Pflichtenheft

λurora

The Lambda Calculus IDE

Alexander von Heyden, Iuliia Patrusheva
Max Nowak, Nikolai Polley
Randy Seng, Younis Bensalah

2017-11-29

Inhaltsverzeichnis

1. Einleitung

Der Lehrstuhl für Programmierparadigmen am IPD Snelting ist seit Jahren dafür zuständig, Besuchern der gleichnamigen Vorlesung des λ-Kalkül zu vermitteln. Leider hat es sich in den vergangenen Jahren immer wieder gezeigt, dass eine erhebliche Anzahl von Studenten mit den grundlegenden Eigenschaften des λ-Kalküls Probleme haben. Hier versucht λurora anzusetzen. Mit Hilfe einer einfach strukturierten Web App, die vom Design her an gängige IDEs angelehnt ist, soll dem Benutzer die Einführung in das λ-Kalkül erleichert werden. Hierfür wird dem Besucher eine Eingabemaske zur Verfügung gestellt, in die er eigene Terme eingeben kann. Schritt für Schritt kann er nun die Auswertung bis zum finalen Ergebnis verfolgen.

2. Zielsetzung

2.1. Musskriterien

Eingabe **M1**
Implementiert durch: F1 F3

Es gibt ein Eingabefeld um λ-Terme einzugeben.

Input editieren **M2**
Implementiert durch: F1 F6

Der Input kann nur bearbeitet werden, solange das Programm nicht rechnet.

Output nicht editierbar **M3**
Implementiert durch: F33

Der Output kann nicht bearbeitet werden.

Auswertung starten **M4**
Implementiert durch: F14 F32 F40 F41

Der Benutzer kann die Auswertung des eingegebenen λ-Terms starten. Im Ausgabefenster erscheint das Ergebnis in Normalform.

Auswertung terminieren **M5**
Implementiert durch: F16

Der Benutzer kann die Auswertung des λ-Terms jederzeit abbrechen.

Zwischenschritte **M6**
Implementiert durch: F5 F34

Nach der Auswertung des λ-Terms können Zwischenergebnisse ausgegeben werden.

Auswertungsstrategie **M7**
Implementiert durch: F27

Die Auswertungsstrategie bei β-Reduktion von λ-Termen ist die Normalreihenfolge. Weitere Auswertungsstrategien sind in den Wunschkriterien.

2.2. Wunschkriterien

K1 **Zeilennummer**
Implementiert durch: F2

Neben dem Eingabefeld soll ausgeblasst eine Zeilennummerierung dargestellt werden um eine einfache Zeilenzuteilung zu ermöglichen.

K2 **Kommentare einfügen**
Implementiert durch: F8

Einzelne Zeilen in der Eingabe können auskommentiert werden. Diese Zeilen werden dann vom Parser übersprungen.

K3 **Autovervollständigung von Klammern**
Implementiert durch: F4

Wenn der Benutzer eine Klammer öffnet „(“, dann schließt das Programm automatisch diese Klammer mit „)“.

K4 **Syntax-Highlighting**
Implementiert durch: F9 F10 F11

Funktionsnamen aus der Standard- oder Benutzerbibliothek werden farbig geschrieben um eine einfache Erkennung zu ermöglichen. Zur besseren Übersichtlichkeit werden λs und „.“ Punkte fett geschrieben. Beim Hovern über einer Klammer wird das entsprechende Klammerpaar markiert.

K5 **Erweitertes Syntax-Highlighting**
Implementiert durch: F12 F13

Der nächste auszuwertende Redex wird farblich hinterlegt. Außerdem wird der Term, der bei einer β-Reduktion von einer Abstraktion reduziert wurde, mit einer dritten Farbe im nächsten Schritt markiert.

K6 **Pretty Print**
Implementiert durch: F7

Die Ausgabe der Auswertungen von λ-Termen wird schön formatiert. Insbesondere wird das korrekte Einrücken der Zeilen gewährleistet.

K7 **Standardbibliothek**
Implementiert durch: F20 F24 F25 F26

Bibliothek, in der vordefinierte Funktionen und Church-Zahlen verfügbar sind. Durch Schlüsselwörter kann der Benutzer diese direkt in seinen Code einbinden.

Benutzerbibliothek **K8**
Implementiert durch: F21 F22 F23 F24 F25 F26

Bibliothek, in die der Benutzer eigene Funktionen definieren, speichern und löschen kann. Durch Schlüsselwörter kann der Benutzer diese direkt in seinen Code einbinden.

Weitere Auswertungsstrategien **K9**
Implementiert durch: F28 F29 F30

Zusätzliche Auswertungsstrategien sind „Call by Name“ und „Call by Value“.

Benutzerdefinierte Auswahl **K10**
Implementiert durch: F30 F31

Eigene Auswahl bei der β-Reduzierung eines Redex.

Mehrere Schritte auswerten **K11**
Implementiert durch: F15

Auf Wunsch kann eine benutzerdefinierte Anzahl von β-Reduktionen durchgeführt und angezeigt werden.

Anzahl angezeigter Zwischenschritte **K12**
Implementiert durch: F36

Der Benutzer kann die Anzahl der angezeigten Zwischenschritte frei wählen.

Pause-Button **K13**
Implementiert durch: F17

Der „Pause“-Button unterbricht die laufende Berechnung. Nachdem das Programm pausiert hat, kann der Benutzer sich die bisher berechneten Zwischenschritte ansehen.

Continue-Button **K14**
Implementiert durch: F18

Der „Continue“-Button setzt eine pausierte Berechnung fort.

Share-Button **K15**
Implementiert durch: F19 F56

Ein „Share"-Button ermöglicht das Erstellen von Short-URLs sowie das Kopieren dieser in den Zwischenspeicher.

K16 **Öffnen in neuem Tab**
Implementiert durch: F54

Im Kontextmenü zu den einzelnen Schritten kann der aktuelle Schritt als Eingabe in einem neuen Browser-Tab geöffnet werden.

K17 **LaTeX export**
Implementiert durch: F55

λ-Terme aus dem Text-Editor oder der Schrittliste können als LaTeX-Snippet exportiert werden. Bis zu einer gewissen Größe ist es auch möglich, den kompletten Rechenweg mit zu exportieren.

K18 **Night Mode**
Implementiert durch: F38

Die Funktion „Night Mode" ändert das äußere Erscheinungsbild der Applikation zu einem dunkleren Farbschema.

K19 **Schleifenerkennung**
Implementiert durch: F42 F43

Das Programm soll einfache Schleifen, wie z.B. (λx. x x) (λx. x x), erkennen. Insbesondere kann es aber weiterhin vorkommen, dass komplexere Schleifen nicht erkannt werden. Außerdem soll das Programm nicht die Berechnung abbrechen wenn eine Schleife gefunden wurde, sondern nur dem Nutzer die Option gegeben werden die Berechnung abzubrechen. Wenn der Benutzer will kann er auch gefundene Endlosschleifen weiterlaufen lassen.

K20 **λurora Tutorial**
Implementiert durch: F57

Das Tutorial erklärt und zeigt dem Benutzer, wie man die Applikation λurora nutzt.

K21 **Sprachenauswahl im Programm**
Implementiert durch: F37

Der Benutzer soll neben der Default-Sprache Englisch auch die Möglichkeit haben, Deutsch, Französisch, oder Russisch auszuwählen.

K22 **De Bruijn-Indizes**
Implementiert durch: F35

Ausgabe des Programmes kann wahlweise mit De Bruijn-Indizes erfolgen.

Lokale Sessions **K23**
Implementiert durch: F44 F45 F46 F47 F48

Die Applikation merkt sich im localStorage des Webbrowsers die Historie von vorherigen Eingaben. Der Benutzer kann außerdem diese Historie verändern.

Tastatur-Shortcuts **K24**
Implementiert durch: F49 F50 F51 F52 F53

λurora stellt Tastatur-Shortcuts zum vereinfachten Ausführen von Features bereit.

