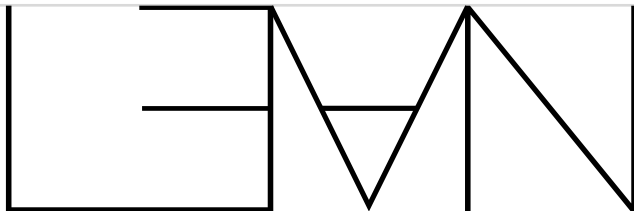




Theorembeweiserpraktikum

Informationen zur Projektpräsentation, Abschlussarbeiten

Jakob von Raumer, Sebastian Ullrich | SS 2022



THEOREM PROVER

Anmeldung zum Modul

Noch bis 26.07. 23:59, Campus-System

Organisation

- Ergebnisse werden vor Mitgliedern des Lehrstuhls vorgestellt
- 27. Juli 2021, 11:00 Uhr, Seminarraum 010 im Informatik-Gebäude
- Jede Gruppe hat **15 Minuten** Zeit
- Folien vorher per Mail an uns schicken

Zur Präsentation an sich

- Beweisideen deutlich machen
- Beweismethode nennen (Induktion, Rekursion, ...)
- Nicht unbedingt jeden einzelnen Fall bei Induktionsbeweisen durchnudeln

Zur Präsentation an sich

- Beweisideen deutlich machen
- Beweismethode nennen (Induktion, Rekursion, ...)
- Nicht unbedingt jeden einzelnen Fall bei Induktionsbeweisen durchnudeln

Als bekannt darf vorausgesetzt werden:

- Lean, Grundzüge von interaktiven Theorembeweisern wie induktive Prädikate, Taktikbeweise, ...
- Big-Step-Semantiken, Typsysteme, Konstantenfaltung grundsätzlich

spezifische Syntax, Taktiken, abhängige Typen eher nicht

Inhaltsvorschlag für das Semantik-Projekt

Benedikt, Joscha, Martin

- Syntax und Semantik vorstellen (kurz)
- Definition der Konstantenfaltung und -propagation
- Semantikerhaltung und Beschleunigungsaussage (nur Aussagen)
- Typisierungsprädikat und -aussage
- Erfahrungen, Feedback, Statistiken (LoC, Anzahl der Lemmas)

Achim, Benjamin, Julian

- Abhängig typisierter Ansatz
- Definition der Konstantenfaltung und -propagation
- Semantikerhaltung und Beschleunigungsaussage
- Determinismusbeweis
- Erfahrungen, Feedback, Statistiken (LoC, Anzahl der Lemmas)

Inhaltsvorschlag für das Semantik-Projekt

Jonas, Pascal, Tim

- Semantikerhaltung und Beschleunigungsaussage
- Idempotenzbeweis
- Techniken zur Beweisminimierung
- Erfahrungen, Feedback, Statistiken (LoC, Anzahl der Lemmas)

Lean-Codesnippets in Folien

Wegen Unicode-Zeichen gar nicht so einfach. Vier Optionen:

- Das Paket listings verwenden
- Das Python-Programm Pygments und xelatex statt pdflatex verwenden
- Gehighlighteten Code auf Google Slides kopieren
- Zugeschnittene Screenshots verwenden

Informationen zu ersteren Optionen:

https://leanprover.github.io/lean4/doc/syntax_highlight_in_latex.html.

Offene Masterarbeiten

- Formalisierungsprojekte für Mathlib gern gesehen
- Gerne auch in Lean 3, wo es schon mehr Automatisierung gibt
- Entweder in Algebra, Topologie oder Kategorientheorie oder mit zusätzlichem Mathematiker
- Entwicklungen von themenspezifischen Taktiken
- Tooling für Lean 4, mehr Interaktivität für die Infoview

