Übungsaufgaben Theoretische Informatik und Logik

INF-B-290

TU Dresden

Sommersemester 2024

Diese Notizen sind während meiner Vorbereitung für die Prüfung *Theoretische Informatik und Logik* entstanden.

- Link zur Vorlesung
- Link zu den Online-Vorlesungen (Sommersemester 2021)

Wenn Lösungen mit **TODO** markiert sind, sind diese nicht richtig ausgearbeitet und sollten eigentlich später nochmal genauer betrachtet werden.

Disclaimer: Das ganze hat für die Note 3,0 gereicht.

1. Übungsblatt

Aufgabe B

> Zeigen Sie: Wenn es möglich ist, für zwei beliebige Turing-Maschinen zu entscheiden, ob sie dieselbe Sprache akzeptieren, so ist es auch möglich, für beliebige Turing-Maschinen zu entscheiden, ob sie die leere Sprache akzeptieren.

gesucht: Erkennt die Turing-Maschine M die leere Sprache $L(M) = \emptyset$?

- Konstruktion einer Turing-Maschine $M_\emptyset$, die für jedes Wort w ablehnt $\Rightarrow L(M_\emptyset) = \emptyset$
- Entscheide, ob M und $M_\emptyset$ die gleiche Sprache erkennen:
  - M und $M_\emptyset$ erkennen die gleiche Sprache $\Rightarrow L(M) = \emptyset$
  - M und $M_\emptyset$ erkennen **nicht** die gleiche Sprache $\Rightarrow L(M) \neq \emptyset$

Aufgabe 1.1

> Zeigen Sie folgende Aussagen:
> **c)** $|\mathbb{R}| \neq |\mathbb{N}|$

siehe Cantors zweites Diagonalargument

> Zeigen Sie folgende Aussagen:
> **d)** für jede nicht-leere endliche Menge Σ ist Σ^* abzählbar unendlich.

besser $f : \mathbb{N} \to \Sigma^*$:

- $\Sigma = \{a_1, a_2, \ldots, a_n\}$, $w \in \Sigma^*$
- $|w| = 0$
  - $1 \mapsto w = \varepsilon$

  $(|\ldots| = n^0 = 1)$
- $|w| = 1$ (falls $|a_1| = |a_2| = \ldots = |a_n| = 1$)
  - $2 \mapsto w = a_1$
  - $3 \mapsto w = a_2$
  - $\ldots$
  - $n + 1 \mapsto w = a_n$

  $(|\ldots| = n)$
- $|w| = 2$ (falls $|a_1| = |a_2| = \ldots = |a_n| = 1$)
  - $n + 2 \mapsto w = a_1 a_1$
  - $n + 3 \mapsto w = a_2 a_1$
  - $\ldots$
  - $2n + 1 \mapsto w = a_n a_1$
  - $2n + 2 \mapsto w = a_1 a_2$
  - $2n + 3 \mapsto w = a_1 a_2$
  - $\ldots$
  - $3n + 1 \mapsto w = a_n a_2$
  - $3n + 2 \mapsto w = a_1 a_3$
  - $\ldots$
  - $(n + 1)n + 1 \mapsto w = a_n a_n$

  $(|\ldots| = n^2)$
- $\ldots$

wahrscheinlich falsch:

1. $\Sigma^* = \{\varepsilon\} \cup \Sigma \cup \Sigma^2 \cup \ldots$
2. $|\Sigma| = |\mathbb{N}|$
3. $|\Sigma^*| = 1 + |\mathbb{N}| + |\mathbb{N} \times \mathbb{N}| + |\mathbb{N} \times \mathbb{N} \times \ldots| + \ldots$
4. $|\mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$
   $\Rightarrow |\mathbb{N} \times \mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$
   $\Rightarrow \ldots$
5. $|\mathbb{N}| + |\mathbb{N}| = |\mathbb{N}|$
6. $|\Sigma^*| = |\mathbb{N}|$

Aufgabe 1.2

Geben Sie für folgende Sprachen Aufzähler an:
a) $L_1 = \{3n | n \in \mathbb{N}\}$, wobei die Ausgabe unär kodiert sein soll

turingmachine.io

```
input: "11"
blank: _
start state: mark_input_end
table:
  mark_input_end:
    1: {R: mark_input_end}
```

```
    _: {write: S, L: move_left}
  move_left:
    [1, S]: L
    _: {R: take1}
  take1:
    1: {write: _, R: append3}
    S: {write: _, R: done}
  append3:
    [1, S]: R
    _: {write: 1, R: append2}
  append2:
    _: {write: 1, R: append1}
  append1:
    _: {write: 1, L: move_left}
  done:
```

(siehe auch 1.4 b)

> Geben Sie für folgende Sprachen Aufzähler an:
> **b)** $L_2 = \{a^n b^n | n \in \mathbb{N}\}$

turingmachine.io

```
input: "111"
blank: _
start state: move_right
table:
  move_right:
    1: R
    [_, a]: {L: prepend_a}
  prepend_a:
    [a, b]: L
    _: {R: done}
    1: {write: a, R: append_b}
  append_b:
    [a, b]: R
    _: {write: b, L: prepend_a}
  done:
```

Aufgabe 1.3

> **a)** Konstruieren Sie eine Turing-Maschine A_{mul}, welche die Multiplikation zweier natürlicher Zahlen implementiert. Dabei sollen sowohl die Eingaben als auch die Ausgabe unär kodiert sein.

turingmachine.io

```
input: "11_111"
blank: _
start state: skip_right_a
table:
  skip_right_a:
```

```
  1: R
  _: {R: skip_right_b}
skip_right_b:
  1: R
  _: {write: S, L: skip_left_b}
# move left over the inputs
skip_left_b:
  1: L
  _: {L: skip_left_a}
skip_left_a:
  1: L
  _: {R: take1}
# subtract one from a
take1:
  1: {write: _, R: skip_input}
  _: {R: cleanup}
# skip right over inputs and outputs
skip_input:
  [_, 1]: R
  S: {R: skip_output}
skip_output:
  1: R
  _: {L: skip_output_to_b}
# add 1 to output for every 1 in b and turn 1 to 2 in b
skip_output_to_b:
  1: L
  S: {L: flip_b}
flip_b:
  2: L
  1: {write: 2, R: append_1}
  _: {R: unflip_b}
append_1:
  [2, S, 1]: {R: append_1}
  _: {write: 1, L: skip_output_to_b}
# turn 2 to 1 in b
unflip_b:
  2: {write: 1, R: unflip_b}
  S: {L: skip_left_b}
# erase up to and including "S"
cleanup:
  [_, 1]: {write: _, R: cleanup}
  S: {write: _, R: done}
done:
```

Aufgabe 1.4

Auf Folie 27 der 2. Vorlesung vom 11.4.2024 wird innerhalb des Widerspruchsbeweises zur Berechenbarkeit der Busy-Beaver-Funktion eine Turingmaschine M_{*2} mit Alphabet $\{x, \}$ verwendet, welche die Funktion $x^n \to x^{2n}$ berechnet.
b) Geben Sie eine Einband-Turingmaschine über dem Alphabet $\{x, \}$ an, die die Funktion $x^n \to x^{2n}$ berechnet.

turingmachine.io

```
input: "xxxx"
blank: _
start state: add_space
table:
  add_space:
    x: {write: _, L: prepend_x}
    _: {R: skip_x0_right_x}
  prepend_x:
    _: {write: x, L: add_space}
  skip_x0_right_x:
    x: {R: skip_x0_right_0}
    _: {L: fill_space_left_0}
  skip_x0_right_0:
    _: {R: skip_x0_right_x}
    x: {L: add_space}
  fill_space_left_0:
    _: {write: x, L: fill_space_left_x}
  fill_space_left_x:
    x: {L: fill_space_left_0}
    _: {R: cleanup}
  cleanup:
    x: {write: _, R: done}
  done:
```

2. Übungsblatt

Aufgabe C

> Zeigen Sie, dass $\{1\}^*$ unentscheidbare Teilmengen besitzt.

Die Potenzmenge einer abzählbar unendlichen Menge ist überabzählbar.

Cantors Diagonalargument:

- $\mathcal{P}(A) = \{S_1, S_2, S_3, \ldots\}$
- $T = \{a_i | a_i \in A, a \notin S_i\}$
- $\Rightarrow T$ enthält a_i, aber S_i nicht $\Rightarrow T \neq S_i$
  - $\Rightarrow T$ enthält a_1, aber S_1 nicht $\Rightarrow T \neq S_1$
  - $\Rightarrow T$ enthält a_2, aber S_2 nicht $\Rightarrow T \neq S_2$
  - $\ldots$
- $\Rightarrow T \notin \mathcal{P}(A)$
- $\Rightarrow \mathcal{P}(A)$ ist überabzählbar.

Aufgabe D

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **d)** Die Ackermannfunktion ist total und damit LOOP-berechenbar.

siehe 3. Vorlesung, Seite 17ff

Aufgabe 2.1

Zeigen Sie, dass folgende Funktionen $f : \mathbb{N}^2 \to \mathbb{N}$ LOOP-berechenbar sind:
a) $f(x, y) := \max(x - y, 0)$

While simulator

```
result := x + 0
LOOP y DO
  result := result - 1
END
```

Zeigen Sie, dass folgende Funktionen $f : \mathbb{N}^2 \to \mathbb{N}$ LOOP-berechenbar sind:
b) $f(x, y) := x \cdot y$

```
result := 0
LOOP x DO
  LOOP y DO
    result := result + 1
  END
END
```

Zeigen Sie, dass folgende Funktionen $f : \mathbb{N}^2 \to \mathbb{N}$ LOOP-berechenbar sind:
c) $f(x, y) := \max(x, y)$

While simulator

```
result := x
LOOP x DO
  y := y - 1
END
LOOP y DO
  result := result + 1
END
```

Zeigen Sie, dass folgende Funktionen $f : \mathbb{N}^2 \to \mathbb{N}$ LOOP-berechenbar sind:
d) $f(x, y) := \text{ggT}(x, y)$

While simulator, Euklidischer Algorithmus

(Beim Verwenden von `IF x != 0 THEN` scheint es einen Bug zu geben und das ganze Programm einfach nicht zu laufen.)

```
result := b
# <if a > 0>
nonzero := 0
LOOP a DO
  nonzero := 1
END
```

```
LOOP nonzero DO
# </if a > 0>
  LOOP b DO
    # <if b > 0>
    nonzero := 0
    LOOP b DO
      nonzero := 1
    END
    LOOP nonzero DO
    # </if b > 0>
      # now we are inside WHILE b != 0
      aminusb := a
      LOOP b DO
        aminusb := aminusb - 1
      END
      # <if aminusb > 0>
      nonzero := 0
      LOOP aminusb DO
        nonzero := 1
      END
      else := 1
      LOOP nonzero DO
      # </if aminusb > 0>
        # if a > b
        LOOP b DO
          a := a - 1
        END
        else := 0
      END
      LOOP else DO
        # if b <= a
        LOOP a DO
          b := b - 1
        END
      END
    END
  END
  result := a
END
```

Aufgabe 2.3

Zeigen Sie, dass es keine Many-One-Reduktion vom Halteproblem P_{halt} von Turing-Maschinen auf das Leerheitsproblem
$P_{\text{leer}} := \{\text{enc}(M) | L(M) = \emptyset\}$
von Turing-Maschinen gibt.

- P_{halt} ist semi-entscheidbar und unentscheidbar. Daraus folgt P_{halt} ist nicht co-semi-entscheidbar. (Wäre P_{halt} semi-entscheidbar und $\neg P_{\text{halt}}$ ebenfalls semi-entscheidbar (gleichbedeutend mit P_{halt} ist co-semi-entscheidbar), wäre P_{halt} entscheidbar.)

- P_{halt} ist *nicht co-semi-entscheidbar* und $P_{\text{halt}} \leq_m P_{\text{leer}}$
  $\Rightarrow P_{\text{leer}}$ ist *nicht co-semi-entscheidbar*
  $\Leftrightarrow \neg P_{\text{leer}} := \{\text{enc}(M) | L(M) \neq \emptyset\}$ (das Komplement zu P_{leer}) ist nicht semi-entscheidbar
- Um einen Widerspruch herbeizuführen, ist zu zeigen, dass $\neg P_{\text{leer}}$ semi-entscheidbar ist, bspw. mit einer Reduktion $\neg P_{\text{leer}} \leq_m P_{\text{halt}}$.

  Wikipedia verweist auf Introduction to Automata Theory, Languages, and Computation (3rd Edition). In Kapitel 9.3.2 Turing Machines That Accept the Empty Language (Seite 410) wird P_{leer} als L_e eingeführt und $L_{ne} = \neg P_{\text{leer}} = \{\text{enc}(M) | L(M) \neq \emptyset\}$.

  Theorem 9.8 beweist, dass $\neg P_{\text{leer}}$ semi-entscheidbar ist (L_u ist in Kapitel 9.2.3 (Seite 403) als die Sprache der Universellen Turing-Maschine definiert ($L_u = \{\text{enc}(M, w) | M$ ist eine Turing-Maschine, $w \in L(M)\}$).):

  - Konstruktion einer **nicht-deterministische** Turing-Maschine M_{Orakel} mit der Eingabe $\text{enc}(M) \in \neg P_{\text{leer}}$
  - M_{Orakel} nutzt seine nicht-deterministischen Fähigkeiten, um $w \in L(M)$ zu raten.
  - w wird auf das Band geschrieben.
  - Halteproblem:
    * Die Turing-Maschine M wird auf w simuliert.
    * Akzeptiert M, so akzeptiert auch M_{Orakel}.

  Sollte M also irgendeine Eingabe akzeptieren ($L(M) \neq \emptyset$), rät M_{Orakel} diese und akzeptiert somit die Eingabe $\text{enc}(M)$.

  Damit ist $\neg P_{\text{leer}}$ semi-entscheidbar.

  Widerspruch: Es kann somit keine Many-One-Reduktion $P_{\text{halt}} \leq_m P_{\text{leer}}$ geben.

siehe auch:

- "Berechenbarkeit #33 - Epsilon-Halteproblem und Leerheitsproblem sind unentscheidbar" von @NLogSpace (ab 8min)
- "Berechenbarkeit #36 - Semi-Entscheidbarkeit" von @NLogSpace

Aufgabe 2.4

> Zeigen Sie, dass jede semi-entscheidbare Sprache L auf das Halteproblem P_{halt} many-one-reduziert werden kann.

Dazu muss eine totale turing-berechenbare Funktion $f : L \to P_{\text{halt}}$ gefunden werden. Da L semi-entscheidbar ist existiert eine Turing-Maschine M_L, die alle Worte $w \in L$ akzeptiert und ewig läuft, wenn $w \notin L$.

Idee: $f(w)$ mit $w \in L$ konstruiert einer Turing-Maschine M_w, die zuerst das Band löscht, dann $w \in L$ auf das Band schreibt und anschließend das Halteproblem $\text{enc}(M_L, w) \in P_{\text{halt}}$ simultiert.

Wenn M_L auf w hält, hält auch M_w und die Eingabe $w \in L$ wird akzeptiert. Ist $w \notin L$ läuft M_L und damit auch M_w ewig.