Verfügbare Tastatur-Shortcuts

Shortcut	Beschreibung
\	λ
`Shift` + $\hookleftarrow$	Run
`Shift` + `T`	Reset
`Shift` + `P`	Pause
`Shift` + `N`	Step

Mobile-Darstellung **K25**
Implementiert durch: F39

Das Programm soll stark reduziert auch auf mobilen Endgeräten laufen

2.3. Abgrenzungskriterien

Berechnungen laufen Client-seitig **A1**

Der Client führt das Programm aus. Alle Funktionen, die etwas berechnen oder speichern müssen, werden auf der Hardware des Clients ausgeführt.

Keine Server-Komponente **A2**

Der Server ist nur für die Verbreitung des Programmcodes zuständig. Er selber speichert keinerlei Daten oder führt Aktionen für das Programm aus.

Kein typisiertes λ-Kalkül **A3**

Das Programm kann nur β-Reduktion des untypisierten λ-Kalküls durchführen. Auch eine einfache Erweiterungsschnittstelle, um z.B. Auswertungen des typisierten λ-Kalküls zu erlauben, ist nicht geplant und wird nicht bei Entwicklung des Programms in Betracht gezogen.

A4 Applikation bietet keine Projekt-Unterstützung

Das Programm unterstützt keine Klassen oder anderweitig mehr als eine Eingabe oder mehrere Ausgabe. Jede Instanz des Programmes ist nur auf die Nutzung durch eine Person ausgelegt. Es bietet keine Möglichkeiten zur Gruppenarbeit, lediglich durch die Share-Funktion und den LaTeX Export kann Code geteilt werden.
Eine Versionskontrolle oder generelles Zusammenarbeiten über zwei Clients mit dem gleichen Programm ist nicht möglich.

3. Produkteinsatz

Wenn ein Interessierter das Programm benutzen will, dann benötigt er eine Internetanbindung. Er benötigt außerdem einen PC mit folgenden Spezifikationen: Wenn er Windows benutzt, dann Windows 7 oder höher, wenn er Mac benutzt dann OS X Mavericks 10.9 oder höher oder wenn er Linux benutzt Ubuntu 14.04 oder höher, Debian 8 oder höher, oder Fedora24 oder höher. Er braucht zusätzlich einen SSE2-fähigen Intel Pentium 4-Prozessor oder besser.

3.1. Anwendungsbereiche

Der Anwendungsbereich umfasst Studenten und wissenschaftlichen Mitarbeiter, die sich mit dem λ-Kalkül beschäftigen. λurora erlaubt dem Benutzer λ-Terme zu berechnen und sich dabei einzelne Schritte der β-Reduktion anzeigen zu lassen. Durch vorhandene Tastatur-Shortcuts und eine mit grundlegenden Funktionen ausgestattete Standardbibliothek, soll die Benutzung der λurora Web App den Benutzer an eine normale IDE erinnern, sodass schneller das gewünschte Ergebnis erreicht werden kann.

3.2. Zielgruppen

Zur Zielgruppe gehören vor allem Informatik-Studenten und wissenschaftliche Mitarbeiter, die sich mit dem λ-Kalkül beschäftigen und bereits gewisse Grundkenntnisse des λ-Kalküls besitzen. Auch anderweitig Interessierte sollen das Programm benutzen können.

λurora ist für Einzelpersonen gedacht, eine Gruppenarbeit mit dem Programm (z.B. simultanes Bearbeiten) ist nicht möglich.

Für die Benutzung von λurora werden Grundkenntnisse der Internetnutzung vorausgesetzt.

Die Sprache der Benutzeroberfläche ist Englisch, daher sind Grundkenntnisse der Sprache von Vorteil.

3.3. Betriebsbedingungen

λurora muss auf einem Web Server (z.B. Apache) gehostet werden und kann an beliebig viele Benutzer ausgeliefert werden.

4. Produktumgebung

λurora ist eine Client seitige Anwendung.

4.1. Software

- Es wird eines der gängigen Betriebssysteme benötigt:
  - Windows 7 oder höher
  - OS X Mavericks 10.9 oder höher
  - Ubuntu 14.04 oder höher, Debian 8 oder höher, Fedora24 oder höher
- Es wird ein gängiger Webbrowser benötigt, der HTML5, CSS3 und JavaScript unterstützt:
  - Firefox (ab Version 57)
  - Google Chrome (ab Version 62.0)
  - Safari (ab Version 11.0.0)

4.2. Hardware

- Es wird ein an das Internet angeschlossener Rechner benötigt.
- Des weiteren wird ein SSE2-fähiger Intel Pentium 4-Prozessor oder besser benötigt.

4.3. Orgware

- Während dem Projekt werden folgende Dokumente erstellt und dauernd gepflegt:
  - Pflichtenheft
  - λurora Handbuch

5. Funktionale Anforderungen

5.1. Eingabe

F1 **Eingabefeld**
Getestet durch: T3 T4 T6 Implementiert: M1 M2

Es gibt ein Eingabefeld, dessen Inhalt als Eingabe für λurora dient.

F2 **Zeilennummern**
Getestet durch: T3 T4 T6 Implementiert: K1

Links neben dem Eingabefeld befinden sich Zahlen, die die Nummer der Zeile angeben. Diese sind nicht editierbar.

F3 **Eingabefeld bearbeiten**
Getestet durch: T4 T6 Implementiert: M1

Der Benutzer kann den Inhalt des Eingabefelds über die Tastatur bearbeiten.

F4 **Autovervollständigung von Klammern**
Getestet durch: T4 T6 Implementiert: K3

Wenn der Benutzer eine Klammer öffnet „(“, fügt λurora automatisch eine passende schließende Klammer ein „)“.

F5 **Anzahl von dargestellten Schritten auswählen**
Getestet durch: T6 Implementiert: M6

Der Benutzer kann einstellen wie viele Schritte er sehen will, wenn er auf den „3-Punkte“-Button drückt um die Schritte zu sehen.

F6 **Keine Änderungen während der Auswertung**
Getestet durch: T4 Implementiert: M2

Der Benutzer kann das Eingabefeld nur genau dann editieren, wenn keine Berechnung am laufen ist.

F7 **Pretty Print**
Getestet durch: T9 Implementiert: K6

Wenn eine Eingabe länger als die Zeilenlänge ist, dann wird der Zeilenumbruch wenn mög-

lich nach einem Redex gemacht damit der Redex nicht in zwei Teile gerissen wird. In der folgenden Zeile ist dann die Eingabe eingerückt um zu verdeutlichen, dass es sich immer noch um die gleiche Eingabe wie in der Zeile davor handelt.

Kommentare **F8**
Getestet durch: T4 Implementiert: K2

Der Benutzer kann mit „#“ eine Zeile bis zum nächsten Zeilenumbruch auskommentieren. Der Kommentar wird dann farbig kenntlich gemacht.

5.2. Highlighting

Highlighting von Lambdas und Punkten **F9**
Getestet durch: T5 Implementiert: K4

Lambda-Zeichen „λ“ und Punkte „.“ werden fett geschrieben.

Highlighting von Klammern **F10**
Getestet durch: T5 Implementiert: K4

Wenn der Eingabezeiger adjazent zu einer öffnenden oder schließenden Klammer steht, werden beide Klammern des Paares farbig hervorgehoben.

Highlighting von Funktionsnamen **F11**
Getestet durch: T5 Implementiert: K4

Während der Benutzer Eingaben tätigt, versucht das Programm Funktionsnamen aus der Standardbibliothek oder Benutzerbibliothek zu erkennen und farblich hervorzuheben. Wenn ein Funktionsname gefunden wurde, dann wird er farbig geschrieben.

Highlighting des nächsten auszuwertenden Redex **F12**
Getestet durch: T5 Implementiert: K5

In jedem Schritt wird im Schrittfeld der nächste auszuwertende Redex farblich hinterlegt. Die Abstraktion des auszuwertenden Redex wird mit einer Farbe markiert und der Term, der von der Abstraktion reingezogen wird, bekommt eine andere Farbe.

Highlighting des zuletzt ausgeführten Redex **F13**
Getestet durch: T5 Implementiert: K5

In jedem Schritt wird der Term, der in der Berechnung für diesen Schritt von einer Abstraktion eingefügt wurde, farbig markiert.

