schreiben von komplienbaren Efigif. Odesen

Latex, Erklärung wie

1. Ziel: Erstellung Latex File

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Entwicklung LATEX-Generierung

- im *LTEX-Modus* Schreiben von kompilierbaren LTEX-Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

- 1. Ziel: Erstellung Latex File
- 2. alles, was gemacht wird
- 3. Lerneffekt, Skript

Entwicklung

LATEX-Generierung



- im *LTEX-Modus* Schreiben von kompilierbaren LTEX-Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten
- manuelles Starten und Beenden

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

Entwicklung #Ex-Generating

im BTgX-Modus Schreiben von komplierbaren bTgX-Dateien
 dokumentierte Ausführung von Algorithmen oder Drucken von Objekten

- 1. Ziel: Erstellung Latex File
- 2. alles, was gemacht wird
- 3. Lerneffekt, Skript
- 4. manuell starten + beenden

Fazit Gainsief

Entwicklung LATEX-Generierung

- im *LTEX-Modus* Schreiben von kompilierbaren LTEX-Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten
- manuelles Starten und Beenden
- Herausforderungen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

Entwicklung #T_CX-Generators

im MD/Modus Schneiben von kompilierbaren MD/M-Datelen
 dokumentierbe Ausführung von Algorithmen oder Drucken von Objekten
 manuelles Starten und Beenden
 Hetrausforderungen

- 1. Ziel: Erstellung Latex File
- 2. alles, was gemacht wird
- 3. Lerneffekt, Skript
- 4. manuell starten + beenden
- 5. Welche Herausforderungen?

HEINRICH HEINE

Entwicklung LATEX-Generierung

- im *LTEX-Modus* Schreiben von kompilierbaren LATEX-Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten
- manuelles Starten und Beenden
- Herausforderungen
 - Sonderzeichen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Entwicklung
Entwicklung

im Eligit Modus Schreiben von kompilierbaren Eligit Dateien
 dokumenteren Ausführung von Appriltmen oder Drucken von Objekten
 masselalen Stefens und Reunden

Entwicklung

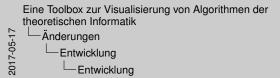
- Herausforderungen

- 1. Ziel: Erstellung Latex File
- 2. alles, was gemacht wird
- 3. Lerneffekt, Skript
- 4. manuell starten + beenden
- 5. Welche Herausforderungen?
- 6. Sonderzeichen

faire of figer

Entwicklung LATEX-Generierung

- im LaTEX-Modus Schreiben von kompilierbaren LaTEX-Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten
- manuelles Starten und Beenden
- Herausforderungen
 - Sonderzeichen
 - Automaten



im Effgit Abodus Schreiben von kompilierbaren Effgit Dateien
 dokumenteren Ausführung von Algorifferen oder Drucken von Chijekten
 masselalen Sterlen und Reund

Entwicklung

Herausforderungen

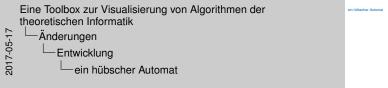
Sonderzeichen
 Automaten

- 1. Ziel: Erstellung Latex File
- 2. alles, was gemacht wird
- 3. Lerneffekt, Skript
- 4. manuell starten + beenden
- 5. Welche Herausforderungen?
- 6. Sonderzeichen
- 7. generisch Automaten

Einleitung Stand Vorher Änderungen Live-Demo Fazit Eine Toolbox zur Visualisierung von Al

ein hübscher Automat





waagerecht, alphanumerisch, Winkel

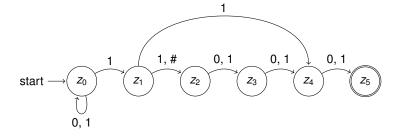
Einleitung Stand Vorher Änderungen

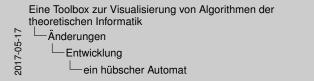
Live-Demo

Fazit



ein hübscher Automat





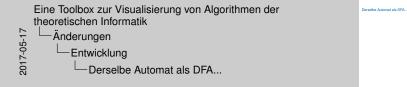


waagerecht, alphanumerisch, Winkel

Derselbe Automat als DFA...



