Exkurs: NP-Vollständigkeit

Entscheidung der Erfüllbarkeit ist NP-vollständig

Ein (Entscheidungs-)Problem L liegt in der Klasse $\boldsymbol{P}$, wenn es durch einen deterministischen Algorithmus in polynomieller Zeit lösbar ist (= in $O(n^k)$ für „Größe“ n des Problems und Konstante k).

L liegt in $\boldsymbol{NP}$, wenn es durch einen nicht-deterministischen Algorithmus in polynomieller Zeit lösbar ist
(= durch einen nichtdet. Algo., der an allen Entscheidungspunkten richtig „rät“,
= durch einen det. Algo. erst in exponenzieller Zeit).

Vermutung(!): $P \neq NP$

L heißt **NP-vollständig**, gdw. $L \in \boldsymbol{NP}$, und L ist **NP-schwer** (*NP hard*), d.h. jedes (andere) Problem in $\boldsymbol{NP}$ kann mit polynomiellem Aufwand auf L reduziert werden.
(Reduktion = Umrepräsentation, z.B. formuliere TSP als Inferenzproblem)

Hilfreiche/Fundamentale Sätze

Deduktionssatz

Für beliebige A und B: $A \models B$ gdw. $(A \Rightarrow B)$ allgemeingültig ist

Beweis direkt aus Definitionen der Folgerung und der Implikation

Satz vom Widerspruchsbeweis

Für beliebige A und B: $A \models B$ gdw. $(A \land \neg B)$ inkonsistent ist

Beweis: Umformung des Deduktionsatzes

Monotoniesatz

Für beliebige $\mathcal{A}$, A und B: wenn $\mathcal{A} \models A$ dann $\mathcal{A} \cup \{B\} \models A$

Beweis über Betrachtung der Modelle

Endlichkeitssatz

Eine (möglicherweise unendliche) Formelmenge ist inkonsistent, gdw. sie eine endliche inkonsistente Teilmenge hat.

Beweis Richtung „→“ nichttrivial! (s. z.B. Schöning)

Äquivalenzen

Zwei Formeln A und B sind **äquivalent**, Notation: $A \equiv B$, gdw. sie dieselben Modelle haben.

NB: $\equiv$ ist kein Junktor <u>in</u> der Logik, sondern Meta-Symbol!

$A \wedge B \equiv B \wedge A$	**Kommutativität** (auch für $\vee$)
$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$	**Assoziativität** (auch für $\vee$)
$\neg(\neg A) \equiv A$	**Doppelte Negation**
$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$	**Kontraposition**
$\neg(A \wedge B) \equiv \neg A \vee \neg B$	**deMorgansche Regel** (analog f. $\vee$)
$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$	**Distributivität** $\wedge$ ü. $\vee$ (analog $\vee$,$\wedge$)

Beweis je durch Aufstellen der W'tafeln (gleiche Einträge je in allen Zeilen)

KNF und DNF

Ein **Literal** ist eine Aussagevariable ohne oder mit 1 Negation.
Eine **Klausel** ist eine Disjunktion von Literalen.

Bspl: P Literal; $\neg Q$ Literal; $\neg\neg P$ kein Literal; $(\neg Q \vee P)$ Klausel

Eine Formel ist in **konjunktiver** (bzw. **disjunktiver**) **Normalform** (**KNF**, bzw. **DNF**), wenn sie eine Konjunktion von Disjunktionen von Literalen (bzw. Disjunktion von Konjunktionen) ist.
KNF „übersetzt" man auch als **Klauselnormalform**.

Beispiele:

KNF: $\neg P \wedge (\neg Q \vee R) \wedge (\neg R \vee S \vee \neg P)$

DNF: $(\neg Q \wedge P \wedge P) \vee \neg P \vee (\neg P \wedge S \wedge R)$

Notation: Die leere Klausel (enthält 0 Literale) schreiben wir $\square$

KNF und DNF sind duale Formen. Im Folgenden Konzentration auf KNF!

Umformung von Formeln in KNF

… durch Verwendung der Äquivalenzen Folie 37 nach Rezept:

1. Löse $\Leftrightarrow$ und $\Rightarrow$ auf
   (s. Folie 29: $P \Leftrightarrow Q$ zu $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$ und $P \Rightarrow Q$ zu $\neg P \vee Q$)
2. Bringe alle Negationszeichen direkt vor die Variablen;
   (deMorgansche Regeln)
   löse dabei doppelte Negation immer auf
3. Multipliziere aus und fasse zusammen, bis KNF fertig
   (Distributivität)

Beispiel: Überführe in KNF: $(P \wedge (Q \Rightarrow R)) \Rightarrow S$ | an der Tafel |

Tafelbeispiel

$$
\begin{aligned}
\big(P \wedge (Q \Rightarrow R)\big) \Rightarrow S \quad &\equiv \quad \big(P \wedge (\neg Q \vee R)\big) \Rightarrow S \\
&\equiv \quad \neg\big(P \wedge (\neg Q \vee R)\big) \vee S \\
&\equiv \quad \big(\neg P \vee \neg(\neg Q \vee R)\big) \vee S && \text{de M.} \\
&\equiv \quad \big(\neg P \vee (\neg\neg Q \wedge \neg R)\big) \vee S && \text{de M.} \\
&\equiv \quad \big(\neg P \vee (Q \wedge \neg R)\big) \vee S \\
&\equiv \quad \big((\neg P \vee Q) \wedge (\neg P \vee \neg R)\big) \vee S && \text{Distr.} \\
&\equiv \quad \big(S \vee (\neg P \vee Q)\big) \wedge \big(S \vee (\neg P \vee \neg R)\big) && \text{Komm.+Distr.} \\
&\equiv \quad (S \vee \neg P \vee Q) \wedge (S \vee \neg P \vee \neg R)
\end{aligned}
$$

Mehr zur KNF

Existenz

Zu jeder AL-Formel gibt es äquivalente Formeln in KNF.

