Planen und Suchen

Vorlesung SS 2003

Reinhold Letz (TU München)

Täglich 16.15 - 17.45 Uhr von 10.3.02 - 21.3.02

Inhalte der Vorlesung

- Was ist Planen?
- Planungsverfahren (Situationskalkül, STRIPS)
- Klassische Probleme des Planens (Frame-Problem)
- Methoden der Übersetzung in entscheidbare Logiken
- Aussagenlogische Entscheidungsverfahren
- Entscheidungsverfahren für Quantifizierte Booolesche Logik
- Optimierungs- und Implementierungstechniken

Was ist Planen?

Im Bereich Wissensbasierte Systeme/Künstliche Intelligenz/Robotik:

- Modellierung der Entscheidungsfindung in komplizierten Umgebungen
- Entwicklung effizienter Algorithmen zur Entscheidungsfindung
- Betonung allgemeiner Repräsentationsmethoden und allgemeiner Lösungsansätze

Typen von Planungsproblemen:

- Wege- und Bewegungsplanung (Kontrolle autonomer Roboter, Geometrie)
- Scheduling (Ordnung und Auswahl vordefinierter Aktionen)
- In dieser Vorlesung: sog. Task planning

Ein typisches Planungsproblem

Ein Beispiel:

Ein Bauer will einen Wolf, eine Ziege und einen großen Kohlkopf über einen Fluß transportieren. Er hat dafür ein Boot zur Verfügung, in welches außer ihm selbst lediglich noch ein weiteres Objekt hineinpaßt. Außerdem muß der Bauer beim Transport berücksichtigen, daß, wann immer er an einem Ufer nicht anwesend ist, möglicherweise der Wolf die Ziege frißt oder die Ziege den Kohlkopf.

Mit welcher Folge von Transportaktionen kann der Bauer alle Objekte unbeschädigt auf die andere Seite bringen?

Zentrale Trias beim Planen:

1. Ausgangssituation
2. Zielsituation
3. Aktionen

Kernkonzepte im Planen

Determinismus vs. Nichtdeterminismus:

- Deterministisches Planen: Der Anfangszustand und jede Folge von Aktionen bestimmt eindeutig den momentanen Weltzustand
- Mögliche Quellen des Nichtdeterminismus: Unvollständigkeit des Wissens, nichtdeterministische Aktionen

Beobachtbarkeit (der Folgen von Aktionen):

- Relevant im nichtdeterministischen Fall und bei mehr als einem Anfangszustand
- Zwei Alternativen:
  - Planfindung gekoppelt an Planausführung
  - Erzeugung eines bedingten Planes (mit Alternativen)
- Volle Beobachtbarkeit vs. partielle Beobachtbarkeit

Kernkonzepte im Planen (2)

Zeit:

- Zumeist diskrete (Einheits-)Zeit: Zeit getaktet, jede Aktion dauert genau eine Zeiteinheit, keine graduelle Veränderung, keine Feinanalyse von Aktionen
- Kompliziertere Zeitmodelle möglich, erfordern aber weit grösseren Modellierungsaufwand. Deshalb meistens Abbildung in diskretes Modell (graduelle Änderungen kodiert in Zustandsbeschreibung)

Planqualität:

- Oft ist man zufrieden mit dem Erreichen eines Zielzustandes
- Qualitätsmerkmale eines Planes: Länge, Kosten, Erfolgswahrscheinlichkeit

Geschichte des Planens

Beginn in den 1960ern:

- Programme zur Simulation menschlicher Problemlösungsfähigkeiten
- General Problem Solver von Newell und Simon: Zustandsraumsuche und Benutzung einer Heuristik zur Anstandsbewertung zwischen aktuellem und Zielzustand (unvollständig)

Ende der 1960er:

- Vorschlag von Green, Theorembeweiser zur Plankonstruktion einzusetzen
- Auf Grund der mangelnden Leistungsfähigkeit der damaligen Theorembeweiser aufgegeben zugunsten spezialisierter Planungsalgorithmen

Geschichte des Planens (2)

Anfang der 1970er:

- Entwicklung des STRIPS-Systems von Fikes und Nilsson: historisch einflussreichstes Planungssystem
- Zustände als Mengen von Formeln repräsentiert
- Operatoren ändern diese Mengen durch Wegnahme bzw. Hinzunahme: add list, delete list,
- Ausführbarkeit der Operatoren durch preconditions gesteuert
- Zur Steuerung: Heuristiken wie in General Problem Solver

Geschichte des Planens (3)

Mitte der 1970er:

- Hierarchisches Planen von Sacerdoti: Benutzung einer hierarchischen Planstruktur
  - Hauptaufgabe wird in kleinere Teilaufgaben aufgeteilt, die rekursiv gelöst werden.
  - Jede Teilaufgabe kann durch unterschiedliche Methoden gelöst werden, je weniger, desto effizienter die Plansuche
  - Typischerweise Einsatz problemspezifischer Problemlöser und Heuristiken
- Planen mit partiellen Ordnungen von Sacerdoti, McAllester, Rosenblitt: Inkrementelle Konstruktion eines Plans ausgehend von Anfangszustand und zu erreichenden Zielen durch:
  - Entweder: Hinzunahme einer Aktion, die eines der Ziele erreicht oder eine der Vorbedingungen erfüllt
  - Oder: Hinzunahme einer Ordnungsbedingung (-constraint) zur Auflösung möglicher Konflikte zwischen Aktionen im bisherigen Plan

Geschichte des Planens (4)

1990er:

- Graphplan planner von Blum und Furst: Beginn der Wegbewegung vom partial-order-Ansatz, zum Teil wieder zum total-order-Ansatz, der vorher als zu ineffizient betrachtet wurde

- Planen als (aussagenlogische) Erfüllbarkeit von Kautz und Selman: Wiederbelebung der deduktiv- und logikbasierten Ansätze durch die Verbesserung der Lösungsverfahren für die Aussagenlogik

Übersichtspapier von Allen u.a. 1990

Aussagenlogik

Syntax der Aussagenlogik

- Gegeben eine Menge $\mathcal{P}$ von *Aussagenvariablen*.
- Die *(Formeln der) Aussagenlogik* für $\mathcal{P}$ sei(en) definiert durch $PL(\mathcal{P}) :=$

$$\mathcal{P} \mid \top \mid \bot \mid \neg PL \mid (PL \wedge PL) \mid$$
$$(PL \vee PL) \mid (PL \rightarrow PL) \mid (PL \leftrightarrow PL)$$

- Evtl. werden Klammern eingespart, die Präzedenz der Operatoren sei dann wie folgt festgelegt: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Beispiel:

  ((es_regnet $\rightarrow$ strasse_nass) $\wedge$ $\neg$ strasse_nass) $\rightarrow$ ($\neg$ es_regnet)

  ist eine Formel der Aussagenlogik $PL(\{\text{es\_regnet, strasse\_nass}\})$.

Semantik der Aussagenlogik

Interpretation

- Eine *Interpretation* oder *Belegung* für die Aussagenlogik $PL(\mathcal{P})$ ist eine Abbildung $\iota : \mathcal{P} \longrightarrow \{\mathbf{w}, \mathbf{f}\}$, wobei $\{\mathbf{w}, \mathbf{f}\}$ *Wahrheitswertmenge* heisst.

Fortsetzung einer Interpretation auf Formeln

Induktiv definiert wie folgt:

$\mathcal{I}$ *erfüllt* $p \in \mathcal{P}$, geschrieben $\mathcal{I} \models p$ gdw $\iota(p) = \mathbf{w}$,

$\mathcal{I} \models \top$ und $\mathcal{I} \not\models \bot$,

$\mathcal{I} \models (A \wedge B)$ gdw $\mathcal{I} \models A$ und $\mathcal{I} \models B$,

$\mathcal{I} \models (A \vee B)$ gdw $\mathcal{I} \models A$ oder $\mathcal{I} \models B$,

$\mathcal{I} \models (A \rightarrow B)$ gdw $\mathcal{I} \not\models A$ oder $\mathcal{I} \models B$,

$\mathcal{I} \models (A \leftrightarrow B)$ gdw $\mathcal{I} \models (A \rightarrow B)$ und $\mathcal{I} \models (B \rightarrow A)$.

Semantik der Aussagenlogik (2)

Modellbegriff der Aussagenlogik

Gegeben eine Formel(menge) F der Aussagenlogik $PL(\mathcal{P})$:

- F heisst *erfüllbar* gdw es eine Interpretation ι für $PL(\mathcal{P})$ gibt, die (alle Formeln in) F erfüllt; ι heisst dann auch *Modell* für F.
- Eine Formel(menge), die kein Modell hat, heisst *unerfüllbar*.
- F heisst *(allgemein)gültig* gdw jede Interpretation für $PL(\mathcal{P})$ F erfüllt.

Beispiele

- Die Formel (es_regnet $\rightarrow$ strasse_nass) $\wedge$ $\neg$ strasse_nass

  ist erfüllbar, aber nicht allgemeingültig.
- $F =$ ((es_regnet $\rightarrow$ strasse_nass) $\wedge$ $\neg$ strasse_nass) $\rightarrow$ $\neg$ es_regnet

  ist allgemeingültig.
- Dementsprechend ist die Formel $\neg F$ unerfüllbar.

Semantik der Aussagenlogik (3)

Implikation und Äquivalenz
Gegeben zwei Formel(-menge)n F und G einer Aussagenlogik $PL(\mathcal{P})$:

- F *impliziert* G: $F \models G$ gdw jedes Modell für F eine Modell für G ist.
- F und G sind *äquivalent*: $F \equiv G$ gdw F und G sich gegenseitig implizieren.
- F und G sind *erfüllbarkeitsäquivalent* gdw beide Formeln erfüllbar oder beide unerfüllbar sind.

Satz: Aus Äquivalenz folgt Erfüllbarkeitsäquivalenz, aber nicht umgekehrt.

Zusammenhang zwischen Meta- und Objektsprache
Gegeben zwei Formeln F und G einer Aussagenlogik $PL(\mathcal{P})$:

- $F \models G$ gdw die Formel $F \rightarrow G$ allgemeingültig ist
- $F \equiv G$ gdw die Formel $F \leftrightarrow G$ allgemeingültig ist

Semantik der Aussagenlogik (4)

Wichtige semantische Eigenschaften

- Absorption:
$$F \vee (F \wedge G) \equiv F$$
$$F \wedge (F \vee G) \equiv F$$
$$\top \wedge F \equiv F$$
$$\top \vee F \equiv \top$$
$$\bot \wedge F \equiv F$$
$$\bot \vee F \equiv \bot$$

- Idempotenz:
$$F \vee F \equiv F$$
$$F \wedge F \equiv F$$

- Kommutativität:
$$F \vee G \equiv G \vee F$$
$$F \wedge G \equiv G \wedge F$$

- Assoziativität:
$$F \vee (G \vee H) \equiv (F \vee G) \vee H$$
$$F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$$

Zentrale Probleme der Aussagenlogik

Beweis der Erfüllbarkeit einer Formel

Gegeben eine aussagenlogische Formel F:

- Zeige, dass F erfüllbar ist.
- Gebe ein Modell ι für F an. (Belegung ι als Zeuge/Beweisobjekt)
- Probleme sind NP-vollständig.

Beweis der Allgemeingültigkeit/Unerfüllbarkeit einer Formel

Gegeben eine aussagenlogische Formel F:

- Zeige, dass F allgemeingültig ist.
- Zeige, dass $\neg F$ unerfüllbar ist.
- Probleme sind coNP-vollständig.
- Gibt es immer „kurze“ Beweise für Allgemeingültigkeit/Unerfüllbarkeit?
  Wenn nein, dann gilt: NP $\neq$ coNP, und damit P $\neq$ NP.

Bezüge zur Komplexitätstheorie

NP bzw. NPTIME:

Klasse aller Sprachen, die von einem nichtdeterministischen Algorithmus in polynomieller Zeit (bzgl. der Eingabegrösse) akzeptiert werden

- Konkretisierung Algorithmus: Turingmaschine oder Registermaschine
- Eine Sprache L_0 ist NP-hart, wenn es für alle Sprachen $L \in NP$ eine polynomielle Transformation Θ von L nach L_0 gibt derart, dass $w \in L$ gdw $\Theta(w) \in L_0$.
- Eine Sprache L_0 ist NP-vollständig, wenn L_0 NP-hart und $L_0 \in NP$ ist.

P: Klasse aller Sprachen, die von einem deterministischen Algorithmus in polynomieller Zeit akzeptiert werden

coNP: Klasse aller Sprachen L derart, dass $\{w : w \notin L\}$ in NP

Offene Fragen:

- P =? NP bzw. P =? coNP
- NP =? coNP

Bezüge zur Komplexitätstheorie (2)

PSPACE bzw. NPSPACE:

Klasse aller Sprachen, die von einem deterministischen bzw. nichtdeterministischen Algorithmus auf polynomiellem Platz (bzgl. der Eingabegrösse) akzeptiert werden

Resultate:

- PSPACE = NPSPACE (Savitch)
- coPSPACE = PSPACE (Immerman et. al.)