5.3. Buttons

F14 **Run**
Getestet durch: T4 T6 Implementiert: M4

Der „Run"-Button startet die Auswertung des Inputs. Diese veranlasst den Parser dazu, die λ-Ausdrücke im Eingabefeld einzulesen.

F15 **Step**
Getestet durch: T10 Implementiert: K11

Der „Step"-Button führt eine benutzerdefinierte Anzahl von β-Reduktionen durch und gibt diese als Zwischenergebnis aus. Es wird kein vorzeitiges Ergebnis ausgegeben.

F16 **Reset**
Getestet durch: T4 Implementiert: M5

Der Benutzer kann eine laufende Berechnung über einen „Reset"-Button abbrechen. Alle Berechnungsschritte werden verworfen und nichts aus der bisherigen Berechnung wird gespeichert. Der Input bleibt erhalten.

F17 **Pause**
Getestet durch: T4 Implementiert: K13

Der Benutzer kann auf den „Pause"-Button drücken um die laufende Berechnung zu unterbrechen und sich die bisherigen Schritte anzeigen zu lassen. Er kann allerdings, im Gegensatz zum Reset-Button, die Berechnung weiterlaufen lassen.

F18 **Continue**
Getestet durch: T4 Implementiert: K14

Der Benutzer kann mit dem „Continue"-Button eine mit Pause unterbrochenen Berechnung fortsetzen lassen.

F19 **Share**
Getestet durch: T7 T8 Implementiert: K15

Der Benutzer kann mithilfe des „Share"-Buttons eine Short-URL erstellen, welche den Input und die Benutzerbibliothek enthält. Wenn ein anderer Benutzer diesen Link in seinem Browser lädt, dann öffnet sich das Programm mit gleichem Input und Benutzerbibliothek.

5.4. Bibliotheken

Standardbibliothek **F20**
Getestet durch: T6 Implementiert: K7

Der Benutzer kann Funktionen aus der Standardbibliothek im Eingabefeld verwenden.

Benutzerbibliothek **F21**
Getestet durch: T6 Implementiert: K8

Funktionen aus der Benutzerbibliothek können analog zu Funktionen der Standardbibliothek verwendet werden. Es spielt für die Eingabe keine Rolle, ob eine Funktion in der Standard- oder in der Benutzerbibliothek ist.

Erweitern der Benutzerbibliothek **F22**
Getestet durch: T6 Implementiert: K8

Der Benutzer kann Funktionen zur Benutzerbibliothek hinzufügen.

Entfernen aus der Benutzerbibliothek **F23**
Getestet durch: T6 Implementiert: K8

Der Benutzer kann Funktionen aus der Benutzerbibliothek entfernen.

Definition einer Funktion **F24**
Getestet durch: T6 Implementiert: K7 K8

Eine definierte Funktion der Standard- oder Benutzerbibliothek hat einen Namen und entspricht genau einem λ-Term.

Funktionsnamen-Konvention **F25**
Getestet durch: T6 Implementiert: K7 K8

Der Name einer Funktion beginnt ausnahmslos mit „$“.

Äquivalenz von Funktionen und deren Definition **F26**
Getestet durch: T6 Implementiert: K7 K8

Die Eingabe von definierten Funktionen aus der Standard- und Benutzerbibliothek ist semantisch äquivalent zur Eingabe deren entsprechenden Definition als λ-Term.

5.5. Auswertungsstrategien

F27 **Normalreihenfolge**
Getestet durch: T4 T6 T7 Implementiert: M7

λ-Terme werden standardmäßig in Normalreihenfolge ausgewertet. Das heißt, dass immer der linkeste Redex zuerst ausgewertet wird.

F28 **Call-by-Name**
Getestet durch: T7 Implementiert: K9

λ-Terme können gemäß der Auswertungsstrategie „Call-by-Name“ ausgewertet werden.

F29 **Call-by-Value**
Getestet durch: T7 Implementiert: K9

λ-Terme können gemäß der Auswertungsstrategie „Call-by-Value“ ausgewertet werden.

F30 **Auswahl der Auswertungsstrategie**
Getestet durch: T7 Implementiert: K9 K10

Der Benutzer kann zwischen den drei Auswertungsstrategien „Normalreihenfolge“, „Call-by-Name“ und „Call-by-Value“ wählen, oder selbst den nächsten auszuwertenden Redex auswählen.

F31 **Benutzerdefinierte Redex Wahl**
Getestet durch: T7 Implementiert: K10

Im Step-Modus kann der Benutzer selbst einen Redex auswählen, der als nächstes ausgewertet wird.

5.6. Ausgabe

F32 **Ausgabefeld**
Getestet durch: T4 T6 Implementiert: M4

Falls die Berechnung terminiert, steht das Ergebnis im Ausgabefeld.

F33 **Ausgabefeld nicht editierbar**
Getestet durch: T4 Implementiert: M3

Der Inhalt des Ausgabefelds kann nicht bearbeitet werden.

β-Reduktionen angezeigen **F34**
Getestet durch: T6 Implementiert: M6

In der GUI wird dem Benutzer nach einer erfolgreichen Berechnung ein „3-Punkte"-Button angezeigt. Wenn er auf diesen klickt, werden ihm die ausgewählte Anzahl von β-Reduktionsschritten angezeigt, die durchgeführt wurden, um auf das Ergebnis zu kommen.

De Bruijn-Indizes **F35**
Getestet durch: T7 Implementiert: K22

Der Benutzer kann einstellen, dass die λ-Ausdrücke mit De Bruijn-Indizes ausgegeben werden.

5.7. Darstellung

Anzahl der angezeigten Schritte **F36**
Getestet durch: T10 Implementiert: K12

Der Benutzer kann auswählen, wie viele Zwischenschritte angezeigt werden sollen.

Sprache ist auswählbar **F37**
Getestet durch: T1 Implementiert: K21

Der Benutzer kann aus einer Auswahl von menschlichen Sprachen, die Sprache der Benutzeroberfläche einstellen.

Night Mode **F38**
Getestet durch: T3 Implementiert: K18

Das äußere Erscheinungsbild der Applikation kann über die „Night mode"-Funktion zu einem dunklen Farbschema geändert werden. Dies kann über das Hamburger-Menü ein- oder ausgestellt werden.

Mobile Darstellung **F39**
Getestet durch: T3 Implementiert: K25

Falls erkannt wird, dass das Programm von einem mobilen Endgerät aus aufgerufen wird, wird eine vereinfachte Version der GUI angezeigt. Hier ist ist kein bearbeiten des Inputs möglich, lediglich der vorhandene Input, wie z.B. der durch eine geteilte URL erhaltene, wird ausgewertet.

5.8. Hintergrundablauf

F40 **β-Reduktion**
Getestet durch: T4 T6 Implementiert: M4

Das Programm kann anhand eines vorhandenen λ-Ausdrucks und einer gegebenen Auswertungsstrategie eine korrekte β-Reduktion durchführen.

F41 **Auswertung beendet**
Getestet durch: T4 T6 Implementiert: M4

Das Programm erkennt, falls keine weiteren β-Reduktionen mehr möglich sind.

F42 **Schleifenerkennung**
Getestet durch: T4 Implementiert: K19

Eine laufende Berechnung wird automatisch auf Schleifen überprüft. Wenn eine Schleife gefunden wurde, warnt das Programm davor.

F43 **Kein Timeout**
Getestet durch: T4 Implementiert: K19

Ein vom Benutzer eingegebenes Programm kann potentiell unendlich laufen.

5.9. History

F44 **Historie**
Getestet durch: T8 Implementiert: K23

Eine ausgeführte Benutzereingabe wird automatisch in die Historie eingefügt.

F45 **Permanenz der Historie**
Getestet durch: T8 Implementiert: K23

Einträge in der Historie werden nach schließen des Browsers nicht gelöscht.

F46 **Wiederherstellen der Historie**
Getestet durch: T8 Implementiert: K23

Der Benutzer kann Eingaben aus der Historie wiederherstellen.

Löschen der Historie **F47**
Getestet durch: T8 Implementiert: K23

Der Benutzer kann die Einträge der Historie löschen.

Benutzerbibliothek in der Historie **F48**
Getestet durch: T8 Implementiert: K23

Eine vom Benutzer definierte Funktion wird automatisch in die Historie eingefügt.

