

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Isabel Wingen

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf

18.05.2017

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen

Benutzerebene
Build-Prozess
Entwicklung
Bugs und Testen

4 Live-Demo

5 Fazit

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Einleitung

└─ Gliederung

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen
Benutzerebene
Build-Prozess
Entwicklung
Bugs und Testen

4 Live-Demo

5 Fazit

Einleitung

- Weiterführung der Bachelorarbeit von Fabian Ruhland

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Einleitung

└─ Einleitung

2017-05-17

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

Einleitung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Einleitung

1. Weiterführung der Arbeit von Fabian Ruhland
2. in Java geschrieben
3. Toolbox zur Visualisierung von Automaten und Grammatiken

Einleitung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Einleitung

1. Weiterführung der Arbeit von Fabian Ruhland
2. in Java geschrieben
3. Toolbox zur Visualisierung von Automaten und Grammatiken
4. Fokus: Automaten und Compilerbau

Einleitung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Einleitung

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

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

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

2017-05-17

└─ Einleitung

└─ Einleitung

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

- 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

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Einleitung

└─ Motivation

2017-05-17

Warum weiterentwickeln?

1. Funktionsumfang erweitern

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Einleitung

└─ Motivation

2017-05-17

Warum weiterentwickeln?

1. Funktionsumfang erweitern
2. Mehr Algorithmen für Grammatiken
3. Chomsky-Normalform oder CYK

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Motivation

Warum weiterentwickeln?

1. Funktionsumfang erweitern
2. Mehr Algorithmen für Grammatiken
3. Chomsky-Normalform oder CYK
4. Kommentierte Ausführung der Algorithmen
5. Latex

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Motivation

Warum weiterentwickeln?

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

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Einleitung

└ Motivation

Warum weiterentwickeln?

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

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen

Motivation

Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen
 - komfortablere Benutzeroberfläche

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Einleitung

└─ Motivation

Motivation
Warum weiterentwickeln?

- Erweiterung des Funktionsumfangs
 - Algorithmen für Grammatiken
 - Kommentierte Ausführung & \LaTeX -Generierung
- Verbesserung der Benutzerfreundlichkeit
 - Änderung rückgängig machen
 - komfortablere Benutzeroberfläche

Warum weiterentwickeln?

1. Funktionsumfang erweitern
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

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Stand Vorher

└─ Gliederung

Gliederung

1 Einleitung

2 **Stand Vorher**

3 Änderungen

- Benutzerebene
- Build-Prozess
- Entwicklung
- Bugs und Testen

4 Live-Demo

5 Fazit

Datentypen

- (kontextfreie) Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

• (kontextfreie) Grammatiken

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Datentypen

Datentypen

- (kontextfreie) Grammatiken
- (endliche) Automaten

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

- (kontextfreie) Grammatiken
- (endliche) Automaten
- 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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Datentypen

Datentypen

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

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

- Grammatiken

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

• Grammatiken

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

- Grammatiken
 - Anzeigen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Datentypen

Datentypen

• Grammatiken
• Anzeigen

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

- Grammatiken
 - Anzeigen
 - Bearbeiten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Datentypen

Datentypen

- Grammatiken
 - Anzeigen
 - Bearbeiten

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

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

2017-05-17

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

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

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Datentypen

Datentypen

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren

Datentypen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Datentypen

2017-05-17

Datentypen

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

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. epsilon-Übergänge entfernen
10. Vervollständigen
11. minimalisieren



2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Stand Vorher

└─ GUI

GUI

GUI Grammar Modus, Menü wechseln

GUI

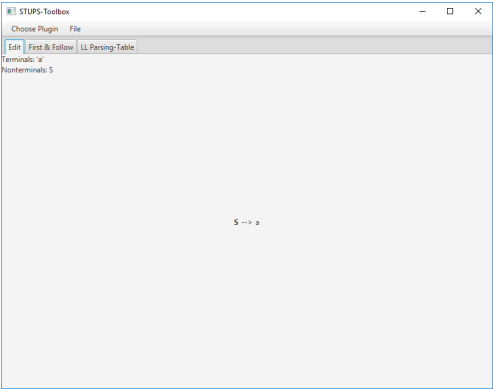


Figure: Die GUI hatte zwei Modi, Grammar und Automaton. Über die Bedienleiste lässt sich zwischen den Modi wechseln.

