Klauselmengen (I)

Atom: $A \in V$

Literal: A, $\neg A$

Klausel: Disjunktion von Literalen
$[\neg]A_1 \vee [\neg]A_2 \vee \ldots \vee [\neg]A_n$ ($n \geq 0$).

Klauselmenge: Menge (Konjunktion) von m Klauseln ($m \geq 0$).

Belegung: Funktion $\varphi : V \to \{0, 1\}$.

Klauselmengen (II)

$A \in V$ heißt wahr unter φ, falls $\varphi(A) = 1$.

$\neg A$ heißt wahr unter φ, falls A unter φ falsch ist.

Eine Klausel Γ heißt wahr unter φ, falls *mindestens ein* Literal von Γ unter φ wahr ist.

Eine Klauselmenge M heißt wahr unter φ, falls *alle* Klauseln von M unter φ wahr sind.

Klauselmengen (III)

Eine Klausel(menge) heißt *allgemeingültig*, falls sie unter jeder Belegung wahr ist.

Eine Klausel(menge) heißt *erfüllbar*, falls eine Belegung existiert, unter der sie wahr ist. (Anderenfalls heißt sie unerfüllbar.)

Problem: *Ist eine gegebene Klauselmenge erfüllbar?*

Komplexität (I)

Wie untersucht man, ob eine Klauselmenge erfüllbar ist?

Wertetabelle:

ineffizient: Laufzeit für unerfüllbare Klauselmengen immer exponentiell (bzgl. der Größe der Eingabe).

Geht es schneller (garantiert polynomiell) ?

Komplexität (II)

Komplexitätsklassen für Entscheidungsprobleme (Ja/Nein-Probleme):

P = Menge aller Probleme, die von einem Programm entschieden werden können, dessen Laufzeit polynomiell bzgl. der Größe der Eingabe ist.

NP = Menge aller Probleme, die von einem nichtdeterministischen Programm entschieden werden können, dessen Laufzeit polynomiell bzgl. der Größe der Eingabe ist.

Nichtdeterministisches Programm:

„Programm mit Rechen- und Rateschritten“

Komplexität (III)

Ein nichtdeterministisches Programm *prog* entscheidet ein Problem Q falls:

- Falls $Q(x)$ gilt, dann liefert *prog* bei Eingabe x bei richtigem Raten die Antwort „$Q(x)$ gilt“.
- Falls $Q(x)$ nicht gilt, dann liefert *prog* bei Eingabe x garantiert die Antwort „$Q(x)$ gilt nicht“.

Komplexität (IV)

$\mathrm{P} \subseteq \mathrm{NP}$: offensichtlich.

$\mathrm{P} \supseteq \mathrm{NP}$: bis heute unbekannt.

NP-vollständige Probleme:

die „schwersten“ Probleme in NP

z. B.: Erfüllbarkeit (SAT), Knotenüberdeckung, Hamiltonscher Zyklus, Partition [Garey & Johnson]

Cook (1971): $\mathrm{SAT} \in \mathrm{P} \;\Leftrightarrow\; \mathrm{P} = \mathrm{NP}$.

Komplexität (V)

Wie untersucht man, ob eine Klauselmenge erfüllbar ist?

Geht es garantiert polynomiell?

Wahrscheinlich nicht.

Geht es deutlich schneller als Wertetabelle?

Vereinfachungen (I)

$\vee$, $\wedge$ assoziativ und kommutativ

$\Rightarrow$ Umsortieren von Literalen und Klauseln erlaubt.

Leere Klauselmenge ist allgemeingültig (Tautologie)

Klausel $A \vee \neg A \vee \Gamma$ ist allgemeingültig (Tautologie)

$\Rightarrow$ Klausel kann aus der Klauselmenge gestrichen werden.

Vereinfachungen (II)

Klausel $[\neg]A \vee [\neg]A \vee \Gamma$ ist wahr unter φ, genau dann wenn $[\neg]A \vee \Gamma$ unter φ wahr ist.

$\Rightarrow$ wiederholte Literale können aus einer Klausel gestrichen werden.