5.10. Shortcuts

Lambda-Shortcut **F49**
Getestet durch: T11 Implementiert: K24

Durch Eingabe von \ wird ein λ ausgegeben.

Run-Shortcut **F50**
Getestet durch: T11 Implementiert: K24

Drücken von `Shift` + ↩ ist äquivalent zum Drücken des „Run“-Buttons.

Reset-Shortcut **F51**
Getestet durch: T11 Implementiert: K24

Drücken von `Shift` + `T` ist äquivalent zum Drücken des „Reset“-Buttons.

Pause-Shortcut **F52**
Getestet durch: T11 Implementiert: K24

Drücken von `Shift` + `P` ist äquivalent zum Drücken des „Pause“-Buttons.

Step-Shortcut **F53**
Getestet durch: T11 Implementiert: K24

Drücken von `Shift` + `N` ist äquivalent zum Drücken des „Schritt“-Buttons.

5.11. Sonstige Funktionalität

In neuem Tab öffnen **F54**

Getestet durch: T8 Implementiert: K16

Ein vom Benutzer ausgewählter Schritt kann in einem neuen Browser-Tab zum Editieren geöffnet werden.

F55 **LaTeX-Export**
Getestet durch: T2 Implementiert: K17

Der Benutzer kann entweder die Eingabe oder die Eingabe mit Zwischenschritten und Ergebnis in Latexsyntax exportieren. Dies wird automatisch ins Clipboard kopiert, damit der Benutzer dies selbsständig in ein LaTeX Dokument einfügen kann.

F56 **E-Mail Link**
Getestet durch: T8 Implementiert: K15

Der Benutzer kann eine Short-URL erstellen, welche den Input und die Benutzerbibliothek enthält und diese anschließend als E-Mail versenden.

F57 **Tutorial**
Getestet durch: T9 Implementiert: K20

Durch drücken des „Hilfe“-Buttons erscheint ein Tutorial, das dem Benutzer das Programm erklärt und auf Features hinweist.

6. Nicht-Funktionale Anforderungen

Korrekte Darstellung in Google Chrome **N1**

In „Google Chrome“ 62 wird das Programm korrekt und ohne Fehler jeglicher Art dargestellt.

Benutzbarkeit **N2**

Das Programm soll auch ohne Tutorial verständlich bedienbar sein.

Aussehen **N3**

Die Grafische Oberfläche soll schlicht aber modern wirken.

Geschwindigkeit **N4**

Das Programm soll sich schnell und responsiv anfühlen. Die Aktion die ein Button ausführt soll schnell erfolgen.

Programmsprache **N5**

Die Standard Sprache des Programmes ist Englisch.

Erweiterte Sprachenauswahl **N6**

Das Programm ist außerdem in den Sprachen Deutsch, Russisch und Französisch verfügbar.

Hinzufügen neuer Sprachen **N7**

Die Programmarchitektur soll einfaches Hinzufügen neuer, menschlicher Sprachen unterstützen.

Browserkompatibilität **N8**

Das Programm funktioniert auf dem Webbrowser „Google Chrome“ 62.

Erweiterte Browserkompatibilität **N9**

Das Programm funktioniert auf aktuellen Webbrowsern, die Javascript unterstützen.

Hash-Aufwand **N10**

Die Berechnung der Hashwerte ist nicht mehr als 10% der Gesamtrechenzeit in Anspruch.

N11 Standardbibliothek

Die Standardbibliothek kann bis zu 50 vordefinierte Funktionen beinhalten.

N12 Benutzerbibliothek

Die Benutzerbibliothek kann bis zu 25 selbst definierte Funktionen beinhalten.

N13 Begrenzte Zwischenergebnisse

Die Anzahl der sichtbaren Zwischenergebnisse ist auf 100 beschränkt.

N14 Begrenzter LaTeX export

Der LaTeX-Export für die komplette Rechnung kann nur ausgeführt werden, falls weniger als 50 Zwischenergebnisse vorliegen.

N15 Punkt Eingabe

Zur Eingabe des Punktes innerhalb einer Abstraktion darf lediglich das Punktzeichen „." verwendet werden.

N16 Eingabe von Zahlen

Zahlen dürfen nur als reine Ziffernfolgen eingegeben werden. Es darf kein „$" davor gesetzt werden.

N17 Variablennamen

Der Name einer Variable darf lediglich aus Kleinbuchstaben bestehen. Sonderzeichen, Zahlen, und Großbuchstaben sind nicht zugelassen.

N18 Funktionsnamen

Der Name von Funktionen, unabhängig davon, wo sie definiert sind, beginnt ausnahmslos mit „$". Danach ist die Funktion mit Kleinbuchstaben zu schreiben. Fehlt das „$" wird der Name als Variable interpretiert.

7. Tests

Sprachenauswahl **T1**
Testet: F37

T1.1 **Stand:** Das Programm befindet sich im Default-Zustand.
Aktion: Der Benutzer drückt auf den „ENG“-Knopf.
Reaktion: Es geht ein Menü auf mit verfügbaren menschlichen Sprachen.

T1.2 **Stand:** Zustand aus dem vorherigen Schritt.
Aktion: Der Benutzer drückt auf „Russian“.
Reaktion: Die Das Menü verschwindet und die Sprache des gesamten Programms ändert sich auf Russisch.

Teilen-Funktion **T2**
Testet: F55

T2.1 **Stand:** Es steht eine gültige Eingabe im Eingabefeld. Es werden mindestens 3 Zwischenschritte angezeigt.
Aktion: Der Benutzer hovert über einen Schritt.
Reaktion: Es wird der Schrittemenüknopf angezeigt.

T2.2 **Stand:** Zustand aus vorherigem Schritt.
Aktion: Benuzer klickt auf Schrittemenüknopf.
Reaktion: Es öffnet sich das zu dem Schritt zugehörige Schrittemenü.

T2.3 **Stand:** Zustand aus dem vorherigen Schritt.
Aktion: Der Benutzer klickt auf „LaTeX“ im Schrittemenüknopf.
Reaktion: Der LaTeX-Code für diesen einen Schritt wird in die Zwischenablage des Benutzers kopiert. Es wird ein Pop-up angezeigt als Bestätigung. Nach hoechstens 5 Sekunden ist es verschwunden.

Darstellung in Chrome **T3**
Testet: N1 F1 F2 F39 F38

T3.1 **Stand:** Das Programm liegt auf einem Server.
Aktion: Der Benutzer tippt die lokale IP in den Browser „Google Chrome“ ein.
Reaktion: Die Webseite wird geöffnet und wird korrekt dargestellt je nach Browserauflösung.

T3.2 **Stand:** Das Programm liegt auf einem Server.
Aktion: Der Benutzer tippt die Adresse in „Google Chrome“ für mobile Geräte ein.
Reaktion: Die mobile Version der Webseite wird geöffnet und korrekt dargestellt.

T3.3 **Stand:** Das Programm ist im Default State.
Aktion: Der Benutzer drückt auf den „Nightmode“-Button
Reaktion: Das Programm wechselt im Design in den Nachtmodus. Das bisher helle Eingabefenster wird dunkel und die Schrift hell.

T4 Eingabe und korrektes Ergebnis

Testet: F1 F2 F3 F4 F6 F42 F43 F14 F27 F40 F41 F32 F16 F17 F18 F33 F8

T4.1 **Stand:** Das Programm ist geöffnet und es ist im Default-State
Aktion: Der Benutzer gibt den Term „λx.x“ in das Eingabefeld ein und drückt auf Run
Reaktion: Das Programm kann keine β-Reduktion durchführen und gibt als Ausgabe λx.x

T4.2 **Stand:** Das Programm ist geöffnet und es ist im Default-State
Aktion: Der Benutzer gibt in den Editor „(λx.x) z “ ein. Dannach drückt er auf den Button „Run“
Reaktion: Das Programm führt eine β-Reduktion aus und gibt als Ausgabe „z“ aus. Es wird auch angezeigt, dass es nur eine β-Reduktion durchgeführt hat.