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ GUI

GUI



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

GUI Grammar Modus, Menü wechseln

Plugin-System

Modularer Aufbau

- Funktionen sind über sog. Plugins eingebunden

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Plugin-System

2017-05-17

Plugin-System

Modularer Aufbau
• Funktionen sind über sog. Plugins eingebunden

1. Funktionen = Plugins

Plugin-System

Modularer Aufbau

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Plugin-System

2017-05-17

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
3. automatisch einbinden

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Stand Vorher

└─ Plugin-System

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

1. Funktionen = Plugins
2. Implementiert bestimmtes Interface
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

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

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

2017-05-17

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
- je Typ nur eine Instanz

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Stand Vorher

Plugin-System

2017-05-17

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
- je Typ nur eine Instanz

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

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
- 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

2017-05-17

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
- je Typ nur eine Instanz
- leichte Weiterentwicklung ohne Einarbeitung in den Legacy-Code möglich

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

Probleme



Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

- └ Stand Vorher
 - └ Probleme

Probleme Weiterentwicklung
Kurz erläutern, dann weiter

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

- Modifizierbarkeit der Objekte

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Stand Vorher

└─ Probleme

2017-05-17

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

1. Objekte waren modifizierbar

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.
• Modifizierbarkeit der Objekte

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

2017-05-17

└─ Stand Vorher

└─ Probleme

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.
• Modifizierbarkeit der Objekte
• Plugins verändern das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.

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

2017-05-17

└─ Stand Vorher

└─ Probleme

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

Probleme

Aber: Für die Weiterentwicklung ergaben sich einige Probleme.

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

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

2017-05-17

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

Probleme

- Aber:** Für die Weiterentwicklung ergaben sich einige Probleme.
- Modifizierbarkeit der Objekte
 - Plugins verändern das Objekt. Keine einfache Speicherung von Vorgängerversionen möglich.
 - keine Vergleichbarkeit
 - Zwei Nichtterminale mit gleichem Namen waren nicht gleich

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

2017-05-17

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

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Stand Vorher

└─ Probleme

2017-05-17

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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 verändern 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

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

2017-05-17

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

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

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

└─ Probleme

2017-05-17

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
 - Erstellung einer JAR nicht möglich

Probleme Weiterentwicklung

Kurz erläutern, dann weiter

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

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen

Benutzerebene
Build-Prozess
Entwicklung
Bugs und Testen

4 Live-Demo

5 Fazit

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Änderungen

└─ Gliederung

Änderungen

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen
Benutzerebene
Build-Prozess
Entwicklung
Bugs und Testen