Fazit

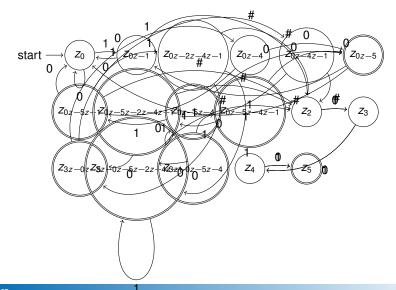


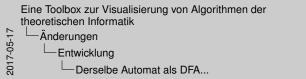
dieser Automato DFA



Fazit

Derselbe Automat als DFA...







dieser Automato DFA

Gliederung

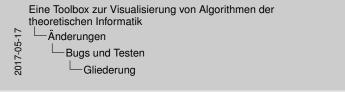
HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

- Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess

Bulld-Prozess
Entwicklung
Bugs und Testen

- 4 Live-Demo
- **5** Fazit







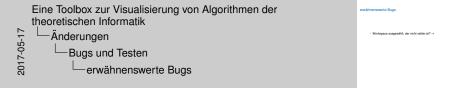
einige Herausforderungen, Bugs + Testen

HEINRICH HEINE

Fazit

erwähnenswerte Bugs

Workspace ausgewählt, der nicht valide ist? →



1. Workspace nicht valide?



erungen Liv



Fazit

erwähnenswerte Bugs

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!



Workspace susgewählt, der nicht valide ist? →
 der ganze Ordner wird gelöscht!

- 1. Workspace nicht valide?
- 2. Ordner gelöscht

Einleitung Stand Vorher Änderungen



Fazit

Live-Demo

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? →



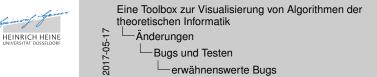
- 1. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten



Fazit

erwähnenswerte Bugs

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? →
- Exception!



- Workspace ausgewählt, der nicht valide ist? →
der ganze Ordner wird gelöscht!
ganz großen Automaten anzeigen lassen? →
Exception!

- 1. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!

Einleitung Stand Vorher Änderungen

erwähnenswerte Bugs



Fazit

Live-Demo

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? \rightarrow
- Exception!
- Grammatik eingegeben, die Schwachsinn ist? →

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Bugs und Testen
erwähnenswerte Bugs

Workspace ausgewählt, der nicht valide ist? →
der ganze Ordner wird gelüscht!
ganz großen Automaten anzeigen lassen? →
Exception!
Grammatik eingegeben, die Schwachsinn ist? →

- 1. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!
- 5. ungültige Grammatik

erwähnenswerte Bugs



Fazit

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? →
- Exception!
- Grammatik eingegeben, die Schwachsinn ist? →
- · Grammatik weg!

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Characterischen Informatik

Anderungen

Bugs und Testen

erwähnenswerte Bugs

Workspace ausgewählt, der nicht valide ist? →
der ganze Ordere wird gelücht!
ganz großen Automaten anzeigen lassen? →
Exception!
Crammalk eingegeben, die Schwachsinn ist? →
Crammalk eingeben, die Schwachsinn ist? →

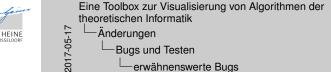
- I. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!
- 5. ungültige Grammatik
- 6. CNF Algo löscht alles

erwähnenswerte Bugs



Fazit

- Workspace ausgewählt, der nicht valide ist? →
- der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? \rightarrow
- Exception!
- Grammatik eingegeben, die Schwachsinn ist? →
- · Grammatik weg!

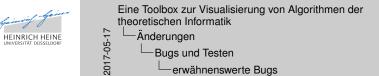


Workspace ausgewählt, der nicht valide ist? -der ganze Ordner wird geläscht!
ganz großen Automaten anzeigen lassen? -Exception!
Grammatik eingegeben, die Schwachsinn ist? --

- I. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!
- 5. ungültige Grammatik
- 6. CNF Algo löscht alles

Fazit

- Workspace ausgewählt, der nicht valide ist? →
- · der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? →
- Exception!
- Grammatik eingegeben, die Schwachsinn ist? →
- · Grammatik weg!