Offene Fragen:

- P =? PSPACE
- NP =? PSPACE

Aussagenlogische Entscheidungsverfahren

Semantisch motivierte Verfahren
arbeiten durch systematisches Überprüfen der/aller Interpretationen

- Wahrheitstafelmethode
- Semantische Bäume (Davis/Loveland/Logemann-Verfahren)
- Binäre Entscheidungsdiagramme

Beweistheoretische Verfahren (Logikkalküle)
prüfen systematisch, ob die Formel beweisbar/widerlegbar ist

- Tableaukalküle
- Resolution
- „Starke“ Kalküle:
  - Axiomatische Kalküle (Frege/Hilbert-Kalküle)
  - Natürliches Schliessen
  - Sequenzenkalkül

Eine deterministische Plansprache (DPLAN)

Modellierung von Weltzuständen: Gegeben eine Aussagenlogik PL, ein (Welt-)Zustand ist eine Interpretation für PL

Beschreibung von Aktionen:

- Anwendbarkeit einer Aktion: Vorbedingung
- Was ist der Folgezustand nach Ausführung der Aktion?
- Repräsentation als Paar: $\langle c, e \rangle$, c Vorbedingung , e Wirkung/Effekt der Aktion
- Vorbedingung c ist beliebige (aussagen-)logische Formel

Beschreibung von Effekten:

- Effekt e soll kodieren, welche Sachverhalte wahr werden
- Im aussagenlogischen Fall:
  - welche $p \in \mathcal{P}$ sollen wahr, welche falsch werden
  - Notation als eingeschränkte aussagenlogische Formeln: $e = p \wedge \neg q$ bedeute: nach der Aktion ist p wahr und q falsch

Wirkungen oder Effekte

Allgemeine Syntax deterministischer Effekte:

1. $\top$ ist ein Effekt
2. Falls e ein Effekt ist und p eine aussagenlogische Variable, die nicht in e vorkommt, dann sind $e \wedge p$ und $e \wedge \neg p$ Effekte

Anwendung einer Aktion auf einen Zustand:

Eine Aktion $\langle c, e \rangle$ heisst anwendbar in einem Zustand s gdw $s(c) = \mathbf{w}$

Resultat eines Effektes e auf eine Situation s, geschrieben $[e]_s$, definiert wie folgt:

1. $[l]_s = \{l\}$ falls l ein Literal
2. $[e_1 \wedge e_2]_s = [e_1]_s \cup [e_2]_s$

Folgezustand $\alpha(s)$ eines Zustandes s nach einer anwendbaren Aktion $\alpha = \langle c, e \rangle$:

$\alpha(s)(p) = \mathbf{w}$ bzw. $\mathbf{f}$ falls p bzw. $\neg p$ in $[e]_s$, ansonsten $\alpha(s)(p) = s(p)$

Deterministische Planungsprobleme

Deterministisches Planungsproblem: $\langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$:

- $\mathcal{P}$ Menge aussagenlogischer Variablen,
- Anfangszustand: $\mathcal{I}$ Belegung für $\mathcal{P}$
- Aktionsoperatoren: $\mathcal{O}$ Aktionen
- Zielformel: $\mathcal{G}$ Formel über $\mathcal{P}$

Plan für ein deterministisches Planungsproblem:

Gegeben sei ein deterministisches Planungsproblem $\Phi = \langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$.

Ein Plan für Φ ist eine Folge von Aktionen $\alpha_1, \ldots, \alpha_n$ aus $\mathcal{O}$ derart, dass $\alpha_n(\cdots \alpha_1(\mathcal{I}) \cdots) \models \mathcal{G}$.

Modellierung des Beispiels

Notation: b, w, z, k : Bauer, Wolf, Ziege bzw. Kohlkopf ist am Ausgangsufer

Mögliche Aktionen (Negation durch Überstreichen abgekürzt):

- Bauer allein: $\langle b, \bar{b}\rangle$, $\langle \bar{b}, b\rangle$
- Bauer mit Objekt: $\langle b \wedge w, \bar{b} \wedge \bar{w}\rangle$, $\langle \bar{b} \wedge \bar{w}, b \wedge w\rangle$, $\langle b \wedge k, \bar{b} \wedge \bar{k}\rangle$, $\langle \bar{b} \wedge \bar{k}, b \wedge k\rangle$, $\langle b \wedge z, \bar{b} \wedge \bar{z}\rangle$, $\langle \bar{b} \wedge \bar{z}, b \wedge z\rangle$

Sichere Aktionen:

- Gefahr am Ausgangsufer bzw. Zielufer: $ga \leftrightarrow (z \wedge (w \vee k))$ bzw. $gz \leftrightarrow (\bar{z} \wedge (\bar{w} \vee \bar{k}))$
- Bauer allein: $\alpha_1 = \langle b \wedge \bar{ga}, \bar{b}\rangle$, $\alpha_2 = \langle \bar{b} \wedge \bar{gz}, b\rangle$
- Bauer mit Objekt:
  $\alpha_3 = \langle b \wedge w \wedge (\bar{w} \rightarrow \bar{ga}), \bar{b} \wedge \bar{w}\rangle$, $\alpha_4 = \langle \bar{b} \wedge \bar{w} \wedge (w \rightarrow \bar{gz}), b \wedge w\rangle$,
  $\alpha_5 = \langle b \wedge k \wedge (\bar{k} \rightarrow \bar{ga}), \bar{b} \wedge \bar{k}\rangle$, $\alpha_6 = \langle \bar{b} \wedge \bar{k} \wedge (k \rightarrow \bar{gz}), b \wedge k\rangle$,
  $\alpha_7 = \langle b \wedge z \wedge (\bar{z} \rightarrow \bar{ga}), \bar{b} \wedge \bar{z}\rangle$, $\alpha_8 = \langle \bar{b} \wedge \bar{z} \wedge (z \rightarrow \bar{gz}), b \wedge z\rangle$

Planen mit STRIPS

STRIPS [Fikes,Nilsson, 1971]:

- Beschreibungssprache: Datenlogik (Prädikatenlogik ohne Funktionszeichen)
- Zustand: Menge von variablenfreien Atomformeln mit closed world assumption: wenn eine Atomformel nicht angegeben ist, gilt ihre Negation

  Beispiel: $\{ontable(b), ontable(c), on(a,b), clear(a), clear(c)\}$
- (Schematische) Operatoren (mit Variablen): 4-Tupel aus positiven Vorbedingungen, negativen Vorbedingungen, positiven Wirkungen, negativen Wirkungen

  Beispiel (Variablen grossgeschrieben):

$$move(X,Y,Z) = \langle\ \{clear(X), clear(Z), on(X,Y)\}, \emptyset,$$
$$\{on(X,Z), clear(Y)\},$$
$$\{on(X,Y), clear(Z)\}\rangle$$

Planen mit STRIPS (2)

Weitere Eigenschaften von STRIPS:

- Aktion: Grundinstanzierter Operator
- Plan: Folge von Aktionen
- Zusatzannahme: verschiedene Variablen bezeichnen verschiedene Objekte

Resultat $\alpha(s)$ einer Aktion $\alpha = \langle p, n, p', n' \rangle$ im Zustand s:

$\alpha(s) = (s \cup p') \setminus n'$, falls $p \subseteq s$ und $n \cap s = \emptyset$; sonst $\alpha(s) = s$

Beispiel:

$$s = \{ontable(b), ontable(c), on(a,b), clear(a), clear(c)\}$$

$$\begin{aligned}\alpha = \langle\ & \{clear(a), clear(c), on(a,b)\}, \emptyset, \\ & \{on(a,c), clear(b)\}, \\ & \{on(a,b), clear(c)\}\rangle\end{aligned}$$

$$\alpha(s) = \{ontable(b), ontable(c), \underline{on(a,c)}, clear(a), \underline{clear(b)}\}$$

Beispiel Planen mit STRIPS

$$
\begin{aligned}
move(X,Y,Z) = \quad & \langle \{clear(X), clear(Z), on(X,Y)\}, \emptyset, \\
& \{on(X,Z), clear(Y)\}, \\
& \{on(X,Y), clear(Z)\}\rangle \\
stack(X,Y) = \quad & \langle \{clear(X), clear(Y), ontable(X)\}, \emptyset, \\
& \{on(X,Y)\}, \{ontable(X), clear(Y)\}\rangle \\
unstack(X,Y) = \quad & \langle \{clear(X), on(X,Y)\}, \emptyset, \\
& \{ontable(X), clear(Y)\}, \{on(X,Y)\}\rangle
\end{aligned}
$$

Anfangszustand: $\{ontable(c), clear(c), ontable(b), on(a,b), clear(a)\}$

Zielzustand: $\{on(a,b), on(b,c)\}$

Plan: $unstack(a,b), stack(b,c), stack(a,c)$

Aussagenlogisches STRIPS

PSTRIPS = aussagenlogisches STRIPS:

- Beschreibungssprache eingeschränkt auf Aussagenlogik
- STRIPS kann polynomiell auf PSTRIPS reduziert werden, falls es nur polynomiell viele Instanzen der Operatoren gibt (gilt z.B. bei fester Operatorstelligkeit)
- Vorgehen: Volle Instanzierung der Operatoren und Abbildung jedes Grundatoms auf eine aussagenlogische Variable
- PSTRIPS kann polynomiell auf DPLAN abgebildet werden:
  - Jeder PSTRIPS-Zustand s ist eine Menge von Atomen;
    definiere entsprechenden DPLAN-Zustand: $s'(a) = \mathbf{w}$ gdw $a \in s$
  - Jeder PSTRIPS-Operator α ist ein 4-Tupel von Atomenmengen $\langle p, n, p', n' \rangle$;
    definiere entsprechenden DPLAN-Operator:
    $\alpha' = \langle \bigwedge p \wedge \neg \bigvee n, \bigwedge (p' \cup \{\neg a : a \in n'\} \rangle$

Schwierigkeit von Planungsproblemen

Planexistenzproblem:

Gegeben sei ein deterministisches Planungsproblemen $\Phi = \langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$

Gibt es einen Plan für Φ?

Existenz kurzer Pläne:

Gegeben sei ein deterministisches Planungsproblem $\Phi = \langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$ und eine natürliche Zahl n.

Gibt es einen Plan für Φ der Länge n oder kürzer?

Resultate:

- Das Planexistenzproblem ist PSPACE-vollständig.
- Das Problem der Existenz kurzer Pläne ist PSPACE-vollständig.
- Das Problem der Existenz kurzer Pläne, wobei n polynomiell in der Eingabegrösse, ist NP-vollständig.

Schwierigkeit von Planungsproblemen (2)

Beweis der PSPACE-Vollständigkeit des Planexistenzproblems:

- Man zeigt leicht, dass das Problem sich auf polynomiellem Platz nichtdeterministisch lösen lässt: analog der Lösung des Erreichbarkeitsproblems in Graphen exponentieller Grösse

- PSPACE-Härte: Entwicklung einer generischen Transformation, die jedes Problem X aus PSPACE und jede Instanz $x \in X$ in eine Instanz y eines Planexistenzproblems Y überführt, und zwar in polynomieller Zeit und derart, dass $x \in L(X)$ gdw ein Plan für y existiert.

Planen als Theorembeweisen

Grundidee:

- Übersetzung eines Planexistenz- oder Planfindungsproblems P in eine logische Formel F derart, dass P lösbar gdw F allgemeingültig ist.
- Verschiedene Ansätze bzgl. der logischen Zielsprache und der Art der Übersetzung

Übersetzung in Prädikatenlogik:

- Modellierung verschiedener Zustände durch Zustandsterme, die jedem Literal als zusätzliches Argument hinzugefügt werden müssen
- Modellierung von Zuständen als Konjunktionen von Grundliteralen mit demselben Zustandsterm
- Modellierung von Aktionen durch Konditionale, die die Erreichbarkeit des durch den Operator definierten Zustandes formulieren
- Schlussendliche Formel:
  Konjunktion der Aktionen und der Initialformel $\longrightarrow$ Zielformel

Planen als Theorembeweisen (2)

Beispiel für Übersetzung in Prädikatenlogik aus STRIPS:

- STRIPS-Zustand $s = \{ontable(b), ontable(c), on(a,b), clear(a), clear(c)\}$
  Entsprechende Zustandsformel: $F(S)$ (Variablen grossgeschrieben)
  $ontable(b,S) \wedge ontable(c,S) \wedge on(a,b,S) \wedge clear(a,S) \wedge clear(c,S) \wedge$
  $\neg ontable(a,S)$ (u.U. Explizitmachung der closed world assumption)
- STRIPS-Operator:

  $$move(X,Y,Z) = \langle\ \{clear(X), clear(Z), on(X,Y)\}, \emptyset,$$
  $$\{on(X,Z), clear(Y)\}, \{on(X,Y), clear(Z)\}\ \rangle$$

  Entsprechende Operatorformel:

  $$\forall XYZS(\ (clear(X,S) \wedge clear(Z,S) \wedge on(X,Y,S) \wedge$$
  $$S' = plan(move(X,Y,Z),S)) \rightarrow$$
  $$(on(X,Z,S') \wedge clear(Y,S') \wedge \neg on(X,Y,S') \wedge \neg clear(Z,S')))$$

Planen als Theorembeweisen (3)

Beispiel für Übersetzung in Prädikatenlogik aus STRIPS:

- Schlussendliche Formel Φ: Konjunktion der Initialzustandsformel(s_0) und der Operatorformeln $\rightarrow \exists S\ Zielzustandsformel(S)$

Rahmenproblem (Frame problem):

- Falls Plan existiert, gilt nicht, dass Φ allgemeingültig ist.
- Was fehlt, sind die sog. Rahmenaxiome
- Ein Rahmenaxiom muss besagen, dass alle Sachverhalte, die in einem Zustand s gelten und nicht von der entsprechenden Aktion berührt werden, auch im Folgezustand s' gelten.

Planen als Theorembeweisen (4)

Beispiel für Rahmenaxiome:

- Rahmenaxiome für $move(X, Y, Z)$:
  $ontable(b, S) \rightarrow ontable(b, plan(move(X, b, Z), S))$
  $ontable(c, S) \rightarrow ontable(c, plan(move(X, Y, c), S))$
  $\ldots$

Probleme der Rahmenaxiome:

- Prädikatenlogische Formel wächst stark an
- Länge des zu findenden Beweises nimmt stark zu

Konsequenz: Ansatz in der Praxis nicht erfolgreich

Lösungsansatz: Verwendung von Ressourcen-Logiken (analog Lineare Logik) (Bibel, Fronhöfer, Hölldobler, et.al.)