4 Live-Demo

5 Fazit

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

2017-05-17

- Änderungen
 - Benutzerebene
 - Gliederung

Gliederung

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



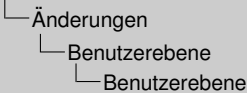
Benutzerebene

GUI

- Gestaltung durch CSS-Stylesheet

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



Benutzerebene

• Gestaltung durch CSS-Stylesheet

zunächst zur GUI

1. css-Stylesheet

Benutzerebene GUI

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

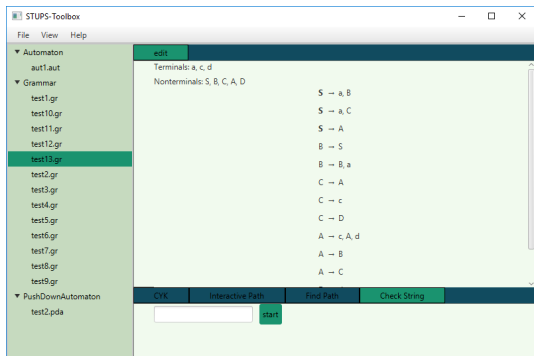


Figure: die GUI in der Toolbox 2.0

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Benutzerebene
Benutzerebene

2017-05-17

Benutzerebene
GUI

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



Figure: die GUI in der Toolbox 2.0

zunächst zur GUI

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

Benutzerebene GUI

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

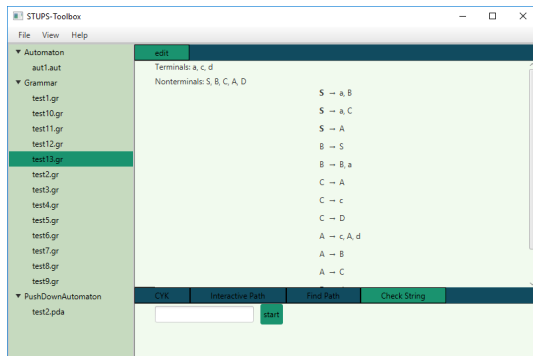


Figure: die GUI in der Toolbox 2.0

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Änderungen
Benutzerebene
Benutzerebene

2017-05-17

Benutzerebene
GUI

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

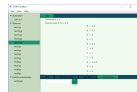


Figure: die GUI in der Toolbox 2.0

zunächst zur GUI

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

Benutzerebene GUI

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

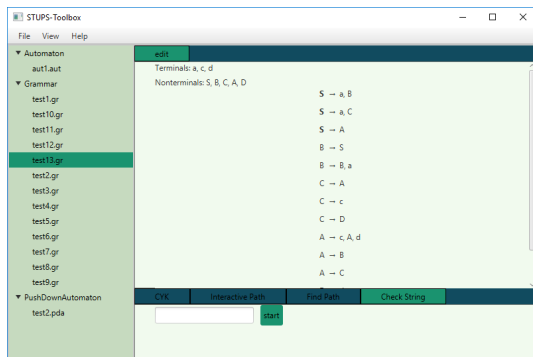


Figure: die GUI in der Toolbox 2.0

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen
Benutzerebene
Benutzerebene

Benutzerebene
GUI

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

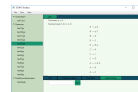


Figure: die GUI in der Toolbox 2.0

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

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen

Benutzerebene

Benutzerebene

Benutzerebene
Neu hinzugefügte Features

* dokumentierte Durchführung der Algorithmen → auch in \LaTeX

neu hingefügte Features

1. dokumentierte Durchführung der Algorithmen Latex, Lerneffekt, Skripte

Benutzerebene

Neu hinzugefügte Features

- dokumentierte Durchführung der Algorithmen \rightarrow auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen

Benutzerebene

Benutzerebene

Benutzerebene
Neu hinzugefügte Features

- dokumentierte Durchführung der Algorithmen \rightarrow auch in \LaTeX
- 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

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen

neu hingefügte Features

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

Benutzerebene

Neu hinzugefügte Features

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen
- Automatisches Speichern der Änderung beim Beenden

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen
- Automatisches Speichern der Änderung beim Beenden

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

Benutzerebene

Neu hinzugefügte Features

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- 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

2017-05-17

Änderungen
└─ Benutzerebene
 └─ Benutzerebene

- dokumentierte Durchführung der Algorithmen → auch in \LaTeX
- paralleles Arbeiten auf mehreren Objekten
- Rückgängig machen
- Automatisches Speichern der Änderung beim Beenden
- Kellerautomaten als neuer Datentyp

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. Kellerautomaten neu



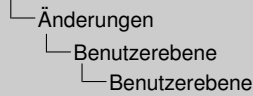
Benutzerebene

Algorithmen

- Kellerautomaten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



Benutzerebene
Algorithmen

• Kellerautomaten

welche Algorithmen wurden implementiert?

1. Zu Kellerautomaten:

Benutzerebene

Algorithmen

- Kellerautomaten
 - Umwandeln zu Grammatik

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Benutzerebene
 └─ Benutzerebene

Benutzerebene
Algorithmen

- Kellerautomaten
 - Umwandeln zu Grammatik

welche Algorithmen wurden implementiert?

1. Zu Kellerautomaten:
2. Kellerautomat zu Grammatik

Benutzerebene

Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Benutzerebene
 └─ Benutzerebene

Benutzerebene
Algorithmen

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

welche Algorithmen wurden implementiert?