3. Übungsblatt

Aufgabe E

> Geben Sie eine Turing-Maschine $A_{\mod 2}$ an, die die Funktion $f : N \to N$ mit $f(x) = (x \mod 2)$ berechnet. Stellen Sie dabei die Zahlen in unärer Kodierung dar.

turingmachine.io

```
input: "11111"
blank: _
start state: even
table:
  even:
    1: {write: _, R: odd}
    _: {R: undo_move_right}
  odd:
    1: {write: _, R: even}
    _: {write: 1, R: undo_move_right}
  undo_move_right:
    _: {L: done}
  done:
```

Aufgabe G

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **a)** Die Menge der Instanzen des Postschen Korrespondenzproblems, welche eine Lösung haben, ist semi-entscheidbar.

wahr, das *PCP* ist semi-enscheidbar. Wenn ein *PCP* mit n "Dominosteinen" eine Lösung der Länge m hat, dann kann diese Lösung mittels Brute-Force in $O(n^m)$ gefunden werden und die Turing-Maschine hält.

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **b)** Das Postsche Korrespondenzproblem ist bereits über dem Alphabet $\Sigma = \{a, b\}$ nicht entscheidbar.

wahr, **TODO**

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **c)** Es ist entscheidbar, ob eine Turingmaschine nur Wörter akzeptiert, die Palindrome sind. (Ein Palindrom ist ein Wort $w = (a_1, \ldots, a_n)$ mit $(a_1, \ldots, a_n) = (a_n, \ldots, a_1)$.)

falsch, **TODO** $P_{\text{halt}} \leq_m P_{\text{Palindrom}}$

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **d)** P_{halt} ist semi-entscheidbar.

wahr

$P_{\text{halt}} := \{\text{enc}(M, w) | M \text{ hält auf } w\}$

Konstruktion einer Turing-Maschine M_{halt} mit der Eingabe enc(M, w) als Semi-Entscheider für P_{halt}:

- lösche das Band w
- schreibe w auf das Band
- simuliere M auf w
- akzeptiert M, dann war enc$(M, w) \in P_{\text{halt}}$
- hält M nicht, hält auch M_{halt} nicht
- $\Rightarrow$ **semi-entscheidbar**

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
e) Es ist nicht entscheidbar, ob die von einer deterministischen Turing-Maschine berechnete Funktion total ist.

wahr: Ist f total? $\Leftrightarrow$ Hält M_f für w? $\Leftrightarrow$ enc$(M_f, w) \in P_{\text{halt}}$

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
f) Es gibt reguläre Sprachen, die nicht semi-entscheidbar sind.

Aufgabe 3.1

Besitzen folgende Instanzen P_i des Postschen Korrespondenzproblems Lösungen? Begründen Sie Ihre Antwort.

a) $P1$		
a	$abaa$	ab
aaa	ab	b

Lösung für $s_a = (1, 0), |s_a| = 2$

Besitzen folgende Instanzen P_i des Postschen Korrespondenzproblems Lösungen? Begründen Sie Ihre Antwort.

b) $P2$		
ab	baa	aba
aba	aa	baa

womöglich, aber keine Lösung für $|s_b| < 16$

Besitzen folgende Instanzen P_i des Postschen Korrespondenzproblems Lösungen? Begründen Sie Ihre Antwort.

c) $P3$			
bba	ba	ba	ab
b	baa	aba	bba

womöglich, aber keine Lösung für $|s_c| < 12$

```
import multiprocessing
from functools import cache

post = [
    # ("a", "aaa"),
    # ("abaa", "ab"),
    # ("ab", "b"),
    ("ab", "aba"),
    ("baa", "aa"),
    ("aba", "baa"),
    # ("bba", "a"),
    # ("ba", "baa"),
    # ("ba", "aba"),
    # ("ab", "bba"),
]


@cache
def path_to_strings(path):
    if not path:
        return ([], [])
    a, b = path_to_strings(path[:-1])
    i = path[-1]
    return (a + [post[i][0]], b + [post[i][1]])


def find_solution(worker_num, q, finished):
    depth = 0
    while True:
        path = q.get()
        if len(path) > depth:
            depth = len(path)
            print("worker", worker_num, "depth:", depth)
        assert depth < 20
        # generate both strings
        top, bottom = path_to_strings(path)
        if "".join(top) == "".join(bottom):
            finished.put(path)
            break
        # create new paths from the curent path plus each tuple
        for i in range(len(post)):
            q.put(path + (i,))


# intial paths of length 1 each starting with another PCP tuple
q = multiprocessing.Queue()
for i in range(len(post)):
    q.put((i,))

# start 8 processes that will write their results to finished
finished = multiprocessing.Queue()
workers = [
```

```
    multiprocessing.Process(target=find_solution, args=(i, q, finished))
    for i in range(8)
]
for worker in workers:
    worker.start()
try:
    # wait for the first result
    result = finished.get()
finally:
    # kill all workers
    for worker in workers:
        worker.kill()
        worker.join()
# print result
print(result)
top, bottom = path_to_strings(result)
print(top, "==", bottom)
```

Aufgabe 3.2

> Zeigen Sie, dass das Postsche Korrespondenzproblem über einem einelementigen Alphabet entscheidbar ist.

Jede Instanz des Postschen Korrespondenzproblem über einem einelementigen Alphabet $\Sigma = \{a\}$ ist durch eine Menge $\{p_{(i_1,j_1)}, p_{(i_2,j_2)}, \ldots, p_{(i_n,j_n)}\}$, wobei $p_{(i,j)} = (a^i, a^j)$, gegeben. Die Lösung ist berechenbar, genau dann wenn eine Lösung $s = (s_1, s_2, \ldots, s_n)$ existiert:

- $(a^{i_1})^{s_1} \cdot (a^{i_2})^{s_2} \cdot \ldots \cdot (a^{i_n})^{s_n} = (a^{j_1})^{s_1} \cdot (a^{j_2})^{s_2} \cdot \ldots \cdot (a^{j_n})^{s_n}$
  (Die Reihenfolge der Elemente in der Lösung p_{i_k,j_k} ist beliebig vertauschbar, da $\Sigma = \{a\}$.)
- Die erzeugten Worte sind gleich, wenn sie die gleiche Länge haben.
  $\Rightarrow i_1 s_1 + i_2 s_2 + \ldots + i_n s_n = j_1 s_1 + j_2 s_2 + \ldots + j_n s_n$
- $\Rightarrow (i_1 - j_1)s_1 + (i_2 - j_2)s_2 + \ldots + (i_n - j_n)s_n = 0$
- Lösbarkeit:
  - $i_k = j_k$ (triviale Lösung):
    - $s_k = 1$
    - $s_{k'|k' \neq k} = 0$
  - $i_{k_1} > j_{k_1}$ **und** $i_{k_2} < j_{k_2}$:
    - $s_{k_1} = j_{k_2} - i_{k_2}$
    - $s_{k_2} = i_{k_1} - j_{k_1}$
    - $s_{k'|k' \neq k_1, k' \neq k_2} = 0$

    $\Rightarrow +(i_{k_1} - j_{k_1})(j_{k_2} - i_{k_2}) + (i_{k_2} - j_{k_2})(i_{k_1} - j_{k_1}) = 0$
    $\Leftrightarrow -(i_{k_1} - j_{k_1})(i_{k_2} - j_{k_2}) + (i_{k_2} - j_{k_2})(i_{k_1} - j_{k_1}) = 0$
  - sonst alle $i_k > j_k$ oder alle $i_k < j_k$: $\nexists s_k > 0$

Es ist also lediglich zu prüfen, ob die Instanz

- **ein Element** $p_{(i_k,i_k)}$ **beinhaltet oder**
- **zwei Elemente**
  - $p_{(i_{k_1},j_{k_1})}$ **mit** $i_{k_1} > j_{k_1}$ **und**
  - $p_{(i_{k_2},j_{k_2})}$ **mit** $i_{k_2} < j_{k_2}$

beinhaltet.

Beispiele:

1. $P_1 = \{p_{(5,2)}, p_{(1,3)}\}$
   $\Rightarrow 5s_1 + 1s_2 = 2s_1 + 3s2$
   $\Rightarrow 3s_1 - 2s_2 = 0$
   $\Rightarrow s_1 = 2, s_2 = 3$
2. $P_2 = \{p_{(1,2)}, p_{(2,3)}\}$
   $\Rightarrow 1s_1 + 2s_2 = 2s_1 + 3s2$
   $\Rightarrow 1s_1 + 1s_2 = 0$
   $\Rightarrow \nexists s_1 > 0, s_2 > 0$

Aufgabe 3.3

Es sei $T := \{\text{enc}(M) | M$ ist eine Turing-Maschine, welche w^R akzeptiert, falls sie w akzeptiert$\}$, wobei w^R das zu w umgekehrte Wort ist. Zeigen Sie, dass T nicht entscheidbar ist.

Beweis: $P_{\text{halt}} \leq_m T$

Redunktionsfunktion $f : P_{\text{halt}} \to T$

- $f(\text{enc}(M, w))$ für $\text{enc}(M, w) \in P_{\text{halt}}$ konstruiert eine Turing-Maschine M_T
- M_T simuliert M auf w
- Da $w \in L(M)$, akzeptiert M w.
- M_T wird erweitert, dass M_T auch w_R akzeptiert.
- $\Rightarrow L(M_T) = \{w, w^R\}$
  $\Rightarrow \text{enc}(M_T) \in T$

Da P_{halt} nicht entscheidbar ist und $P_{\text{halt}} \leq_m T$ existiert, ist T nicht entscheidbar.

Aufgabe 3.4

Zeigen Sie, dass weder das Äquivalenzproblem $P_{\text{äquiv}}$ für Turing-Maschinen noch dessen Komplement $\neg P_{\text{äquiv}}$ semi-entscheidbar ist, wobei

- $P_{\text{äquiv}} := \{\text{enc}(M_1)\#\#\text{enc}(M_2) | L(M_1) = L(M_2)\}$,
- $\neg P_{\text{äquiv}} := \{\text{enc}(M_1)\#\#\text{enc}(M_2) | L(M_1) \neq L(M_2)\}$.

Zeigen Sie dazu, dass $P_{\text{halt}} \leq_m P_{\text{äquiv}}$ und $P_{\text{halt}} \leq_m \neg P_{\text{äquiv}}$ gilt. Weshalb zeigt dies die Aussage?

$P_{\text{halt}} \leq_m P_{\text{äquiv}}$

- $f(\text{enc}(M, w)) := \text{enc}(M)\#\#\text{enc}(M)$ für $\text{enc}(M, w) \in P_{\text{halt}}$
- $\text{enc}(M)\#\#\text{enc}(M) \in P_{\text{äquiv}}$, da $L(M) = L(M)$

$P_{\text{halt}} \leq_m \neg P_{\text{äquiv}}$

- $f(\text{enc}(M, w)) := \text{enc}(M)\#\#\text{enc}(M')$ für $\text{enc}(M, w) \in P_{\text{halt}}$
- M' ist eine Turing-Maschine, die ein $w' \notin L(M)$ akzeptiert, z.B.:

- $\Sigma_{M'} := \Sigma_M \cup \{\wedge\}$ mit $\wedge \notin \Sigma_M$
- M' erwartet als erstes Zeichen $\wedge$, löscht dieses, geht nach rechts
- M' simuliert M
- $\Rightarrow L(M') \neq L(M)$, weil jedes Wort $w \in L(M')$ mit $\wedge$ beginnt und damit $w \notin L(M)$

• enc(M)##enc$(M') \in \neg P_{\text{äquiv}}$, da $L(M) \neq L(M')$

P_{halt} ist nicht co-semi-entscheidbar
$\Rightarrow P_{\text{äquiv}}$ ist nicht co-semi-entscheidbar
$\Rightarrow \neg P_{\text{äquiv}}$ ist nicht semi-entscheidbar

$\neg P_{\text{halt}}$ ist nicht co-semi-entscheidbar
$\Rightarrow \neg P_{\text{äquiv}}$ ist nicht co-semi-entscheidbar
$\Rightarrow P_{\text{äquiv}}$ ist nicht semi-entscheidbar

4. Übungsblatt

Aufgabe H

Sei L eine unentscheidbare Sprache. Zeigen Sie:
a) L hat eine Teilmenge $T \subseteq L$, die entscheidbar ist.

$T = \emptyset$ mit Entscheider M_T: M_T hat nur einen Zustand, der ein nicht-akzeptierender Endzustand ist.

Sei L eine unentscheidbare Sprache. Zeigen Sie:
b) L hat eine Obermenge $O \supseteq L$, die entscheidbar ist.

$O = \Sigma^*$ mit Entschieder M_O: M_O hat nur einen Zustand, der ein akzeptierender Endzustand ist.

Sei L eine unentscheidbare Sprache. Zeigen Sie:
c) Es gibt jeweils nicht nur eine sondern unendlich viele entscheidbare Teilmengen bzw. Obermengen wie in (a) und (b).

Jede überabzählbare Menge $L(M)$ hat unendlich viele abzählbare Teilmengen $T \subseteq L(M)$. Wenn T abzählbar ist, dann ist T entscheidbar.

$O = (\Sigma \cup \Sigma_+)^*$ mit $\Sigma \cap \Sigma_+ = \emptyset$ (Das Alphabet von O wird um weitere Zeichen, die nicht in Σ liegen, erweitert.): Es existieren unendlich viele Σ_+.

Aufgabe 4.1

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
a) Falls P $\neq$ NP gilt, dann auch P $\cap$ NP $\neq \emptyset$

falsch, für eine deterministische Turing-Maschine in *P* lässt sich auch immer eine nicht-deterministische Turing-Maschine in *NP* konstruieren.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
b) Es gibt Probleme, die *NP*-hard, aber nicht *NP*-vollständig sind.

wahr, Gegenbeispiel P_{halt}: Jedes Problem in *NP* kann auf P_{halt} reduziert werden $\Rightarrow$ $P_{\text{halt}} \in$ NP-hard, aber $P_{\text{halt}} \notin$ NP (siehe Ackermann-Funktion, Busy-Beaver, ...) und damit nicht *NP*-complete.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
c) Polynomielle Reduzierbarkeit ist nicht transitiv.

falsch, Many-One-Reduktionen sind transitiv: aus $A \leq_m B$ mit $f_a : A \to B$ und $B \leq_m C$ mit $f_b : B \to C$ folgt $f_{a,b} : A \to C$ mit $f_{a,b}(x) := f_b(f_a(x))$. Eine polynomielle Reduktion ist eine Many-One-Reduktion mit polynomieller Laufzeit. Eine Verkettung von polynomiellen Funktion erzeugt auch wieder eine polynomielle Funktion.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
d) Ist $L_2 \in$ P und $L_1 \leq_p L_2$, dann ist auch $L_1 \in$ P.

wahr, für L_2 existiert eine deterministische Turing-Maschine mit polynomieller Laufzeit M_2 als Entscheider. Jedes Wort in w_1 in L_1 lässt sich entscheiden, indem man es mit einer Funktion f in ein Wort w_2 in L_2 überführt und anschließend der Entscheider M_2 simuliert. Sowohl f als auch M_2 laufen in polynomieller Zeit, also ist L_1 in *P* entscheidbar.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
e) Ist L_1 eine *NP*-vollständige Sprache und gilt $L_1 \leq_p L_2$, dann ist auch L_2 eine *NP*-vollständige Sprache.

wahr, da L_1 polynomiell entscheidbar ist und L_1 mit einer polynomiellen Funktion in L_2 überführt werden kann, muss auch der Entscheider für L_2 ebenfalls polynomiell lange laufen. Damit liegt L_2 in *NP*. Da L_1 aber auch in *NP*-hard liegt, muss also auch L_2 in *NP*-hard liegen. Da L_2 in *NP* und *NP*-hard liegt, liegt es in *NP*-complete.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
f) Ist L_2 eine *NP*-vollständige Sprache und gilt $L_1 \leq_p L2$, dann ist auch L_1 eine *NP*-vollständige Sprache.

falsch, $L_1 \in$ NP $\setminus$ NP-complete ist möglich.