Planen als Theorembeweisen (5)

Weiterer Ansatz zur Vermeidung von Rahmenaxiomen:

- Explizite Darstellung eines Weltzustandes in einer Atomformel:
- Explizite Modellierung der Erreichbarkeitsrelation durch Angabe von Konditionalformeln zwischen Atomformeln

Realisierung für aussagenlogische Plansprache:

- Seien $p_1, \ldots, p_n$ die Aussagenvariablen des Planungsproblems
- Jeder Zustand (Belegung) kann durch ein n-Tupel über 0,1 beschrieben werden, d.h. als Bitvektor
- Jede Aktion kann beschrieben werden durch eine Formel der Form:
  $E(b_1, \ldots, b_n) \rightarrow E(b'_1, \ldots, b'_n)$
- Durch die Verwendung von Variablen kann
  a) das Rahmenproblem vermieden werden
  b) und Operatoren schematisch dargestellt werden

Planen als Theorembeweisen (6)

Modellierung des Transport-Beispiels (in Prolog-Format):

```
<- situation(r,r,r,r).

situation(l,l,l,l).

situation(B1,K,W,Z) <- different(B,B1), situation(B,K,W,Z),
                       save(B1,K,W,Z).

situation(B1,K1,W1,Z1) <- different(B,B1), one_different(K,W,Z,K1,W1,Z1),
                          situation(B,K,W,Z), save(B1,K1,W1,Z1).

different(l,r). different(r,l).

one_different(K,W,Z,K1,W,Z) <- different(K,K1).
one_different(K,W,Z,K,W1,Z) <- different(W,W1).
one_different(K,W,Z,K,W,Z1) <- different(Z,Z1).

save(X,Y,Z,X). save(X,X,X,Y).
```

Planen als Theorembeweisen (7)

Zusammenfassung des Ansatzes:

- U.U. werden Planungsprobleme in logische Sprachen übersetzt, deren Entscheidungsproblem schwerer ist als das gegebene Planungsproblem
- erster vorgestellter Ansatz: Sprache i.A. unentscheidbar
- Bit-Vektor-Ansatz: i.A. coNEXPTIME-vollständig

Planen als aussagenlogische Erfüllbarkeit

Grundidee:

- Übersetzung eines Planexistenz- oder Planfindungsproblems P in eine aussagenlogische Formel F derart, dass P lösbar gdw F erfüllbar ist.
- Grundsätzliche Beschränkung: Übersetzung kann nicht polynomiell sein, wenn NP $\neq$ PSPACE

Vorgehen:

- Lösung von Planproblemen mit polynomieller Anzahl von Aktionen
- Finden von kürzesten Plänen: Binäre Suche über Planlänge

Planen als aussagenlogische Erfüllbarkeit (2)

Übersetzung in aussagenlogische Erfüllbarkeit:

- Jeweils Übersetzung eines Planproblems mit fester Aktionszahl $n-1$
- Logische Repräsentation verschiedener Weltzustände durch Ver$n$fachung der aussagenlogischen Variablen p^j zu $p_1^j, \ldots, p_n^j$
- Formulierung des Effektes einer Aktion $\alpha^j = \langle c^j, e^j \rangle$ $(1 \leq j \leq m)$ durch $n-1$ Formeln:

$$\alpha_i^j \leftrightarrow (e_{i+1}^j \wedge \bigwedge \{(p_i \leftrightarrow p_{i+1}) : p_{i+1} \text{ kommt nicht in } e_{i+1}^j \text{ vor}\}) \quad (1 \leq i < n)$$

- Kombination der Effekte durch Disjunktion aller α_i^j mit demselben unteren Index:

$$\alpha_i^1 \vee \cdots \vee \alpha_i^m \quad (1 \leq i < n)$$

- Notwendigkeit der Vorbedingung für einen Effekt:

$$\alpha_i^j \rightarrow c_i^j \quad (1 \leq i < n)$$

  bzw.: falls k äquivalente Effekte existieren, k-fache Disjunktion ihrer Vorbedingungen im Konsequent

Wahrheitstafelmethode

Aufgabe: Entscheidung der Erfüllbarkeit einer Formel

- Am Beispiel (es_regnet $\rightarrow$ strasse_nass) $\wedge \neg$ strasse_nass
- Zur Abkürzung: r $\hat{=}$ es_regnet, n $\hat{=}$ strasse_nass
- Erstellen einer Tafel mit allen Belegungen
- „Auswerten“ der Formel für jede Belegung

r	n	(	r	$\rightarrow$	n	)	$\wedge$	$\neg$	n
w	**w**		**w**	**w**	**w**		**f**	**f**	**w**
w	**f**		**w**	**f**	**f**		**f**	**w**	**f**
f	**w**		**f**	**w**	**w**		**f**	**f**	**w**
f	**f**		**f**	**w**	**f**		**w**	**w**	**f**

Effizienz des Verfahrens

- Jeder einzelne Auswertungsschritt ist linear (Zeit) in der Grösse von F
- Die Anzahl der Zeilen ist exponentiell in der Anzahl der Variablen von F
- Platzbedarf: naiv exponentiell, linear bei Wiederverwendung von Zeilen

Formeln in Klauselform

Definition der Klauselform

- Literal := a | $\neg$ a (a $\in \mathcal{P}$)
- Klausel := $\bot$ | $l \vee c$ (l Literal, c Klausel)
- Klauselformel := $\top$ | $c \wedge F$ (c Klausel, F Klauselformel)

Mengennotation für Klauselformeln

- Nutzt Idempotenz, Kommutativität und Assoziativität von $\vee$ und $\wedge$
- Klausel c als „disjunktive Menge“ von Literalen:
  $\iota(c) = \mathbf{w}$ gdw es ein Literal $l \in c$ gibt mit $\iota(l) = \mathbf{w}$
- Klauselformel F als Menge von Klauseln:
  $\iota(F) = \mathbf{w}$ gdw für alle Klauseln $c \in F$ gilt: $\iota(c) = \mathbf{w}$

Satz: Zu jeder aussagenlogischen Formel gibt es eine äquivalente Formel in Klauselform.

Semantische Bäume

- Optimierung der Wahrheitstafelmethode
- Schrittweise Partitionierung der Belegungsmenge (Äste)
- Typischerweise für Klauselformeln

Definition Semantischer Baum für Klauselmenge S:

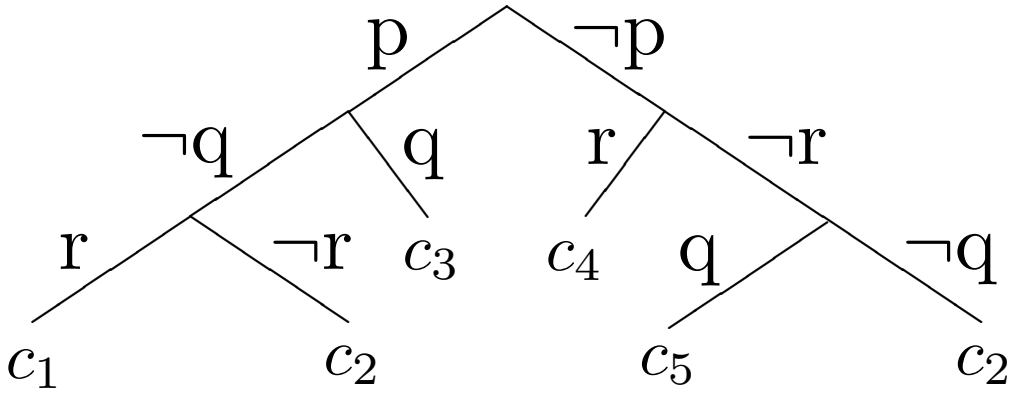
Binärbaum, dessen Kanten mit Literalen und dessen Blattknoten möglicherweise mit Klauseln aus S markiert sind wie folgt:

- Jedes Paar von Kanten, die aus einem gemeinsamen Knoten entspringen, ist mit einer Variablen p, die in S vorkommt, bzw. mit $\neg p$ markiert, wobei p nicht bereits auf dem Ast vorkommt.
- jeder Blattknoten kann mit einer Klausel $c \in S$ markiert sein, falls die *Komplemente* aller Literale in c auf dem Ast vorkommen.

Eine Semantischer Baum für S heisst *geschlossen*, falls jeder Blattknoten mit einer Klausel (aus S) markiert ist.

Semantische Bäume (2)

Beispiel: Ein Geschlossener semantischer Baum für die Klauselmenge $\{c_1, c_2, c_3, c_4, c_5\}$



c_1: q $\vee$ $\neg$r
c_2: q $\vee$ r
c_3: $\neg$p $\vee$ $\neg$q
c_4: p $\vee$ $\neg$ r
c_5: p $\vee$ $\neg$q $\vee$ r

Satz: Eine Klauselmenge ist unerfüllbar gdw es einen geschlossenen semantischen Baum für sie gibt.

Beweis: Einfach über die Partitionierung der Belegungsmenge und die Semantik von Klauselmengen

Semantische Bäume (3)

Gesättigter semantischer Baum für S: kein Erweiterungsschritt mehr möglich, d.h. jeder Ast ist entweder durch eine Klausel aus S geschlossen oder er hat maximale Länge und kann nicht durch eine Klausel aus S geschlossen werden

Modelle: Wenn eine Formelmenge S erfüllbar ist, dann gilt für jeden gesättigten semantische Baum T für S folgendes: Die Literalmengen M auf den offenen Ästen von T entsprechen genau den Modellen von S; genauer: für alle M: $l \in M$ gdw $\iota(l) = \mathbf{w}$.

Bezug zur Wahrheitstafelmethode:

- Äste entsprechen Modellen bzw. Modellmengen
- Vorteil gegenüber Wahrheitstafelmethode: Geschlossene Äste müssen nicht maximale Länge haben

Semantische Bäume (4)

Teilbelegung ι

Durch Astliterale M gegebene partielle Belegung wie folgt:

$$\iota(p) = \begin{cases} \mathbf{w} & \text{falls } p \in M \\ \mathbf{f} & \text{falls } \neg p \in M \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Partielle Evaluierung von S und Anwendung der Absorptionsregeln für $\bot$ und $\top$

Gegeben durch Astliterale M definierte Teilbelegung $\iota : V \to \{\mathbf{w}, \mathbf{f}\}$:

- Ersetzung aller Variablen $p \in V$ durch $\top$ bzw. $\bot$, wenn $\iota(p) = \mathbf{w}$ bzw. $\mathbf{f}$
- Löschung aller Klauseln, die $\top$ oder $\neg\bot$ enthalten
- Löschung aller Literale $\bot$ und $\neg\top$ aus den Klauseln
- Resultierende Klauselmenge bezeichnen wir mit $M(S)$
- Ast M kann abgeschlossen werden, wenn $M(S)$ die leere Klausel enthält

Semantische Bäume (5)

Möglichkeit der effizienten Implementation

- Baumabarbeitung mittels Tiefenverfahren und Backtracking
- Deaktivierung/Reaktivierung von wahren Klauseln, d.h. $M(c) = \top$
- Dekrementierung/Inkrementierung von Zählern $\ell(c)$ an Klauseln c mit $M(c) \neq \top$: $\ell(c) = |M(c)|$

Effizienz des Verfahrens (im worst case):

- Linearer Platzbedarf
- Jeder Expansionschritt/Kontraktionsschritt hat linearen Zeitbedarf
- Anzahl der Schritte: exponentiell

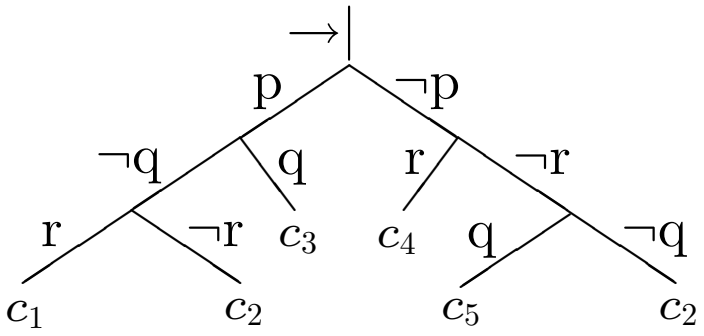
Man beachte:

- Variablenfolge auf Ast bestimmt die Grösse des semantischen Baums
- Bei verschiedenen geschlossenen semantischen Bäumen für dieselbe Klauselmenge u.U. exponentielle Grössenunterschiede

Abarbeitung eines Semantischen Baums

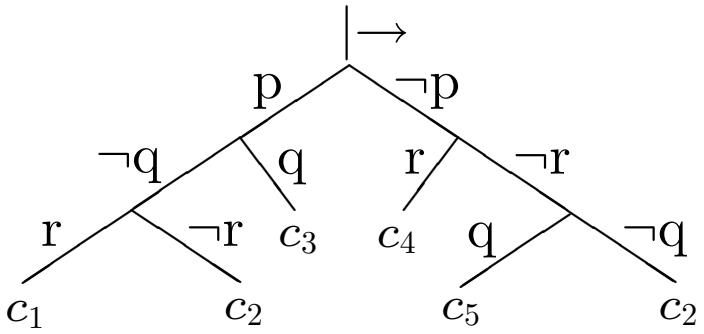
Astliterale	$\{q,\neg r\}$	$\{q,r\}$	$\{\neg p,\neg q\}$	$\{p,\neg r\}$	$\{p,\neg q,r\}$	Aktion
	2	2	2	2	3	Start
p^l	2	2	1	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l,r^l$	0	$\top$	$\top$	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Uparrow l$
$p^l,\neg q^l,\neg r^r$	$\top$	0	$\top$	$\top$	$\top$	$\Downarrow r$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Uparrow r$
p^l	2	2	1	$\top$	$\top$	$\Uparrow l$
p^l,q^r	$\top$	$\top$	0	$\top$	$\top$	$\Downarrow r$
p^l	2	2	1	$\top$	$\top$	$\Uparrow r$

...



