Funktionsplotter - Interaktive Funktionsvisualisierung

Projektübersicht

Der **Funktionsplotter** ist ein leistungsstarkes Werkzeug zur Analyse und Visualisierung mathematischer Funktionen. Er ermöglicht es, mathematische Ausdrücke einzugeben, sie zu analysieren und grafisch darzustellen. Das Projekt bietet folgende Hauptfunktionen:

- Auswertung mathematischer Ausdrücke
- Generierung und Visualisierung abstrakter Syntaxbäume (AST)
- Grafische Darstellung von Funktionen in einem kartesischen Koordinatensystem
- Unterstützung verschiedener mathematischer Operationen und Funktionen

Diese interaktive Dokumentation demonstriert die Funktionalitäten des Projekts und gibt Einblicke in die Implementation.

Funktionalitäten

Implementierte Funktionen

- **Ausdruckseingabe**: Funktionen können in Infix-Notation eingegeben werden (z.B. `x^2 + 3*x`) und RPN-Notation (z.B. `x 2 ^ 3 * +`)
- **Taschenrechnermodus**: Sofortige Auswertung von Ausdrücken ohne Variablen
- **AST-Generierung**: Erzeugung eines abstrakten Syntaxbaums für jeden Ausdruck
- **AST-Visualisierung**: Darstellung des AST mittels DOT/Graphviz
- **Funktionsdarstellung**: Zeichnung von Funktionen in einem kartesischen Koordinatensystem
- **Mehrfarbige Darstellung**: Verschiedene Funktionen können in unterschiedlichen Farben dargestellt werden
- **Umfangreiche Unterstützung mathematischer Operationen**: `+`, `-`, `*`, `/`, `^`, `sqrt`, `log`, `ln`, `sin`, `cos`, `tan`, `pi`, `π`, `e` usw.
- **Bedingte Ausdrücke**: Unterstützung von bedingten Ausdrücken (z.B. `x <= 0 ? 0 : x * x`)

Geplante Funktionalitäten

- **Interaktives Zoomen/Verschieben**: Dynamische Anpassung des sichtbaren Bereichs im Koordinatensystem
- **Parametrische Funktionen**: Unterstützung von Funktionen mit anpassbaren Parametern über Schieberegler
- **Logarithmische/Lineare Achsen**: Auswählbare Achsenskalierung

Beispiel: Funktionsvisualisierung

Der folgende Code demonstriert die grundlegende Verwendung des Funktionsplotters:

```
    // Input-Felder für Funktionen und Intervall
    Clerk.markdown("### Eingabeparameter anpassen");

    // Eingabefelder für Parameter ohne Aktualisierung
    Clerk.markdown("**Intervall:** (Format: [min;max]) Bitte
geben Sie kein größeres Intervall als [-40;40] ein.");
    String interval = "[-10;10]";  // interval Update
    Clerk.write(Interaction.input("./demo.java", "// Interval
Update", "interval = \"$\";", interval));

    Clerk.markdown("**Funktionen eingeben:**");

    Clerk.markdown("Funktion f(x) - <span
style='color:red'>rot</span>:");
    String fx = "exp(x -1)"; // fx Update
    Clerk.write(Interaction.input("./demo.java", "// fx Update",
"String fx = \"$\";", fx));

    Clerk.markdown("Funktion g(x) - <span
style='color:blue'>blau</span>:");
    String gx = "x <= 0 ? 0 : x * x";  // gx Update
    Clerk.write(Interaction.input("./demo.java", "// gx Update",
"gx = \"$\";", gx));

    Clerk.markdown("Funktion h(x) - <span
style='color:green'>grün</span>:");
    String hx = "cos(x)";  // hx Update
    Clerk.write(Interaction.input("./demo.java", "// hx Update",
"hx = \"$\";", hx));
```

```
    // FunctionPlotter mit dem Intervall initialisieren
    FunctionPlotter plotter = new FunctionPlotter(interval);

    // AST (Abstrakter Syntaxbaum) für jede Funktion
visualisieren
    Clerk.markdown("### Abstrakte Syntaxbäume (AST)");

    Clerk.markdown("**f(x) = " + fx + "**");
    plotter.drawExpressionAST(fx);    // Zeigt die interne
Struktur des Ausdrucks

    Clerk.markdown("**g(x) = " + gx + "**");
    plotter.drawExpressionAST(gx);

    Clerk.markdown("**h(x) = " + hx + "**");
    plotter.drawExpressionAST(hx);

    // Funktionen mit unterschiedlichen Farben im
Koordinatensystem zeichnen
    Clerk.markdown("### Funktionsgraphen");
    plotter.plotFunction(fx, 255, 0, 0);  // Rot für die
Funktion f(x)
    plotter.plotFunction(gx, 0, 0, 255);  // Blau für die
Funktion g(x)
    plotter.plotFunction(hx, 0, 255, 0);  // Grün für die
Funktion h(x)

    // Ergebnis als SVG-Grafik ausgeben
    plotter.writeTurtle();
```