Gegenbeispiel:

- $L_2 =$ SAT $\in$ NP-complete
- $L_1 =$ HORN-SAT $\in$ P
- $L_1 \subseteq L_2 \Rightarrow$ Reduktion mit $O(1)$

Aufgabe 4.2

Zeigen Sie, dass das Wortproblem deterministischer endlicher Automaten in LogSpace liegt:
ist $P_{\text{DFA}} := \{\text{enc}(A)\#\#\text{enc}(w) | A$ ist ein DFA, der w akzeptiert$\}$, dann gilt $P_{\text{DFA}} \in$ LogSpace

Der Speicherbedarf ist mit $O(\log|\text{enc}(A)\#\#\text{enc}(w)|)$ beschränkt. Der Automat ist wie folgt definiert $A = (Q, \Sigma, \delta, q_0, F)$. Um P_{DFA} zu simulieren, muss die aktuelle Position in w und der aktuelle Zustand von A gespeichert werden. Da der Automat A deterministisch ist, kann er sich immer nur in genau einem $q \in Q$ befinden. Also $i \in \{0, 1, \ldots, m-1\}$ mit $Q = \{q_0, q_1, \ldots, q_{m-1}\}$ und $j \in \{0, 1, \ldots, n-1\}$ mit $w = w_0 w_1 \ldots w_{n-1}$ zu speichern. Solange i und j nicht unär kodiert sind, ist $|i| \sim \log i$ mit $|i|$ als Länge der Kodierung der Zahl i.

Aufgabe 4.3

Wir betrachten das folgende Problem K: Gegeben sind zwei gerichtete Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sowie eine Zahl $k \in N$. Gefragt ist, ob es Teilmengen $V_1' \subseteq V_1$ und $V_2' \subseteq V_2$ gibt, so dass $|V_1'| = |V_2'| = k$ ist und es eine Bijektion $f : V_1' \to V_2'$ gibt, so dass gilt
$(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2$.
a) Zeigen Sie $K \in$ NP.

K ermittelt, ob G_1 und G_2 isomorphe Subgraphen mit k Knoten enthalten. $K \in$ NP genau dann, wenn eine Lösung V_1', V_2', f in polynomieller Zeit überprüft werden kann:

```
def check(V1_, V2_, f):
    assert len(V1_) == len(V2_)
    for u in V1_:
        for v in V1_:
            u_v_in_E1 = False
            for e in E1:
                if (u, v) == e:
                    u_v_in_E1 = True
            if u_v_in_E1:

                u2 = f(u)
                u2_in_V2 = False
                for x in V2_:
                    if u2 == x:
                        u2_in_V2 = True
                assert u2_in_V2

                v2 = f(v)
                v2_in_V2 = False
                for x in V2_:
                    if v2 == x:
                        v2_in_V2 = True
                assert v2_in_V2

                u2_v2_in_E2 = False
                for e in E2:
                    if (u2, v2) == e:
                        u2_v2_in_E2 = True
                assert u2_v2_in_E2
```

`check(V1_, V2_, f)` hat eine Laufzeit $O(|V_1'|^2\,(|E_1| + 2(|f| + |V_2'|) + |E_2|))$ mit f der Laufzeit von $O(|V_1'| \cdot |V_2'|)$. Damit ist `check(V1_, V2_, f)` polynomiell. Überprüfung der Lösung ist in polynomieller Zeit möglich, damit ist $K \in$ NP.

Wir betrachten das folgende Problem K: Gegeben sind zwei gerichtete Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sowie eine Zahl $k \in N$. Gefragt ist, ob es Teilmengen $V_1' \subseteq V_1$ und $V_2' \subseteq V_2$ gibt, so dass $|V_1'| = |V_2'| = k$ ist und es eine Bijektion $f : V_1' \to V_2'$ gibt, so dass gilt
$(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2$.
b) Zeigen Sie, dass K ein *NP*-schweres Problem ist. Zeigen Sie dafür, dass das Problem *CLIQUE* auf *K* in polynomieller Zeit reduzierbar ist.

CLIQUE sucht für einen ungerichteten Graph $G = (V, E)$ und $k \in \mathbb{N}$ einen Subgraph $G' = (V', E')$ mit $V' \subseteq V$, $|V'| = k$ und $E' := \{\{u, v\} | u, v \in V', u \neq v, \{u, v\} \in E\}$, so dass jeder Knoten des Subgraphs mit jedem anderen Knoten des Subgraphs verbunden ist.

CLIQUE $\leq_p K$:

- gegeben: ungerichteter Graph $G_{\text{CLIQUE}} = (V_{\text{CLIQUE}}, E_{\text{CLIQUE}})$ und $k_{\text{CLIQUE}} \in \mathbb{N}$
- G_{CLIQUE} ist ungerichtet $\Rightarrow E^{\rightarrow}_{\text{CLIQUE}} := \bigcup_{\{u,v\} \in E_{\text{CLIQUE}}} \{(u, v), (v, u)\}$ ($O(n)$)
- Führe K aus:
  - $G_1 := (V_{\text{CLIQUE}}, E^{\rightarrow}_{\text{CLIQUE}})$ ($O(1)$)
  - $G_2 := (V_{\text{CLIQUE}}, E_{\text{total}})$ mit $E_{\text{total}} = \{(u, v) | u \in V, v \in V\}$ ($O(n^2)$)
  - $k := k_{\text{CLIQUE}}$ ($O(1)$)
  - $K \in$ NP sucht also einen Subgraph, der sowohl in G_1 als auch G_2 enthalten ist. (Da $V_1 = V_2 = V$ ist $f(x) = x$ und spart die Bijektion der Knoten.) Da in G_2 alle Knoten miteinander verbunden sind, ist der Subgraph nur in G_1 enthalten, wenn dort auch alle Knoten des Subgraphs miteinander verbunden sind.
  - Ergebnis: $G_1' = (V_1', E_1')$ mit $E_1' \subseteq E^{\rightarrow}_{\text{CLIQUE}}$, jeder Knoten in V_1' ist mit jedem anderen Knoten in V_1' verbunden
- *CLIQUE* ist auch wieder ein ungerichteter Graph, alss muss E_1' noch umgewandelt werden, um die Lösung von *CLIQUE* zu erhalten: $\{\{u, v\} | (u, v) \in E^{\rightarrow}_{\text{CLIQUE}}\}$ ($O(n^2)$)

5. Übungsblatt

Aufgabe J

Zeigen Sie, dass *NP* unter Kleene-Stern abgeschlossen ist.

Es ist zu zeigen $\forall L \in$ NP : $L^* \in$ NP

gegeben: M entscheidet in $O(p(|w|))$, ob $w \in L$ oder $w \notin L$

gesucht: nicht-deterministische Turing Maschine M^*, die in $O(p^*(|w|))$ entscheidet, ob $w \in L^*$ oder $w \notin L^*$

Beweis:

- nicht-deterministische Turing-Maschine M^* rät Aufteilung von $w \in L^*$ in $w = w_1 w_2 \dots w_n$ mit $w_k \in L$
  in $O(p_w(|w|))$ überprüfbar
- M^* akzeptiert das leere Wort ε
- M^* simuliert M für jedes w_k in $O(p(|w_1|) + \dots + p(|w_n|))$
  - akzeptiert M jedes $w_k \Rightarrow M^*$ akzeptiert
  - sonst M^* lehnt ab

Aufgabe 5.1

Wir betrachten das folgende Problem K: Gegeben eine aussagenlogische Formel φ mit n Variablen, gibt es eine erfüllende Belegung von φ, bei der mindestens die Hälfte aller in φ vorkommenden Variablen mit *"true"* belegt sind?
a) Formalisieren Sie dieses Problem als Sprache und zeigen Sie, dass $K \in$ NP gilt.

$L_K := \{(\varphi, b) | b$ ist eine Belegung der aussagenlogische Formel bei der mindestens die Hälfte aller in φ vorko

nicht-deterministische Turing-Maschine:

- rät (φ, b)
- prüft in polynomieller Zeit, ob φ erfüllt ist
- prüft in polynomieller Zeit, ob mindestens die Hälfte aller in φ vorkommenden Variablen mit *"true"* belegt sind

Wir betrachten das folgende Problem K: Gegeben eine aussagenlogische Formel φ mit n Variablen, gibt es eine erfüllende Belegung von φ, bei der mindestens die Hälfte aller in φ vorkommenden Variablen mit *"true"* belegt sind?
b) Zeigen Sie, dass K ein NP-schweres Problem ist.

$P_{\text{SAT}} \leq_m K$ ($P_{\text{SAT}} \in$ NP-complete $\subseteq$ NP-hard), Reduktionsfunktion:

- Eingabe enc$(\varphi, b) \in P_{\text{SAT}}$ mit $b = (b_1, b_2, \ldots, b_{|b|})$
- erweitere φ um $|b|$ weitere Variablen, die alle mit *"true"* belegt sein müssen:
  - $\varphi_K := \varphi \wedge \bigwedge_{i=1}^{|b|} a_i = \varphi \wedge a_1 \wedge \ldots \wedge a_{|b|}$
  - $b_K := (b_1, b_2, \ldots, b_{|b|}, a_1 \mapsto \top, a_2 \mapsto \top, \ldots, a_{|b|} = \top)$
    $\Rightarrow$ die Hälfte aller Variablen in b_K ist mit $\top$ belegt
- $\Rightarrow$ enc$(\varphi_k, b_K) \in K$
- Da $K \in$ NP-hard und nicht $K \in$ NP-complete gefragt war, ist die Laufzeit der Reduktion egal, wird wohl aber doch polynomiell sein.

Aufgabe 5.2

Im folgenden *Solitaire*-Spiel haben wir ein Spielbrett der Größe $m \times m$ gegeben. Als Ausgangsposition liegt auf jeder der m^2 Positionen entweder ein blauer Stein, ein roter Stein, oder gar nichts. Das Spiel wird nun so gespielt, dass Steine vom Brett genommen werden bis in jeder Spalte nur noch Steine einer Farbe liegen, und in jeder Zeile mindestens ein Stein liegen bleibt. In diesem Fall ist das Spiel gewonnen. Es ist möglich, dass man ausgehend von einer Ausgangsposition das Spiel nicht gewinnen kann.
a) Formalisieren Sie das Problem, für eine gegebene Ausgangsposition im Solitaire-Spiel zu entscheiden, ob es möglich ist, das Spiel zu gewinnen, als ein Entscheidungsproblem *SOLITAIRE*.

$P_{\text{SOLITAIRE}} = \{\text{enc}(B) | B$ ist eine Ausgangsposition, die gewonnen werden kann$\}$

Im folgenden *Solitaire*-Spiel haben wir ein Spielbrett der Größe $m \times m$ gegeben. Als Ausgangsposition liegt auf jeder der m^2 Positionen entweder ein blauer Stein, ein roter Stein, oder gar nichts. Das Spiel wird nun so gespielt, dass Steine vom Brett genommen werden bis in jeder Spalte nur noch Steine einer Farbe liegen, und in jeder Zeile mindestens ein Stein liegen bleibt. In diesem Fall ist das Spiel gewonnen. Es ist möglich, dass man ausgehend von einer Ausgangsposition das Spiel nicht gewinnen kann.
b) Zeigen Sie, dass SOLITAIRE $\in$ NP gilt.

nicht-deterministische Turing-Maschine:

- Rät ausgehend vom Ausgangsbrett eine erfüllende End-Position.
- Falls keine Endposition existiert, dann akzeptiere *nicht*.
- Prüfe für jede der m Spalten, ob alle Steine die gleiche Farbe haben. $O(m^2)$
- Prüfe für jede der m Zeilen, ob diese mindestens einen Stein enthält. $O(m^2)$
- In $O(m^2)$ Schritten lässt sich aus der Endposition das Ausgangsbrett wiederherstellen, indem die Steine, so wie sie auf dem Ausgangsbrett lagen, wieder zur Endposition hinzugefügt werden ohne bereits liegende Steine zu ersetzen.

Für eine gegebene Endposition lässt sich also in $O(m^2)$ überprüfen, ob diese zu der Ausgangsposition passt $\Rightarrow P_{\text{SOLITAIR}} \in$ NP

Im folgenden *Solitaire*-Spiel haben wir ein Spielbrett der Größe $m \times m$ gegeben. Als Ausgangsposition liegt auf jeder der m^2 Positionen entweder ein blauer Stein, ein roter Stein, oder gar nichts. Das Spiel wird nun so gespielt, dass Steine vom Brett genommen werden bis in jeder Spalte nur noch Steine einer Farbe liegen, und in jeder Zeile mindestens ein Stein liegen bleibt. In diesem Fall ist das Spiel gewonnen. Es ist möglich, dass man ausgehend von einer Ausgangsposition das Spiel nicht gewinnen kann.
c) Zeigen Sie, dass *SOLITAIRE* ein *NP*-schweres Problem ist, indem Sie zeigen, dass *3-SAT* in polynomieller Zeit auf *SOLITAIRE* reduzierbar ist.

$P_{\text{3-SAT}} \leq_p P_{\text{SOLITAIRE}}$ **TODO**

Aufgabe 5.3

Sei Σ ein Alphabet und $A, B \subseteq \Sigma^*$. Wir sagen, dass A auf B in logarithmischen Platz reduzierbar ist, und schreiben $A \leq_\ell B$, falls es eine Many-One-Reduktion von A nach B gibt, die in logarithmischen Platz berechenbar ist. Begründen Sie: Gilt $A \leq_\ell B$ und $B \leq_\ell C$, dann gilt auch $A \leq_\ell C$.
Anmerkung: Bei dieser Aufgabe ist nicht nach einem vollständigen Beweis, sondern eher nach einer Beweisidee gefragt.

Problem: lediglich das Arbeitsband der Reduktion g bzw. M_g ist mit $O(\log n)$ beschränkt, die Ausgabe von g bzw. M_g kann aber $O(p(n))$ sein.

Musterlösung Wir zeigen, dass für zwei logspace-berechenbare Funktion $f, g\colon \Sigma^* \to \Sigma^*$ auch $f \circ g$ logspace-berechenbar ist. Seien dazu M_f und M_g Turing-Maschinen, die mit logarithmischer Platzbeschränkung die Funktionen f und g berechnen.