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

Benutzerebene

Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

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

welche Algorithmen wurden implementiert?

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

Benutzerebene

Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

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

welche Algorithmen wurden implementiert?

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

2017-05-17

- └─ Änderungen
 - └─ Benutzerebene
 - └─ Benutzerebene

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

welche Algorithmen wurden implementiert?

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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

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

welche Algorithmen wurden implementiert?

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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

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

welche Algorithmen wurden implementiert?

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 λ -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

2017-05-17

- └─ Änderungen
 - └─ Benutzerebene
 - └─ Benutzerebene

- 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

welche Algorithmen wurden implementiert?

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 λ -Regeln
8. Entfernen von einfachen Regeln
9. Umwandeln zu Kellerautomat

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Benutzerebene

- 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

welche Algorithmen wurden implementiert?

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- Änderungen
 - Benutzerebene
 - Benutzerebene

- 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

welche Algorithmen wurden implementiert?

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
11. automatisches Suchen

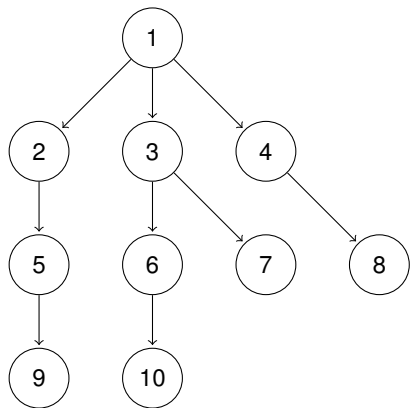


Breitensuche

2017-05-17

- Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
 - Änderungen
 - Benutzerebene
 - Breitensuche

Breitensuche



Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

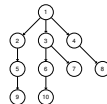
Änderungen

Benutzerebene

Breitensuche

2017-05-17

Breitensuche



Umsetzung der Algorithmen

- Umsetzung oberflächlich genau wie im Info IV-Skript

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Benutzerebene

└ Umsetzung der Algorithmen

Umsetzung der Algorithmen

• Umsetzung oberflächlich genau wie im Info IV-Skript

1. oberflächlich Info 4 Skript

Umsetzung der Algorithmen

- Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung für Computer notwendig

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Benutzerebene

└ Umsetzung der Algorithmen

Umsetzung der Algorithmen

- Umsetzung oberflächlich genau wie im Info IV-Skript
- aber: Anpassung für Computer notwendig

1. oberflächlich Info 4 Skript
2. Anpassung Computer

Umsetzung der Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Benutzerebene
 - └ Umsetzung der Algorithmen

Umsetzung der Algorithmen

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

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

Umsetzung der Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen

Benutzerebene

Umsetzung der Algorithmen

Umsetzung der Algorithmen

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

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

Umsetzung der Algorithmen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- Änderungen

- Benutzerebene

- Umsetzung der Algorithmen

Umsetzung der Algorithmen

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

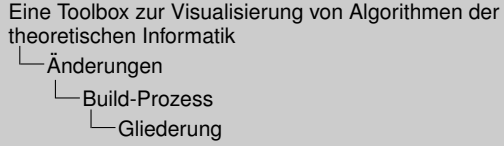
1. oberflächlich Info 4 Skript
2. Anpassung Computer
3. Beispiel: Finden von Zykeln
4. Für Menschen klar
5. Computer Tiefensuche

Gliederung

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



2017-05-17



- Gliederung
- Einleitung
 - Stand Vorher
 - Änderungen**
 - Benutzerebene
 - Build-Prozess**
 - Entwicklung
 - Bugs und Testen
 - Live-Demo
 - Fazit



Build-Prozess

- Umstieg von Makefile auf Gradle

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

2017-05-17

Änderungen
└─ Build-Prozess
 └─ Build-Prozess

Build-Prozess

• Umstieg von Makefile auf Gradle

1. BuildProzess
2. makeFile Gradle

Build-Prozess

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Build-Prozess
 └─ Build-Prozess