Edit Code

Für jede dieser Funktionen wird zuerst der abstrakte Syntaxbaum (AST) visualisiert, der die interne Struktur des mathematischen Ausdrucks zeigt. Danach werden die Funktionen im kartesischen Koordinatensystem gezeichnet, wobei jede Funktion eine eigene Farbe erhält (Rot, Blau und Grün).

Am Ende wird die fertige Visualisierung ausgegeben.

Eingabeparameter anpassen

Intervall: (Format: [min;max]) Bitte geben Sie kein größeres Intervall als [-40;40] ein.

Interval Update [-10;10] Send

Funktionen eingeben:

Funktion f(x) - rot:

fx Update exp(x -1) Send

Funktion g(x) - blau:

gx Update x <= 0 ? 0 : x * x Send

Funktion h(x) - grün:

hx Update cos(x) Send

Bedingte Funktionen

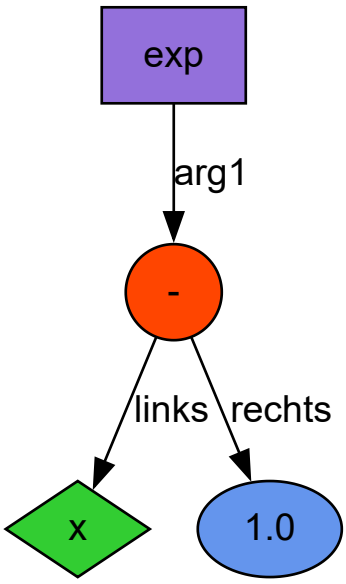
Der Funktionsplotter unterstützt bedingte Ausdrücke mit der Syntax `bedingung ? ausdruck_wenn_wahr : ausdruck_wenn_falsch`.

Beispiele:

- `x <= 0 ? 0 : x * x` - Gibt 0 zurück, wenn $x \leq 0$ ist, sonst x^2
- `x > 0 ? sin(x) : cos(x)` - Gibt sin(x) zurück, wenn $x > 0$ ist, sonst cos(x)

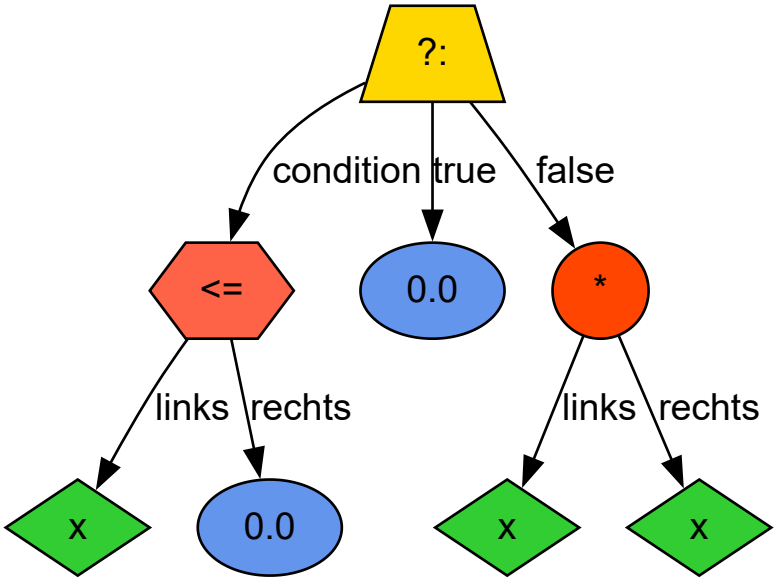
Abstrakte Syntaxbäume (AST)

f(x) = exp(x -1)