Eine erste Idee, eine Turing-Maschine M zu erhalten, die $f \circ g$ berechnet, ist, zuerst M_g auf der Eingabe aufzurufen, das Zwischenergebnis zu speichern, und dann M_f auf diesem Zwischenergebnis laufen zu lassen. **Diese Idee funktioniert jedoch nicht:** zwar benutzt M_g bei Eingabe $w \in \Sigma^*$ nur zusätzlich logarithmischen Platz zur Berechnung von $g(w)$. Dieses Ergebnis kann jedoch polynomiell groß sein in der Länge von w – und damit exponentiell in der Größe des zur Verfügung stehenden Platzes, der ja logarithmisch in der Größe von w beschränkt ist. Damit kann das Zwischenergebnis $g(w)$ nicht vollständig gespeichert werden und dieser Ansatz funktioniert nicht.

Wir können aber diese Idee so modifizieren, dass sie funktioniert! Dazu berechnen wir die Zeichen von $g(w)$ "on demand": wir verändern M_g so, dass sie nur das k-te Symbol von $g(w)$ berechnet. Dies kann erreicht werden, indem M_g mit einem weiteren Zähler p versehen wird, der um eins hochgezählt wird, wann immer M_g ein Symbol ausgeben möchte. Ist der Wert von p gleich k, gibt M_g das entsprechende Zeichen aus und hält an.

Um $f(g(w))$ in *LogSpace* zu berechnen, gehen wir nun wie folgt vor: wir simulieren die Berechnung von M_f. Wann immer diese Berechnung ein Symbol von $g(w)$ lesen möchte, simulieren wir M_g wie oben beschrieben. Beide Simulationen können in logspace durchgeführt werden, und damit kann auch $f(g(w))$ in logspace berechnet werden.

Diese Berechnung von $f(g(w))$ ist recht ineffizient, da Symbole von $g(w)$ möglicherweise mehrfach berechnet werden. Wir haben aber potentiell nicht genug Platz, um das gesamte Wort $g(w)$ zu speichern. Wir tauschen also "Platz gegen Zeit".

Aufgabe 5.4

PCP-k ist das folgende Entscheidungsproblem:
Gegeben: eine Zahl $k \in \mathbb{N}$ in unärer Kodierung und eine Instanz P des Postschen Korrespondenzproblems, d.h. eine endliche Folge von Wortpaaren

$$P = (x_1, x_2), \ldots, (x_n, y_n)$$

über einem Alphabet Σ, also $x_i, y_i \in \Sigma^+$ für $1 \leq i \leq n$.
Gefragt: Gibt es eine Lösung für P mit maximaler Länge k? Oder genauer: Gibt es eine Folge von Zahlen $i_1, \ldots, i_\ell$, so dass gilt:

$$x_{i_1} \ldots x_{i_\ell} = y_{i_1} \ldots y_{i_\ell}$$

wobei $0 < \ell \leq k$ ist und $i_j \in \{1, \ldots, n\}$ für alle $j = 1, \ldots, \ell$?
a) Zeigen Sie, dass *PCP-k* entscheidbar ist.

PCP ist semi-entscheidbar. Da zusätzlich mit k noch ein Abbruchkriterium gegeben ist, lassen sich in $O(n^k)$ alle möglichen Kombinationen durchprobieren. (Für jedes (x_{i_j}, y_{i_j}) mit $j \in \{1, \ldots, k\}$ kann aus n *Dominosteinen* des *PCP*-Problems gewählt werden.) Wenn also in den n^k möglichen Kombinationen keine Lösung gefunden wurde, kann eine Lösung des *PCP-k* ausgeschlossen werden. $\Rightarrow$ *PCP-k* ist entscheidbar.

PCP-k ist das folgende Entscheidungsproblem:
Gegeben: eine Zahl $k \in \mathbb{N}$ in unärer Kodierung und eine Instanz P des Postschen Korrespondenzproblems, d.h. eine endliche Folge von Wortpaaren

$$P = (x_1, x_2), \ldots, (x_n, y_n)$$

über einem Alphabet Σ, also $x_i, y_i \in \Sigma^+$ für $1 \leq i \leq n$.
Gefragt: Gibt es eine Lösung für P mit maximaler Länge k? Oder genauer: Gibt es eine Folge von Zahlen $i_1, \ldots, i_\ell$, so dass gilt:

$$x_{i_1} \ldots x_{i_\ell} = y_{i_1} \ldots y_{i_\ell}$$

wobei $0 < \ell \leq k$ ist und $i_j \in \{1, \ldots, n\}$ für alle $j = 1, \ldots, \ell$?
b) Zeigen Sie, dass *PCP-k* in *NP* liegt.

Eine Lösung $(i_1, \ldots, i_\ell)$ des allgemeinen *PCP* lässt sich in polynomieller Zeit verifizieren. Damit lässt sich auch *PCP-k* mit $\ell \leq k$ in polynomieller Zeit verifizieren. $\Rightarrow$ PCP-k $\in$ NP

6. Übungsblatt

- P $\subsetneq$ Exp, LogSpace $\subsetneq$ PSpace
- LogSpace $\subseteq$ P $\subseteq$ PSpace $\subsetneq$ Exp
- DTIME(f) $\subseteq$ DSPACE(f)
- DSPACE(f) $\subseteq$ DTIME($2^{O(f)}$)
- NTIME(f) $\subseteq$ NSPACE(f)
- NSPACE(f) $\subseteq$ DTIME($2^{O(f)}$)
- LogSpace $\subseteq$ NLogSpace, P $\subseteq$ NP, PSpace $\subseteq$ NPSpace, Exp $\subseteq$ NExp
- P $\subseteq$ PSpace, NP $\subseteq$ NPSpace
- NLogSpace $\subseteq$ P, NPSpace $\subseteq$ Exp
- PSpace $=$ NPSpace
- NLogSpace $\subsetneq$ NPSpace, NP $\subsetneq$ NExp
- P $\subseteq$ NP $\cap$ coNP
- "Eine Sprache ist *NP*-schwer, wenn jede Sprache in *NP* polynomiell darauf reduzierbar ist."
- NP-complete $=$ NP $\cap$ NP-hard
- $P_{\text{halt}} \in$ NP-hard, $P_{\text{halt}} \notin$ NP
- NL $=$ coNL
- *PSpace*-hard

Aufgabe 6.1

Zeigen Sie folgende Aussagen:
a) Ist $L_2 \in$ PSpace und gilt $L_1 \leq_p L_2$, so ist auch $L_1 \in$ PSpace.

- $L_1 \in L_2$ bedeutet, dass die Reduktion in polynomieller **Zeit** stattfindet.
- Die (polynomielle) Reduktion f_p mit $O_t(n^p)$ kann also höchstens $O_s(n^p)$ (polynomiellen) Speicher benutzen.
- $L_1 \in$ LogSpace:
  - Entscheider mit Reduktion: $M_2(f_p(w_1))$ mit $O(p \cdot \log n)$
  - **Widerspruch:** Wenn $L_1 \in$ LogSpace und $L_1 \leq_p L_2$, dann wäre $L_2 \in$ LogSpace. ($O(p \cdot \log n) \not\approx O(n^{d_2})$)

- $L_1 \in$ PSpace:
  - Entscheider mit Reduktion: $M_2(f_p(w_1))$ mit $O(n^{p \cdot d_2})$
  - $O(n^{d_1}) := O(n^{p \cdot d_2})$
- Mit einer polynomiellen Reduktion kann nur von *PSpace* nach *PSpace* transformiert werden und nicht von *LogSpace* in *PSpace*.

Zeigen Sie folgende Aussagen:
b) Ist L_1 ein *PSpace*-schweres Problem, und gilt $L_1 \leq_p L_2$, dann ist auch L_2 ein *PSpace*-schweres Problem.

- L_1 ist *PSpace*-hard: $\forall K \in$ PSpace : $K \leq_p L_1$ mit $f_K : K \to L_1$
- $L_1 \leq_p L_2$ mit $f_2 : L_1 \to L_2$
- $\Rightarrow$ mit $f_2(f_K(w))$ lässt sich jedes $w \in K$ zu L_2 reduzieren
- $\forall K \in$ PSpace : $K \leq_p L_2$ mit $f_K \circ f_2 : K \to L_2$
- Die Verkettung polynomieller Funktionen ist polynomiell.

Aufgabe 6.2

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
a) Jedes PSpace-schwere Problem ist *NP*-schwer.

falsch, es existieren Probleme $K \in$ PSpace, $K \notin$ NP für die keine Reduktion $K \leq_p K_{\text{NP}} \in$ NP-hard existiert. ($K \leq_p K_{\text{PSpace}} \in$ PSpace-hard muss existieren.)

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
b) Es gibt kein *NP*-schweres Problem, welches in *PSpace* liegt.

falsch, Gegenbeispiel: QBF $\in$ NP-hard, QBF $\in$ PSpace

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
c) Jedes *NP*-vollständige Problem liegt in *PSpace*.

wahr, NP-complete $\subseteq$ NP $\subseteq$ PSpace

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
d) Es gilt NP $=$ PSpace, wenn es ein *PSpace*-schweres Problem in *NP* gibt.

wahr, wäre ein *PSpace*-schweres Problem $P_{\text{NP}} \in$ NP, dann würde die Reduktion $P_{\text{PSpace}} \leq_p P_{\text{NP}}$ für $P_{\text{PSpace}} \in$ PSpace alle Probleme P_{PSpace} von *PSpace* in *NP* reduzieren.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
e) Wenn P $\neq$ NP gilt, dann gibt es kein *NP*-schweres Problem in *P*.

wahr, wäre ein *NP*-schweres Problem $P_{\text{NP-hard}} \in$ P, dann würde die Reduktion $P_{\text{NP}} \leq_p P_{\text{NP-hard}}$ für $P_{\text{NP}} \in$ NP alle Probleme P_{NP} von *NP* in *P* reduzieren.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
f) Sei L ein *PSpace*-vollständiges Problem. Dann gilt $L \in \text{P} \Leftrightarrow \text{P} = \text{PSpace}$.

wahr

- $L \in$ PSpace-complete und wenn $L \in$ P, dann würde die Reduktion $P_{\text{PSpace}} \leq_p L$ für $P_{\text{PSpace}} \in$ PSpace alle Probleme P_{PSpace} von *PSpace* in *P* reduzieren.
- Wenn P $=$ PSpace, dann wäre $L \in$ PSpace-complete $\subseteq$ PSpace $=$ P.

Aufgabe 6.3

Zeigen Sie, dass PSpace unter Komplement, Durchschnitt, Vereinigung, Konkatenation und Kleene-Stern abgeschlossen ist.

$\forall L_1, L_2 \in$ PSpace :

- $\neg L_1 \in$ PSpace
  - **TODO**
- $L_1 \cap L_2 \in$ PSpace
  - akzeptiere genau dann, wenn M_1 akzeptiert *und* M_2 akzeptiert
- $L_1 \cup L_2 \in$ PSpace
  - akzeptiere genau dann, wenn M_1 akzeptiert *oder* M_2 akzeptiert
- $\{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\} \in$ PSpace
  - PSpace $=$ NPSpace
  - rate $w_1 \in L_1$ und $w_2 \in L_2$
  - akzeptiere, wenn M_1 w_1 akzeptiert und M_2 w_2 akzeptiert
- $L_1^* \in$ PSpace
  - PSpace $=$ NPSpace
  - siehe Aufgabe J

Aufgabe 6.4

Wir betrachten das japanische Spiel Gomoku, welches von zwei Spielern X und O auf einem 19×19-Brett gespielt wird. Die Spieler setzen abwechselnd ihre Steine auf das Brett, und derjenige Spieler, der zuerst fünf Steine in einer Reihe (horizontal, vertikal, oder diagonal) gelegt hat, gewinnt. Spieler X beginnt.

Verallgemeinertes Gomoku wird statt auf einem Brett fester Größe auf einem beliebigen $n \times n$-Brett gespielt. Eine Position in diesem Spiel ist eine Belegung der Felder des Spielbretts mit Steinen der Spieler X und O, wie sie in einem wirklichen Spiel auftreten könnte. Sei

GM $:= \{\text{enc}(B) | B$ ist eine Position im verallgemeinerten Gomoku, in der X eine Gewinnstrategie hat$\}$

wobei enc(B) die zeilenweise Kodierung der Position B über einem festen Alphabet ist. Argumentieren Sie, warum GM $\in$ PSpace gilt.

TODO

Aufgabe 6.5

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
a) $\exists p_1.p_1$

wahr, $\exists$ verlangt, dass eine mögliche Belegung für p_1 existiert für die Formel p_1 wahr ist. Eine Möglichkeit ist $p_1 = \top$.

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
b) $\forall p_1.p_1$

falsch, $\forall$ verlangt, dass für alle möglichen Belegungen für p_1 die Formel p_1 wahr ist. Gegenbeispiel: $p_1 = \bot$

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
c) $\exists p_1.\bot$

falsch, es existiert kein p_1 für das $\bot$ wahr ist.

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
d) $\forall p_1.\exists p_2.p_2 \rightarrow p_1$

wahr, für jedes p_1 ist die Formel $\exists p_2.p_2 \rightarrow p_1$ mit $p_2 = \bot$ erfüllt.

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
e) $\forall p_1.\exists p_2.\forall p_3.(p_1 \vee p_2) \wedge p_3$

falsch, für alle p_1 existiert zwar $p_2 = \top$, damit $p_1 \vee p_2$ wahr ist, aber für $p_3 = \bot$ ist $(p_1 \vee p_2) \wedge p_3$ nicht wahr.

Welche der folgenden Quantifizierten Booleschen Formeln (QBFs) sind wahr? Begründen Sie Ihre Antwort.
f) $\forall p_1.\forall p_2.\exists p_3.\forall p_4.((p_1 \wedge p_2) \rightarrow p_4) \vee \neg p_3$

wahr, mit $p_3 = \bot$ ist $((p_1 \wedge p_2) \rightarrow p_4) \vee \neg p_3$ immer erfüllt.

7. Übungsblatt

Aufgabe M

Wir betrachten folgende Position im Tic-Tac-Toe:

X		
	O	
O		X

Angenommen, Spieler X ist am Zug. Beschreiben Sie eine Gewinnstrategie für X.

X		**X**
	O	
O		X

Aufgabe N

Zeigen Sie, dass für jedes *PSpace*-vollständige Problem L auch das Komplement L ein *PSpace*-vollständiges Problem ist.

Die Entscheider M_L für $L \in$ PSpace-complete $\subseteq$ PSpace darf zum Entscheiden eines Wortes w höchstens $p(|w|)$ Speicher nutzen. Der Entscheider $M_{\neg L}$ simuliert M_L auf und kehrt anschließend dessen Ergebnis um. Damit braucht $M_{\neg L}$ genauso viel Speicher wie M_L.

Aufgabe O

Zeigen Sie: ist P $=$ NP, dann sind alle Sprachen $L \in P \setminus \{\emptyset, \Sigma^*\}$ *NP*-vollständig.

TODO

Aufgabe 7.1

Zeigen Sie, dass das Wortproblem für linear beschränkte Turingmaschinen (*LBA*)

$$P_{\text{LBA}} := \{\text{enc}(M)\#\#\text{enc}(w) |\ M \text{ ist ein LBA, der } w \text{ akzeptiert}\}$$

ein *PSpace*-vollständiges Problem ist.
Zur Erinnerung (aus Formale Systeme): Ein linear beschränkte Turingmaschine (linear bounded automaton, *LBA*) ist eine nichtdeterministische Turingmaschine, die den Lese-/Schreibkopf nicht über das letzte Eingabezeichen hinaus bewegen kann. Versucht sie das, so bleibt der Kopf stattdessen an der letzten Bandstelle stehen.