Build-Prozess

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

1. BuildProzess
2. makeFile Gradle
3. Workaround Sablecc

Build-Prozess

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen
└─ Build-Prozess
 └─ Build-Prozess

Build-Prozess

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

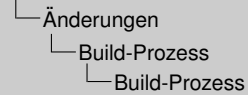
1. BuildProzess
2. makeFile Gradle
3. Workaround Sablecc
4. SableCC Source Resource, selber Ordner

Build-Prozess

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



Build-Prozess

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

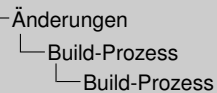
1. BuildProzess
2. makeFile Gradle
3. Workaround Sablecc
4. SableCC Source Resource, selber Ordner
5. Probleme getResource()

Build-Prozess

- 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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

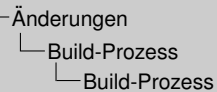
- Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource
 - Probleme mit Gradle und *getResource()*
 - Lösung: Gradle-Skript, welches Dateien verschiebt und einige Zeilen im Code anpasst

Build-Prozess

- 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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



1. 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

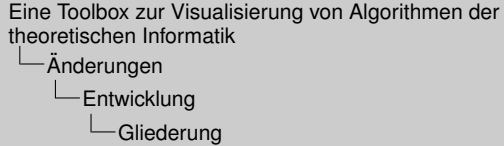
- Umstieg von Makefile auf Gradle
- kleines Workaround wegen SableCC nötig
 - SableCC mischt Source und Resource
 - Probleme mit Gradle und *getResource()*
 - Lösung: Gradle-Skript, welches Dateien verschiebt und einige Zeilen im Code anpasst
- ausführbare Jar kann nun per Plugin erstellt werden

Gliederung

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



2017-05-17



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



Entwicklung

Unmodifizierbarkeit I

- Wunsch, vollführte Änderungen rückgängig zu machen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

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

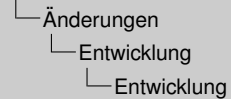
Entwicklung

Unmodifizierbarkeit I

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

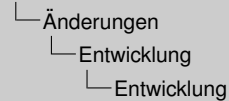
Entwicklung

Unmodifizierbarkeit I

- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt

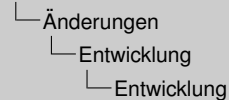
Entwicklung

Unmodifizierbarkeit I

- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt
- Lösung: Unmodifizierbarkeit

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- Wunsch, vollführte Änderungen rückgängig zu machen
- hierfür Referenz auf Vorgängerversion
- Problem: Plugins verändern das Objekt
- Lösung: Unmodifizierbarkeit

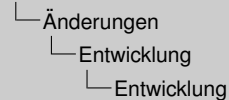
Entwicklung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

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

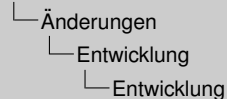
Entwicklung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- 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



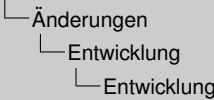
Entwicklung

Unmodifizierbarkeit II

- Vorteile

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik



Entwicklung
Unmodifizierbarkeit II

• Vorteile

1. Umständlich? Viele Vorteile

Entwicklung

Unmodifizierbarkeit II

- Vorteile
 - Objektreferenz auf Vorgänger direkt im Konstruktor

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

1. Umständlich? Viele Vorteile
2. Objektref Vorgänger Konstruktor

- Vorteile
 - Objektreferenz auf Vorgänger direkt im Konstruktor

Entwicklung

Unmodifizierbarkeit II

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

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

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

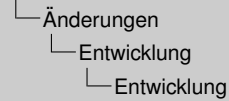
Entwicklung

Unmodifizierbarkeit II

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

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

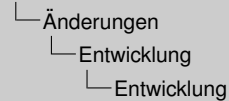
Entwicklung

Unmodifizierbarkeit II

- Vorteile
 - 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

2017-05-17



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

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

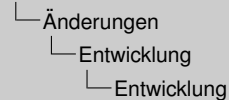
Entwicklung

Unmodifizierbarkeit II

- Vorteile
 - 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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- Vorteile
 - 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