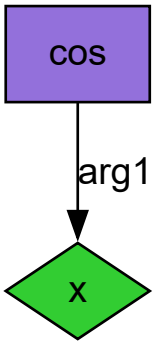
Ausdruck: `exp(x -1)`

g(x) = x <= 0 ? 0 : x * x



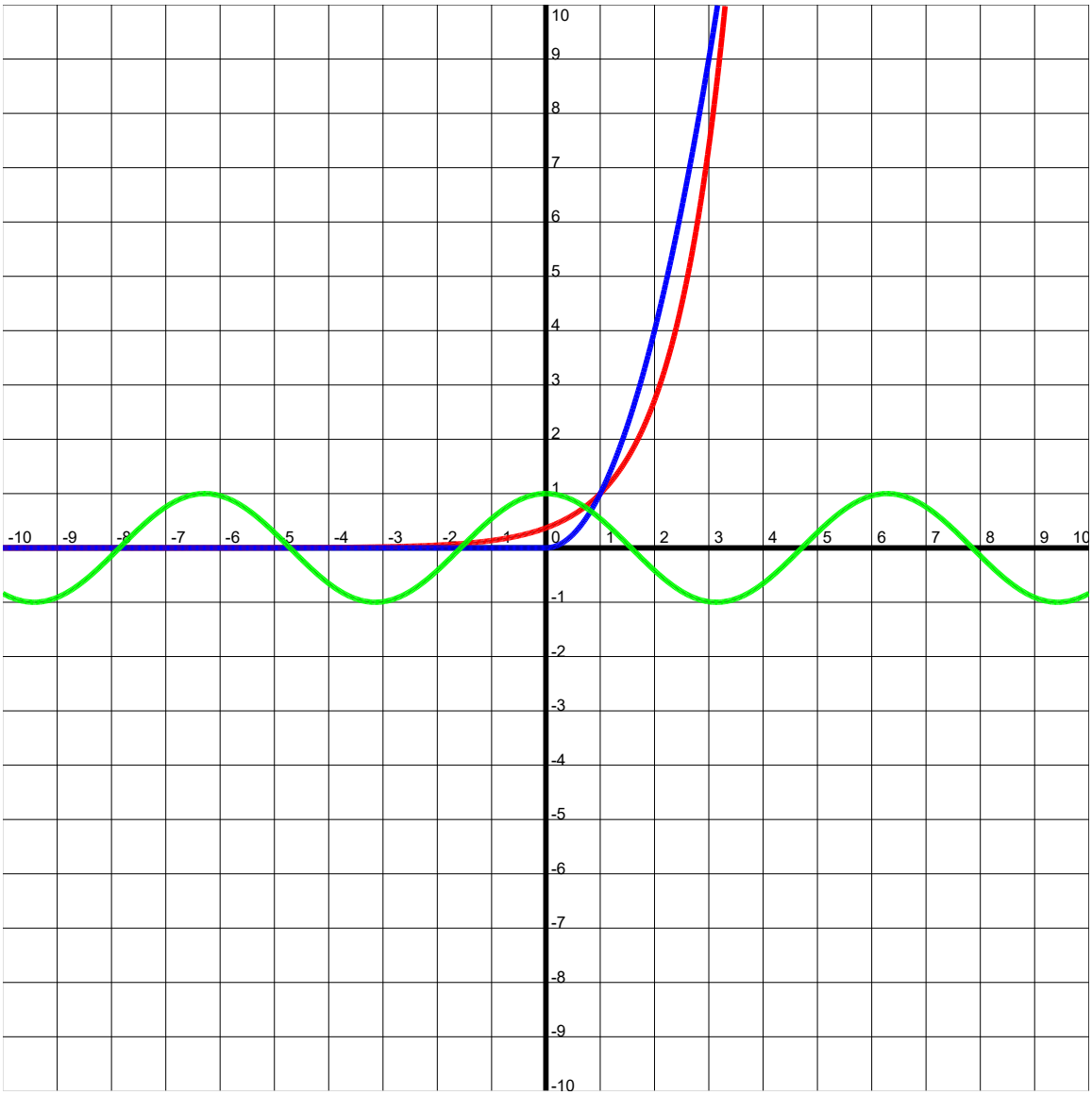
Ausdruck: `x <= 0 ? 0 : x * x`

h(x) = cos(x)



Ausdruck: `cos(x)`

Funktionsgraphen


10
9
8
7
6
5
4
3
2
1
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10