Musterlösung Wir zeigen zuerst $P_{\text{LBA}} \in$ PSpace. Dafür muss zuerst geprüft werden, ob eine gegeben Eingabe von der Form enc(M)##enc(w) ist, wobei M ein *LBA* ist. Dabei ist leicht zu sehen, dass in polynomieller Zeit geprüft werden kann, ob enc(M) die Kodierung einer Turing-Maschine ist. Um zu sehen, dass M ein *LBA* ist, genügt es dann zu prüfen, ob M niemals ein $\llcorner$ überschreibt, und bei Lesen des $\llcorner$ dieses nicht überschreibt und den Lesekopf nach links bewegt. Auch dies kann in polynomieller Zeit geprüft werden.

Als nächstes muss ein Entscheider für P_{LBA} prüfen, ob $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ die Eingabe w akzeptiert. Dazu beobachten wir, dass die Maschine M höchstens $|Q| \cdot n \cdot |\Gamma|^n$ Konfigurationen durchläuft, bevor sie in eine Schleife gerät. Es genügt also, die Maschine M für höchstens $|Q| \cdot n \cdot |\Gamma|^n$ Schritte zu simulieren um zu entscheiden, ob M das Wort w erkennt. Die Simulation von M selbst benötigt nur linear Platz (da M ein *LBA* ist), und hinzu kommt noch Platz für einen Zähler, dessen Wert höchstens $|Q| \cdot n \cdot |\Gamma|^n$ ist und der deswegen höchstens Platz $\log |Q| + \log n + n \cdot \log |\Gamma|$ benötigt. Dies gesamte Simulation kann also in polynomiellen Platz ausgeführt werden.

Wir zeigen nun, dass P_{LBA} *PSpace*-hart ist. Sei dazu $L \in$ PSpace und sei M eine polynomiell platzbeschränkte Turing-Maschine, die L entscheidet. Sei p ein Polynom, welches den Platzverbrauch von M beschränkt. Ohne Einschränkung sei dabei $p(n) \geq n$ für alle $n \in \mathbb{N}$. Definiere dann pad(w) durch

$$\text{pad}(w) = w\bot^k,$$

wobei $\bot$ ein neues Symbol ist, so dass $|\text{pad}(w)| = p(|w|)$ gilt.

Sei nun M' die Turing-Maschine, welche M auf der Eingabe simuliert und dabei das $\bot$-Zeichen wie behandelt. Dies kann erreicht werden, indem in der Übergangsfunktion von M jedes Vorkommen von durch $\bot$ ersetzt wird, und zusätzlich noch die Transition hinzugefügt wird, dass M' bei Lesen von den Lesekopf nach links bewegt und wieder schreibt. Dann ist M' ein *LBA* und es gilt

$$M' \text{ akzeptiert } w \iff M \text{ akzeptiert } w.$$

Also ist

$$f(\text{enc}(w)) = \text{enc}(M')\#\#\text{enc}(\text{pad}(w))$$

eine polynomiell zeitbeschränkte Reduktion von L auf P_{LBA}. Da L beliebig gewählt war folgt, dass P_{LBA} *PSpace*-hart ist.

TODO

Aufgabe 7.2

Begründen Sie folgende Aussagen:
a) Ist P $=$ NP, dann ist NP $=$ coNP.

- $L \in$ NP $\Leftrightarrow \neg L \in$ coNP
- $L \in$ P $\Leftrightarrow \neg L \in$ P

- P $=$ NP
  $\Rightarrow L \in$ NP $\Leftrightarrow L \in$ P
  $\Leftrightarrow \neg L \in$ P $\Leftrightarrow \neg L \in$ NP und $\neg L \in$ coNP

Begründen Sie folgende Aussagen:
b) Ist P $\neq$ NP, dann gilt P $\neq$ coNP, L $\neq$ NP und P $\neq$ PSpace.

- Für jedes Problem $L \in$ NP liegt sein Komplement $\neg L \in$ coNP.
  Wäre P $=$ coNP, dann wäre $\neg L \in$ P, aber $L \in$ P $\Leftrightarrow \neg L \in$ P.
  Das würde bedeutet $L \in$ NP $\Rightarrow L \in$ P, da $\neg L \in$ P.
  $\Rightarrow$ NP $\subseteq$ P
  Mit P $\subseteq$ NP folgt P $=$ NP
  Widerspruch: P $=$ coNP kann also nicht gelten.
- L $\subseteq$ P und P $\neq$ NP $\Rightarrow$ L $\neq$ NP
- P $\subset$ NP $\subseteq$ PSpace $\Rightarrow$ P $\subset$ PSpace $\Rightarrow$ P $\neq$ PSpace

Aufgabe 7.3

Wir betrachten folgendes Scheduling-Problem: Gegeben sind Prüfungen $P_1, \ldots, P_k$ und Studierende $S_1, \ldots, S_\ell$, so dass jede Prüfung von einer bestimmten Menge von Studierenden abgelegt wird. Die Aufgabe ist, die Prüfungen so in Zeitslots zu legen, dass niemand zwei Prüfungen im selben Zeitslot ablegen muss. Formalisieren Sie die Frage, ob solch ein Prüfungsplan mit höchstens h Zeitslots möglich ist, als eine formale Sprache und zeigen Sie, dass diese *NP*-vollständig ist. Nutzen Sie dazu die Tatsache, dass Färbbarkeit von Graphen *NP*-vollständig ist.

$P := \{P_1, \ldots, P_k\}$
$S := \{S_1, \ldots, P_\ell\}$
$L := \{$enc$(P, S, h)|$ Es gibt eine Zuordnung von ℓ Studenten auf k Prüfungen in h Zeitslots ohne Überschneidungen$\}$

L ist *NP*-complete, wenn $P_{\text{Graph-Coloring}} \leq_p L$:

- ungerichteter Graph $G = (V, E)$ und die Anzahl der Farben k von $P_{\text{Graph-Coloring}}$
- $P := V$
- für jede Kante $\{v_{i_1}, v_{i_2}\} \in V$:
  - ein Schüler S_i hat die Prüfungen P_{i_1} und P_{i_2}
- $h := k$
- Löse das Scheduling-Problem.

Polynomielle Reduktion mit $O(|E|)$ existiert, damit ist $L \in$ NP-complete.

Aufgabe 7.4

Zeigen Sie, dass folgendes Problem unentscheidbar ist: Gegeben eine Turing-Maschine M und eine Zahl $k \in \mathbb{N}$, ist M eine $O(n^k)$-zeitbeschränkte Turing-Maschine?

TODO

8. Übungsblatt

Aufgabe P

Geben Sie für die Formel

$$F = \forall x.\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$$

wobei c_1, c_2 Konstanten sind, folgendes an:
a) die Menge aller Teilformeln;

1. $p(c_1, z)$
2. $q(x, c_2, z)$
3. $p(c_2, y)$
4. $q(x, c_2, z) \vee p(c_2, y)$
5. $p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y))$
6. $\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$
7. $\forall x.\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$

Geben Sie für die Formel

$$F = \forall x.\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$$

wobei c_1, c_2 Konstanten sind, folgendes an:
b) die Menge aller Terme;

- Variablen: x, y, z
- Konstanten: c_1, c_2
- Funktionen: $\emptyset$

Geben Sie für die Formel

$$F = \forall x.\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$$

wobei c_1, c_2 Konstanten sind, folgendes an:
c) die Menge aller Variablen, mit Unterscheidung freier und gebundener Variablen;

- gebundene Variablen: x, y
- freie Variablen: z

Geben Sie für die Formel

$$F = \forall x.\exists y.(p(c_1, z) \wedge (q(x, c_2, z) \vee p(c_2, y)))$$

wobei c_1, c_2 Konstanten sind, folgendes an:
d) ein Interpretation I und eine Zuweisung Z für I, so dass $I, Z \models F$.

- $I = (\Delta^I, \cdot^I)$
  - $\Delta^I = \{a, b, \ldots\}$
  - $c_1^I = a$
  - $c_2^I = b$
  - $p^I = \{(a, z) | z \text{ belieblig}\} \cup \{(b, y) | y \text{ beliebig}\}$
  - $q^I = \emptyset$

- $Z = \{z \mapsto \ldots\}$ Die Belegung von z ist beliebig.

$\forall x.\exists y.($	$p(c_1, z)$	$\wedge($	$q(x, c_2, z)$	$\vee$	$p(c_2, y)$	$))$
$\forall x.\exists y.($	$p(a, z)$	$\wedge($	$q(x, b, z)$	$\vee$	$p(b, y)$	$))$
$\forall x.\exists y.($	$\top$	$\wedge($	$\bot$	$\vee$	$\top$	$))$

Aufgabe Q

Zeigen Sie die folgenden Aussagen:
a) Es gilt $\{F\} \models G$ genau dann, wenn $F \to G$ allgemeingültig ist.

$\{F\} \models G$: jedes Modell von F ist auch ein Modell von G

$I \not\models F$	$I \not\models G$	erfüllt
$I \not\models F$	$I \models G$	erfüllt
$I \models F$	$I \not\models G$	nicht erfüllt
$I \models F$	$I \models G$	erfüllt

TODO

siehe 14. Vorlesung, Seite 9: Formeln interpretieren

Zeigen Sie die folgenden Aussagen:
b) Es gilt $\{F_1, \ldots, F_k\} \models G$ genau dann, wenn $\bigwedge_{i=1}^{k} F_i \to G$ allgemeingültig ist.

siehe Deduktionstheorem auf Wikipedia

TODO

Aufgabe R

Seien F, G Formeln und x eine Variable. Zeigen Sie, dass dann gilt $\exists x.(F \to G) \equiv \forall x.F \to \exists x.G$.

- $\forall x.F \to \exists x.G$
- $\neg\forall x.F \vee \exists x.G$
- $\exists x.\neg F \vee \exists x.G$
- Es handelt sich um zwei verschiedene x, für $\vee$ ist aber egal, ob eine der Bedingungen vom eigenen x erfüllt werden oder ein allgemeines x eine der Bedingungen erfüllt, deshalb können die Existenzquantoren hier zusammengefasst werden.
- $\exists x.(\neg F \vee G)$
- $\exists x.(F \to G)$

Aufgabe 8.1

Gegeben sei ein Universum aus verschiedenen Personen und Speisen und ein Prädikat mag, so dass mag(x, y) ausdrückt "x mag y".
a) "Übersetzen" Sie folgende prädikatenlogische Formeln in natürlichsprachlich formulierte Aussagen:

1. $\forall x.\text{mag}(x, \text{Eiscreme})$.
2. $\forall x.\exists y.\text{mag}(x, y)$.
3. $\forall x.\forall y.\text{mag}(x, y) \vee \neg\text{mag}(x, y)$.

1. Jeder mag Eiscreme.
2. Jeder mag irgendwas.
3. **TODO**

Gegeben sei ein Universum aus verschiedenen Personen und Speisen und ein Prädikat mag, so dass mag(x, y) ausdrückt "x mag y".
a) "Übersetzen" Sie folgende natürlichsprachlich formulierte Aussagen in prädikatenlogische Formeln:

1. Tom mag keinen Fisch.
2. Jeder, der Pizza mag, mag auch Spaghetti.
3. Es gibt niemanden, der alles mag.

1. $\neg\text{mag}(\text{Tom}, \text{Fisch})$
2. $\forall x.\text{mag}(x, \text{Pizza}) \rightarrow \text{mag}(x, \text{Spaghetti})$
3. $\forall x.\exists y.\neg\text{mag}(x, y)$

Aufgabe 8.2

Welche der angegebenen Strukturen sind Modelle der folgenden Formel?

$$\forall x.p(x, x) \wedge \forall x, y.((p(x, y) \wedge p(y, x)) \rightarrow x \approx y) \wedge \forall x, y, z.((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))$$

a) I_1 mit Grundmenge $\mathbb{N}$ und $p^{I_1} = \{(m, n) | m < n\}$;

I_1 ist kein Modell, da $\forall x.p(x, x)$ ($p(a, b) := (a < b)$) nicht erfüllt ist.

Welche der angegebenen Strukturen sind Modelle der folgenden Formel?

$$\forall x.p(x, x) \wedge \forall x, y.((p(x, y) \wedge p(y, x)) \rightarrow x \approx y) \wedge \forall x, y, z.((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))$$

b) I_2 mit Grundmenge $\mathbb{N}$ und $p^{I_2} = \{(n, n + 1) | n \in \mathbb{N}\}$;

I_2 ist kein Modell, da $\forall x.p(x, x)$ ($p(a, b) := (a + 1 = b)$) nicht erfüllt ist.

Welche der angegebenen Strukturen sind Modelle der folgenden Formel?

$$\forall x.p(x, x) \wedge \forall x, y.((p(x, y) \wedge p(y, x)) \rightarrow x \approx y) \wedge \forall x, y, z.((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))$$

c) I_3 mit Grundmenge $\mathbb{N}$ und $p^{I_3} = \{(m, n) | m$ teilt $n\}$;

I_3 ist ein Modell.

Welche der angegebenen Strukturen sind Modelle der folgenden Formel?

$$\forall x.p(x,x) \wedge \forall x,y.((p(x,y) \wedge p(y,x)) \rightarrow x \approx y) \wedge \forall x,y,z.((p(x,y) \wedge p(y,z)) \rightarrow p(x,z))$$

d) I_4 mit Grundmenge Σ^* für ein Alphabet Σ und $p^{I_4} = \{(x,y)|x \text{ ist Präfix von } y\}$;

I_4 ist ein Modell.

Welche der angegebenen Strukturen sind Modelle der folgenden Formel?

$$\forall x.p(x,x) \wedge \forall x,y.((p(x,y) \wedge p(y,x)) \rightarrow x \approx y) \wedge \forall x,y,z.((p(x,y) \wedge p(y,z)) \rightarrow p(x,z))$$

e) I_5 mit Grundmenge $\mathcal{P}(M)$ für eine Menge M und $p^{I_5} = \{(X,Y)|X \subseteq Y\}$;

I_5 ist ein Modell.

Aufgabe 8.3

a) Geben Sie je eine erfüllbare Formel in Prädikatenlogik mit Gleichheit an, so dass alle Modelle

1. höchstens drei,
2. mindestens drei,
3. genau drei

Elemente in der Grundmenge besitzen.

1. **TODO**
2. **TODO**
3. **TODO**

b) Geben Sie je eine erfüllbare Formel in Prädikatenlogik mit Gleichheit an, so dass das zweistellige Prädikatensymbol p in jedem Modell als der Graph einer

1. injektiven Funktion,
2. surjektiven Funktion,
3. bijektiven Funktion

interpretiert wird. (Der Graph einer Funktion $f : A \rightarrow B$ ist die Relation $\{(x,y) \in A \times B | f(x) = y\}$.)