T4.3 **Stand:** Das Programm ist geöffnet und es ist im Default-State
Aktion: Der Benutzer gibt
(λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z)) ind den Editor ein und drückt auf run
Reaktion: Das Programm gibt λs.λz. (s (s (s (s z))) aus. (Dieser Test gibt 4 aus, wenn das Wunschkriterium Standardbibliothek vollständig implementiert ist)

T4.4 **Stand:** Das Programm ist im Default-State
Aktion: der Benutzer gibt in den Editor (λx.xx)(λx.xx) ein
Reaktion: das Programm erkennt die Endlosschleife und warnt den Benutzer

T4.5 **Stand:** Das Programm berechnet zurzeit einen λ-Term
Aktion: Der Benutzer drückt auf den Reset-Button
Reaktion: Das Programm beendet die Berechnung und gibt kein Ergebnis aus

T4.6 **Stand:** Das Programm berechnet zurzei einen λ-Term
Aktion: Der Benutzer will das Eingabefeld editieren
Reaktion: Dies ist nicht möglich

T4.7 **Stand:** Das Programm berechnet zurzeit einen λ-Term
Aktion: Der Benutzer betätigt den Pause-Button
Reaktion: Das Programm hört auf β-Redutkionen zu berechnen. Es ist allerdings möglich auf den Continue Button zu drücken um weiter zu Rechnen

T4.8 **Stand:** Das Programm ist im Default-State
Aktion: Der Benutzer hat nach seiner Eingabe „((λx.x)(“in dem Editor stehen
Reaktion: Das Programm gibt die Fehlermeldung aus, dass dies kein gültiger λ-Term ist

T4.9 **Stand:** Das Programm ist im Default State
Aktion: Der Benutzer drückt „(“
Reaktion: Das Programm schreibt automatisch „)“ um die Klammer zu schließen

T4.10 **Stand:** Das Programm kann in jedem Status sein
Aktion: Der Benutzer will das Ausgabefeld editieren
Reaktion: Dies ist unmöglich

T4.11 **Stand:** Das Programm ist im Default-State
Aktion: Der Benutzer gibt in den Editor ein „#dies ist ein test (λz.z) a “ , drückt dann enter und schreibt in der zweiten Zeile
„(λ x. x) y“ und drückt dannach auf Run
Reaktion: Das Programm überspringt den Lambdaterm in der Kommentarzeile und berechnet nur den aus der unteren Zeile. Das Programm gibt y aus. Außerdem ist der Kommentar farblich markiert.

Testen von Highlighting **T5**
Testet: F9 F10 F11 F12 F13

T5.1 **Stand:** Das Programm ist im Default state
Aktion: Der Benutzer gibt „plus“ in den Editor ein
Reaktion: Das Programm erkennt plus aus der Standardbibliothek und markiert es farbig

T5.2 **Stand:** Das Programm ist im Default-State.
Aktion: Der Benutzer gibt in den Editor
(λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z)) ein
Reaktion: Die λs und die „Punkte" sind fett markiert. Wenn der Benutzer über eine Klammer hovert dann wird die jeweils öffnende bzw. schließende Klammer markiert

T5.3 **Stand:** Der letzte Test wurde eingegeben und es wurde auf Run gedrückt.
Aktion: Der Benutzer schaut sich die ersten zwei Steps an. Sie sind jeweils
(λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z))
(λm.λs.λz.(λs.λz.s(s z)) s (m s z)) (λs.λz.s(s z))
Reaktion: In dem ersten Schritt wird die Abstraktion (λn.λm.λs.λz.n s (m s z)) mit Farbe A markiert. Term (λs.λz.s(s z)) wird mit Farbe B markiert um zu zeigen, dass A B reduziert. Im zweiten Schritt wird (λs.λz.s(s z)) welches gerade eingefügt wurde mit Farbe 3 markiert um zu zeigen, dass es im letzten Schritt eingefügt wurde. Mit Farbe A wird (λm.λs.λz.(λs.λz.s(s z)) s (m s z)) markiert und mit Farbe B der rechte (λs.λz.s(s z)) um zu zeigen dass im nächsten Schritt B von eingesetzt wird. Dies widerholt sich in den nächsten Schritten.

T6 Testen der Standard- und Benutzerbibliotheken

Testet: F1 F2 F3 F4 F14 F25 F20 F21 F22 F24 F27 F40 F41 F32 F23 F26 F34 F5

T6.1 **Stand:** Das Programm ist im Default-State.
Aktion: Der Benutzer drückt im Feld Benutzerbibliothek auf den „Plus"-Button.
Reaktion: Es öffnet sich ein Popup, indem der Benutzer eine Funktion definieren kann.

T6.2 **Stand:** Das Programm ist im Default-State.
Aktion: Der Benutzer drückt auf den „Plus" Button im Feld „Benutzerbibliothek" und gibt „first" und λp.p(λa.λb.a) ein.
Reaktion: Wenn er auf den Button drückt öffnet sich ein Popup-Fenster in dem der Benutzer als Name der neuen Funktion Pair eingeben kann und als λ-Term den von ihm gewünschten λ-Term eingibt. Dieser Name wird mit dem λ-Term gespeichert und man kann im Editor anstelle des λ-Terms den Namen eingeben.

T6.3 **Stand:** Der vorherige Test wurde ausgeführt.
Aktion: Der Benutzer gibt in den Editor „first (pair a b)“ ein. Dannach drückt er auf Run.
Reaktion: Das Programm erkennt pair aus seiner Standardbibliothek und ihm ist λa.λb.λf.f a b zugewiesen. Das Programm erkennt first aus dem im letzten Testfall hinzugefügten λ-Term und kann nun β-Reduktion druchführen. Das Programm gibt als Ergebnis a aus.

T6.4 **Stand:** Die vorherigen zwei Tests wurden ausgeführt.
Aktion: Der Benutzer löscht den vorher neu definierten „first“ Term aus der Benutzerbibliothek.
Reaktion: Der Name „first“ verschwindet aus der Liste der Namen in der Benutzerbibliothek und dannach wird die Funktion „first“ nicht mehr im Editor erkannt und gehighlightet.

T6.5 **Stand:** Das Programm ist geöffnet und findet sich im Default-State. An der rechten Seite ist die Standardbibliothek implementiert.
Aktion: Der Benutzer gibt in den Editor „$plus 2 2“ ein und drückt auf den Buttorn „Run“.
Reaktion: Das Programm erkennt das Wort „plus“ aus der Standardbibliothek und ersetzt es durch „λn.λm.λs.λz.n s (m s z)“ die 2en erkennt er jeweils als die Churchzahl 2 und ersetzt sie durch „λs.λz.s(s z)“. Dann führt das Programm β-Reduktion aus und bekommt als Ergebnis„λs.λz. (s (s (s (s z))))“ dies erkennt es ist die Churchzahl 4 und gibt dies aus.

T6.6 **Stand:** Der vorherige Test wurde soeben durchgeführt.
Aktion: Der Benutzer drückt auf den „3-Punkte“-Button, um die gewünschte Anzahl von Schritten zu sehen, die bei der β-Reduktion durchgeführt werden.
Reaktion:

Das Programm zeigt folgende Schritte :
(λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z))
(λm.λs.λz.(λs.λz.s(s z)) s (m s z)) (λs.λz.s(s z))
λs.λz.(λs.λz.s(s z)) s ((λs.λz.s(s z)) s z)
λs.λz.(λz.s(s z)) ((λs.λz.s(s z)) s z)
λs.λz. (s (s ((λs.λz.s(s z)) s z)))
λs.λz. (s (s ((λz.s(s z)) z)))
λs.λz. (s (s (s (s z))))

T6.7 **Stand:** Der Benutzer hat erneut „$plus 2 2“ in den Editor eingegeben.
Aktion: Der Benutzer wählt aus, dass er nur 5 Schritte sehen will, er drückt auf „Run“ und wartet, bis das Ergebnis zu sehen ist, anschließend drückt er auf den „3-Punkte“-Button.
Reaktion: Es werden nun die ersten 5 Schritte angezeigt, wenn er mehr sehen will muss er erneut auf den Button drücken.

T6.8 **Stand:** Das Programm ist im Default state.
Aktion: Der Benutzer gibt einmal „plus 2 2“ in den Editor ein. Einmal gibt er (λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z)) in den Editor ein. Beidesmal drückt er auf run und schaut auf das Ergebnis
Reaktion: Das Ergebnis ist in beiden Fällen das Gleiche.

