Algorithmisches Beweisen LAB

Einschub: Testen der Algorithmen

Luc Spachmann

FSU Jena

16.05.2022

- Bisher: Ausführung der Algorithmen auf einzelnen Instanzen
- Heute: Allgemeine Testumgebung
- Idee:
  - Direkter Vergleich verschiedener Algorithmen
  - Performance auf theoretisch schweren Formeln
  - Performance auf zufälligen Formeln

Details

- Eingabe: Algorithmus als Black Box
- Eingabe: Art des Tests
- Mehrfacher Aufruf des Algorithmus
- Parsen des Outputs
- Ausgabe: Graphische Darstellung der Laufzeiten/Entscheidungen/etc.
- Abbruch, falls der Algorithmus zu lange dauert

Pidgeonhole Principle

- PHP_n^{n+1} hat Variablen $x_{i,j}, i \in [n+1], j \in [n]$
- $x_{i,j}$ gibt an, ob Taube i in Loch j sitzt
- Klauseln:

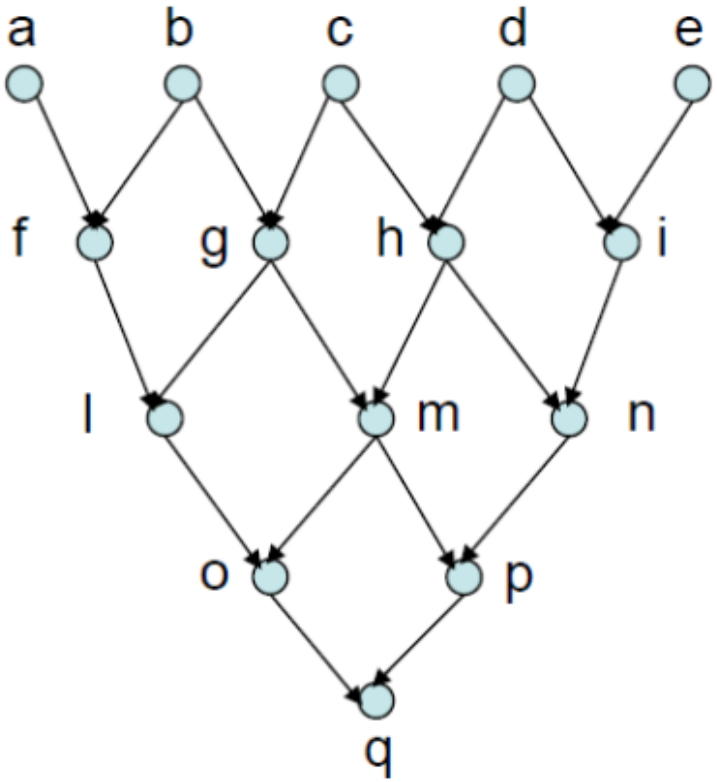
$$\bigvee_{j \in [n]} x_{i,j} \quad \text{für alle } i \in [n+1]$$

und

$$\neg x_{i_1,j} \vee \neg x_{i_2,j}$$

für alle paarweise verschiedenen $i_1, i_2 \in [n+1]$ und alle $j \in [n]$.

Pebbling Formeln auf pyramidalen Graphen



Pebbling Formeln auf pyramidalen Graphen

- Betrachten nur n mit $\frac{k(k+1)}{2}$
- Sonst: Größte Zahl $< n$ dieser Form
- Variablen: $x_{v,c}$ für alle $v \in V, c \in \{B, W\}$
- Klauseln:

$x_{v,B} \vee x_{v,W}$	für alle Quellknoten v (a-e in Bsp)
$\neg x_{u,a} \vee \neg x_{w,b} \vee x_{v,B} \vee x_{v,W}$	für alle $v \in V, a, b \in \{B, W\}$, u, w sind Vorgänger von v
$\neg x_{v,B}, \neg x_{v,W}$	für Zielknoten v q in Bsp

Aufgabe

- Implementierung der Testumgebung
- Algorithmen: Hornformeln, DPLL, CDCL (später)
- Formeln: Zufällig, PHP, Pebbling, ggf. noch andere
- Vergleich der verschiedenen Metriken (Dauer, Speicher, Entscheidungen, Propagations, etc.)
- Beispiel: Ausgabe und Plot der Laufzeiten von DPLL für PHP_n^{n+1} für $1 < n < 10$