1. $\forall x_1 \in A, x_2 \in A, y \in B.((p(x_1,y) \wedge p(x_2,y)) \rightarrow x_1 \approx x_2$
2. $\forall y \in B.\exists x \in A.p(x,y)$
3. $(\forall x_1 \in A, x_2 \in A, y \in B.((p(x_1,y) \wedge p(x_2,y)) \rightarrow x_1 \approx x_2) \wedge (\forall y \in B.\exists x \in A.p(x,y))$

Aufgabe 8.4

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
a) Sind Γ und Γ' Mengen von prädikatenlogischen Formeln, dann folgt aus $\Gamma \subseteq \Gamma'$ und $\Gamma \models F$ auch $\Gamma' \models F$.

wahr **TODO**

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **b)** Jede aussagenlogische Formel ist eine prädikatenlogische Formel.

wahr, Aussagenlogik ist Prädikatenlogik ohne Prädikate, Variablen oder Quantoren

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **c)** Eine prädikatenlogische Formel F ist genau dann allgemeingültig, wenn $\neg F$ unerfüllbar ist.

wahr **TODO**

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **d)** Es gilt $\{\forall x, y.(p(x, y) \rightarrow p(y, x)), \forall x, y, z.((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))\} \models \forall x.p(x, x)$

- $\forall x, y.(p(x, y) \rightarrow p(y, x))$ bedeutet, dass p kommuntativ ist
  $(x, y) \in p^I \Leftrightarrow (y, x) \in p^I$
- $\forall x, y, z.((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))$ bedeutet, dass p transitiv ist
  $(x, y) \in p^I$ und $(y, z) \in p^I \Rightarrow (x, z) \in p^I$
- $\forall x.p(x, x)$ bedeutet, dass p reflexiv ist
  $(x, x) \in p^I$
- Für alle $(x, y) \in p^I$ muss gelten, dass auch $(y, x) \in p^I$ und durch die Transitivität von $(x, y) \in p^I$ und $(y, x) \in p^I$ folgt $(x, x) \in p^I$.

wahr

9. Übungsblatt

Aufgabe S

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **a)** Jede Formel in Pränexform ist in Skolemform.

falsch, Formeln in Pränexform können einen $\exists$-Quantor enthalten.

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **b)** Jede Formel in Skolemform ist in Pränexform.

wahr, alle $\forall$-Quantoren stehen am Anfang der Formel.

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **c)** Jede Formel ist äquivalent zu einer bereinigten Formel.

wahr, gebundene Variablen können umbenannt werden.

> Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
> **d)** Jede Formel ist äquivalent zu einer bereinigten Formel in Pränexform.

wahr, bei Umwandlung in Pränexform finden nur äquivalenzerhaltende Operation statt.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
e) Jede Formel ist äquivalent zu einer bereinigten Formel in Skolemform.

falsch, eine Formel in Skolemform ist nur erfüllbarkeits-äquivalent.

Aufgabe T

Formalisieren Sie die folgenden Aussagen in Prädikatenlogik:
a) Jeder Drache ist glücklich, wenn alle seine Drachen-Kinder fliegen können.

$\forall d.((\text{drache}(d) \land \forall k.(\text{kind}(k,d) \rightarrow \text{fliegen}(k))) \rightarrow \text{glücklich}(d))$

Formalisieren Sie die folgenden Aussagen in Prädikatenlogik:
b) Grüne Drachen können fliegen.

$\forall d.((\text{drache}(d) \land \text{grün}(d)) \rightarrow \text{fliegen}(d))$

Formalisieren Sie die folgenden Aussagen in Prädikatenlogik:
c) Ein Drache ist grün, wenn er Kind mindestens eines grünen Drachen ist.

$\forall d.((\text{drache}(d) \land (\exists e.(\text{drache}(e) \land (\text{grün}(e) \land \text{kind}(d,e))))) \rightarrow \text{grün}(d))$

Formalisieren Sie die folgenden Aussagen in Prädikatenlogik:
d) Alle grünen Drachen sind glücklich.
Zeigen Sie, dass die letzte Aussage aus den ersten drei folgt.

$\forall d.((\text{drache}(d) \land \text{grün}(d)) \rightarrow \text{glücklich}(d)$

Aufgabe 9.1

Formalisieren Sie Bertrand Russells Barbier-Paradoxon

> Der Barbier rasiert genau diejenigen Personen, die sich nicht selbst rasieren.

als eine prädikatenlogische Formel und zeigen Sie, dass diese unerfüllbar ist.

Nehmen Sie dafür an, dass der Barbier durch eine Konstante b repräsentiert wird und dass für die Relation * *rasiert y* durch ein zweistelliges Prädikatensybol r ausgedrückt wird.

- $\forall x.\neg r(x,x) \leftrightarrow r(b,x)$
- Für $x := b$:
  - $\neg r(b,b) \leftrightarrow r(b,b)$
  - r kann also nicht definiert werden, um die Formel zu erfüllen.

Aufgabe 9.2

Bestimmen Sie zu jeder der folgenden Formeln eine äquivalente bereinigte Formel in Pränexform.
a) $\forall x.(p(x,x) \leftrightarrow \neg\exists y.q(x,y))$

1. $\forall x.(p(x,x) \leftrightarrow \neg\exists y.q(x,y))$
2. Äquivalenz auflösen: $\forall x.((p(x,x) \rightarrow \neg\exists y.q(x,y)) \wedge (\neg\exists y.q(x,y) \rightarrow p(x,x)))$
3. Bereinigung: $\forall x.((p(x,x) \rightarrow \neg\exists y_1.q(x,y_1)) \wedge (\neg\exists y_2.q(x,y_2) \rightarrow p(x,x)))$
4. Implikationen auflösen: $\forall x.((\neg p(x,x) \vee \neg\exists y_1.q(x,y_1)) \wedge (\exists y_2.q(x,y_2) \vee p(x,x)))$
5. $\neg\exists x.A \Leftrightarrow \forall x.\neg A$:
   $\forall x.((\neg p(x,x) \vee \forall y_1.\neg q(x,y_1)) \wedge (\exists y_2.q(x,y_2) \vee p(x,x)))$
6. Quantoren nach vorne ziehen: $\forall x.\forall y_1.\exists y_2.((\neg p(x,x) \vee \neg q(x,y_1)) \wedge (q(x,y_2) \vee p(x,x)))$

Bestimmen Sie zu jeder der folgenden Formeln eine äquivalente bereinigte Formel in Pränexform.
b) $\forall x.p(f(x,x)) \vee (q(x,z) \rightarrow \exists x.p(g(x,y,z)))$

1. $\forall x.p(f(x,x)) \vee (q(x,z) \rightarrow \exists x.p(g(x,y,z)))$
2. Bereinigung: $\forall x_1.p(f(x_1,x_1)) \vee (q(x_1,z) \rightarrow \exists x_2.p(g(x_2,y,z)))$
3. Implikation auflösen: $\forall x_1.p(f(x_1,x_1)) \vee (\neg q(x_1,z) \vee \exists x_2.p(g(x_2,y,z)))$
4. Quantoren nach vorne ziehen: $\forall x_1.\exists x_2.p(f(x_1,x_1)) \vee (\neg q(x_1,z) \vee p(g(x_2,y,z)))$

Bestimmen Sie zu jeder der folgenden Formeln eine äquivalente bereinigte Formel in Pränexform.
c) $\forall x.p(x) \wedge (\forall y.\exists x.q(x,g(y)) \rightarrow \exists y.(r(f(y)) \vee \neg q(y,x)))$

1. $\forall x.p(x) \wedge (\forall y.\exists x.q(x,g(y)) \rightarrow \exists y.(r(f(y)) \vee \neg q(y,x)))$
2. Bereinigung: $\forall x_1.p(x_1) \wedge (\forall y_1.\exists x_2.q(x_2,g(y_1)) \rightarrow \exists y_2.(r(f(y_2)) \vee \neg q(y_2,x_1)))$
3. Implikationen auflösen: $\forall x_1.p(x_1) \wedge (\neg\forall y_1.\exists x_2.q(x_2,g(y_1)) \vee \exists y_2.(r(f(y_2)) \vee \neg q(y_2,x_1)))$
4. $\neg\forall y_1.\exists x_2.A \Leftrightarrow \exists y_1.\forall x_2.\neg A$:
   $\forall x_1.p(x_1) \wedge (\exists y_1.\forall x_2.\neg q(x_2,g(y_1)) \vee \exists y_2.(r(f(y_2)) \vee \neg q(y_2,x_1)))$
5. Quantoren nach vorne ziehen: $\forall x_1.\exists y_1.\forall x_2.\exists y_2.(p(x_1) \wedge (\neg q(x_2,g(y_1)) \vee (r(f(y_2)) \vee \neg q(y_2,x_1))))$

Aufgabe 9.3

Bestimmen Sie zu jeder der folgenden Formeln eine erfüllbarkeitsäquivalente bereinigte Formel in Skolemform.
a) $p(x) \vee \exists x.q(x,x) \vee \forall x.p(f(x))$

1. $p(x) \vee \exists x.q(x,x) \vee \forall x.p(f(x))$
2. $p(x) \vee \exists x_1.q(x_1,x_1) \vee \forall x_2.p(f(x_2))$
3. $\exists x_1.\forall x_2.(p(x) \vee q(x_1,x_1) \vee p(f(x_2)))$
4. $x_1 \Rightarrow a_1$:
   $\forall x_2.(p(x) \vee q(a_1,a_1) \vee p(f(x_2)))$

Bestimmen Sie zu jeder der folgenden Formeln eine erfüllbarkeitsäquivalente bereinigte Formel in Skolemform.
b) $\forall x.\exists y.q(f(x),g(y)) \wedge \forall x.(p(x,y,y) \vee q(h(y),x))$

1. $\forall x.\exists y.q(f(x),g(y)) \wedge \forall x.(p(x,y,y) \vee q(h(y),x))$
2. $\forall x_1.\exists y_1.q(f(x_1),g(y_1)) \wedge \forall x_2.(p(x_2,y,y) \vee q(h(y),x_2))$
3. $\forall x_1.\exists y_1.\forall x_2.(q(f(x_1),g(y_1)) \wedge (p(x_2,y,y) \vee q(h(y),x_2)))$

4. $y_1 \Rightarrow f_1(x_1)$:
   $\forall x_1.\forall x_2.(q(f(x_1), g(f_1(x_1))) \wedge (p(x_2, y, y) \vee q(h(y), x_2)))$

Bestimmen Sie zu jeder der folgenden Formeln eine erfüllbarkeitsäquivalente bereinigte Formel in Skolemform.
c) $\forall x.\forall x.(p(x) \leftrightarrow q(x, x)) \vee \exists x.\forall y.(q(x, g(y, z)) \wedge \exists z.q(z, z))$

1. $\forall x.\forall x.(p(x) \leftrightarrow q(x, x)) \vee \exists x.\forall y.(q(x, g(y, z)) \wedge \exists z.q(z, z))$
2. $\forall x.\forall x.((p(x) \rightarrow q(x, x)) \wedge (q(x, x) \rightarrow p(x))) \vee \exists x.\forall y.(q(x, g(y, z)) \wedge \exists z.q(z, z))$
3. $\forall x_1.\forall x_2.((p(x_2) \rightarrow q(x_2, x_2)) \wedge (q(x_2, x_2) \rightarrow p(x_2))) \vee \exists x_3.\forall y_1.(q(x_3, g(y_1, z)) \wedge \exists z_1.q(z_1, z_1))$
   ($y \Rightarrow y_1$ ist nicht notwendig, schadet aber auch nicht.)
4. $\forall x_1.\forall x_2.((\neg p(x_2) \vee q(x_2, x_2)) \wedge (\neg q(x_2, x_2) \vee p(x_2))) \vee \exists x_3.\forall y_1.(q(x_3, g(y_1, z)) \wedge \exists z_1.q(z_1, z_1))$
5. $\forall x_1.\forall x_2.\exists x_3.\forall y_1.\exists z_1.(((\neg p(x_2) \vee q(x_2, x_2)) \wedge (\neg q(x_2, x_2) \vee p(x_2))) \vee (q(x_3, g(y_1, z)) \wedge q(z_1, z_1)))$
6. $x_3 \Rightarrow f_{x_3}(x_1, x_2)$:
   $\forall x_1.\forall x_2.\forall y_1.\exists z_1.(((\neg p(x_2) \vee q(x_2, x_2)) \wedge (\neg q(x_2, x_2) \vee p(x_2))) \vee (q(f_{x_3}(x_1, x_2), g(y_1, z)) \wedge q(z_1, z_1)))$
7. $z_1 \Rightarrow f_{z_1}(x_1, x_2, y_1)$:
   $\forall x_1.\forall x_2.\forall y_1.(((\neg p(x_2) \vee q(x_2, x_2)) \wedge (\neg q(x_2, x_2) \vee p(x_2))) \vee (q(f_{x_3}(x_1, x_2), g(y_1, z)) \wedge q(f_{z_1}(x_1, x_2, y_1), f_{z_1}(x_1, x_2, y_1))))$

Aufgabe 9.4

Zeigen Sie, dass Allgemeingültigkeit von Formeln der Prädikatenlogik erster Stufe in Skolemform entscheidbar ist.

Musterlösung Es sei F eine quantorenfreie Formel mit Variablen $x_1, \ldots, x_n$. Dann gilt $\forall x_1, \ldots, x_n.F$ ist allgemeingültig $\Leftrightarrow$ $\exists x_1, \ldots, x_n.\neg F$ ist unerfüllbar $\Leftrightarrow \neg F[x_1/a_1, \ldots, x_n/a_n]$ ist unerfüllbar (Skolemisierung mit Konstanten $a_1, \ldots, a_n$).

Es ist also $\forall x_1, \ldots, x_n.F$ allgemiengültig genau dann, wenn $\neg F[x_1/a_1, ..., x_n/a_n]$ unerfüllbar ist. Letzteres ist aber essentiell eine aussagenlogische Formel, und deren Erfüllbarkeit ist entscheidbar.

10. Übungsblatt

Aufgabe U

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
a) Zwei prädikatenlogische Formeln F und G sind äquivalent, wenn die Formel $F \leftrightarrow G$ allgemeingültig ist.

wahr, **TODO**

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
b) Jede erfüllbare Formel der Prädikatenlogik erster Stufe hat ein endliches Modell.

falsch, Δ^I kann unendlich sein.

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
c) Jede erfüllbare Formel der Prädikatenlogik erster Stufe hat ein abzählbares Modell.

wahr, siehe Satz von Löwenheim-Skolem

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
d) Jede Skolemformel hat höchstens eine Herbrand-Interpretation.

TODO

Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort.
e) Jede Skolemformel hat mindestens ein Herbrand-Modell.

falsch, lediglich jede erfüllbare Skolemform hat ein Herbrand-Modell.

Aufgabe V

Zeigen Sie, dass man das Resolutionsverfahren der Prädikatenlogik erster Stufe auch zum Nachweis von semantischen Konsequenzen nutzen kann, indem Sie die Äquivalenz der folgenden Aussagen nachweisen:
a) $\Gamma \models F$.

TODO

Zeigen Sie, dass man das Resolutionsverfahren der Prädikatenlogik erster Stufe auch zum Nachweis von semantischen Konsequenzen nutzen kann, indem Sie die Äquivalenz der folgenden Aussagen nachweisen:
b) $\Gamma \cup \neg F$ ist unerfüllbar.

TODO

Zeigen Sie, dass man das Resolutionsverfahren der Prädikatenlogik erster Stufe auch zum Nachweis von semantischen Konsequenzen nutzen kann, indem Sie die Äquivalenz der folgenden Aussagen nachweisen:
c) $\bigwedge \Gamma \rightarrow F$ ist allgemeingültig.
Hierbei sei $\bigwedge \Gamma = \gamma_1 \wedge \ldots \wedge \gamma_n$ für $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$.