T6.9 **Stand:** Das Programm ist im Default-State.
Aktion: Der Benutzer gibt „$ minus 1 1“ ein und drückt auf „Run“.
Reaktion: Das Programm erkennt die funktion minus und die churchzahlen und gibt als ergebnis 0 aus. Da das Programm nicht erkennen kann ob es die Churchzahl 0 oder der boolean false ist gibt es immer 0 aus. Im Tutorial des Programmes wird das dem Benutzer erläutert.

T6.10 **Stand:** Das Programm befindet sich im Default-State.
Aktion: Der Benutzer will einen neuen Eintrag in die Benutzerbibliothek einfügen und nennt seinen neuen Term „plus“ (Dieser Term ist schon in der Standardbibliothek definiert.)
Reaktion: Es öffnet sich ein Popup das dem Benutzer sagt, dass er dies nicht darf.

T7 Auswertungsstrategie und De Bruijn

Testet: F27 F28 F29 F30 F31 F19 F35

T7.1 **Stand:** Im Editor steht der Term (λt.λf.f)((λy.(λx.x x))((λx.x)(λx.x)))(λt.λf.f) sonst ist das Programm im Default-State.
Aktion: Der Benutzer lässt den Term ohne eingestellte Auswertungsstrategie (im Default ist das Normalenreihenfolge) und schaut sich den ersten Schritt an. Dann tut er das gleiche mit der Auswertungsstrategie Call-By-Value und mit Call-By-Name.
Reaktion: Bei der Normalreihenfolge und Call-by-Name wird der Redex (λt.λf.f) als erstes reduziert. Bei Call-By-Value wird der Term (λx.x) als erstes reduziert.

T7.2 **Stand:** Im Editor steht der λ-Term (λx. f x x) ((λy. y)z). Die Schrittanzahl für die zu gehenden Steps ist auf 1 eingestellt.
Aktion: Der Benutzer drückt auf den „Step"-Button und drückt auf (λy. y) um diesen Redex als nächstes zu reduzieren.
Reaktion: Das Programm wählt diesen Redex für die β-Reduktion und gibt als nächsten Step „(λx. f x x) z " aus.

T7.3 **Stand:** Das Programm ist im Default-State.
Aktion: Der Benutzer gibt (λy.λx.y x) x a in den Editor ein und drückt auf Run.
Reaktion: Durch die α-Konversion mit De Bruijn-Indizes ist das Ergebnis nicht a a.

Sharing und Speichern **T8**

Testet: F54 F44 F45 F46 F48 F47 F19 F56

T8.1 **Stand:** Das Programm hat einen eingegebenen λ-Term erfolgreich fertig ausgewertet und zeigt die Zwischenergebnisse an.
Aktion: Der Benutzer öffnet im ersten Zwischenergebnis das Kontextmenü uns wählt „To new Tab" aus.
Reaktion: Es öffnet sich ein neuer Browser-Tab, in dem der entsprechende Zwischenschritt nun im Eingabefeld steht und editiert werden kann.

T8.2 **Stand:** Das Programm hat einen λ-Term ausgerechntet
Aktion: Der Benutzer drückt auf den Share button und wählt copy-link
Reaktion: Sofern möglich, wird eine gekürzte URL in sein Clipboard gespeichert

T8.3 **Stand:** Das Programm hat einen λ-Term ausgerechntet
Aktion: Der Benutzer drückt auf den Share button und wählt „E-Mail"
Reaktion: Sofern möglich, wird eine gekürzte URL generiert und diese an den E-Mail-Client des Benutzers übergeben.

T8.4 **Stand:** Der Benutzer hat von vorherigem Test eine Url erhalten
Aktion: Der Benutzer gibt den Link in seine URL Leiste ein
Reaktion: Das Programm öffnet sich mit dem λ-Term bereits in der Eingabe

T8.5 **Stand:** Der Benutzer hat die Funktion first in die Benutzerbibliothek eingefügt. Er hat die Eingabe „(λx.x) y" runnen lassen.
Aktion: Der Benutzer schließt den Browser, fährt den PC herunter, fährt den PC wieder hoch und startet das Programm im selben Browser.
Reaktion: In der Benutzerbibliothek ist immer noch die Funktion „first" und in der Eingabe steht „(λx.x) y"

T8.6 **Stand:** Der vorherige Test wurde ausgeführt
Aktion: Der Benutzer drückt den Button der die History löscht
Reaktion: Die Benutzerbibliothekfuntkionen die aus der History geladen wurden, werden gelöscht. Die Eingabe aus der History wird auch gelöscht

T9 **Tutorial and Prettyprint**
Testet: F57 F7

T9.1 **Stand:** Der Benutzer hat den Term
(λn.λm.λs.λz.n s (m s z)) (λs.λz.s(s z)) (λs.λz.s(s z)) in das Eingabefeld ein und schaut sich den ersten Step an.
Aktion: Bei einer geringen Bildschirmauflösung wird der Lambdaterm nicht in einer Zeile passen. Das Programm verwendet Pretty print
Reaktion: Der Term wird nach einer „)“ Klammer in die neue Zeile gebracht. Jetzt ist der erste Step
(λn.λm.λs.λz.n s (m s z))
(λs.λz.s(s z)) (λs.λz.s(s z))

T9.2 **Stand:** Das Programm ist im Default-State
Aktion: Der Benutzer drückt auf den Hilfe Knopf
Reaktion: Es öffnet sich ein Tutorial, welches die Funktionen des Programmes erklärt

T10 **Ausgabe von Zwischenschritten**
Testet: F36 F15

T10.1 **Stand:** Der Benutzer hat einen λ-Term eingegeben. Der λ-Term sollte mehr als 100 Zwischenschritte haben. Der Benutzer hat den λ-Term auswerten lassen. Lediglich das Ergebnis wurde ausgegeben.
Aktion: Der Benutzer stellt die Anzahl der zu gehenden Schritte auf 15. Der Benutzer drückt auf den „3-Punkte“-Button.
Reaktion: Im Schrittfeld werden 15 Zwischenschritte angezeigt. Der „3-Punkte“-Button ist unter den Zwischenergebnissen sichtbar.

T10.2 **Stand:** Nach dem vorigen Test wurde noch 5 mal auf den „3-Punkte“-Button gedrückt, es sind also 90 Zwischenergebnisse sichtbar.
Aktion: Der Benutzer drückt erneut auf den „3-Punkte“-Button.
Reaktion: Es werden die Zwischenschritte 91 bis 105 der Liste hinzugefügt, die Zwischenschritte 1 bis 5 verschwinden und werden durch einen „3-Punkte“-Button ersetzt.

T10.3 **Stand:** Der Benutzer hat einen λ-Term eingegeben und diesen Auswerten lassen. Dieser λ-Term sollte länger als 50 Zwischenschritte beinhalten. Er stellt die Anzahl der auszugebenden Zwischenergebnisse auf 50 um.
Aktion: Der Benutzer drückt auf den „3-Punkte“-Button.
Reaktion: Im Schrittfeld werden die ersten 50 Zwischenergebnisse angezeigt. Der „3-Punkte“-Button ist unter den Zwischenergebnissen sichtbar.