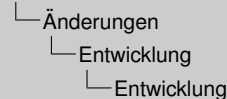
Entwicklung

Unmodifizierbarkeit II

- Vorteile
 - 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 Automaten weiterhin modifizierbar

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- Vorteile
 - 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 Automaten weiterhin modifizierbar

Entwicklung

Vergleichbarkeit

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

Entwicklung
Vergleichbarkeit

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

1. Objekte nicht vergleichbar
2. Für Vergleichbarkeit: Überschreiben *equals()*, *toHash()*

Entwicklung

Vergleichbarkeit

- Überschreiben der *equals()*- und *toHash()*-Methoden
 - ermöglicht Vergleichbarkeit von Komponenten

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

Entwicklung
Vergleichbarkeit

- Überschreiben der *equals()*- und *toHash()*-Methoden
- ermöglicht Vergleichbarkeit von Komponenten

1. Objekte nicht vergleichbar
2. Für Vergleichbarkeit: Überschreiben *equals()*, *toHash()*
3. dadurch Vergleichbarkeit

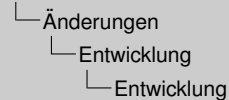
Entwicklung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

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

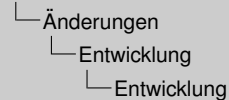
Entwicklung

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*

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

- 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*

- 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

2017-05-17

Änderungen
└─ Entwicklung
└─ Entwicklung

- 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

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

- 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 Äquivalenz zu überprüfen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen
└─ 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

- 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 Äquivalenz zu überprüfen

- 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 Äquivalenz zu überprüfen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

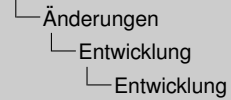
2017-05-17

Änderungen
└─ 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
7. Grammatiken äquivalent?

- 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 Äquivalenz zu überprüfen

- Interface *Storable*



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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Entwicklung
 - └ Entwicklung

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

Entwicklung

Persistierung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └─ Änderungen
 - └─ Entwicklung
 - └─ Entwicklung

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

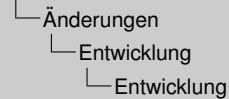
Entwicklung

Persistierung

- Interface *Storable*
- Vorteil: gemeinsamer Super-Typ
- Workspace
- wird beim Beenden auf der Festplatte gespeichert

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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
- wird beim Beenden auf der Festplatte gespeichert

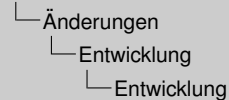
Entwicklung

Persistierung

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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
- wird beim Beenden auf der Festplatte gespeichert
- selbes Verzeichnis wie jar

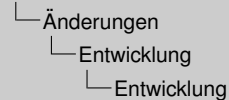
Entwicklung

Persistierung

- Interface *Storable*
- 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

2017-05-17



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
- wird beim Beenden auf der Festplatte gespeichert
- selbes Verzeichnis wie jar
- config Datei

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Änderungen
 └─ Entwicklung
 └─ Entwicklung

Entwicklung
 \LaTeX -Generierung

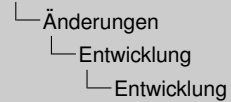
• im \LaTeX -Modus Schreiben von kompilierbaren \LaTeX -Dateien

- im \LaTeX -Modus Schreiben von kompilierbaren \LaTeX -Dateien

Latex, Erklärung wie

1. Ziel: Erstellung Latex File

- im \LaTeX-Modus Schreiben von kompilierbaren \LaTeX -Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten

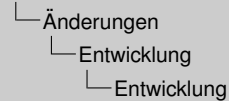


- im \LaTeX-Modus Schreiben von kompilierbaren \LaTeX -Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten

Latex, Erklärung wie

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

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



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

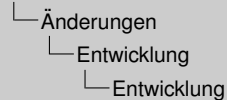
Latex, Erklärung wie

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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

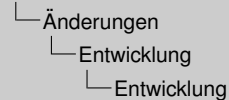


- im \LaTeX -Modus Schreiben von kompilierbaren \LaTeX -Dateien
- dokumentierte Ausführung von Algorithmen oder Drucken von Objekten
- manuelles Starten und Beenden
- Herausforderungen