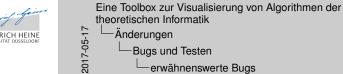
- Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!
- 5. ungültige Grammatik
- 6. CNF Algo löscht alles

erwähnenswerte Bugs



Fazit

- Workspace ausgewählt, der nicht valide ist? →
- · der ganze Ordner wird gelöscht!
- ganz großen Automaten anzeigen lassen? →
- Exception!
- Grammatik eingegeben, die Schwachsinn ist? →
- Grammatik weg!





- 1. Workspace nicht valide?
- 2. Ordner gelöscht
- 3. große Automaten
- 4. ArrayOutOfBoundException!
- 5. ungültige Grammatik
- 6. CNF Algo löscht alles

Einleitung Stand Vorher Änd

Änderungen Live-Demo

Fazit



Testen

Legacy-Code



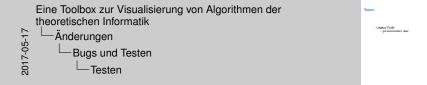
- Testen
- 2. Legacy-Code

Testen



Fazit

- Legacy-Code
 - gut kommentiert, aber



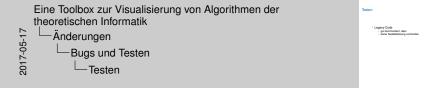
- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung

Testen



Fazit

- Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden



- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben

Testen



- Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden
 - es gibt einige Bugs, die nicht behoben werden konnten





- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen

Testen



- Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden
 - es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Anderungen
Bugs und Testen
Testen



- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree

HEINRICH HEINE

Testen

- · Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden
 - es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?





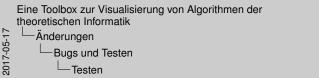
- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz

Testen



Legacy-Code

- · gut kommentiert, aber
- keine Testabdeckung vorhanden
- es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?
 - nicht möglich: Vergleich von Grammatiken/Automaten





- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz
- 8. manuell, kostete Zeit

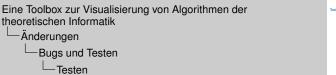
Testen

30 / 37



Legacy-Code

- gut kommentiert, aber
- keine Testabdeckung vorhanden
- es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?
 - nicht möglich: Vergleich von Grammatiken/Automaten
 - manuelles Überprüfen kostete viel Zeit



1. Testen

7-05-17

- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz
- 8. manuell, kostete Zeit
- 9. Latex



Testen



- · Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden
 - es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?
 - nicht möglich: Vergleich von Grammatiken/Automaten
 - manuelles Überprüfen kostete viel Zeit
- PLEX

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen

Bugs und Testen
Testen



1. Testen

7-05-17

- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz
- 8. manuell, kostete Zeit
- 9. Latex
- 10. manuell Kompilierbarkeit

Testen



- Legacy-Code
 - gut kommentiert, aber
 - keine Testabdeckung vorhanden
 - es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?
 - nicht möglich: Vergleich von Grammatiken/Automaten
 - manuelles Überprüfen kostete viel Zeit
- LATEX
 - · manuelles Überprüfen auf Kompilierbarkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik 7-05-17 Änderungen Bugs und Testen └─ Testen

es gibt einige Bugs, die nicht behaben werden konnte

- 1. Testen
- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz
- 8. manuell, kostete Zeit
- 9. Latex
- 10. manuell Kompilierbarkeit
- 11. Sonderfälle, lambda

Testen

30 / 37



Legacy-Code

- gut kommentiert, aber
- keine Testabdeckung vorhanden
- es gibt einige Bugs, die nicht behoben werden konnten
- Algorithmen
 - automatisches Überprüfen von Algorithmen: wird das richtige Ergebnis geliefert?
 - nicht möglich: Vergleich von Grammatiken/Automaten
 - manuelles Überprüfen kostete viel Zeit
- LATEX
 - · manuelles Überprüfen auf Kompilierbarkeit
 - viele Sonderfälle, die alle beachtet werden müssen (X_a_b,#,λ,usw.)