Shortcuts **T11**

Testet: F49 F50 F51 F52 F53

T11.1 **Stand:** Das Programm ist im Default-State
Aktion: Der Benutzer gibt \in den Editor ein
Reaktion: Ein fettgedrucktes Lambdasymbol λ erscheint

T11.2 **Stand:** In dem Eingabefeld steht eine willkürliche Eingabe
Aktion: Der Benutzer drückt gleichzeitg `Shift` + $\hookleftarrow$
Reaktion: Der Shortcut betätigt den Run Knopf und die Standard Run-Operation wird ausgeführt

T11.3 **Stand:** Das Programm berechnet β-Reduktionen
Aktion: Der Benutzer drückt gleichzeitig `Shift` + $\hookleftarrow$
Reaktion: Die Berechnung terminiert wie sie es getan hätte, hätte der Benutzer den „Reset“-Button gedrückt

T11.4 **Stand:** Das Programm berechnetβ-Reduktionen
Aktion: Der Benutzer drückt gleichzeitig `Shift` + `P`
Reaktion: Die Berechnung pausiert wie sie es getan hätte, als hätter der Benutzer auf den „Pause“-Button gedrückt hätte

T11.5 **Stand:** In dem Eingabefeld steht eine willkürliche Eingabe
Aktion: Der Benutzer drückt gleichzeitig `Shift` + `N`
Reaktion: Das Programm zeigt den ersten Schritt. Der Shortcut ist äquivalent zum drücken des „Schritt“-Buttons

8. Entwicklungsumgebung

Benutzte Software die für das Erstellen des Programmes verwendet wird

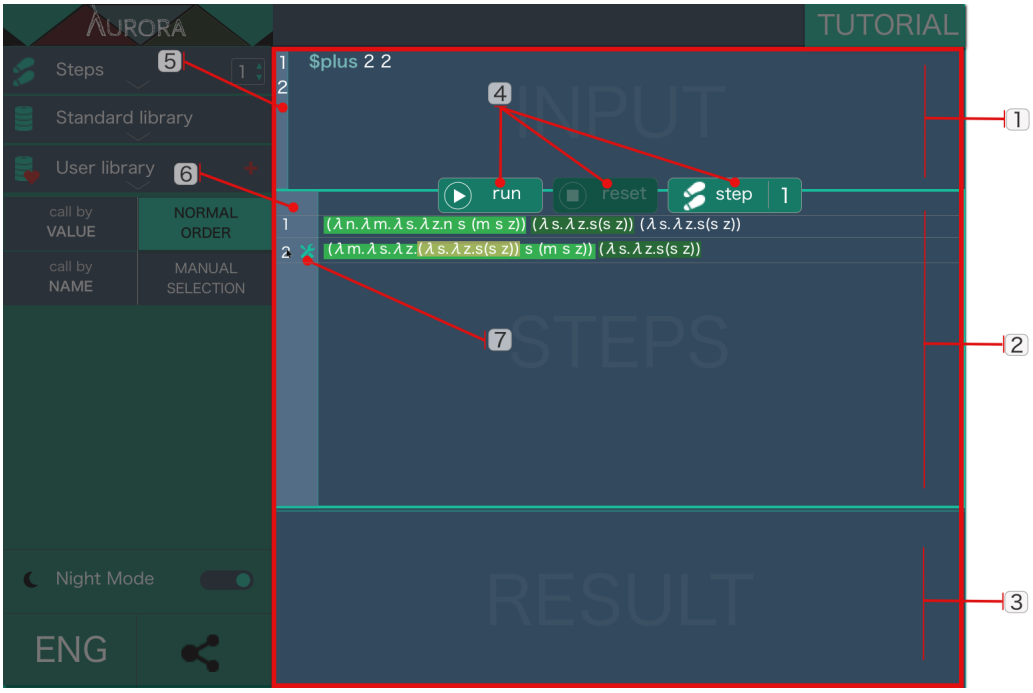
- **Programmiersprache**
  - Java 8
- **Build System**
  - Bazel
  - Apache Ant
- **Editor / IDE**
  Freie Wahl des jeweiligen Entwicklers, benutzt werden unter anderem
  - Eclipse
  - IntelliJ
  - SublimeText
  - Atom
  - Vim
- **Webframework**
  - Google Web Toolkit „GWT“
- **Versionskontrolle**
  - Git auf git.scc.kit.edu. Webanwendung ist Gitlab
- **Dokument Erstellung**
  - LaTeX
- **Testen**
  - JUnit

  - JaCoCo

- **GUI Entwürfe**
  - draw.io
  - Sketch
- **UML und Diagramme zeichnen**
  - UMLet
  - Umbrella
  - BOUML
  - Lucidchart
- **endgültige Programmiersprache auf der Website**
  - JavaScript

A. Seitenentwürfe

A.1. Editor Bereich



(1) Eingabefeld
Hier können die λ-Terme eingegeben werden.

(2) Schrittfeld
Hier könne die einzelnen Zwischenergebnisse der Auswertung angesehen werden.

(3) Ergebnisfeld
Hier wird das Ergebnis der Auswertung angezeigt.

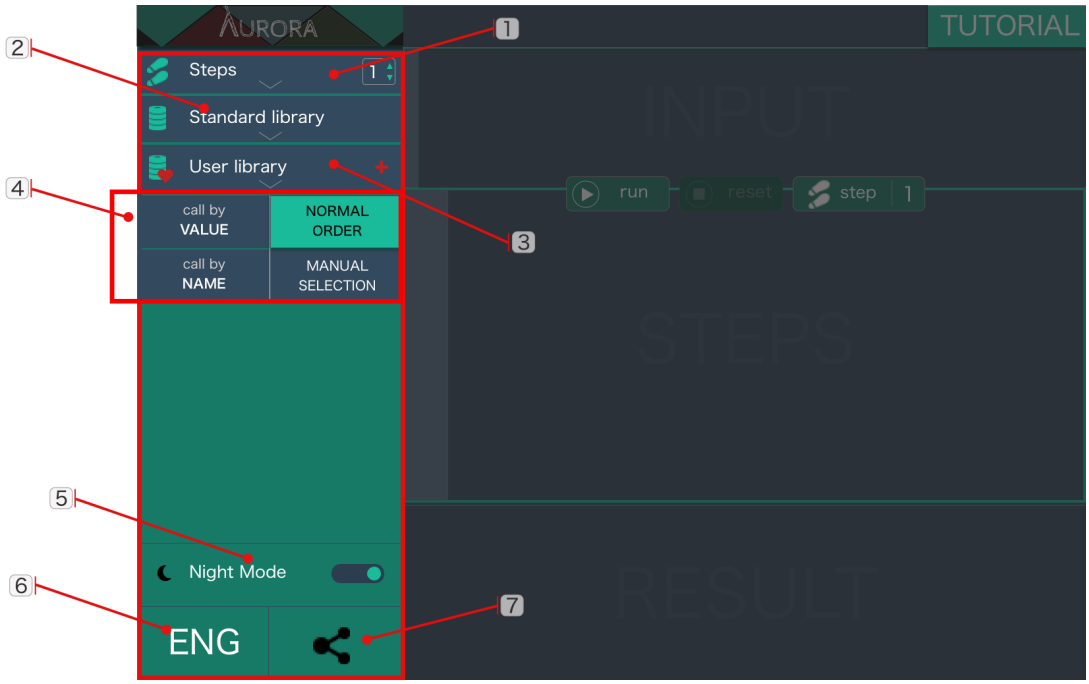
(4) Buttons
Hier kann die Auswertung der Eingabe unter anderem gestartet und angehalten werden.

(5) Zeilenzählung
Zeilenzählung zur besseren Übersicht im Editorfenster.

(6) Zwischenschritt-Zählung
Die Nummer des Zwischenschrittes.

(7) Kontextmenü
Ruft das Kontextmenü zu dem entsprechenden Zwischenschritt auf.

A.2. Sidebar Bereich



(1) Schritte
Hier kann die Anzahl der Zwischenschritte eingestellt werden, die der Ansicht im nächsten Schritt hinzugefügt werden.

(2) Standardbibliothek
Hier kann eine Liste mit allen vordefinierten Funktionen eingesehen werden.

(3) Benutzerbibliothek
Hier kann eine Liste mit allen benutzerdefinierte Funktionen eingesehen werden. Auf dem „Plus"-Button können Funktionen selbst definiert werden.