Latex, Erklärung wie

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

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



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

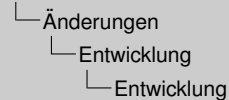
Latex, Erklärung wie

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

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17



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

Latex, Erklärung wie

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



ein hübscher Automat

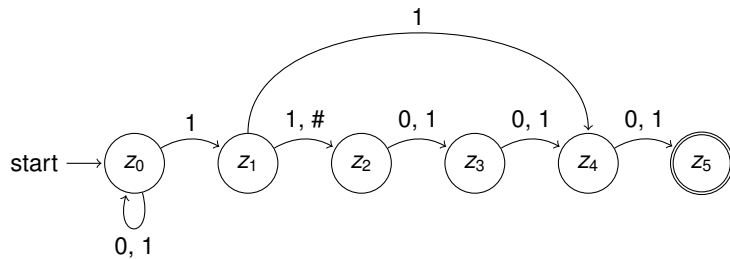
2017-05-17

- Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik
 - └ Änderungen
 - └ Entwicklung
 - └ ein hübscher Automat

ein hübscher Automat

waagrecht, alphanumerisch, Winkel

ein hübscher Automat



Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

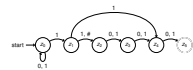
└ Änderungen

└ Entwicklung

└ ein hübscher Automat

2017-05-17

ein hübscher Automat



waagrecht, alphanumerisch, Winkel



Derselbe Automat als DFA...

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

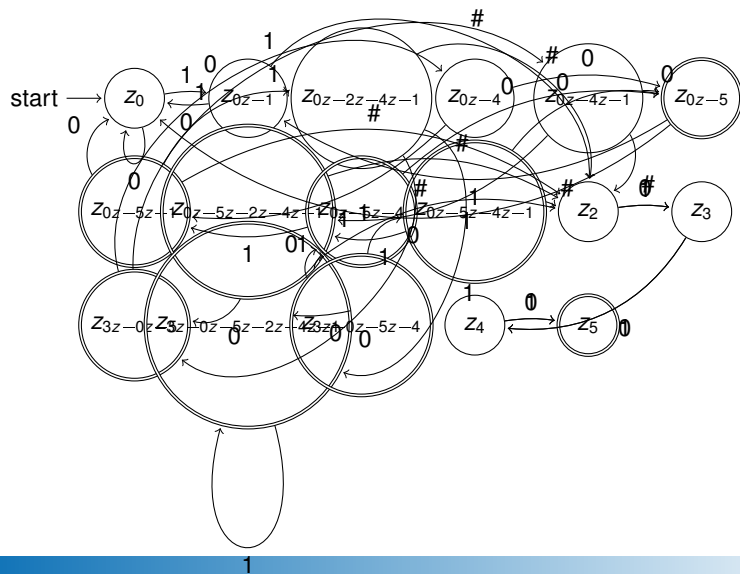
2017-05-17

- └ Änderungen
 - └ Entwicklung
 - └ Derselbe Automat als DFA...

Derselbe Automat als DFA...

dieser Automato DFA

Derselbe Automat als DFA...



Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└ Änderungen

└ Entwicklung

└ Derselbe Automat als DFA...

2017-05-17

Derselbe Automat als DFA...



dieser Automato DFA

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen

Benutzerebene

Build-Prozess

Entwicklung

Bugs und Testen

4 Live-Demo

5 Fazit

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Änderungen

└─ Bugs und Testen

└─ Gliederung

einige Herausforderungen, Bugs + Testen

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen

Benutzerebene

Build-Prozess

Entwicklung

Bugs und Testen

4 Live-Demo

5 Fazit

erwähnenswerte Bugs

- Workspace ausgewählt, der nicht valide ist? →

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ erwähnenswerte Bugs

erwähnenswerte Bugs

• Workspace ausgewählt, der nicht valide ist? →

1. Workspace nicht valide?

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ erwähnenswerte Bugs

erwähnenswerte Bugs

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

1. Workspace nicht valide?
2. Ordner gelöscht

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ erwähnenswerte Bugs

erwähnenswerte Bugs

- 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

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Bugs und Testen

└ erwähnenswerte Bugs

erwähnenswerte Bugs

- 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. ArrayOutOfBoundsException!