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen

Bugs und Testen

Testen

1. Testen

7-05-17

- 2. Legacy-Code
- 3. gut Kommentare, keine Testabdeckung
- 4. Bugs Automaten, nicht behoben
- 5. Algorithmen
- 6. manche Algorithmen: ja, Bspl LambdaFree
- 7. kein Überprüfen Äquivalenz
- 8. manuell, kostete Zeit
- 9. Latex
- 10. manuell Kompilierbarkeit
- 11. Sonderfälle, lambda

Gliederung

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- 5 Fazit



Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
Live-Demo
Gliederung

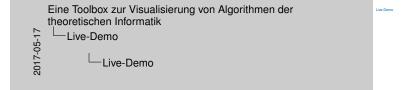
Einleitung Stand Vorher Änderungen

Live-Demo

Fazit

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Live-Demo



Gliederung



2017-05-17

- 1 Einleitung
- 2 Stand Vorher
- 3 Änderungen
 Benutzerebene
 Build-Prozess
 Entwicklung
 Bugs und Testen
- 4 Live-Demo
- 5 Fazit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Gliederung

Gliederung

Einleitung Stand Vorher

Änderungen

Live-Demo

Fazit



Herausforderungen

Unmodifizierbarkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Herausforderungen

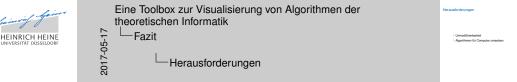
Herausforderungen

- 1. Unmodifizierbarkeit
- $2. \ \ \, \text{nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem}$

Fazit

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen f
 ür Computer umsetzen



Herausforderungen

- 1. Unmodifizierbarkeit
- 2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
- 3. Algorithmen Computer
- 4. Skript Info 4 entnomment, für Menschen, bedurften Anpassung



Einleitung Stand Vorher Änderungen Live-Demo

Herausforderungen



- Algorithmen f
 ür Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes

Eine Toolbox zur Visualisierung von Algorithmen der Herausforderunger theoretischen Informatik -Fazit

Fazit

-Herausforderungen

Herausforderungen

- 1. Unmodifizierbarkeit
- 2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
- 3. Algorithmen Computer
- 4. Skript Info 4 entnomment, für Menschen, bedurften Anpassung
- 5. Weiterentwicklung
- 6. doch nötig Code anzugucken
- 7. nicht einfacher oder schwerer, komplett anders



Einleitung Stand Vorher Änderungen Live-Demo

Fazit

Herausforderungen



- Algorithmen f
 ür Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex

34 / 37

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Herausforderungen

Herausforderungen

- 1. Unmodifizierbarkeit
- 2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
- 3. Algorithmen Computer
- 4. Skript Info 4 entnomment, für Menschen, bedurften Anpassung
- 5. Weiterentwicklung
- 6. doch nötig Code anzugucken
- 7. nicht einfacher oder schwerer, komplett anders
- 8. Latex
- 9. viele Probleme, viele Sonderfälle
- 10. bemüht, alle abzudecken.

Einleitung Stand Vorher Änderungen Live-Demo

Fazit

Herausforderungen



- Algorithmen f
 ür Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex
- Testen

34 / 37



Herausforderungen

- 1. Unmodifizierbarkeit
- 2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
- 3. Algorithmen Computer
- 4. Skript Info 4 entnomment, für Menschen, bedurften Anpassung
- 5. Weiterentwicklung
- 6. doch nötig Code anzugucken
- 7. nicht einfacher oder schwerer, komplett anders
- 8. Latex
- 9. viele Probleme, viele Sonderfälle
- 10. bemüht, alle abzudecken.
- 11. Testen
- 12. keine bestehende Testabdeckung, JUnit
- 13. schwierig, wegen Äquivalenz, hauptsächlich manuell

