Humboldt-Universität zu Berlin
Lehrstuhl Logik in der Informatik
Dr. André Frochaux

16. Januar 2026

Logik-Programmierung und Prolog für IMP-Wechselnde

Wintersemester 2025/2026

Übungsblatt 12

Abgabe: *bis 26. Januar 2026, 13.00 Uhr*

Aufgabe 1: (20 Punkte)

Lesen Sie Kapitel 8 aus dem Buch „Learn Prolog Now!“.
Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Sie können zur Lösung dieser Aufgabe und der Zusatzaufgabe alle Prolog-Module verwenden, die Sie unter*

https://hu.berlin/impprolog

vorfinden. Dies gilt insbesondere für die Module `tseitin.pl` *und* `dpll.pl`.[1]

(a) Implementieren Sie ein Prädikat `sat/1`, so dass eine Anfrage

```
?- sat(F).
```

für eine aussagenlogische Formel `F` genau dann erfolgreich ist, wenn `F` erfüllbar ist.

Hinweise: Ihr Prädikat soll zu der Formel `F` zuerst eine *erfüllbarkeitsäquivalente* 3-*KNF* konstruieren, und anschließend deren Erfüllbarkeit mit dem DPLL-Algorithmus testen. Es macht nichts, wenn Ihr Prädikat für eine erfüllbare Formel mehrfach

```
true.
```

ausgibt.

(b) Für die folgenden Teilaufgaben betrachten wir *geordnete und beschriftete Binärbäume* (analog zu Blatt 6) mit einer leicht geänderten Termrepräsentation.[2] Das folgende Prädikat implementiert eine *pre-order-Traversierung* eines gegebenen Binärbaums `B` in Prolog und veranschaulicht die leicht geänderte Termrepräsentation:

```
preorder(node(T, nil, nil), [T]) :- !.
preorder(node(T, L, R), [T|X]) :-
    preorder(L, LX), preorder(R, RX), append(LX, RX, X).
```

Schreiben Sie eine *Definite Clause Grammar mit einem zusätzlichen Argument*, so dass die Anfrage

```
?- pt(B, X, []).
```

für einen Binärbaum `B` und eine Liste `X` genau dann erfüllt ist, wenn `X` eine pre-order-Traversierung von `B` ist.

[1] Verfügbar ab 19.01.26 ca. 20:00 Uhr.
[2] Dennoch haben innere Knoten weiterhin exakt zwei Kinder.

— *auf der nächsten Seite geht's weiter* —

(c) Wir nennen einen Binärbaum einen *binären Suchbaum*, wenn seine Knoten nur mit Ganzzahlen beschriftet sind und für jeden seiner inneren Knoten `node(Z, L, R)` gilt: Jeder Knoten in `L` ist mit einer Zahl kleiner `Z`, und jeder Knoten in `R` mit einer Zahl größer `Z` beschriftet.

Schreiben Sie eine *Definite Clause Grammar mit drei zusätzlichen Argumenten*, so dass die Anfrage

```
?- bst(B, Min, Max, X, []).
```

genau dann erfüllt ist, wenn `X` die pre-order-Traversierung für den *binären Suchbaum* `B` ist, und zusätzlich `Min` und `Max` die kleinste, beziehungsweise größte Ganzzahl ist, die als Beschriftung in `B` vorkommt.

Hinweis: Nutzen Sie die Möglichkeit, Ihrer Definite Clause Grammar mit Hilfe der Notation {...} zusätzliche Ziele hinzuzufügen. Verwenden Sie gegebenfalls das eingebaute Prädikat `integer/1`.

Aufgabe 2: (20 Zusatzpunkte)

Für Vergleiche von SAT-Solvern werden 3-KNF oft im sogenannten DIMACS-Format angegeben.[3] Implementieren Sie ein Prädikat `sat_dimacs/1`, welches als Argument den Namen einer Datei erhält, so dass beispielsweise die Anfrage

```
?- sat_dimacs('knf.cnf').
```

genau dann erfolgreich ist, wenn die in der Datei `knf.cnf` repräsentierte 3-KNF erfüllbar ist. Ist diese nicht erfüllbar, soll das Ergebnis `false.` sein.

Sie können in Ihrer Implementation davon ausgehen, dass die aufgerufene Datei im aktuellen Verzeichnis existiert und dem DIMACS-Standard entspricht.

Unter `https://hu.berlin/impprolog` finden Sie auch Beispieldateien im DIMACS-Format.

[3] So waren auch auf der SAT Competition 2025 [Link: `http://www.satcompetition.org/`] die 2011 festgelegten Regeln gültig, welche auch für uns das DIMACS-Format definieren sollen (vgl. `https://satcompetition.github.io/2025/benchmarks.html`).