Abarbeitung eines Semantischen Baums (2)

Astliterale	$\{q,\neg r\}$	$\{q,r\}$	$\{\neg p,\neg q\}$	$\{p,\neg r\}$	$\{p,\neg q,r\}$	Aktion
...						
p^l	2	2	1	$\top$	$\top$	$\Uparrow r$
$\neg p^r$	2	2	$\top$	1	2	$\Downarrow r$
$\neg p^r, r^l$	1	$\top$	$\top$	0	$\top$	$\Downarrow l$
$\neg p^r$	2	2	$\top$	1	2	$\Uparrow l$
$\neg p^r, \neg r^r$	$\top$	1	$\top$	$\top$	1	$\Downarrow r$
$\neg p^r, \neg r^r, q^l$	$\top$	$\top$	$\top$	$\top$	0	$\Downarrow l$
$\neg p^r, \neg r^r$	$\top$	1	$\top$	$\top$	1	$\Uparrow l$
$\neg p^r, \neg r^r, \neg q^r$	$\top$	0	$\top$	$\top$	$\top$	$\Downarrow r$
	2	2	2	2	3	$\Uparrow r \Uparrow r \Uparrow r$



Resolution

Resolution basiert auf der Mengendarstellung von Klauselformeln

- Kalkül zur Herleitung von Klauseln (*Resolventen*) aus Paaren von Klauseln (*Elternklauseln*)
- Seien c_1, c_2 Literalmengen und a eine aussagenlogische Variable, die sog. *Resolventenvariable*.

Resolutionsregel:
$$\frac{c_1 \cup \{a\} \qquad \{\neg a\} \cup c_2}{(c_1 \setminus \{a\}) \cup (c_2 \setminus \{\neg a\})}$$

Beispiel:
$$\frac{a \vee b \vee c \qquad a \vee \neg b \vee \neg d}{a \vee c \vee d} \qquad \frac{\{a, b, c\} \qquad \{a, \neg b, \neg d\}}{\{a, c, d\}}$$

Man beachte das durch die Mengendarstellung bedingte „Merging“.

Korrektheit der Resolution: Jede Resolvente wird von der Menge ihrer Elternklauseln impliziert.

Resolution (2)

Resolutionsherleitung einer Klausel c_n aus einer Menge von Klauseln S:
Folge $D = c_1, \ldots, c_n$ von Klauseln derart, dass für jede Klausel c_k in D gilt:

- $c_k \in S$ oder
- es gibt Klauseln c_i, c_j in D mit $i, j < k$ und c_k ist Resolvente von c_i, c_j.

Resolutionswiderlegung von S:
Resolutionsherleitung der *leeren Klausel* $\emptyset$ aus S

Beispiel: $S = \{\{\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{n}\}\}$.

Eine Resolutionswiderlegung von S: $\{\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{n}\}, \{\mathsf{n}\}, \emptyset$

(Widerlegungs-)Vollständigkeit der Resolution: Zu jeder unerfüllbaren Klauselmenge S gibt es eine Resolutionswiderlegung.

Man beachte, dass nicht jede Klausel c, die von einer Klauselmenge impliziert wird, auch mittels Resolution hergeleitet werden kann.

Resolution (3)

Resolution als Entscheidungsverfahren

Gegeben eine aussagenlogische Formel F und die Frage, ob F erfüllbar ist:

- Transformiere F in eine (erfüllbarkeits-)äquivalente Klauselmenge S.
- Bilde die Menge S^* aller mittels Resolution aus S herzuleitenden Klauseln, die sog. Sättigungsmenge von S.
- $\emptyset \notin S^*$ gdw S erfüllbar ist.
- Beachte: Im Erfüllbarkeitsfall liefert das Verfahren aber kein Modell.

Effizienz des Verfahrens: S^* ist im worst-case exponentiell in S.

Vollständigkeitserhaltende Optimierungen

- Subsumptionslöschung: Lösche jede *subsumierte* Klausel, d.h. jede Klausel, die eine echte Obermenge einer hergeleiteten Klausel ist.
- Tautologielöschung: Lösche jede *tautologische* Klausel, d.h. jede Klausel, die eine aussagenlogische Variable und ihre Negation enthält.

Geordnete Resolution

Direkte Einschränkungen der Resolventenbildung

Sei $\prec$ eine totale Ordnung auf den Variablen.

- Eine Resolventenbildung erfolgt nur, wenn die Resolventenvariable maximal in beiden Elternklauseln ist.
- Geordnete Resolution ist für jede Ordnung widerlegungsvollständig.
- I.A. ist die Sättigungsmenge S^* aber immer noch exponentiell in S.

Beispiel für Effizienzgewinn durch geordnete Resolution

Gegeben eine Zahl n, sei $S = \{\{\neg a_i, a_{i+1}\} : 1 \leq i \leq n\}$.

- Bei normaler Resolution ist $S^* = \{\{\neg a_i, a_j\} : 1 \leq i < j \leq n+1\}$.
- Bei geordneter Resolution und $a_i \prec a_{i+1}$, ist $S^* = S$.

Kalkülvergleich über Minimale Beweislängen

Problem der geordneten Resolution:

- Es gibt bestimmte Formelklassen mit kurzen (i.e. polynomiellen) Resolutionswiderlegungen aber nur superpolynomiell (z.B. exponentiell) langen geordneten Resolutionswiderlegungen.
- Man sagt dann: Die geordnete Resolution kann die normale Resolution nicht *polynomiell simulieren*.

Polynomielle Simulation von Kalkülen

Ein Logikkalkül K_1 *simuliert* einen Logikkalkül K_2 *polynomiell* gdw wenn es ein Polynom p gibt derart, dass folgendes gilt: Für jeden Beweis D_2 der Allgemeingültigkeit bzw. Unerfüllbarkeit einer Formel im Kalkül K_2 existiert ein Beweis D_1 der Allgemeingültigkeit bzw. Unerfüllbarkeit der Formel in K_1 und $p(|D_2|) < |D_1|$. ($|D|$ bezeichne die Länge eines Beweises D)

Normalerweise ist man sogar an *direkter* ($p(n) = c_1 + n$) oder *linearer* ($p(n) = c_1 + c_2 \times n$) Simulierbarkeit interessiert. (c_i feste Konstanten)

Minimale Beweislängen der vollen Resolution

Satz: Es gibt bestimmte unerfüllbare Klassen von Klauselformeln, für die keine polynomiellen Resolutionswiderlegungen existieren. [A. Haken, 1985]

- Betrachtete Formelklasse formuliert das sog. „pigeonhole"-Prinzip

Schubfach-Prinzip:

In n Schubfächer passen keine $n+1$ Objekte, wenn in jedes Schubfach nur ein Objekt passt.

Aussagenlogische Formalisierung des Schubfach-Prinzips für festes n

- p_j^i bedeute: Objekt i ist in Schubfach j
- In jedes Schubfach passt nur ein Objekt:
  $S_1 = \{\neg p_j^i \vee \neg p_j^k : 1 \leq j \leq n, 1 \leq i < k \leq n+1\}$
- Jedes Objekt ist in einem Schubfach:
  $S_2 = \{p_1^i \vee \cdots \vee p_n^i : 1 \leq i \leq n+1\}$
- $S_1 \cup S_2$ hat keine Resolutionswiderlegung polynomieller Länge.
- In anderen Kalkülen haben die Schubfach-Formeln polynomielle Beweise.

Die Lineare Resolution

Einschränkung der Struktur einer Resolutionsherleitung
Lineare Resolutionsherleitung einer Klausel c_n aus einer Klauselmenge S:
Folge $D = c_1, \ldots, c_n$ von Klauseln derart, dass für jede Klausel c_k in D gilt:

- $c_k \in S$ oder
- es gibt Klausel c_i in D mit $i < k$ und c_k ist Resolvente von c_i, c_{k-1}.

Beispiel: $S = \{\{\mathsf{p},\mathsf{q}\}, \{\mathsf{p},\neg\mathsf{q}\}, \{\neg\mathsf{p},\mathsf{q}\}, \{\neg\mathsf{p},\neg\mathsf{q}\}\}$.

Lineare Resolutionswiderlegung von S (ab $i = 5$): $\{\neg\mathsf{p}\}, \{\mathsf{q}\}, \{\mathsf{p}\}, \emptyset$

Satz: Die lineare Resolution ist widerlegungsvollständig.

Aber: Die lineare Resolution ist völlig ungeeignet als Basis eines Entscheidungsverfahrens. Warum?

Offenes Problem: Kann die lineare Resolution die volle Resolution polynomiell simulieren?

Die Input-Resolution

Eine Einschränkung der linearen Resolution

Eingabe- oder *Input-Resolutionsherleitung*: Folge $D = c_1, \ldots, c_n$ von Klauseln derart, dass für jede Klausel c_k in D gilt: $c_k \in S$ (Eingabemenge) oder es gibt ein c_i in D mit $i < k$ und c_k ist Resolvente von c_i, c_{k-1} und $c_i \in S$.

Satz: Die Input-Resolution ist widerlegungsvollständig für die *Hornklausellogik*, aber nicht für die volle Klausellogik.

Hornklausellogik: Jede Klausel hat maximal ein positives Literal.

Gegenbeispiel für Nicht-Horn-Fall: $\{\{\mathsf{p},\mathsf{q}\}, \{\mathsf{p},\neg\mathsf{q}\}, \{\{\neg\mathsf{p},\mathsf{q}\}, \{\neg\mathsf{p},\neg\mathsf{q}\}\}$.

Satz: Die aussagenlogische Hornklausellogik ist linear entscheidbar.

Bemerkung zur Eingabegrösse: Als *realistische Grösse* einer Klauselmenge S nehme man z.B. $g = \Sigma_{c \in S}|c|$ oder evtl. $g \log g$, da die Annahme eines unendlichen Alphabets für die Variablen unrealistisch ist.

Satz: Die aussagenlogische Hornklausellogik ist P-vollständig.

Die Unit-Resolution

Einschränkung der Resolutionsregel, nicht der Herleitung:

- *Einer-* oder *Unit-Klausel*: Klausel mit genau einem Literal
- *Einer-* oder *Unit-Resolutionsregel*: eine der Elternklauseln ist Einerklausel

Satz: Die Unit-Resolution ist widerlegungsvollständig für die Hornklausellogik, aber nicht für die volle Klausellogik.

Genauer: Klauselmenge ist Input-widerlegbar gdw sie Unit-widerlegbar ist.

Beachte: Im Gegensatz zur Input-Resolution ist die Unit-Resolution als Basis eines Entscheidungsverfahrens für die aussagenlogische Hornklausellogik geeignet.

Wichtige Eigenschaft der Unit-Resolution: Jede Unit-Resolvente subsumiert eine ihrer Elternklauseln c_i.

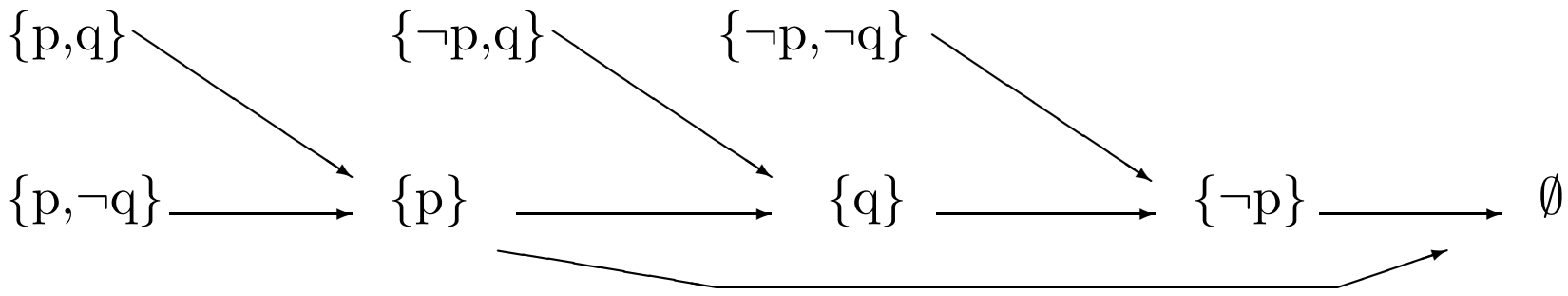
$\Rightarrow$ c_i kann damit gelöscht werden oder besser: das Resolventenliteral destruktiv aus c_i entfernt werden, ohne eine neue Klausel zu bilden.

Resolutions-Graphen

Resolutionsgraph = Graphische Darstellung einer Herleitung:
Resolutionsherleitung als *gerichteter azyklischer Graph (DAG)* $\langle V, E \rangle$:

- Jeder Knoten $v \in V$ ist mit einer Klausel $c(v)$ markiert und
- $c(v) \in S$ (Eingabemenge) oder v hat Vorgänger v_1, v_2 derart, dass $c(v)$ Resolvente von $c(v_1)$ und $c(v_2)$

Beispiel: Lineare Resolutionsherleitung als Graph



Die Baum-Resolution

Baum-Resolution: Resolutionsgraph ist ein Baum

- Bedeutet für Resolutionsherleitung $D = c_1, \ldots, c_n$: Jedes Klauselvorkommnis c_i in D, das keine Eingabeklausel ist, kann lediglich einmal als Elternklausel verwendet werden („Aufbrauchen“ von Resolventen)
- Von beliebigem Resolutionsgraph zu Resolutionsbaum: Rekursives Kopieren der Untergraphen, bis Graph ein Baum ist
- Als Resolutionsverfeinerung ist die Baum-Resolution unsinnig, aber als Vergleichsformat für andere Verfahren (z.B. semantische Bäume) sehr sinnvoll

Satz: Die Baum-Resolution kann die volle Resolution nicht polynomiell simulieren.

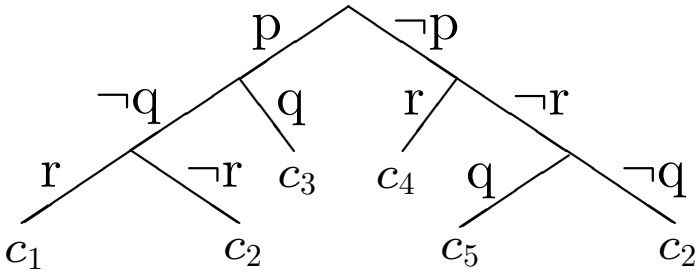
Satz: Die lineare Resolution kann die Baum-Resolution linear simulieren.