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Bugs und Testen

└ erwähnenswerte Bugs

erwähnenswerte Bugs

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

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

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Bugs und Testen

└ erwähnenswerte Bugs

erwähnenswerte Bugs

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

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

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└ Änderungen

└ Bugs und Testen

└ erwähnenswerte Bugs

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

erwähnenswerte Bugs

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

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen

- └ Bugs und Testen

- └ erwähnenswerte Bugs

erwähnenswerte Bugs

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

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

erwähnenswerte Bugs

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

Änderungen

Bugs und Testen

erwähnenswerte Bugs

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

erwähnenswerte Bugs

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



Testen

- Legacy-Code

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

1. Testen
2. Legacy-Code

Testen

- Legacy-Code
 - gut kommentiert, aber

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └─ Änderungen
 - └─ Bugs und Testen
 - └─ Testen

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

Testen

- Legacy-Code
 - gut kommentiert, aber

Testen

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

Testen

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

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

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

- └ Änderungen
 - └ Bugs und Testen
 - └ 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

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

- └ Änderungen
 - └ 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

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

- 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?

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

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

- 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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

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

- 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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

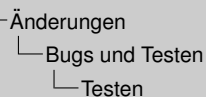
- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

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

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

- 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



- 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

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

- 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

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

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

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

- 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
- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$
 - manuelles Überprüfen auf Kompilierbarkeit
 - viele Sonderfälle, die alle beachtet werden müssen (X_a_b,#,\lambda, usw.)

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

- └ Änderungen
 - └ Bugs und Testen
 - └ Testen

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
- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$
 - manuelles Überprüfen auf Kompilierbarkeit
 - viele Sonderfälle, die alle beachtet werden müssen (X_a_b,#,\lambda, usw.)

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

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

2017-05-17

└ Live-Demo

└ Gliederung

Gliederung

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



Live-Demo

2017-05-17

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Live-Demo

└─ Live-Demo

Live-Demo

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

2017-05-17

└─Fazit

└─Gliederung

Gliederung

1 Einleitung

2 Stand Vorher

3 Änderungen

- Benutzerebene
- Build-Prozess
- Entwicklung
- Bugs und Testen

4 Live-Demo

5 Fazit

Herausforderungen

- Unmodifizierbarkeit

Eine Toolbox zur Visualisierung von Algorithmen der
theoretischen Informatik

└─ Fazit

└─ Herausforderungen

2017-05-17

Herausforderungen

- Unmodifizierbarkeit

Herausforderungen

1. Unmodifizierbarkeit
2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Fazit

└─ Herausforderungen

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen

Herausforderungen

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

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Fazit

└─ Herausforderungen

2017-05-17

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes

Herausforderungen

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

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

└─ Fazit

└─ Herausforderungen

2017-05-17

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex

Herausforderungen

1. Unmodifizierbarkeit
2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
3. Algorithmen Computer
4. Skript Info 4 entnommen, 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.

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex
- Testen

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

Fazit

Herausforderungen

2017-05-17

Herausforderungen

- Unmodifizierbarkeit
- Algorithmen für Computer umsetzen
- Weiterentwicklung eines bestehenden Programmes
- Latex
- Testen

Herausforderungen

1. Unmodifizierbarkeit
2. nicht so schwer, aber zeitaufwendig. fehlende Testabdeckung Problem
3. Algorithmen Computer
4. Skript Info 4 entnommen, 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

- Codequalität verbessert

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

- Codequalität verbessert
- alle gewünschten Features konnten umgesetzt werden

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

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

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

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

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

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

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

- durchaus noch mehr Potential

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

- durchaus noch mehr Potential
- Turing-Maschinen

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

Ausblick

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─Fazit

└─Ausblick

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

Ausblick

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

Ausblick

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

Eine Toolbox zur Visualisierung von Algorithmen der theoretischen Informatik

2017-05-17

└─ Fazit

└─ Ausblick

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

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

Vielen Dank für die Aufmerksamkeit!