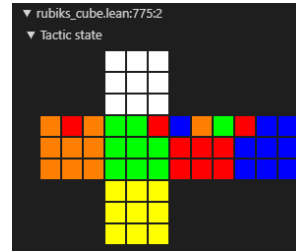
Offene Masterarbeiten – String Diagrams

- VSCode kann mit beliebigen Widgets erweitert werden
- Sudoku- und Zauberwürfel-Widget für Lean 3:

```

403 1003 0013
41  have c44 : s.f (4, 4) = 4 := by naked_single.
42  have c43 : s.f (3, 3) = 3 := by box_logic.
43  have c48 : s.f (8, 8) = 8 := by box_logic.
44  have p32 : s.solver 0 2 2 2 8 := by box_logic.
45  have c88 : s.f (8, 8) = 8 := by pencil_with p32.
46  clear p4.
47  have p33 : s.solver 1 0 2 4 4 := by box_logic.
48  have p34 : s.solver 0 0 7 4 := by box_logic_with p33.
49  have c47 : s.f (4, 7) = 6 := by box_logic.
50  have c78 : s.f (7, 8) = 6 := by box_logic.
51  have c52 : s.f (5, 2) = 6 := by pencil_with p8.
52  clear p6.
53  have c58 : s.f (5, 8) = 1 := by naked_single.
54  have c32 : s.f (3, 2) = 1 := by pencil_with p4.
55  clear p3.
56  have p27 : s.double 3 0 4 9 := by naked_single.
57  have p28 : s.scipio 3 2 2 9 := by naked_single.
58  have p29 : s.double 3 7 2 9 := by naked_single.
59  have c38 : s.f (3, 8) = 8 := by naked_single_with p27 p28 p29.
60  have c88 : s.f (8, 8) = 8 := by box_logic.
61  have c43 : s.f (4, 3) = 8 := by box_logic.
62  have c22 : s.f (2, 2) = 8 := by pencil_with p32.
63  clear p32.
64  have c38 : s.f (3, 8) = 4 := by box_logic.
65  have c82 : s.f (8, 2) = 9 := by box_logic.
66  have c28 : s.f (2, 8) = 9 := by pencil_with p27.
67  have c88 : s.f (8, 8) = 2 := by row_logic.
68  have c28 : s.f (2, 8) = 4 := by pencil_with p32.
69  clear p32.
70  have c18 : s.f (1, 8) = 9 := by row_logic.
71  have c42 : s.f (4, 2) = 2 := by naked_single.
72  have c33 : s.f (3, 3) = 4 := by pencil_with p28.
73  have c82 : s.f (8, 2) = 4 := by col_logic.
74  have c48 : s.f (4, 8) = 9 := by row_logic.
75  have c43 : s.f (4, 3) = 1 := by row_logic.
76  have c38 : s.f (3, 8) = 1 := by row_logic.
77  have c27 : s.f (2, 7) = 2 := by pencil_with p28.
78  have c38 : s.f (3, 8) = 9 := by row_logic.
79  have c33 : s.f (3, 3) = 5 := by row_logic.
80  have c53 : s.f (5, 3) = 2 := by row_logic.
  
```

3	7	8	6	5	4	1	
5	1	7	2	8	6	3	4
2	6	8	9	1	3	7	5
49	249	7	5	3	6	29	8
5	3		4			6	7
1	8	6	5	9	7	4	3
6	3		7	5	4	8	1
7	5	3	8		4		6
8			6	9	3	7	5



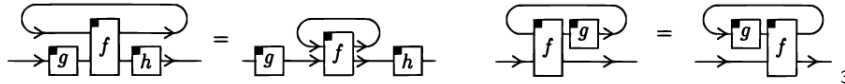
1

2

¹Quelle: Markus Himmel, <https://github.com/TwoFX/sudoku>

²Quelle: Kendall Frey, <https://github.com/kendfrey/rubiks-cube-group>

Offene Masterarbeiten – String Diagrams



- Monoidale Kategorien: Enthalten neben Komposition von Morphismen auch paralleles Produkt von Morphismen
- Verwendet zur Modellierung von Parallelismus, Quantencomputing, Knotentheorie
- String-Diagrams als graphischer Kalkül und nützliches Tool
- Taktiken zur Arbeit mit Monoidalen Kategorien

³Quelle: Peter Selinger, A Survey of Graphical Languages for Monoidal Categories