Semantische Bäume und Resolution

Satz: Zu jedem geschlossenen semantischen Baum einer Klauselmenge S gibt es eine Baum-Resolutionswiderlegung gleicher oder kleinerer Baumgrösse.

Beweis: Umformung eines geschlossenen semantischen Baums minimaler Grösse in einen Resolutionsbaum

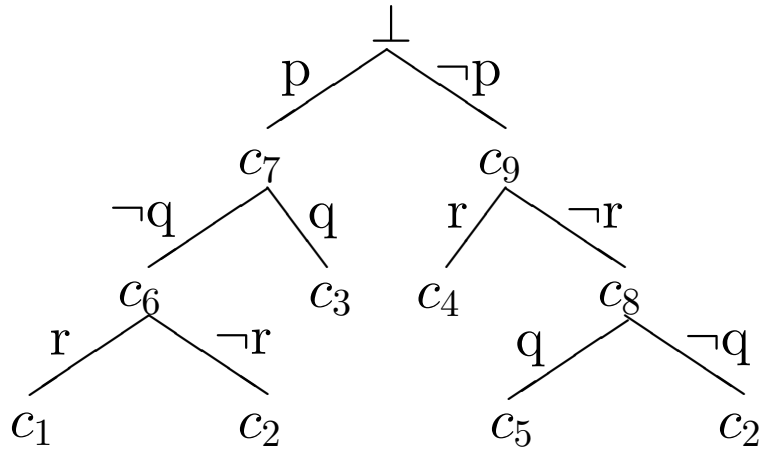
Beispiel:



c_1: q $\vee$ $\neg$r
c_2: q $\vee$ r
c_3: $\neg$p $\vee$ $\neg$q
c_4: p $\vee$ $\neg$ r
c_5: p $\vee$ $\neg$q $\vee$ r

Resolventen:

c_6: q
c_7: $\neg$p
c_8: p $\vee$ r
c_9: p



Semantische Bäume und Resolution (2)

Korollare:

- Die Resolution kann den Kalkül der semantischen Bäume quadratisch simulieren.
- Der Kalkül der semantischen Bäume kann die Resolution nicht polynomiell simulieren.

Aber: Im average case ist der Kalkül der semantischen Bäume der Resolution überlegen

- Linearer Platzbedarf
- Effizientere Kalkülschritte
- Bessere Steuerung der Beweisfindung durch „lineare“ Strategien und heuristisch gesteuerte Variablen- und Zeichenauswahl

Steuerung Semantischer Baum-Verfahren

Lineare Strategien:

- Unit propagation: Verzweige nach Atom p, falls eine Unitklausel p oder ¬p existiert, und schliesse den ¬p- bzw. den p-Ast.
- Purity propagation: Verzweige nach Atom p, das nur in einer Polarität, p bzw. ¬p, vorkommt, und schliesse den ¬p- bzw. den p-Ast.

Zwei Formen des (don't care) Indeterminismus: Heuristisch gesteuert

- Variablenauswahl: welche Variable als nächstes?
- Zeichenauswahl: in welchen Ast zuerst, p oder ¬p?
- Nach Häufigkeit des Vorkommens
- In kurzen Klauseln
- Nach verschiedenen (z.t. magischen) Kriterien

Implementation Semantischer Baum-Verfahren

Datenstrukturen

1. Atomliste: Liste von Atomen.

2. Atom:

Nil / + / −	Pos-Klauseln(Atom)	Neg-Klauseln(Atom)

3. Klausel:

Nil / Atom	Aktuelle Länge	Pos-Atome(Klausel)	Neg-Atome(Klausel)

4. Literal := Atom | ¬ Atom

5. Interpretation: Feld von Atomen

Implementation Semantischer Baum-Verfahren (2)

```
Prozedur: verzweige:
if    unevaluiertes-atom-in(Atomliste):
      Atom := wähle-Atom(Atomliste); Literal := wähle-Literal(Atom);
      widerlege(Literal); widerlege(∼Literal)
else  print(Erfüllbar); break

Prozedur: widerlege(Literal):
Atom := atom(Literal);
if    Literal = Atom:
      Klauseln := Neg-Klauseln(Atom); True-Klauseln := Pos-Klauseln(Atom)
else  Klauseln := Pos-Klauseln(Atom); True-Klauseln := Neg-Klauseln(Atom);
dekrementiere-zähler(Klauseln);
if    not klausel-der-Länge-0-in(Klauseln):
      markiere-als-evaluiert(Atom,Literal); deaktiviere(True-Klauseln,Atom);
      verzweige;
      markiere-als-unevaluiert(Atom); reaktiviere(True-Klauseln,Atom);
inkrementiere-zähler(Klauseln)
```

Implementation Semantischer Baum-Verfahren (3)

Weitere Datenstrukturen zur Effizienzsteigerung:

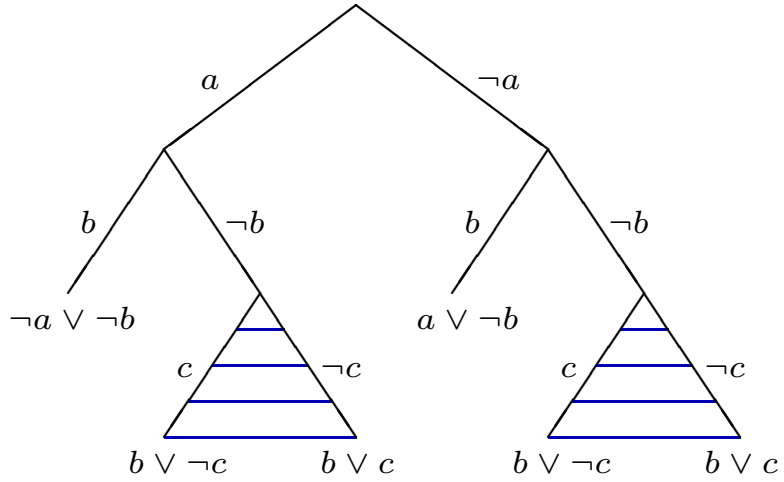
- Liste der Klauseln mit aktueller Länge 1
- Aktuelle Häufigkeit der Literale (Problem: effiziente dynamische Aktualisierung)

Neuer Ansatz zur deutlichen Effizienzsteigerung (System CHAFF)

- Keinerlei Dekrementierung der Klausellänge
- Stattdessen Auszeichnung zweier Wächterliterale pro Klausel
- Reduktion der Listen Pos-Klauseln(Atom), Neg-Klauseln(Atom) auf Klauseln, in denen $\sim$Atom Wächterliteral (Reduktion von $O(\Sigma_{c \in S}|c|)$ auf $O(|S|)$)
- Neues Vorgehen, wenn l Blattliteral und $\sim l$ Wächterliteral in Klausel c:
  - Wenn c ohne Wächter unevaluiertes Literal k enthält: k neuer Wächter von c
  - Andernfalls hat c eine aktuelle Länge ≤ 1
- Nachteil: Literalgewichte nicht aktuell, damit Einschränkung der Möglichkeiten heuristischer Suchraumreduktion

Das Duplikationsproblem in Semantischen Bäumen

Das Problem des Mehrfachvorkommens gleicher Unterbäume:

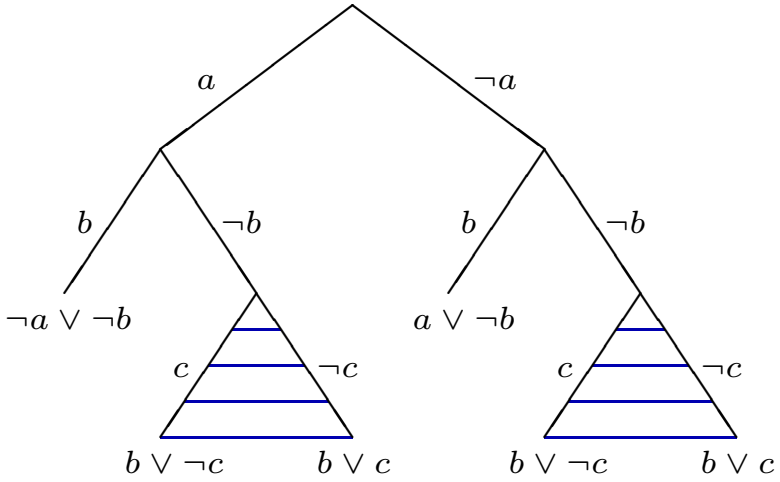


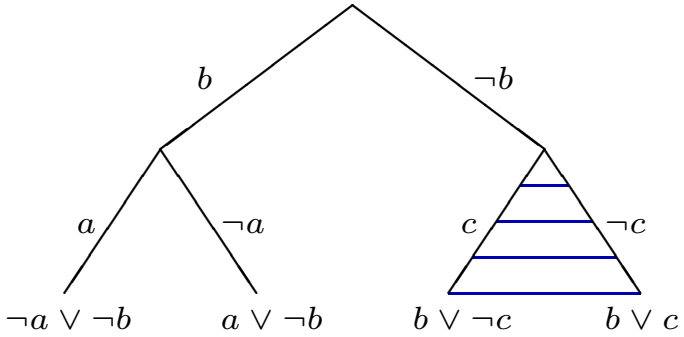
Lösungsansätze:

- Änderung der Variablenordnung
- Wiederverwendung von Unterbäumen

Das Duplikationsproblem in Semantischen Bäumen (2)

Änderung der Variablenselektion:





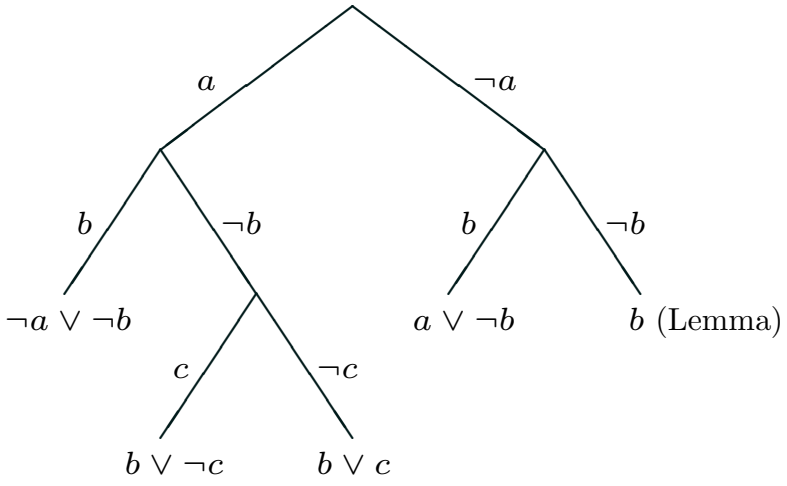
- Aber wie findet man die beste Variablenselektion?
  Dieses Problem ist NP-hart.
- Ausserdem gilt: Doppelvorkommnisse lassen sich i.A. nicht vermeiden, denn minimale semantische Bäume entsprechen Resolutionsbäumen

Semantische Bäume mit Lemmas

Erweiterung des Kalküls der semantischen Bäume:

- Versuch der Simulation der vollen Resolution
- Durch Extraktion von Resolventen (Lemmas) aus gelösten Unterbäumen

Semantischer Baum



Klauseln

$\neg a \vee \neg b$

$b \vee \neg c$

$b \vee c$

$a \vee \neg b$

b (Lemma)

$\neg a$ (Lemma)

Binäre Entscheidungsdiagramme (BDDs)

- Effizienter Formalismus zur Manipulation Boolescher Funktionen und zur möglichst kompakten Darstellung endlicher Mengen
- Jede aussagenlogische Formel F, die die Aussagenvariablen $p_1, \ldots, p_n$ enthält, kann aufgefasst werden als n-stellige Boolesche Funktion von Belegungen der $p_1, \ldots, p_n$ auf die Menge der Wahrheitswerte
- Anforderungen an Repräsentationsformalismus:
  - Möglichst effiziente Manipulation (z.B. Vereinigung, Schnitt, Komplement) der Repräsentationsobjekte
  - Möglichst einfacher Vergleich zweier Repräsentationsobjekte, die dieselbe Menge darstellen
- Beliebige aussagenlogische Formeln sind zu allgemein für effiziente Manipulationen und Vergleich
- Erfolgreicher Kompromiß (auch in der Praxis des Model Checking):
  Binäre geordnete Entscheidungsdiagramme (OBDDs)

Binäre Entscheidungsdiagramme (BDDs)

Grundlage: logisches Konnektiv Ite (if then else)

- Definition: $\mathsf{Ite}(A, B, C)$ gdw $((A \rightarrow B) \wedge (\neg A \rightarrow C))$
- Es gilt weiterhin: $\mathsf{Ite}(A, B, C) \equiv ((A \wedge B) \vee (\neg A \wedge C))$
- Da $(A \rightarrow B) \equiv \mathsf{Ite}(A, B, \top)$, ist die Junktorenmenge $\{\mathsf{Ite}, \top, \bot\}$ aussagenlogisch vollständig.
- Gegeben eine Menge $\mathcal{P}$ aussagenlogischer Variablen.
  Die Menge der *BDD-Formeln (in Baum-Form)* für $\mathcal{P}$ ist wie folgt definiert:

$$BDD(\mathcal{P}) := \quad \bot \quad | \quad \top \quad | \quad \mathsf{Ite}(\mathcal{P}, BDD, BDD)$$

- Jede BDD-Formel F kann als *binärer Entscheidungsbaum* bzw. als *binärer Entscheidungsgraph* dargestellt werden.

Binäre Entscheidungsdiagramme (BDDs)

Grundlage: Binäre Entscheidungsbäume

- Jede BDD-Formel F kann als *binärer Entscheidungsbaum* dargestellt werden, der wie folgt definiert ist.
- Jeder Teilformel der Form $\mathsf{Ite}(p, B, C)$ von F entspricht ein markierter Binärbaum:
  - dessen Wurzel ist mit p markiert,
  - von ihr gehen zwei Kanten aus, eine ist mit 1 markiert (die zum Baum für B führt) und eine mit 0 (die zum Baum für C führt),
  - der Baum für $\top$ und $\bot$ ist ein Knoten, der mit 1 bzw. 0 markiert ist.
- Der so beschriebene Baum ist *der Entscheidungsbaum* der BDD-Formel.