Stand Vorher

Änderungen

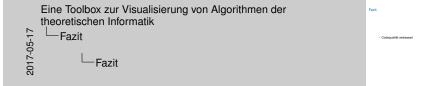
Live-Demo

Fazit



Fazit

Codequalität verbessert



- 1. Fazit
- 2. Codequalität: Aufgeräumt, MVC, Testabdeckung

Stand Vorher

Änderungen

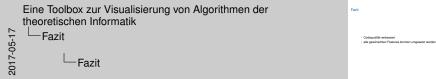
Live-Demo

Symmet of

Fazit

Fazit

- Codequalität verbessert
- alle gewünschten Features konnten umgesetzt werden



- 1. Fazit
- 2. Codequalität: Aufgeräumt, MVC, Testabdeckung
- 3. gewünschte Features umgesetzt

Stand Vorher

Änderungen

Live-Demo

Fazit



Fazit

- Codequalität verbessert
- alle gewünschten Features konnten umgesetzt werden
- GUI übersichtlich und gut zu bedienen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

-Fazit

Codequalität verbessert
 alle gewünschten Features konnten umgesetzt werden
 GUII übersichtlich und gut zu bedienen

- 1. Fazit
- 2. Codequalität: Aufgeräumt, MVC, Testabdeckung
- 3. gewünschte Features umgesetzt
- 4. GUI übersichtlich, gut zu bedienen

Stand Vorher

Änderungen

Live-Demo

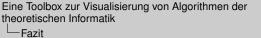
omo

Fazit



Fazit

- Codequalität verbessert
- alle gewünschten Features konnten umgesetzt werden
- GUI übersichtlich und gut zu bedienen
- trotz kleiner Bugs ein gut gelungenes Programm



/I-co-/II

Fazit

Codequalität verbessert
 alle gewünschen Features konnten umgesetzt werder
 GUI übersichtlich und gut zu bedienen
 troft kleiner Buns ein mit neitneueren Progressen.

- 1. Fazit
- 2. Codequalität: Aufgeräumt, MVC, Testabdeckung
- 3. gewünschte Features umgesetzt
- 4. GUI übersichtlich, gut zu bedienen
- 5. trotz kleiner Bugs, gut gelungen

Einleitung Stand Vorher Änderungen

Live-Demo **Fazit**



Ausblick

durchaus noch mehr Potential

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Ausblick

- 1. mehr Potential
- 2. Turing-Maschinen
- 3. Programmaufruf mit Parametern
- 4. Kontextsensitive Grammatiken

Einleitung Stand Vorher Änderungen Live-Demo

HEINRICH HEINI UNIVERSITÄT DÜSSELDOR

Fazit

Ausblick

- durchaus noch mehr Potential
- Turing-Maschinen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Ausblick

- 1. mehr Potential
- 2. Turing-Maschinen
- 3. Programmaufruf mit Parametern
- 4. Kontextsensitive Grammatiken

Änderungen Einleitung Stand Vorher

Live-Demo

Fazit



Ausblick

- · durchaus noch mehr Potential
- Turing-Maschinen
- Programmaufruf mit Parametern



- 1. mehr Potential
- 2. Turing-Maschinen
- 3. Programmaufruf mit Parametern
- 4. Kontextsensitive Grammatiken

Änderungen Einleitung Stand Vorher

Live-Demo

Fazit



Ausblick

36 / 37

- · durchaus noch mehr Potential
- Turing-Maschinen
- Programmaufruf mit Parametern
- kontextsensitive Grammatiken



- 1. mehr Potential
- 2. Turing-Maschinen

-Ausblick

- 3. Programmaufruf mit Parametern
- 4. Kontextsensitive Grammatiken

Turing-Maschinen Programmaufruf mit Parametern kontextuansitive Grammatikan

Einleitung Stand Vorher

Fazit

HEINRICH HEIN UNIVERSITÄT DÜSSELDO

Vielen Dank für die Aufmerksamkeit!

Änderungen

Live-Demo

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
Fazit

Vielen Dank für die Aufmerksamkeit!