Grundlagen der Künstlichen Intelligenz

28. Aussagenlogik: DPLL-Algorithmus

Malte Helmert

Universität Basel

2. Mai 2014

Aussagenlogik: Überblick

Motivation

Motivation für Aussagenlogik

- Aussagenlogik erlaubt Repräsentation von Wissen und Schlussfolgerungen auf Grundlage dieses Wissens
- viele Anwendungsprobleme direkt kodierbar, z. B.:
  - Constraint-Satisfaction-Probleme aller Art
  - Schaltkreisentwurf und -verifikation
- viele Probleme verwenden Logik als einen Bestandteil, z. B.:
  - Handlungsplanung
  - General Game Playing
  - Beschreibungslogik-Anfragen (Semantic Web)

Aussagenlogik: algorithmische Fragestellungen

wesentliche Fragestellungen:

- Schlussfolgern ($\Theta \models \varphi$?):
  Folgt aus Formeln Θ die Formel φ logisch?
- Äquivalenz ($\varphi \equiv \psi$):
  Sind Formeln φ und ψ logisch äquivalent?
- Erfüllbarkeit (SAT):
  Ist Formel φ erfüllbar? Falls ja, finde eine erfüllende Belegung.

Das Erfüllbarkeitsproblem

Das Erfüllbarkeitsproblem (SAT)

Gegeben:

aussagenlogische Formel in konjunktiver Normalform (KNF)

Üblicherweise repräsentiert als Paar $\langle V, \Delta \rangle$:

- V Menge von Aussagevariablen (Propositionen)
- Δ Menge von Klauseln über V
  (Klausel = Menge von Literalen v bzw. $\neg v$ mit $v \in V$)

Gesucht:

- erfüllende Belegung der Formel (Modell)
- oder Beweis, dass keine erfüllende Belegung existiert

SAT ist ein berühmtes NP-vollständiges Problem
(Cook 1971; Levin 1973).

Relevanz von SAT

- Unter SAT versteht man oft auch das Erfüllbarkeitsproblem für allgemeine Logikformeln (statt Einschränkung auf KNF).
- Allgemeines SAT ist auf den KNF-Fall zurückführbar (Aufwand für Umformung ist $O(n)$)
- Alle zuvor genannten Logikprobleme sind auf SAT zurückführbar (Aufwand für Umformung ist $O(n)$)

$\rightsquigarrow$ SAT-Algorithmen sehr wichtig und sehr intensiv erforscht

dieses und nächstes Kapitel: SAT-Algorithmen

Systematische Suche: DPLL

SAT vs. CSP

SAT kann als Constraint-Satisfaction-Problem aufgefasst werden:

- CSP-Variablen = Aussagevariablen
- Wertebereiche = $\{\mathbf{F}, \mathbf{T}\}$
- Constraints = Klauseln

Allerdings haben wir hier oft Constraints,
die mehr als zwei Variablen betreffen.

Wegen Verwandtschaft alle CSP-Ideen auf SAT anwendbar:

- Suche
- Inferenz
- Variablen- und Werteordnungen

Der DPLL-Algorithmus

Der DPLL-Algorithmus (Davis/Putnam/Logemann/Loveland) entspricht Backtracking mit Inferenz bei CSPs.

- rekursiver Aufruf DPLL(Δ, I)
  für Klauselmenge Δ und partielle Belegung I
- Ergebnis ist erfüllende Belegung, die I erweitert;
  unsatisfiable, wenn keine solche Belegung existiert
- oberster Aufruf als DPLL($\Delta, \emptyset$)

Inferenz in DPLL:

- simplify: nachdem der Variablen v der Wert d zugewiesen wird, werden alle Klauseln vereinfacht, die über v sprechen
  $\rightsquigarrow$ entspricht Forward Checking (für mehrstellige Constraints)
- Unit Propagation: Variablen, die in Klauseln ohne weitere Variablen (Einheitsklauseln) auftreten, werden sofort belegt (entspricht minimum remaining values-Variablenordnung)

Der DPLL-Algorithmus: Pseudo-Code

function DPLL(Δ, I):

if $\Box \in \Delta$: [Es gibt eine leere Klausel $\rightsquigarrow$ unerfüllbar]
    return unsatisfiable
else if $\Delta = \emptyset$: [keine Klauseln übrig $\rightsquigarrow$ Belegung I erfüllt die Formel]
    return I
else if there exists a unit clause $\{v\}$ or $\{\neg v\}$ in Δ: [Unit Propagation]
    Let v be such a variable, d the truth value that satisfies the clause.
    $\Delta' := \text{simplify}(\Delta, v, d)$
    return $\text{DPLL}(V, \Delta', I \cup \{v \mapsto d\})$
else: [Splitting Rule]
    Select some variable v which occurs in Δ.
    for each $d \in \{\mathbf{F}, \mathbf{T}\}$ in some order:
        $\Delta' := \text{simplify}(\Delta, v, d)$
        $I' := \text{DPLL}(V, \Delta', I \cup \{v \mapsto d\})$
        if $I' \neq$ **unsatisfiable**
            return I'
    return unsatisfiable

Der DPLL-Algorithmus: simplify

function simplify(Δ, v, d)

Let ℓ be the literal on v that is satisfied by $v \mapsto d$.
Let $\bar{\ell}$ be the complementary (opposite) literal to ℓ.
$\Delta' := \{C \mid C \in \Delta \text{ s.t. } \ell \notin C\}$
$\Delta'' := \{C \setminus \{\bar{\ell}\} \mid C \in \Delta'\}$
return Δ''

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

2a. $X \mapsto \mathbf{F}$
   $\{\{Y\}, \{\neg Y\}\}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