Binäre Entscheidungsdiagramme (BDDs)

Geordnete reduzierte BDD-Formeln und binäre Entscheidungsbäume

- Sei $<$ eine Ordnung auf den aussagenlogischen Variablen $\mathcal{P}$.
- Eine BDD-Formel F ist in *geordneter Form (bzgl. $<$)* oder ist eine $OBDD$-Formel gdw für jede Teilformel $\mathsf{Ite}(p, B, C)$ von F und jede Teilformel $\mathsf{Ite}(q, B', C')$ von B und C gilt: $p < q$.
- Eine BDD-Formel F ist in *reduzierter Form* gdw wenn sie keine Teilformel der Form $\mathsf{Ite}(p, B, B)$ enthält.
- Der einer $OBDD$-Formel in reduzierter Form (bzgl. $<$) entsprechende binäre Entscheidungsbaum heisst *geordneter reduzierter binärer Entscheidungsbaum (bzgl. $<$)*.

Satz: Zu jeder aussagenlogischen Formel über einer beliebigen vollständigen Junktorenmenge und jeder Ordnung $<$ auf den aussagenlogischen Variablen gibt es genau eine logisch äquivalente $OBDD$-Formel in reduzierter Form bzgl. $<$.

Binäre Entscheidungsdiagramme (BDDs)

Erzeugung geordneter reduzierter BDD-Formeln

- Hier für aussagenlogische Formeln über $\{\rightarrow, \top, \bot\}$
- Vorgehen: zunächst wiederholte Anwendung der sog. *Shannon-Expansion:*

$$A \equiv \mathsf{Ite}(p, A[p/\top], A[p/\bot])$$

  auf Teilformeln, die nicht in geordneter BDD-Form, wobei p die kleinste Variable bzgl. $<$ in A
- Dann Reduzierung: durch Boolesche Vereinfachungen wie:

$$\mathsf{Ite}(p, B, B) \equiv B \text{ oder } (\bot \rightarrow \top) \equiv \top$$

Binäre Entscheidungsdiagramme (BDDs)

„Sharing“ in BDD-Formeln und Entscheidungsbäumen

- In einer BDD-Formel kann dieselbe Teilformel mehrmals vorkommen.
- Diese Redundanz lässt sich durch Einführung von *Abkürzungen* (Namen für komplexe Teilformeln) beseitigen
- Eine BDD-*Formel mit Abkürzungen* ist eine BDD-Formel über einem erweiterten Variablenalphabet $\mathcal{P} \cup \{\delta_1, \ldots, \delta_n\}$ plus eine Liste von Abkürzungen der Form

$$\delta_i := \mathsf{Ite}(p, B, C),$$

  wobei $\mathsf{Ite}(p, B, C)$ eine BDD-Formel über $\mathcal{P} \cup \{\delta_{i+1}, \ldots, \delta_n\}$ ist.
- Der Einführung von Abkürzungen entspricht in der graphischen Darstellung der Übergang von Bäumen zu azyklischen gerichteten Graphen, sog. *binären Entscheidungsdiagrammen*.

Geordnete binäre Entscheidungsdiagramme (OBDDs)

- Ein binäres Entscheidungsdiagramm heisst *minimal* gdw kein Teilgraph mehr als einmal vorkommt und kein Knoten zwei gleiche Nachfolger hat.
- Zu jedem binären Entscheidungsbaum gibt es genau ein minimales binäres Entscheidungsdiagramm.

Satz: Sei $<$ eine Ordnung auf den aussagenlogischen Variablen und B die einer beliebigen aussagenlogischen Formel F und $<$ entsprechende logisch äquivalente reduzierte $OBDD$-Formel F'. Sei B das minimale binäre Entscheidungsdiagramm des binären Entscheidungsbaums von F'. Wir nennen B *das Entscheidungsdiagramm für* F *und* $<$. Dann gilt: Bezüglich einer Variablenordnung $<$ haben alle logisch äquivalenten aussagenlogischen Formeln dasselbe Entscheidungsdiagramm.

- Verschiedene aussagenlogische Formeln können u.U. dieselbe Boolesche Funktion darstellen
- Starke Kanonizität: Für jede Variablenordnung entspricht jeder Booleschen Funktion genau ein minimales geordnetes Entscheidungsdiagramm.

Erzeugung von OBDDs

Erzeugung eines OBDD aus einer aussagenlogischen Formel A mit n Variablen: Initaler Aufruf ist pl2bdd$(A, 1)$.

- Ein innerer BDD-Knoten δ sei implementiert als eine Struktur $\langle i, j, \delta_0, \delta_1 \rangle$, wobei $i \in \mathbb{N}$ der Index, j die j-te Variable in $<$, δ_0 der 0-Nachfolger und δ_1 der 1-Nachfolger von δ.
- Alle neu erzeugten BDD-Knoten werden in einer Struktur $struct$ gespeichert, die initial leer ist.

Prozedur pl2bdd(F, j) : (Formel,nat){

if $j > n$ **then** { **return** evaluate(F) } /* liefert $\boxed{0}$ bzw. $\boxed{1}$ */

else { δ_0 **:=** pl2bdd$(F[v_j/\bot], j+1)$; δ_1 **:=** pl2bdd$(F[v_j/\top], j+1)$;
if Index(δ_0) = Index(δ_1) **then** { **return** δ_0 }
elsif $\exists i$: $\langle i, j, \delta_0, \delta_1 \rangle \in struct$ **then** { **return** $\langle i, j, \delta_0, \delta_1 \rangle$ }
else { δ **:=** $\langle |struct| + 1, j, \delta_0, \delta_1 \rangle$; $struct$ **:=** $struct \cup \{\delta\}$;
return δ }}}

Operationen auf OBDDs

Gegegen zwei OBDDs (bzgl. einer Variablenordnung $<$ von aussagenlogischen Formeln bzw. Booleschen Funktionen F_1 und F_2.

Reduktion: Minimierung der OBDDs

Kombinationen Boolescher Funktionen:

- „Komplement"bildung: Erzeugung des minimalen OBDDs (bzgl. $<$) von $\neg F_1$
- Direkte Bildung der minimalen OBDDs (bzgl. $<$) von $F_1 \rightarrow F_2$, $F_1 \vee F_2$, $F_1 \wedge F_2$, $F_1 \leftrightarrow F_2$ aus den gegebenen OBDDs.

Satz: Für jede der Operationen gibt es Verfahren, deren Zeitkomplexität linear ist in der Grösse der Eingabegraphen.

Satz: Es gibt aussagenlogische Formeln/Boolesche Funktionen, deren OBDD für alle Variablenordnungen exponentiell ist (z.B. die Funktion für den n-ten Output einer Integer-Multiplikationsschaltung)

Operationen auf OBDDs

Anwendung beliebiger Boolescher Funktionen $\circ$ [Bryant, 1986]:

Prozedur BDD-apply$(\circ, \delta, \delta')$: (Operator,BDD,BDD)**{**

if $\delta, \delta' \in \{\boxed{0}, \boxed{1}\}$ **then { return** δ „$\circ$“ δ' **}** /$*$ liefert $\boxed{0}$ bzw. $\boxed{1}$ $*$/

else { j **:=** index(min-var(δ, δ')) **;**
$\langle f_0, f_1 \rangle$ **:=** co-factor(δ, j) **;** $\langle g_0, g_1 \rangle$ **:=** co-factor(δ', j) **;**
δ_0 **:=** BDD-apply$(\circ, f_0, g_0)$ **;** δ_1 **:=** BDD-apply$(\circ, f_1, g_1)$ **;**
if $\exists \delta'' \in struct$ mit $\delta'' = \langle i, j, \delta_0, \delta_1 \rangle$ **then { return** δ'' **}**
else { δ'' **:=** $\langle |struct| + 1, j, \delta_0, \delta_1 \rangle$ **;** $struct$ **:=** $struct \cup \{\delta''\}$ **;**
return δ'' **}}}**

Prozedur co-factor(δ, j) : (BDD,nat)**{**

if $\delta \in \{\boxed{0}, \boxed{1}\}$ **then { return** $\langle \delta, \delta \rangle$ **}**

else { $\langle i, k, \delta_0, \delta_1 \rangle$ **:=** δ **;**
if $k > j$ **then { return** $\langle \delta, \delta \rangle$ **}**
else { return $\langle \delta_0, \delta_1 \rangle$ **}}}**

Anwendungen von OBDDs

- Feststellung der logischen Implikation bzw. Äquivalenz zweier aussagenlogischer Formeln: Komplexität linear in der Grösse der BDDs
- Entscheidungsverfahren für aussagenlogische Erfüllbarkeit:
  - Falls Formel unerfüllbar, liefert Verfahren pl2bdd das BDD $\boxed{0}$.
  - Falls Formel allgemeingültig, liefert Verfahren pl2bdd das BDD $\boxed{1}$.
  - Im worst-case exponentielle Platzkomplexität
- Praktisch effiziente Repräsentation der Menge aller erfüllenden Belegungen einer aussagenlogischen Formel
- Damit praktisch effiziente Repräsentation von Teilmengen aus einer endlichen Grundmenge G
- Praktisch effiziente Repräsentation von Relationen über einer endlichen Grundmenge G
- Praktisch effiziente Repräsentation der Anzahl der erfüllenden Belegungen einer aussagenlogischen Formel

Bedeutung der Variablenordnung in OBDDs

Bei Änderung der Variablenordnung u.U. exponentielle Unterschiede

Beispiel:

- Modellierung eines Vergleichers zweier Bit-Vektoren

$$a = a_1 \cdots a_n \quad \text{und} \quad b = b_1 \cdots b_n.$$

- Bei Variablenordnung $a_1 < b_1 < \cdots < a_n < b_n$: $3n + 2$ Knoten
- Bei Variablenordnung $a_1 < \cdots < a_n < b_1 < \cdots < b_n$: $3 \times 2^n - 1$ Knoten

Satz: Finden einer Ordnung mit kleinstem OBDD ist NP-hart.

Methoden der Ordnungsfestlegung:

- Praktisch erfolgreiche Heuristik zur Ordnungsbestimmung: Zusammengruppierung „abhängiger“ Variablen
- Falls keine offenkundige Ordnung gegeben, in periodischen Abständen dynamischer Wechsel der Ordnung.

Arbeitsweise von OBDD-Verfahren

Klauseln

c_1: $\neg a \vee \neg b$

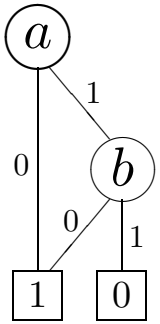
c_2: $b \vee \neg c$

c_3: $b \vee c$

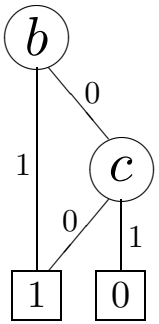
c_4: $a \vee \neg b$

Ordnung

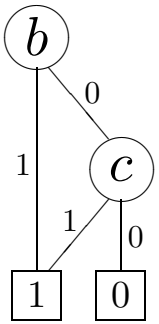
$a < b < c$



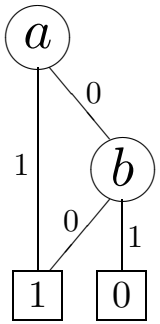
$\mathcal{G}(c_1)$



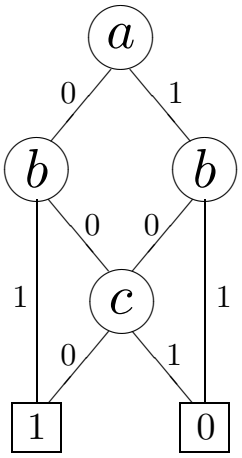
$\mathcal{G}(c_2)$



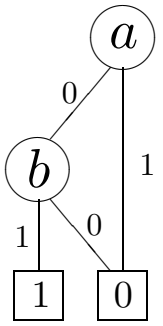
$\mathcal{G}(c_3)$



$\mathcal{G}(c_4)$



$\mathcal{G}(c_1 \wedge c_2)$



$\mathcal{G}(c_1 \wedge c_2 \wedge c_3)$

$\mathcal{G}(c_1 \wedge \cdots \wedge c_4)$

Die Standardparadigmen für SAT-Verfahren

Semantische Bäume [Davis, Loveland, Logemann, 1962]

- Verbesserung der Wahrheitstafelmethode
- Partitionierung der Interpretationen und partielle Evaluierung
- Effiziente Implementierungstechnik (Backtracking-Verfahren)

Geordnete binäre Entscheidungsdiagramme (OBDDs) [Bryant, 1986]

- Formalismus zur Darstellung Boolescher Funktionen
- Verschiedene Funktionalitäten
- Effiziente Modifikationsverfahren (Kombination, Reduktion)

Vergleich der Standardparadigmen

	Semantischer Baum	OBBD
Datenstruktur	Baum	„Dag“
Platzkomplexität	PSPACE	EXPSPACE
Zeitkomplexität	EXPTIME	EXPTIME
Variablenordnung	sehr flexibel	weniger flexibel
Inferenzraten	sehr hoch	weniger hoch
	(Backtracking)	(Saturierung)
Mittlere Beweislänge	sehr lang	weniger lang
	(Baumresolution)	(Geordnete Resolution)

Allerdings gilt: Die Baumresolution und die geordnete Resolution können sich nicht gegenseitig polynomiell simulieren.