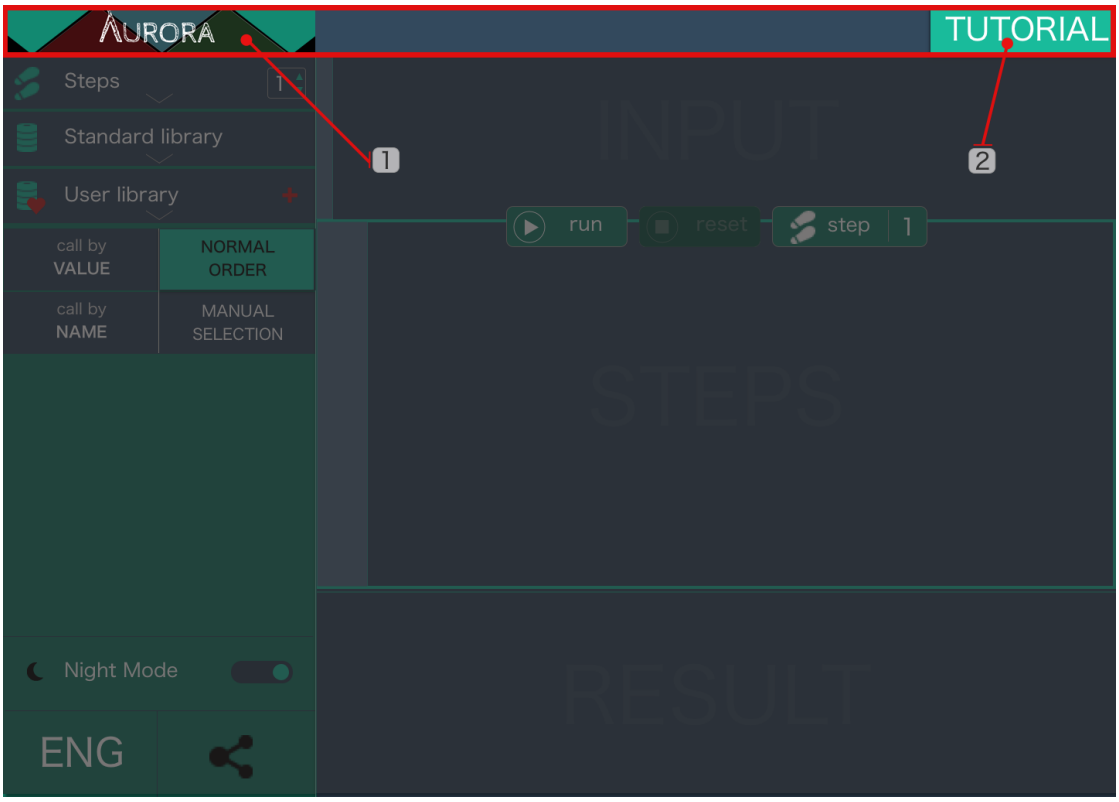
(4) Auswertungsstrategien
Hier kann der Benutzer die Auswertungsstrategie festlegen, die benutzt werden soll.

(5) Nacht Modus
Hier kann der Benutzer den Nacht Modus ein- und ausschalten.

(6) Sprachauswahl
Hier kann der Benutzer zwischen den vorhandenen Programmsprachen wählen.

(7) Teilen
Hier kann der Benutzer auf die Teilenfunktionen des Programms zugreifen.

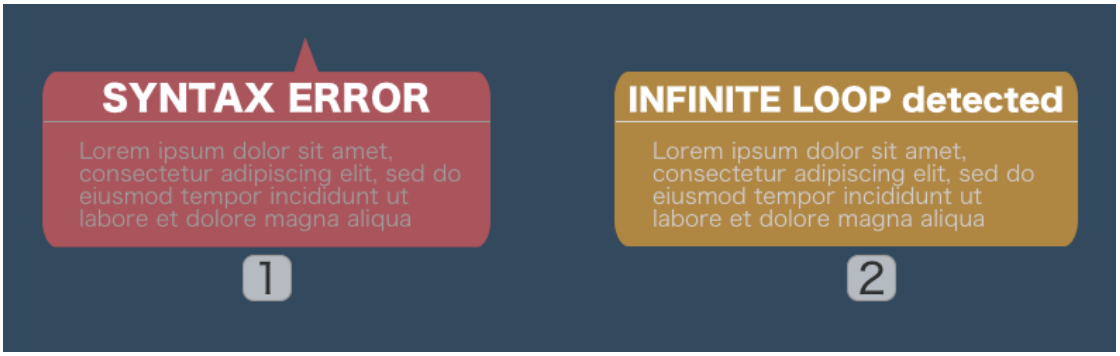
A.3. Top Menu Bereich



(1) λurora
Der Titel der Seite, lädt die Seite neu.

(2) Tutorial
Startet das Tutorial des Programms.

A.4. Pop-ups Bereich



(1) Syntax Fehler
Dieses Popup erscheint, sollte sich in der Eingabe ein Syntax Fehler befinden.

(2) Schleifenerkennung
Dieses Popup erscheint, wenn das Programm eine Endlosschleife erkennt.

B. Glossar

Standardbibliothek
Eine Ansammlung von λ-Termen die einen Funktionsnamen bekommen haben. Die Entwickler von λurora haben die wichtigsten λ-Terme zusammen getragen damit der Benutzer häufig vorkommende λ-Terme nicht selber tippen muss. Der Benutzer kann den Namen der Funktion im Editor verwenden und das Programm ersetzt den Namen automatisch in den gewählten λ-Term.

Benutzerbibliothek
Der Benutzer kann selber Funktionen mit Namen und λ-Term definieren. er kann dann den gewählten Namen im Editor verwenden und der Name wird automatisch durch den λ-Term ersetzt.

Redex
Redex steht für „reducible expression“ oder auf Deutsch „reduzibler Ausdruck“. Ein Redex ist ein Subterm, der durch die Auswertungsstrategien mit β-Reduktion reduzierbar ist.
Zum Beispiel, ist der Term (λx.x) y ist ein Redex. Der Term λx.x aber nicht, da der Term nicht mehr reduzierbar ist.

Normalform
Die Normalform ist ein λ-Term der nicht mehr durch Betareduktion reduzierbar ist. Dies wird im Programm als „Result“ ausgegeben

Auswertungsstrategie
Die Auswertungsstrategie gibt an welcher Redex als nächstes ausgewertet wird und dann mit Beta-Reduktion reduziert wird.

Normalreihenfolge
Der linkeste äußerste Redex wird als erstes ausgewertet und reduziert.

Call-By-Name
Der linkeste äußerste Redex, der nicht von einem λ umgeben ist, wird als erstes ausgewertet und reduziert

Call-By-Value
Der linkeste Redex, der nicht von einem λ umgeben ist und dessen Argument ein Wert ist, wird als erstes ausgewertet und reduziert.

Hover
Mit dem Mauszeiger über einem Element „schewben“, ohne es jedoch zu drücken.

IDE

Integrierte Entwicklungsumgebung (engl. integrated developement environment). Stellt Features bereit, die die Entwicklung von Software vereinfachen soll.

De Bruijn-Indizes
Ermöglicht es, λ-Terme frei von Variablennamen darzustellen.

Clipboard
(dt. Zwischenablage) Speicher, der für das Kopieren von Daten zwischen zwei Programmen gedacht ist. In der Zwischenablage Gespeichertes kann in anderen Programmen einfach eingefügt werden.

localStorage
localStorage ist ein Teil von Web Storage auch „DOM Storage“ genannt. localStorage erlaubt es Daten lokal zu speichern, sogenannte „Local Shared Objects“, auch wenn der Benutzer seinen Browser schließt.

GWT
Google Web Toolkit ist ein Webframework, das erlaubt Webanwendungen in Java zu schreiben. Um dies zu ermöglichen, beinhaltet GWT einen Java zu Javascript Compiler.

Shortcut
(oder auch Tastenkombination) bezeichnet man das gleichzeitige oder aufeinanderfolgende Drücken mehrerer Tasten auf Computertastaturen in einer bestimmten Reihenfolge.

GUI
(engl. graphical user interface, dt. graphische Benutzeroberfläche) bezeichnet eine Form von Benutzerschnittstelle eines Computers. Sie hat die Aufgabe, Anwendungssoftware auf einem Rechner mittels grafischer Symbole, Steuerelemente oder auch Widgets genannt, bedienbar zu machen.

IPD
IPD steht für „Institut für Programmstrukturen und Datenverwaltung“ und beinhaltet mehrere Forschungsgruppen, Programmierparadigmen ist eine davon.

Default-State
Der Default-State ist der Zustand des Programmes in dem es ist, wenn es zum ersten Mal geöffnet wurde und nichts verstellt wurde. Im Default-State wird nichts aus dem localStorage geladen.