2a. $X \mapsto \mathbf{F}$
   $\{\{Y\}, \{\neg Y\}\}$

3a. Unit Propagation: $Y \mapsto \mathbf{T}$
   $\{\square\}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. Unit Propagation: $Y \mapsto \mathbf{T}$
$\{\Box\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. Unit Propagation: $Y \mapsto \mathbf{T}$
$\{\Box\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

3b. Unit Propagation: $Y \mapsto \mathbf{F}$
$\{\}$

Beispiel (1)

$$\Delta = \{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
2. Splitting Rule:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. Unit Propagation: $Y \mapsto \mathbf{T}$
$\{\Box\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

3b. Unit Propagation: $Y \mapsto \mathbf{F}$
$\{\}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{W, \neg X, \neg Y\}, \{X\}, \{Y\}\}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{W, \neg X, \neg Y\}, \{X\}, \{Y\}\}$
2. Unit Propagation: $X \mapsto \mathbf{T}$
   $\{\{W, \neg Y\}, \{Y\}\}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{W, \neg X, \neg Y\}, \{X\}, \{Y\}\}$
2. Unit Propagation: $X \mapsto \mathbf{T}$
   $\{\{W, \neg Y\}, \{Y\}\}$
3. Unit Propagation: $Y \mapsto \mathbf{T}$
   $\{\{W\}\}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{W, \neg X, \neg Y\}, \{X\}, \{Y\}\}$
2. Unit Propagation: $X \mapsto \mathbf{T}$
   $\{\{W, \neg Y\}, \{Y\}\}$
3. Unit Propagation: $Y \mapsto \mathbf{T}$
   $\{\{W\}\}$
4. Unit Propagation: $W \mapsto \mathbf{T}$
   $\{\}$

Beispiel (2)

$$\Delta = \{\{W, \neg X, \neg Y, \neg Z\}, \{X, \neg Z\}, \{Y, \neg Z\}, \{Z\}\}$$

1. Unit Propagation: $Z \mapsto \mathbf{T}$
   $\{\{W, \neg X, \neg Y\}, \{X\}, \{Y\}\}$
2. Unit Propagation: $X \mapsto \mathbf{T}$
   $\{\{W, \neg Y\}, \{Y\}\}$
3. Unit Propagation: $Y \mapsto \mathbf{T}$
   $\{\{W\}\}$
4. Unit Propagation: $W \mapsto \mathbf{T}$
   $\{\}$

Eigenschaften von DPLL

- DPLL ist korrekt und vollständig.
- DPLL erzeugt ein Modell, falls eines existiert.
  - Manche Variablen werden evtl. in der Lösung I nicht belegt; deren Werte können dann beliebig gewählt werden.
- Zeitaufwand im Allgemeinen exponentiell

$\rightsquigarrow$ gute Variablenordnungen in der Praxis wichtig; ebenso zusätzliche Inferenzmethoden, v.a. clause learning

- beste bekannte SAT-Algorithmen basieren auf DPLL

DPLL auf Hornformeln

Hornformeln

wichtiger Spezialfall: Hornformeln

Definition (Hornformel)

Eine Hornklausel ist eine Klausel
mit maximal einem positivem Literal, also von der Form

$$\neg x_1 \vee \cdots \vee \neg x_n \vee y \text{ oder } \neg x_1 \vee \cdots \vee \neg x_n$$

(Der Fall $n = 0$ ist erlaubt.)

Eine Hornformel ist eine aussagenlogische Formel
in konjunktiver Normalform, die nur aus Hornklauseln besteht.

$\rightsquigarrow$ Grundlage von Logikprogrammierung (z.B. PROLOG)
und deduktiven Datenbanken

DPLL auf Hornformeln

Satz (DPLL auf Hornformeln)

Wenn die Eingabeformel φ eine Hornformel ist, dann ist der Zeitaufwand von DPLL polynomiell in der Länge von φ.

Beweis.

Eigenschaften:

1. Wenn Δ eine Hornformel ist, dann ist auch simplify(Δ, v, d) eine Hornformel. (Warum?)

   $\rightsquigarrow$ alle während der Suche von DPLL betrachteten Formeln sind Hornformeln, wenn die Eingabe es ist
2. Jede Hornformel ohne leere oder Einheitsklauseln ist erfüllbar:
   - alle solchen Klauseln enthalten mindestens zwei Literale
   - da Horn: mindestens eines davon negativ
   - Zuweisung **F** an alle Variablen erfüllt die Formel

DPLL auf Hornformeln (Fortsetzung)

Beweis (Fortsetzung).

3. Aus 2. folgt:
   - immer, wenn die Splitting Rule angewandt wird, ist die aktuelle Formel erfüllbar, und
   - immer, wenn dabei eine falsche Entscheidung getroffen wird, wird dies sofort (d. h. nur durch Unit-Propagation-Schritte und Herleiten einer leeren Klausel) erkannt.
4. Deshalb kann der erzeugte Suchbaum für n Variablen nur maximal n viele Knoten enthalten, in denen die Splitting Rule angewandt wird (und der Baum verzweigt).
5. Damit ist der Suchbaum nur polynomiell gross und folglich die Gesamtlaufzeit polynomiell.

$\square$

Zusammenfassung

Zusammenfassung

- Erfüllbarkeit grundlegendes Problem der Aussagenlogik, auf das andere Probleme zurückgeführt werden können
- hier: Erfüllbarkeit für KNF-Formeln
- David-Putnam-Logemann-Loveland-Prozedur (DPLL): systematische Backtracking-Suche mit Unit Propagation als wesentlicher Inferenzmethode
- praktisch erfolgreicher Algorithmus, vor allem in Kombination mit weiteren Ideen wie clause learning
- polynomiell auf Horn-Formeln (= max. ein positives Literal pro Klausel)