Quantifizierte Boolesche Logik

- Mit der Quantifizierten Booleschen Logik bietet sich eine kompaktere Sprache als die reine Aussagenlogik
- Gegeben eine Menge $\mathcal{P}$ von *Aussagenvariablen*.
- Die *(Formeln der) Quantifizierten Boolesche Logik* für $\mathcal{P}$ sei(en) definiert durch:

$$QBF(\mathcal{P}) := \mathcal{P} \mid \bot \mid (QBF \to QBF) \mid \exists \mathcal{P}\, QBF$$

- Dazu übliche weitere Junktoren und den Quantor $\forall := \neg\exists\neg$
- Semantik durch Rückführung auf Aussagenlogik:
  - Eine QBF der Form $\exists p A$ ist wahr gdw $(A[p/\top] \vee A[p/\bot])$ wahr ist
- Für jede aussagenlogische Formel F gilt: $\exists x_1 \cdots x_n F$ ist wahr gdw F erfüllbar
- Zu jeder QBF-Formel gibt es eine äquivalente aussagenlogische Formel, aber QBF-Formeln sind i.A. kompakter.
- Das Problem der Allgemeingültigkeit einer QBF-Formel ist PSPACE-vollständig.

Planen als aussagenlogische Erfüllbarkeit (2)

Übersetzung in aussagenlogische Erfüllbarkeit:

- Jeweils Übersetzung eines Planproblems mit fester Aktionszahl $n-1$
- Logische Repräsentation verschiedener Weltzustände durch Ver$n$fachung der aussagenlogischen Variablen p^j zu $p^j_1, \ldots, p^j_n$
- Formulierung des Effektes einer Aktion $\alpha^j = \langle c^j, e^j \rangle$ $(1 \leq j \leq m)$ durch $n-1$ Formeln:

$$\alpha^j_i \leftrightarrow (e^j_{i+1} \wedge \bigwedge \{(p^j_i \leftrightarrow p^j_{i+1}) : p^j_{i+1} \text{ kommt nicht in } e^j_{i+1} \text{ vor}\}) \quad (1 \leq i < n)$$

- Kombination der Effekte durch Disjunktion aller α^j_i mit demselben unteren Index:

$$\alpha^1_i \vee \cdots \vee \alpha^m_i \quad (1 \leq i < n)$$

- Notwendigkeit der Vorbedingung für einen Effekt:

$$\alpha^j_i \rightarrow c^j_i \quad (1 \leq i < n, 1 \leq j \leq m)$$

  bzw.: falls k äquivalente Effekte existieren, k-fache Disjunktion ihrer Vorbedingungen im Konsequent

Planen als QBF-Erfüllbarkeit (1)

Übersetzung in QBF-Erfüllbarkeit:

- Übersetzung eines Planproblems mit beliebiger Länge $\leq 2^n$
- Um die $\leq$-Bedingung zu erfüllen, nehmen wir zu den Aktionen die triviale Aktion hinzu, die nichts verändert
- Formulierung des Effektes einer Aktion $\alpha^j = \langle c^j, e^j \rangle$ $(1 \leq j \leq m)$ wie in der Aussagenlogik, wobei die a^n die Variablen im ursprünglichen Planungsproblem seien:

$$\alpha^j(p,q) = (e^j[a/q] \wedge \bigwedge\{(p^j \leftrightarrow q^j) : a^j \text{ kommt nicht in } e^j \text{ vor}\}) \quad (1 \leq i < n)$$

- Kombination der Effekte durch Disjunktion aller α_i^j mit demselben unteren Index:

$$D(p,q) = \alpha^1(p,q) \vee \cdots \vee \alpha^m(p,q)$$

- Notwendigkeit der Vorbedingung für einen Effekt:

$$C(p,q) = \bigwedge\{\alpha^j(p,q) \rightarrow c^j[a/p] : 1 \leq j \leq m\}$$

sowie analoge Veränderung falls k äquivalente Effekte existieren

Planen als QBF-Erfüllbarkeit (2)

Definition von Erreichbarkeitsfolgen der Länge $\leq 2^n$:

- $\mathcal{E}_n = \exists p^1 \cdots p^n \exists q^1 \cdots q^n E_n(p,q) \wedge \mathcal{I}[a/p] \wedge \mathcal{G}[a/q]$

  (wobei $\mathcal{I}$ Initial- und $\mathcal{G}$ Zielzustandsformel)
- $E_n(p,q)$ bedeutet Erreichbarkeit in $\leq 2^n$ Schritten
- Definition von E:
  - $E_0(p,q) := D(p,q) \wedge C(p,q)$
  - $E_{i+1}(p,q) := \exists r^1 \cdots r^n \; \forall t \; \exists s^1 \cdots s^n \; \exists l^1 \cdots l^n$

    $(E_i(s,l) \wedge$

    $(t \rightarrow \bigwedge_{1 \leq k \leq n} (\{s^k \leftrightarrow p^k\} \cup \{l^k \leftrightarrow r^k\}) \wedge$

    $(\neg t \rightarrow \bigwedge_{1 \leq k \leq n} (\{s^k \leftrightarrow r^k\} \cup \{l^k \leftrightarrow q^k\}))$
- Das Verfahren erlaubt eine binäre Zerteilung des Gesamtplanes
- $r^1 \wedge \cdots \wedge r^n$ beschreibt den Zwischenzustand
- Die Variable t garantiert, dass beide Teilstrecken existieren, wenn die Formel wahr ist

Planen als QBF-Erfüllbarkeit (3)

Eigenschaften der Übersetzung in QBF:

- Übersetzung ist polynomiell, aber wie verhalten sich QBF-Solver in der Praxis?
- Experimentelle Resultate: Bildformeln sind i.A. sehr schwer für die existierenden QBF-Solver
- Mögliche Gründe:
  - Die existierende QBF-Solver sind schlecht
  - QBF-Formeln sind nun mal schwerer als aussagenlogische Formeln
  - Die QBF-Transformation ist schlecht
- Beobachtung: Angegebene Übersetzung erfordert sehr aufwendige Techniken, um polynomielle Zeitkomplexität zu erhalten, selbst bei Existenz kurzer Pläne
- Alternativen:
  - Analoge Anwendung des aussagenlogischen Vorgehens: Lösen eines Planungsproblems mit n Aussagenvariablen durch parallele Lösung von n QBF-Übersetzungen $\mathcal{E}_1, \ldots, \mathcal{E}_n$
  - Verbesserung der Übersetzung

Planen als QBF-Erfüllbarkeit (4)

Alternative QBF-Übersetzung

- Grundübel der ersten Übersetzung: alle gefundenen Pläne sind exponentiell lang, enthalten aber sehr viele triviale Aktionen
- Ziel: Finden kurzer Pläne in polynomieller Zeit ermöglichen
- Grundidee: triviale Aktion weglassen, dafür Erreichbarkeit verallgemeinern
- Definition von E modifizieren:

$$
\begin{aligned}
E_{i+1}(p,q) := \exists r^1 \cdots r^n \; \forall t \; \exists s^1 \cdots s^n \; \exists l^1 \cdots l^n \\
((E_i(s,l) \wedge \\
(t \rightarrow \textstyle\bigwedge_{1 \leq k \leq n} (\{s^k \leftrightarrow p^k\} \cup \{l^k \leftrightarrow r^k\})) \wedge \\
(\neg t \rightarrow \textstyle\bigwedge_{1 \leq k \leq n} (\{s^k \leftrightarrow r^k\} \cup \{l^k \leftrightarrow q^k\}))) \\
\vee E_0(s,l))
\end{aligned}
$$

- Das Verfahren erlaubt Simulation des aussagenlogischen Verhaltens
- Entwicklung weiterer Übersetzungen und deren Analyse nötig

Steigende Bedeutung von QBF's

Die Quantifizierte Boolesche Logik

- ist ein natürlichen Paradigma zur Charakterisierung der Komplexitätsklasse PSPACE und der polynomiellen Hierarchie
- ermöglicht eine kompakte und natürliche Formulierung vieler Probleme aus den Bereichen: Planen, Abduktion, nichtmonotones Schliessen, Model Checking
- ermöglicht eine Reformulierung von Problemen aus dem Bereich der intuitionistischen Logik, der terminologischen Logiken sowie der Modallogiken

Dementsprechend wurde in neuester Vergangenheit mit der Entwicklung von Entscheidungsverfahren für die Quantifizierte Boolesche Logik (QBF-Solver) begonnen:

Systeme:

- QKN [KleineBuening1995]
- Evaluate [Cadoli1998]
- Decide [Rintanen1999]
- QSOLVE [Feldmann2000]
- QuBE [Giunchiglia2001]
- Semprop [Letz2002]

Paradigmen für QBF-Solver

- Übersetzung in Aussagenlogik (exponentieller Platzbedarf)
- Q-Resolution (exponentieller Platzbedarf)
- BDD-Ansatz (exponentieller Platzbedarf)
- Semantische Bäume (polynomieller Platzbedarf)

Der Ansatz der Verwendung semantischer Bäume ist am robustesten

BDDs für Quantifizierte Boolesche Formeln

Elimination existenzieller BDD-Variablen:

Prozedur BDD-exists(V, δ) : (Variablenindex-Menge,BDD) **{**

if $\delta \in \{\boxed{0}, \boxed{1}\}$ **then { return** δ **}**

else { $\langle i, k, \delta_0, \delta_1 \rangle$ **:=** δ **;**
δ'_0 **:=** BDD-exists(V, δ_0) **;**
δ'_1 **:=** BDD-exists(V, δ_1) **;**
if $k \in V$ **then { return** BDD-apply$(\vee, \delta'_0, \delta'_1)$ **}**
else { return new-node$(k, \delta'_0, \delta'_1)$ **}}}**

Prozedur new-node(k, δ_0, δ_1) : (nat,BDD,BDD)**{**

if $\exists \delta \in struct$ **: (** var$(\delta) = k$ **and** 0-succ$(\delta) = \delta_0$ **and** 1-succ$(\delta) = \delta_1$ **)**
then { return δ **}**
else { δ **:=** $\langle |struct| + 1, k, \delta_0, \delta_1 \rangle$ **;** $struct$ **:=** $struct \cup \{\delta\}$ **;**
then { return δ **}}**

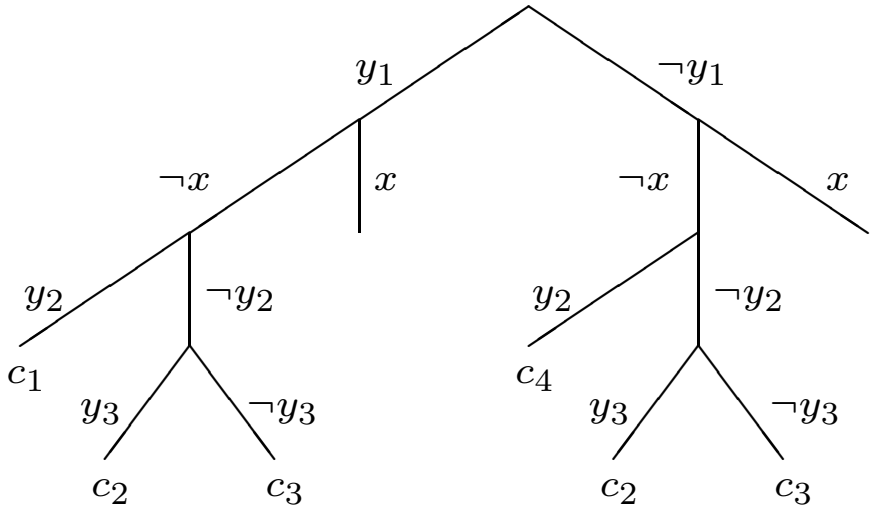
Semantische Bäume für QBFs

- Typischerweise auf QBFs in Pränexform: $Qx_1 \cdots Qx_n F$ wobei F aussagenlogische Formel in konjunktiver Normalform
- Ansatz folgt wieder direkt der Semantik
- Iteratives Aufspalten des Entscheidungsproblems einer Formel der Form $Qx\Phi$ in zwei Teilprobleme $\Phi[x/\top]$ und $\Phi[x/\bot]$
- Unterschied zur Aussagenlogik: ein zusätzlicher Quantor
- Unterscheidung zwischen existenziellen und universellen Variablen und Baumverzweigungen
  - $\exists x\Phi$ ist wahr gdw $\Phi[x/\top]$ oder $\Phi[x/\bot]$ wahr ist
  - $\forall x\Phi$ is wahr gdw $\Phi[x/\top]$ und $\Phi[x/\bot]$ wahr sind

Eine Semantischer Baum für eine QBF

Beispiel 1 Eine falsche Quantifizierte Boolesche Formel:

$$\exists y_1 \forall x \exists y_2 \exists y_3 (\underbrace{(\neg y_1 \vee x \vee \neg y_2)}_{c_1} \wedge \underbrace{(y_2 \vee \neg y_3)}_{c_2} \wedge \underbrace{(y_2 \vee y_3)}_{c_3} \wedge \underbrace{(y_1 \vee \neg x \vee \neg y_2)}_{c_4}).$$



Eine geschlossener semantischer Baum für die QBF

A DPLL-artiges Semantisches Baumverfahren

- Abarbeitung des Baumes mittels Tiefensuche und Abspeichern lediglich eines Astes mit Seiteninformation (linearer Platzbedarf)

- Benutzung der Purity-Regeln: falls die QBF Φ eines Knotens N in einem semantischen Baum ein pures Literal v ($\neg v$) enthält, dann verzweige nach v und
  - wenn v existenziell ist, ignoriere das Teilproblem $\Phi[v/\bot]$ ($\Phi[v/\top]$),
  - when v universell ist, ignoriere das Teilproblem $\Phi[v/\top]$ ($\Phi[v/\bot]$).

- Präfix-geordnete Klausel: Klausel, in der die Literale nach ihrem Vorkommen im Quantorenpräfix aufgeschrieben sind

- Benutzung von Unit-Propagierung: eine präfix-geordnete Klause ist unit, falls sie die Form $y \vee x_1 \vee \cdots \vee x_n$ hat, wobei y existenziell ist und die x_i universell sind; falls eine Unit-Klausel existiert, verzweige nach ihrer (existenziellen) Variable und schliesse einen der neuen Äste durch die Klausel

Zur Entwicklung leistungsfähiger QBF-Solver

- (DPLL-basierte) SAT-solver werden immer leistungsfähiger
- Kann das auch für den QBF-Fall erreicht werden?
- Hauptarbeitsbereiche bei der Verbesserung von SAT-Solvern:

(1) Geschickte Wahl der nächsten Verzweigungsvariable
(2) Effiziente Datenstrukturen und -algorithmen
(3) Lernverfahren (Intelligentes Backtracking, Lemmas, . . .)