TODO

Zeigen Sie, dass man das Resolutionsverfahren der Prädikatenlogik erster Stufe auch zum Nachweis von semantischen Konsequenzen nutzen kann, indem Sie die Äquivalenz der folgenden Aussagen nachweisen:
d) $\bigwedge \Gamma \wedge \neg F$ ist unerfüllbar.
Hierbei sei $\bigwedge \Gamma = \gamma_1 \wedge \ldots \wedge \gamma_n$ für $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$.

TODO

Aufgabe 10.1

Bestimmen Sie jeweils einen allgemeinsten Unifikator für die folgenden Unifikationsprobleme, oder begründen Sie, warum kein allgemeinster Unifikator existiert. Verwenden Sie hierfür den Algorithmus aus der Vorlesung. Dabei sind x, y Variablen und a, b Konstanten.
a) $\{f(x) \doteq g(x,y), y \doteq f(a)\}$

1. $\{f(x) \doteq g(x,y), y \doteq f(a)\}$
2. Eliminierung: $\{f(x) \doteq g(x, f(a)), y \doteq f(a)\}$
3. Löschen, Zerlegung, Orietierung oder Eliminierung nicht möglich

Bestimmen Sie jeweils einen allgemeinsten Unifikator für die folgenden Unifikationsprobleme, oder begründen Sie, warum kein allgemeinster Unifikator existiert. Verwenden Sie hierfür den Algorithmus aus der Vorlesung. Dabei sind x, y Variablen und a, b Konstanten.
b) $\{f(g(x,y)) \doteq f(g(a,h(b)))\}$

1. $\{f(g(x,y)) \doteq f(g(a,h(b)))\}$
2. Zerlegung: $\{g(x,y) \doteq g(a,h(b))\}$
3. Zerlegung: $\{x \doteq a, y \doteq h(b)\}$
4. $\{x \mapsto a, y \mapsto h(b)\}$

Bestimmen Sie jeweils einen allgemeinsten Unifikator für die folgenden Unifikationsprobleme, oder begründen Sie, warum kein allgemeinster Unifikator existiert. Verwenden Sie hierfür den Algorithmus aus der Vorlesung. Dabei sind x, y Variablen und a, b Konstanten.
c) $\{f(x,y) \doteq x, y \doteq g(x)\}$

1. $\{f(x,y) \doteq x, y \doteq g(x)\}$
2. Eliminierung: $\{f(x, g(x)) \doteq x, y \doteq g(x)\}$
3. Orientierung: $\{x \doteq f(x, g(x)), y \doteq g(x)\}$
4. Löschen, Zerlegung, Orietierung oder Eliminierung nicht möglich

Bestimmen Sie jeweils einen allgemeinsten Unifikator für die folgenden Unifikationsprobleme, oder begründen Sie, warum kein allgemeinster Unifikator existiert. Verwenden Sie hierfür den Algorithmus aus der Vorlesung. Dabei sind x, y Variablen und a, b Konstanten.
d) $\{f(g(x),y) \doteq f(g(x),a), g(x) \doteq g(h(a))\}$

1. $\{f(g(x),y) \doteq f(g(x),a), g(x) \doteq g(h(a))\}$
2. Zerlegung: $\{g(x) \doteq g(x), y \doteq a, g(x) \doteq g(h(a))\}$
3. Löschen: $\{g(x) \doteq g(x), g(x) \doteq g(h(a))\}$
4. Zerlegung: $\{y \doteq a, x \doteq h(a)\}$
5. $\{x \mapsto h(a), y \mapsto a\}$

Aufgabe 10.2

Sei p ein k-stelliges Prädikatssymbol und seien $s_1, \ldots, s_k$, $t_1, \ldots, t_k$ Terme. Ferner sei θ eine Substitution. Hierbei bezeichne $\exists[F]$ und $\forall[F]$ jeweils den Existenz- bzw. Allabschluss über alle in F syntaktisch vorkommenden Variablen. Welche der folgenden Aussagen sind richtig? Begründen Sie jeweils Ihre Antwort.

a) Falls $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$ unifizierbar sind, so ist folgende Formel der Prädikatenlogik mit Gleichheit allgemeingültig:

$$\exists[(s_1 \approx t_1) \wedge \ldots \wedge (s_k \approx t_k)]$$

b) Falls $\exists[(s_1 \approx t_1) \wedge \ldots \wedge (s_k \approx t_k)]$ erfüllbar ist, so sind $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$ unifizierbar.

beide wahr, $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$ sind unifizierbar genau dann, wenn es eine Substitution gibt, sodass $p(t_1, \ldots, t_k) \equiv p(s_1, \ldots, s_k)$. Wenn $\exists[(s_1 \approx t_1) \wedge \ldots \wedge (s_k \approx t_k)]$ erfüllt ist, gibt es eine Substitution bei der $s_i \approx t_i$ und damit muss auch $p(t_1, \ldots, t_k) \equiv p(s_1, \ldots, s_k)$ gelten.

Sei p ein k-stelliges Prädikatssymbol und seien $s_1, \ldots, s_k$, $t_1, \ldots, t_k$ Terme. Ferner sei θ eine Substitution. Hierbei bezeichne $\exists[F]$ und $\forall[F]$ jeweils den Existenz- bzw. Allabschluss über alle in F syntaktisch vorkommenden Variablen. Welche der folgenden Aussagen sind richtig? Begründen Sie jeweils Ihre Antwort.
c) Ist θ ein Unifikator für $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$, so ist folgende Formel der Prädikatenlogik mit Gleichheit allgemeingültig:

$$\forall[(s_1\theta \approx t_1\theta) \wedge \ldots \wedge (s_k\theta \approx t_k\theta)]$$

d) Ist $\forall[(s_1\theta \approx t_1\theta) \wedge \ldots \wedge (s_k\theta \approx t_k\theta)]$ allgemeingültig, so ist θ ein Unifikator für $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$.

TODO

beide falsch, wenn $\theta = \{s_1 \mapsto c, t_1 \mapsto c, s_2 \mapsto a_2, t_2 \mapsto b_2, \ldots\}$ mit $a_i \neq b_i$ und $p = \{(c, a_2, a_3, \ldots), (c, b_2, b_3, \ldots)\}$ dann wäre zwar $p(t_1\theta, \ldots, t_k\theta) \equiv p(s_1\theta, \ldots, s_k\theta)$ aber $s_i\theta \neq t_i\theta$ für $i \geq 2$.

Sei p ein k-stelliges Prädikatssymbol und seien $s_1, \ldots, s_k$, $t_1, \ldots, t_k$ Terme. Ferner sei θ eine Substitution. Hierbei bezeichne $\exists[F]$ und $\forall[F]$ jeweils den Existenz- bzw. Allabschluss über alle in F syntaktisch vorkommenden Variablen. Welche der folgenden Aussagen sind richtig? Begründen Sie jeweils Ihre Antwort.
e) Ist $\theta = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$ ein Unifikator für $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$, so ist folgende Formel der Prädikatenlogik mit Gleichheit allgemeingültig:

$$\forall[((x_1 \approx u_1) \wedge \ldots \wedge (x_k \approx u_k)) \rightarrow ((s_1 \approx t_1) \wedge \ldots \wedge (s_k \approx t_k))]$$

TODO

Sei p ein k-stelliges Prädikatssymbol und seien $s_1, \ldots, s_k$, $t_1, \ldots, t_k$ Terme. Ferner sei θ eine Substitution. Hierbei bezeichne $\exists[F]$ und $\forall[F]$ jeweils den Existenz- bzw. Allabschluss über alle in F syntaktisch vorkommenden Variablen. Welche der folgenden Aussagen sind richtig? Begründen Sie jeweils Ihre Antwort.
f) Ist $\theta = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$ ein Unifikator für $p(t_1, \ldots, t_k)$ und $p(s_1, \ldots, s_k)$, so ist folgende Formel der Prädikatenlogik mit Gleichheit allgemeingültig:

$$\forall[((s_1 \approx t_1) \wedge \ldots \wedge (s_k \approx t_k)) \rightarrow ((x_1 \approx u_1) \wedge \ldots \wedge (x_k \approx u_k))]$$

TODO

Aufgabe 10.3

Zeigen Sie mittels prädikatenlogischer Resolution folgende Aussagen:
a) Die Aussage "Der Professor ist glücklich, wenn alle seine Studenten Logik mögen" hat als Folgerung "Der Professor ist glücklich, wenn er keine Studenten hat".

$P(x)$ … x ist ein Professor
$S(x, y)$ … x ist Student von y
$L(x)$ … x mag Logik
$G(x)$ … x ist glücklich

1. "Der Professor ist glücklich, wenn alle seine Studenten Logik mögen"
    1. $\forall p.((P(p) \land \forall s.(S(s,p) \to L(s))) \to G(p))$
    2. $\forall p.(\neg(P(p) \land \forall s.(S(s,p) \to L(s))) \lor G(p))$
    3. $\forall p.((\neg P(p) \lor \neg\forall s.(\neg S(s,p) \lor L(s))) \lor G(p))$
    4. $\forall p.((\neg P(p) \lor \exists s.\neg(\neg S(s,p) \lor L(s))) \lor G(p))$
    5. $\forall p.((\neg P(p) \lor \exists s.(S(s,p) \land \neg L(s))) \lor G(p))$
    6. $\forall p.\exists s.((\neg P(p) \lor (S(s,p) \land \neg L(s))) \lor G(p))$
    7. $\forall p.\exists s.((\neg P(p) \lor S(s,p) \lor G(p)) \land (\neg P(p) \lor \neg L(s) \lor G(p)))$
    8. $\forall p.((\neg P(p) \lor S(f_s(p),p) \lor G(p)) \land ())$
    9. $\{\{\neg P(p), S(f_s(p),p), G(p)\}, \{\neg P(p), \neg L(f_s(p)), G(p)\}\}$
2. "Der Professor ist glücklich, wenn er keine Studenten hat"
    1. $\forall p.((P(p) \land \neg\exists s.S(s,p)) \to G(p))$
    2. $\forall p.(\neg(P(p) \land \neg\exists s.S(s,p)) \lor G(p))$
    3. $\forall p.(\neg P(p) \lor \exists s.S(s,p) \lor G(p))$
    4. $\forall p.\exists s.(\neg P(p) \lor S(s,p) \lor G(p))$
    5. $\forall p.(\neg P(p) \lor S(f_s(p),p) \lor G(p))$
    6. $\{\{\neg P(p), S(f_s(p),p), G(p)\}\}$

Formel 2.6 ist in 1.7 enthalten, bzw. $\{\{\neg P(p), S(f_s(p),p), G(p)\}\} \subseteq \{\{\neg P(p), S(f_s(p),p), G(p)\}, \{\neg P(p), \neg L(f_s(p)$

Zeigen Sie mittels prädikatenlogischer Resolution folgende Aussagen:
b) In Aufgabe T von Übungsblatt 9 folgt die letzte Aussage 4 aus den ersten drei 1-3:

1. Jeder Drache ist glücklich, wenn alle seine Drachen-Kinder fliegen können.
2. Grüne Drachen können fliegen.
3. Ein Drache ist grün, wenn er Kind mindestens eines grünen Drachen ist.
4. Alle grünen Drachen sind glücklich.

Zur Vereinfachung darf hier angenommen werden, dass alle Individuen Drachen sind.

$K(x, y)$ … x ist ein Kind von y
$F(x)$ … x kann fliegen
$H(x)$ … x ist glücklich
$G(x)$ … x ist grün

1. $\forall d.(\forall k.(K(k,d) \to F(k)) \to H(d))$
    1. $\forall d.(\forall k.(K(k,d) \to F(k)) \to H(d))$
    2. $\forall d.\neg(\forall k.(\neg K(k,d) \lor F(k)) \lor H(d))$
    3. $\forall d.(\exists k.\neg(\neg K(k,d) \lor F(k)) \lor H(d))$
    4. $\forall d.(\exists k.(K(k,d) \land \neg F(k)) \lor H(d))$

5. $\forall d.((K(f_k(d), d) \wedge \neg F(f_k(d))) \vee H(d))$
6. $\forall d.((K(f_k(d), d) \vee H(d)) \wedge (\neg F(f_k(d)) \vee H(d)))$
7. $\{\{K(f_k(d), d), H(d)\}, \{\neg F(f_k(d)), H(d)\}\}$

2. $\forall d.(G(d) \rightarrow F(d))$

    1. $\forall d.(\neg G(d) \vee F(d))$
    2. $\{\{\neg G(d), F(d)\}\}$

3. $\forall d.(\exists e.(G(e) \wedge K(d, e)) \rightarrow G(d))$

    1. $\forall d.(\neg \exists e.(G(e) \wedge K(d, e)) \vee G(d))$
    2. $\forall d.(\forall e.\neg (G(e) \wedge K(d, e)) \vee G(d))$
    3. $\forall d.(\forall e.(\neg G(e) \vee \neg K(d, e)) \vee G(d))$
    4. $\forall d.\forall e.(\neg G(e) \vee \neg K(d, e) \vee G(d))$
    5. $\{\{\neg G(e), \neg K(d, e), G(d)\}\}$

Klauseln:

1. $\{K(f_k(d), d), H(d)\}$
2. $\{\neg F(f_k(d)), H(d)\}$
3. $\{\neg G(d), F(d)\}$
4. $\{\neg G(e), \neg K(d, e), G(d)\}$
5. 2 + 3: $\{\neg G(f_k(d)), H(d)\}$
6. 4 + 5: $\{\neg G(d), \neg K(f_k(d), d), H(d)\}$
7. 1 + 6: $\{\neg G(d), H(d)\}$

$\forall d.(G(d) \rightarrow H(d))$ Alle grünen Drachen sind glücklich.

Aufgabe 10.4

Gegeben sind die folgenden Formeln in Skolemform.

$$F = \forall x, y, z.p(x, f(y), g(z, x))$$

$$G = \forall x, y.(p(a, f(a, x, y)) \vee q(b))$$

wobei a und b Konstanten sind.
a) Geben Sie die zugehörigen Herbrand-Universen Δ_F und Δ_G an.

F: keine Konstanten; Funktionen f, g:

1. $\Delta_0 = \{a\}$
2. $\Delta_1 = \Delta_0 \cup \{f(a), g(a, a)\}$
3. $\Delta_2 = \Delta_1 \cup \{$
    - $f(f(a)), f(g(a, a)),$
    - $g(a, f(a)), g(a, g(a, a)),$
    - $g(f(a), a), g(f(a), f(a)), g(f(a), g(a, a)),$
    - $g(g(a, a), a), g(g(a, a), f(a)), g(g(a, a), g(a, a))$

    $\}$
4. ...
5. $\Delta_F = \{f(x), f(f(x)), g(x, y), g(x, f(y)), g(f(x), y), g(f(x), f(y)), \ldots\}$

G Konstanten a, b; Funktionen f:

1. $\Delta_0 = \{a, b\}$
2. $\Delta_1 = \Delta_0 \cup \{f(a,a,a), f(a,a,b), f(a,b,a), f(a,b,b), f(b,a,a), f(b,a,b), f(b,b,a), f(b,b,b)\}$
3. $\Delta_2 = \Delta_1 \cup \{$
   - $f(a,a,f(a,a,a)), f(a,a,f(a,a,b)), f(a,a,f(a,b,a)), \ldots$
   - $f(a,f(a,a,a),a), f(a,f(a,a,b),a), f(a,f(a,b,a),a), \ldots$
   - $f(a,f(a,a,a),f(a,a,a)), \ldots$

   $\}$
4. $\ldots$
5. $\Delta_G = \{a, b, f(a,a,a), f(a,a,b), f(a,b,a), f(a,b,b), f(b,a,a), f(b,a,b), f(b,b,a), f(b,b,b), \ldots\}$

Gegeben sind die folgenden Formeln in Skolemform.

$$F = \forall x, y, z.p(x, f(y), g(z, x))$$

$$G = \forall x, y.(p(a, f(a, x, y)) \vee q(b))$$

wobei a und b Konstanten sind.
b) Geben Sie je ein Herbrand-Modell an oder begründen Sie, warum kein solches existiert.