Wenn Γ wahr ist unter φ, dann ist auch $\Gamma \vee \Delta$ wahr unter φ.

$\Rightarrow$ subsumierte Klausel $\Gamma \vee \Delta$ kann aus der Klauselmenge gestrichen werden.

Wahrheit einer Klauselmenge M unter φ hängt nur von den Werten von φ auf $\mathrm{Var}(M)$ ab.

Vereinfachungen (III)

Einheitsklausel:

Klausel mit genau einem Literal.

Wenn eine Klauselmenge M eine Einheitsklausel A enthält, dann sind äquivalent:

- Es existiert φ, so daß M unter φ wahr ist.
- Es existiert φ', so daß M unter φ' wahr ist und $\varphi'(A) = 1$.

(analog für Einheitsklausel $\neg A$).

Vereinfachungen (IV)

$M[A/1]$ = die Klauselmenge, die man aus M enthält, wenn man

- alle Klauseln, die das Literal A enthalten, streicht, und
- in allen Klauseln, die das Literal $\neg A$ enthalten, $\neg A$ streicht.

(analog für $M[A/0]$).

Es sind äquivalent:

- M ist wahr unter φ' und $\varphi'(A) = 1$.
- $M[A/1]$ ist wahr unter φ' und $\varphi'(A) = 1$.

(analog für $\varphi'(A) = 0$).

Vereinfachungen (V)

Pures Literal:

Literal, dessen Negation in der Klauselmenge nicht vorkommt.

Wenn eine Klauselmenge M ein pures Literal A enthält, dann sind äquivalent:

- Es existiert φ, so daß M unter φ wahr ist.
- Es existiert φ', so daß M unter φ' wahr ist und $\varphi'(A) = 1$.

(analog für pures Literal $\neg A$).

Vereinfachungen (VI)

Wenn eine Klauselmenge ein Atom A enthält, dann sind äquivalent:

- Es existiert φ, so daß M unter φ wahr ist
- Es existiert φ', so daß M unter φ' wahr ist und $\varphi'(A) = 1$,

  oder

  es existiert φ'', so daß M unter φ'' wahr ist und $\varphi''(A) = 0$.

Davis-Putnam-Algorithmus (I)

```
boolean DP(Klauselmenge M) {
    if (M ist leer) {
        return true;
    } else if (M enthält leere Klausel) {
        return false;
    } else if (M enthält Einheitsklausel A) {
        return DP(M[A/1]);
    } else if (M enthält Einheitsklausel ¬A) {
        return DP(M[A/0]);
    } else if (M enthält pures Literal A) {
        return DP(M[A/1]);
    } else if (M enthält pures Literal ¬A) {
        return DP(M[A/0]);
    } else {
        wähle beliebiges Atom A aus Var(M) aus;
        return DP(M[A/1]) || DP(M[A/0]);
    }
}
```


Davis-Putnam-Algorithmus (II)

Terminierung: ja.

Lösung φ des Erfüllbarkeitsproblems:

auf 0 oder 1 gesetzte Atome während des Algorithmus aufsammeln,

Atome, die zum Schluß nicht gesetzt sind: beliebig.

Laufzeit:

im schlimmsten Fall exponentiell
(falls exponentiell viele Verzweigungen).

stark abhängig von Auswahl der Variablen bei Verzweigung
[Kleine Büning & Lettmann].

Davis-Putnam-Algorithmus (III)

Naive Implementierung:

Datenstrukturen (insbesondere M) als lokale Variable der rekursiven Funktion.

Problem: Für zweiten Fall der Verzweigung muß Zustand von M gespeichert werden.

$\Rightarrow$ Platzbedarf pro Pfad (!) quadratisch bzgl. der Anzahl der Variablen.

Davis-Putnam-Algorithmus (IV)

Besser:

Klauselmenge als *globale* Datenstruktur.

Vor rekursivem Aufruf:
Ändere globale Klauselmenge
und merke Änderungen.

Nach rekursivem Aufruf:
Stelle Ausgangszustand wieder her.

Vorteil: Platzbedarf pro Pfad linear.

Nachteil: Implementierung komplizierter.