Beweis Induktion über Formelaufbau nach Syntax; verwende „Rezept"

Uneindeutigkeit

Die äquivalente KNF zu e. gegebenen Formel ist uneindeutig.

Beispiele: $P \wedge Q$ und $Q \wedge P$ sind äquivalente, ungleiche KNF
P und $(P \vee R) \wedge (P \vee \neg R)$ äquivalente, ungleiche KNF

Minimierung

Es gibt effiziente Verfahren z. Minimierung e. gegebenen KNF.

Mengennotation („Klauselmenge")

Im Folgenden werden wir Formeln in KNF wie folgt notieren:

aus $(L_{1,1} \vee \ldots \vee L_{1,n1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,nm})$

wird $\{\{L_{1,1}, \ldots, L_{1,n1}\}, \ldots, \{L_{m,1}, \ldots, L_{m,nm}\}\}$

Nutzen der KNF

- Normalformen erlauben effizientere Verfahren
  - Beispiel Davis/Putnam (gleich)
  - Beispiel Resolution (später)
- Lokale Entscheidung der Erfüllbarkeit:
  - Ein einziges Exemplar von $\square$ signalisiert Inkonsistenz;
  - ist eine Interpretation lokal Modell jeder einzelnen Klausel, ist sie Modell der KNF-Formel insgesamt

Schnelleres *Model Checking* für KNF

1. Entferne Tautologien (Klauseln, in denen P und $\neg P$ vorkommen)
2. Terminiere sofort, wenn jede Klausel ein *true* bewertetes Literal enthält (↪erf'bar) oder wenn mindestens eine Klausel endgültig *false* bewertet ist (↪inkonsistent);
3. Bewerte Variablen, die als Literal pur vorkommen, so, dass das Literal wahr ist
   (**pur**: kommt überall nur negiert oder nur unnegiert vor)
4. Bewerte Variablen, die als Einsklauseln vorkommen, so, dass das Einsklausel-Literal wahr ist
   (**Einsklausel**: Klausel, die aus genau 1 Literal besteht)

Alle vier Regeln sind korrekt!

Erfüllbarkeitsprüfung nach Davis/Putnam

function DPLL-SATISFIABLE?(*s*) **returns** *yes* or *no*

inputs: *s*, a sentence in propositional logic

clauses ← the clause form of *s* with tautologies deleted

symbols ← the list of proposition symbols in *s*

return DPLL (*clauses*, *symbols*,[])

---

function DPLL(*clauses*, *symbols*, *model*) **returns** *yes* or *no*

if every clause in *clauses* is true in *model* **then return** *yes*

if some clause in *clauses* is false in *model* **then return** *no*

P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*–*P*, EXTEND(*P*, *value*, *model*))

P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*–*P*, EXTEND(*P*, *value*, *model*))

P ← FIRST(*symbols*); *rest* ← REST(*symbols*)

return DPLL(*clauses*, *rest*, EXTEND(*P*,1, *model*)) **or** /* Splitting */

DPLL(*clauses*, *rest*, EXTEND(*P*, 0, *model*))

Beispiel Davis/Putnam

Prüfe auf Erf'barkeit: $\{\{P,\neg Q\},\{\neg P,Q\},\{Q,\neg R\},\{\neg Q,R\}\}$

$\{P \mapsto 1\}$ **splitting** $\{P \mapsto 0\}$

$t,\{Q\},\{Q,\neg R\},\{\neg Q,R\}$

$\{\neg Q\},t,\{Q,\neg R\},\{\neg Q,R\}$

$\{P \mapsto 1, Q \mapsto 1\}$ **Einsklausel**

$t,t,t,\{R\}$

$\{P \mapsto 1, Q \mapsto 1, R \mapsto 1\}$ **pur**

t, t, t, t ➥ *yes* ➥ erfüllbar!

Eigenschaften von DPLL

… ist ein Beispiel eines systematischen,
korrekten & vollständigen Verfahrens,
das empirisch konkurrenzfähig ist!

- ☺ Speicherbedarf: $O(mn)$ (m Klauseln über n Variable)
- ☹ Zeitbedarf: $O(2^n)$
  (aber das ist der worst case: Praktisch ist DPLL oft sehr schnell! z.T. mit Heuristiken)
- ☺ korrekt (wenn Modell gefunden, dann stimmt es)
- ☺ vollständig (wenn Modell existiert, findet es das)

Fazit Erfüllbarkeitsprüfung

- Erfüllbarkeit kann in der Aussagenlogik konstruktiv geprüft werden durch den Versuch, ein Modell zu erstellen
- Es gibt dazu systematische (und „lokale“, s.Kap. 3.) Verfahren
- Für endliche Modellmengen sind Verfahren zur Erfüllbarkeitsprüfung Stand der Technik für Aussagelogik
  (mit Repräsentationen durch binäre Entscheidungsdiagramme (*binary decision diagrams*, BDDs) kann man Zustandsmengen bis in die Größenordnung 10^{200} Zuständen analysieren)
- Für unendliche Modellmengen oder ausdrucksstärkere Logiken braucht man allgemeinere Kalküle