- $F = \forall x, y, z.p(x, f(y), g(z, x))$
  - $\Delta^I = \Delta_F$
  - $f^I(t) = t$ mit $t \in \Delta_F$
  - $g^I(t_1, t_2) = t_1$ mit $t_1, t_2 \in \Delta_F$
  - $p^I = \{(t, f(t), g(t, t)) | t \in \Delta_F\}$
- $G = \forall x, y.(p(a, f(a, x, y)) \vee q(b))$
  - $\Delta^I = \Delta_G$
  - $f(t_1, t_2, t_3) = t_1$ mit $t_1, t_2, t_3 \in \Delta_G$
  - $p^I = \Delta_G \times \Delta_G \times \Delta_G$
  - $q^I = \Delta_G$

11. Übungsblatt

Aufgabe 11.1

Gegeben sei die folgende Kino-Datenbank bestehend aus den drei Tabellen Filme, Spielstätten und Kinoprogramm. Geben Sie die nachfolgenden Anfragen jeweils in Form einer prädikatenlogische Formel an:
a) Wer ist der Regisseur von Der Hobbit 1?

$Q = \exists r.(\exists s.\text{Filme}(H_1, r, s))$
$Z = \{H_1 \mapsto \text{"Der Hobbit 1"}\}$

Gegeben sei die folgende Kino-Datenbank bestehend aus den drei Tabellen Filme, Spielstätten und Kinoprogramm. Geben Sie die nachfolgenden Anfragen jeweils in Form einer prädikatenlogische Formel an:
b) Welche Kinos spielen Der Hobbit 1?

$Q = \exists k.(\exists z.\text{Kinoprogramm}(k, H_1, z))$
$Z = \{H_1 \mapsto \text{"Der Hobbit 1"}\}$

Gegeben sei die folgende Kino-Datenbank bestehend aus den drei Tabellen Filme, Spielstätten und Kinoprogramm. Geben Sie die nachfolgenden Anfragen jeweils in Form einer prädikatenlogische Formel an:
c) Gibt es ein Kino welches einen Film von Christopher Nolan zeigt?

$Q = \exists k.(\exists f.(\exists s.\text{Filme}(f, C_{\text{Nolan}}, s) \land \exists z.\text{Kinoprogramm}(k, f, z)))$
$Z = \{C_{\text{Nolan}} \mapsto \text{"Christopher Nolan"}\}$

Gegeben sei die folgende Kino-Datenbank bestehend aus den drei Tabellen Filme, Spielstätten und Kinoprogramm. Geben Sie die nachfolgenden Anfragen jeweils in Form einer prädikatenlogische Formel an:
d) Welche Paare von Schauspielern spielen gemeinsam in mindestens einem Film?

$Q = \exists a, b.(\exists f, r.(\text{Filme}(f, r, a) \land \text{Filme}(f, r, b)) \land (a \not\approx b))$

Gegeben sei die folgende Kino-Datenbank bestehend aus den drei Tabellen Filme, Spielstätten und Kinoprogramm. Geben Sie die nachfolgenden Anfragen jeweils in Form einer prädikatenlogische Formel an:
e) Welche Paare von Schauspielern spielen gemeinsam in genau einem Film?

1. $Q = \exists a, b.((a \not\approx b) \land \exists f_1, r_1.(\text{Filme}(f_1, r_1, a) \land \text{Filme}(f_1, r_1, b) \land \neg\exists f_2, r_2.((f_2 \not\approx f_1) \land \text{Filme}(f_2, r_2, a) \land \text{Filme}(f_2, r_2, b))))$
2. $Q = \exists a, b.((a \not\approx b) \land \exists f_1, r_1.(\text{Filme}(f_1, r_1, a) \land \text{Filme}(f_1, r_1, b) \land \forall f_2, r_2.\neg((f_2 \not\approx f_1) \land \text{Filme}(f_2, r_2, a) \land \text{Filme}(f_2, r_2, b))))$
3. $Q = \exists a, b.((a \not\approx b) \land \exists f_1, r_1.(\text{Filme}(f_1, r_1, a) \land \text{Filme}(f_1, r_1, b) \land \forall f_2, r_2.((f_2 \approx f_1) \lor \neg\text{Filme}(f_2, r_2, a) \lor \neg\text{Filme}(f_2, r_2, b))))$

Aufgabe 11.2

Gegeben sind die folgenden Formeln in Skolemform.

$$F = \forall x, y, z.p(x, f(y), g(z, x))$$

$$G = \forall x, y.(p(a, f(a, x, y)) \lor q(b))$$

wobei a und b Konstanten sind.
Geben Sie die Herbrand-Expansion HE(F) und HE(G) an.

- HE(F)
  - $\Delta_F = \{a, f(a), g(a, a), f(f(a)), \ldots\}$
    - $\Delta_0 = \{a\}$
    - $\Delta_1 = \Delta_0 \cup \{f(a), g(a, a)\}$
    - $\ldots$
  - HE(F) = {
    - $p(a, a, g(a, a))$,
    - $p(f(a), a, g(a, a))$,
    - $p(a, f(a), g(a, a))$,
    - $p(f(a), f(a), g(a, a))$,
    - $p(a, a, g(f(a), a))$,
    - $\ldots$

}

- HE(G)
  - $\Delta_G = \{a, b, f(a,a,a), f(b,a,a), f(a,b,a), f(b,b,a), f(a,a,b), f(f(a,a,a),a,a), f(a,f(a,a,a),a), f(b,f(a$
    - $\Delta_0 = \{a, b\}$
    - $\Delta_1 = \Delta_0 \cup \{f(a,a,a), f(b,a,a), f(a,b,a), f(b,b,a), f(a,a,b), \ldots\}$
    - $\Delta_2 = \Delta_1 \cup \{f(f(a,a,a),a,a), f(a,f(a,a,a),a), f(b,f(a,a,a),a), \ldots\}$
    - $\ldots$
  - HE(G) = {
    - $p(a, f(a,a,a)) \vee q(b)$,
    - $p(a, f(a,b,a)) \vee q(b)$,
    - $p(a, f(a, f(a,a,a), a)) \vee q(b)$,
    - $p(a, f(a,a,b)) \vee q(b)$,
    - $p(a, f(a,b,b)) \vee q(b)$,
    - $p(a, f(a, f(a,a,a), b)) \vee q(b)$,
    - $p(a, f(a,a, f(a,a,a))) \vee q(b)$,
    - $\ldots$

    }

Aufgabe 11.3

Führen Sie für die folgenden Formeln eine Resolution durch, um zu zeigen:
a) Für $F_1 = \forall x.(\text{weg}(x) \rightarrow \text{führtNachRom}(x)) \wedge \forall x.(\text{autobahn}(x) \rightarrow \text{weg}(x)) \wedge \text{autobahn}(\text{a4})$ und $G_1 = \text{führtNachRom}(\text{a4})$ gilt $F_1 \models G_1$.

1. $\{\neg\text{weg}(x_1), \text{führtNachRom}(x_1)\}$
2. $\{\neg\text{autobahn}(x_2), \text{weg}(x_2)\}$
3. $\{\text{autobahn}(\text{a4})\}$
4. 3 + 2: $\{\text{weg}(\text{a4})\}$
5. 1 + 4: $\{\text{führtNachRom}(\text{a4})\}$

Führen Sie für die folgenden Formeln eine Resolution durch, um zu zeigen:
b) Für $F_2 = \forall x.(\text{känguru}(x) \rightarrow \exists y.\text{hatMutter}(x,y)) \wedge \forall z.\forall w.(\text{hatMutter}(z,w) \rightarrow \text{liebt}(w,z))$ und $G_2 = \forall x.(\text{känguru}(x) \rightarrow (\exists y.\text{liebt}(y,x)))$ gilt $F_2 \models G_2$.

$F_2 =$

1. $\forall x.(\text{känguru}(x) \rightarrow \exists y.\text{hatMutter}(x,y)) \wedge \forall z.\forall w.(\text{hatMutter}(z,w) \rightarrow \text{liebt}(w,z))$
2. $\forall x.(\neg\text{känguru}(x) \vee \exists y.\text{hatMutter}(x,y)) \wedge \forall z.\forall w.(\neg\text{hatMutter}(z,w) \vee \text{liebt}(w,z))$
3. $\forall x.\exists y.\forall z.\forall w.((\neg\text{känguru}(x) \vee \text{hatMutter}(x,y)) \wedge (\neg\text{hatMutter}(z,w) \vee \text{liebt}(w,z)))$
4. $\forall x.\forall z.\forall w.((\neg\text{känguru}(x) \vee \text{hatMutter}(x, f_{y_F}(x))) \wedge (\neg\text{hatMutter}(z,w) \vee \text{liebt}(w,z)))$

$G_2 =$

1. $\forall x.(\text{känguru}(x) \rightarrow (\exists y.\text{liebt}(y,x)))$
2. $\forall x.(\neg\text{känguru}(x) \vee (\exists y.\text{liebt}(y,x)))$
3. $\forall x.\exists y.(\neg\text{känguru}(x) \vee (\text{liebt}(y,x)))$
4. $\forall x.(\neg\text{känguru}(x) \vee \text{liebt}(f_{y_G}(x), x))$

Resolution:

1. $\{\neg\text{känguru}(x), \text{hatMutter}(x, f_{y_F}(x))\}$

2. $\{\neg\text{hatMutter}(z, w), \text{liebt}(w, z)\}$
3. 1 + 2: $\{\neg\text{känguru}(x), \text{liebt}(f_{y_F}(x), x)\}$
4. Mit $f_{y_F}(x) = f_{y_G}(x)$ gilt $F_2 \models G_2$.

12. Übungsblatt

Aufgabe 12.1

Wir betrachten das folgende Datalog-Programm P:

$T(x) \leftarrow e(x)$
$T(x) \leftarrow a(x, y) \wedge T(y) \wedge b(x, z) \wedge T(z)$

$e(1), e(2), e(6)$
$a(3, 1), a(4, 3), a(5, 3), a(7, 5)$
$b(3, 2), b(5, 3), b(7, 6)$

a) Geben Sie einen Ableitungsbaum für $T(5)$ an.

- $T(5) \leftarrow a(5, 3) \wedge T(3) \wedge b(5, 3) \wedge T(3)$
  - $T(3) \leftarrow a(3, 1) \wedge T(1) \wedge b(3, 2) \wedge T(2)$
    - $T(2) \leftarrow e(2)$
    - $T(1) \leftarrow e(1)$

Wir betrachten das folgende Datalog-Programm P:

$T(x) \leftarrow e(x)$
$T(x) \leftarrow a(x, y) \wedge T(y) \wedge b(x, z) \wedge T(z)$

$e(1), e(2), e(6)$
$a(3, 1), a(4, 3), a(5, 3), a(7, 5)$
$b(3, 2), b(5, 3), b(7, 6)$

b) Berechnen Sie die Mengen T_P^0, T_P^1, T_P^2, ... An welchem Punkt wird der Grenzwert erreicht?

T_P^{i+1} ... welche $T(x)$ lassen sich mit den bereits berechneten $T(y) \in T_P^i$ berechnen?

- $T_P^0 = \emptyset$
- $T_P^1 = T_P^0 \cup \{T(1), T(2), T(6)\}$
- $T_P^2 = T_P^1 \cup \{T(3)\}$
  - $T(3) \leftarrow a(3, 1) \wedge T(1) \wedge b(3, 2) \wedge T(2)$
- $T_P^3 = T_P^2 \cup \{T(5)\}$
  - $T(5) \leftarrow a(5, 3) \wedge T(3) \wedge b(5, 3) \wedge T(3)$
- $T_P^4 = T_P^3 \cup \{T(7)\}$
  - $T(7) \leftarrow a(7, 5) \wedge T(5) \wedge b(7, 6) \wedge T(6)$
- es gibt keine weiteren Fakten für $T(k)$ mit $k > 7$
- $T_P^n = \{T(1), T(2), T(3), T(5), T(6), T(7)\}$

Aufgabe 12.2

Sei $E \subseteq V \times V$ die Kantenrelation eines gerichteten Graphen $G = (V, E)$ mit (endlicher) Knotenmenge V. Weiter sei e das dazugehörige Datalog-Prädikat mit $e(v_1, v_2) \Leftrightarrow (v_1, v_2) \in E$ für alle $v_1, v_2 \in V$.

Formalisieren Sie die nachfolgenden Probleme als Datalog-Programme, d.h. geben Sie Regeln an, die für einen gegebenen Graphen eine Lösung für das jeweilige Problem liefern.

a) Nicht-Terminiertheit: Alle Paare von Knoten, die über eine Kante miteinander verbunden sind, so dass der Knoten mit der eingehenden Kante wieder verlassen werden kann.

$T(x, y) \leftarrow e(x, y) \wedge e(y, z)$

Sei $E \subseteq V \times V$ die Kantenrelation eines gerichteten Graphen $G = (V, E)$ mit (endlicher) Knotenmenge V. Weiter sei e das dazugehörige Datalog-Prädikat mit $e(v_1, v_2) \Leftrightarrow (v_1, v_2) \in E$ für alle $v_1, v_2 \in V$.

Formalisieren Sie die nachfolgenden Probleme als Datalog-Programme, d.h. geben Sie Regeln an, die für einen gegebenen Graphen eine Lösung für das jeweilige Problem liefern.

b) Erreichbare Knoten: Alle Knoten, die von einem festen Startknoten s erreichbar sind.

$T(s)$
$T(x) \leftarrow e(s, x)$
$T(x) \leftarrow e(y, x) \wedge T(y)$

Sei $E \subseteq V \times V$ die Kantenrelation eines gerichteten Graphen $G = (V, E)$ mit (endlicher) Knotenmenge V. Weiter sei e das dazugehörige Datalog-Prädikat mit $e(v_1, v_2) \Leftrightarrow (v_1, v_2) \in E$ für alle $v_1, v_2 \in V$.

Formalisieren Sie die nachfolgenden Probleme als Datalog-Programme, d.h. geben Sie Regeln an, die für einen gegebenen Graphen eine Lösung für das jeweilige Problem liefern.

c) Alternative Wege: Alle Paare von Knoten, die sowohl über einen Weg der Länge eins als auch über einen Weg der Länge zwei verbunden sind.

$T(x, y) \leftarrow e(x, y) \wedge e(x, z) \wedge e(z, y)$