- Hauptproblem im QBF-Fall: Freiheit der Variablenauswahl ist stark eingeschränkt (i.A. muss die Präfixordnung eingehalten werden) $\Rightarrow$ (1) damit stark eingeschränkt
- (2) kommt zu früh für den QBF-Fall
- Damit bleibt (3)

Intelligentes Backtracking

Intelligentes Backtracking = Abhängigkeitsgesteuertes Backtracking

- Beobachtung: Durch „irrelevante“ Verzweigungen entstehen grosse Redundanzen im Suchraum
- Grundidee: Eine Verzeigungsvariable ist irrelevant wenn ihr Wert bei der Lösung des entsprechenden Teilproblems keine Rolle spielt
- Falls sich eine Verzeigungsvariable als irrelevant herausstellt, muss der zweite Ast nicht untersucht werden, d.h. das Backtracking läuft weiter zurück $\Rightarrow$ Reduktion des Suchraums
- Gesucht: Effiziente Formalisierungen des Konzeptes der Irrelevanz
- Verschiedene Ansätze möglich
- Für existenzielle Variablen analog zum aussagenlogischen Fall
- Für universelle Variablen sind neue Methoden zu entwickeln
- Es zeigt sich: Universeller Fall ist schwieriger zu handhaben

Intelligentes Backtracking für existenzielle Verzweigungen (1)

- Gegeben sei eine partiell evaluierte QBF Φ an einem existenziellen Verzweigungspunkt mit Variable y; angenommen o.B.d.A., wir untersuchen zuerst den Wahrheitsfall von y
- Wenn $\Phi[y/\top]$ falsch ist (durch Unterbaum t belegt), dann muss der $\neg y$-Ast betrachtet werden
- Wenn aber $\neg y$ in keiner der schliessenden Blattklauseln von t vorkommt, dann liefert der Unterbaum t auch einen Beweis für die Falschheit von $\Phi[y/\top]$; damit muss der $\Phi[y/\bot]$-Fall nicht untersucht werden, und es kann der Wert falsch zurückgegeben werden
- Zwei Realisierungen des Ansatzes:
  - Verwendung einer Relevanzmarkierung an jeder Variablen
  - Arbeiten mit Relevanzmengen, die u.U. vereinigt werden müssen
- Erster Ansatz effizienter, aber weniger stark bzgl. Suchraumreduktion wie zweiter Ansatz

Intelligentes Backtracking für existenzielle Verzweigungen (2)

Relevanz durch destruktives Markieren:

- Wenn eine Klausel einen Ast schliesst, markiere die Komplemente aller existenziellen Literale in der Klausel als relevant
- Relevanzprüfung einer Variablen y beim Wechsel ihres Wertes durch Überprüfung ihrer Relevanzmarkierung dann muss der $\neg y$-Ast betrachtet werden

Relevanz durch Rückgabe von Relevanzmengen:

- Wenn eine Klausel einen Ast schliesst, gebe die Menge der Komplemente aller existenziellen Literale in der Klausel als Relevanzmenge zurück
- beim Zurücklaufen Vereinigung der Relevanzmengen, wenn beide Äste falsch sind
- Relevanzprüfung einer Variablen y beim Wechsel ihres Wertes durch Elementschaftstest in der Relevanzmenge

Intelligentes Backtracking für universelle Verzweigungen (1)

Relevanz durch Schnittmengen von Modellen:

- Gegeben sei eine partiell evaluierte QBF Φ an einem universellen Verzweigungspunkt mit Variable x; angenommen o.B.d.A., wir untersuchen zuerst den Wahrheitsfall von x
- Sei M die Menge der Modelle (jeweilige Astliterale) der (offenen) Äste von t
- Sei C die Menge der offenen (d.h. nicht wahren oder falschen) Klauseln in Φ
- Wenn jede Klausel in C ein Literal aus $M \setminus \{x\}$ enthält, dann dann liefert der Unterbaum t auch einen Beweis für die Wahrheit von $\Phi[x/\top]$; damit muss der $\Phi[x/\bot]$-Fall nicht untersucht werden, und es kann der Wert wahr zurückgegeben werden

Weitere Methoden des intelligenten Backtracking für universelle Variablen:

- Relevanz durch Vereinigung von Modellen (teurer, aber mächtiger)
- Beliebig aufwändige Methoden möglich (Abwägung Kosten/Gewinn)

Ein Verdoppelungsproblem in Semantischen Bäumen

Man beachte die Verdoppelung, die aus den identischen Teilbäumen in Beispiel 1 resultiert

Beispiel 2 (eine sehr einfache pathologische Beispielklasse) Für jedes $n \geq 1$, sei F_n die QBF der folgenden Form: $\exists y_{n+1} \forall x_n \exists y_n \cdots \forall x_1 \exists y_1 S$ wobei die Matrix S 4 Klauseln enthält

$$\neg y_{n+1},\ y_1 \vee y_2,\ \neg y_1 \vee y_2,\ y_1 \vee \neg y_2, \text{ plus die } n-1 \text{ Klauseln}$$
$$\neg y_j \vee \neg x_j \vee \neg y_{j+1} \vee x_{j+1} \vee y_{j+2}, \text{ für } 1 \leq j \leq n-1.$$

Die Grösse jedes semantischen Baumbeweises eines F_n ist exponentiell in n (selbst wenn Techniken wie Intelligentes Backtracking benutzt werden).

Performanz von Semprop ohne Lemmas:

- $> 3 \times 10^6$ Äste für F_{30} (33 Klauseln)
- $> 3 \times 10^7$ Äste für F_{35}
- F_{50} kann nicht einmal in einem Tag gelöst werden

Benutzung von Lemmas

Q-resolution [KleineBuening1995]

- $\forall$-Redukt einer präfix-geordneten Klausel $c \vee y \vee x_i \vee \cdots \vee x_n$ ist $c \vee y$
- Gegeben zwei nicht-tautologische Klauseln (als Menge) c_1 und c_2 mit $y \in c_1$ und $\neg y \in c_2$, die Q-Resolvente von c_1 und c_2 ist (als Menge)

$$(c_1' \setminus \{x\}) \cup (c_2' \setminus \{\neg x\})$$

  wobei c_1' und c_2' die $\forall$-Redukte (als Mengen) von c_1 bzw. c_2 sind.
- Q-resolution ist korrekt (und vollständig)
- Integration der Q-resolution in Semantische Bäume: Bilde die Q-Resolvente benachbarter Unterbäume, deren Knotenformeln als falsch erwiesen wurden, falls möglich

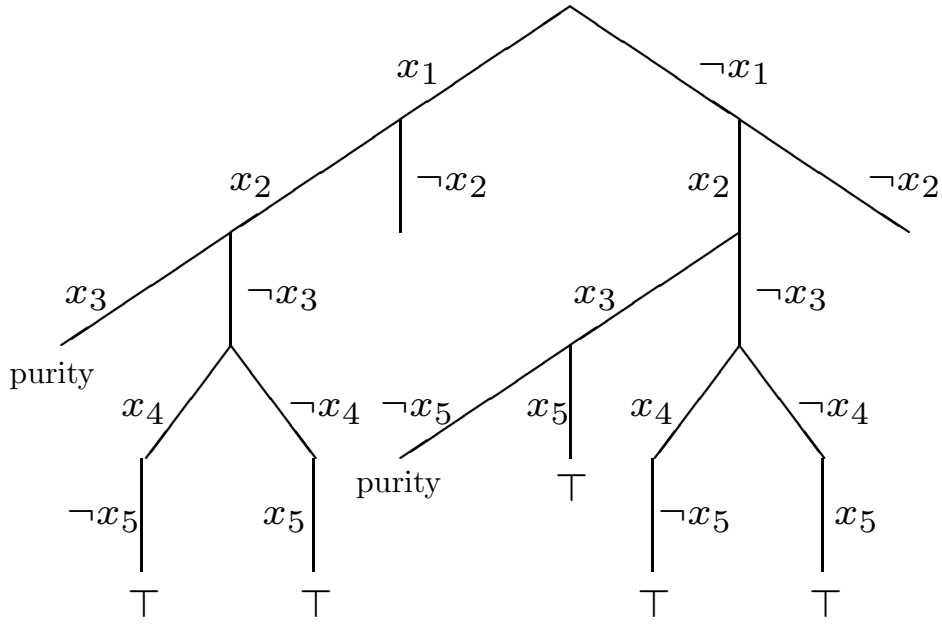
Bei Benutzung von Lemmas kann die Formelklasse F_n in linearer Zeit entschieden werden.

Ein duales Verdoppelungsproblem für Semantische Bäume

Verdoppelung wahrer Unterbäume:

Quantifizierte Boolesche Formel: $\forall x_1 \exists x_2 \forall x_3 \forall x_4 \exists x_5 (c_1 \wedge c_2 \wedge c_3 \wedge c_4 \wedge c_5)$ wobei

$$
\begin{aligned}
c_1 &= x_1 \vee \neg x_3 \vee x_5 \\
c_2 &= \neg x_1 \vee x_2 \vee x_5 \\
c_3 &= \neg x_2 \vee x_4 \vee x_5 \\
c_4 &= x_3 \vee \neg x_4 \vee \neg x_5 \\
c_5 &= x_2 \vee \neg x_3 \vee \neg x_5
\end{aligned}
$$



Vermeidung der Verdoppelung durch Erzeugen und Verwenden von Modellen

Weitere Ausnutzung von Modellen

Phänomen: Existenz variablenunabhängiger Teilformeln

Beispiel:

$\forall x_1 x_3 \cdots x_{n-1} \exists y_2 y_4 \cdots y_n (c_1 \wedge \cdots \wedge c_n)$ wobei
$c_1 = x_1 \vee y_2$, $c_2 = \neg x_1 \vee \neg y_2$, $c_3 = x_3 \vee y_4$, $c_4 = \neg x_3 \vee \neg y_4$, . . . ,
$c_{n-1} = x_{n-1} \vee y_n$, $c_n = \neg x_{n-1} \vee \neg y_n$.

Beobachtung:

Wenn eine partielle Belegung m, die z.B. $\neg y_i$ und x_{i-1} enthält, ein Modell für die Formel ist, dann gilt folgendes: wenn man die Vorzeichen der beiden Literale vertauscht, erhält man ebenfalls ein Modell für die Formel

Zeichenabstraktionsmethode:

Gegeben zwei partielle Belegungen m und m': falls m ein Model von Φ ist und $Vars(m') \subseteq Vars(m)$, $WahreKlauseln(m) \subseteq WahreKlauseln(m')$ sowie die Formel $\Phi \setminus WahreKlauseln(m)$ leer ist, dann ist m' ein Modell für Φ.

Weitere deterministische Planprobleme

Planfindungsprobleme:

- Gegeben sei ein deterministisches Planungsprobleme $\Phi = \langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$
- Ausgabe eines Plans für Φ beliebiger Länge
  - i.A. exponentiell (Problemstellung problematisch)
  - Um bessere Resultate zu erreichen: Erweiterung der Ausgabesprache für Pläne (vgl. Towers of Hanoi)
- Ausgabe eines Plans für Φ der Länge n oder kürzer (in NP)

Erweiterungen der Plansprache:

- Das deterministische Planexistenzproblem für eine Plansprache mit schematischen Operatoren ist EXPSPACE-vollständig. (Beispiel: Operator, der die Werte aller Variablen in einer bestimmten Variablenmenge komplementiert)

- Bei Funktionssymbolen (der Stelligkeit ≥ 2) in der Plansprache werden Planprobleme unentscheidbar

Planen unter Nichtdeterminismus

Zwei Arten von Nichtdeterminismus:

- Initialformel: Anstatt eines Initialzustandes eine Initialformel, d.h. eine Menge von Anfangszuständen
- Nichtdeterministische Operatoren: Aktionen und Zustand bestimmt nicht mehr eindeutig Folgezustand (Aktionen nicht mehr Funktionen auf Zustandsmengen), Aktion kann verschiedene Folgezustände haben

Beobachtbarkeit (der Folgen von Aktionen):

- Relevant im nichtdeterministischen Fall und bei mehr als einem Anfangszustand
- Volle Beobachtbarkeit vs. keine vs. partielle Beobachtbarkeit (absteigende Schwierigkeit der Probleme)
- Realisierung der Planfindung/Planexistenz-Aufgabe: Erzeugung/Beweis der Existenz eines bedingten Planes (mit Alternativen), damit sind Pläne eingeschränkte Programme mit Schleifen und Haltebedingung

Nichtdeterministische Operatoren

Qualitativer und quantitativer Nichtdeterminismus:

- Nichtdeterministische Effekte sind Disjunktionen deterministischer Effekte
- Probabilistische Effekte sind Formeln der Form $w_1 e_1 \mid \cdots \mid w_n e_n$ wobei $\Sigma_{1 \leq i \leq n} w_i = 1$

  Jede Aktion α bildet einen Zustand auf eine Wahrscheinlichkeitsverteilung der Effektvariablen ab

Nichtdeterministische Planungsprobleme mit Beobachtbarkeit:

- Weiterhin Idee: Pläne entscheiden, welche Aktion als nächstes anzuwenden ist
- Kodierung der Beobachtbarkeit: $\Phi = \langle \mathcal{P}, \mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{B} \rangle$ Wobei $\mathcal{B}$ Menge der beobachtbaren Variablen
- Wichtig: Pläne können Schleifen beinhalten und die Ausführung eines Planes kann beliebig lange laufen (Beipiel: Würfle, bis eine 6 kommt!)

Planen bei Einschränkung der Beobachtbarkeit

Einschränkungen der Beobachtbarkeit:

- Keine Beobachtbarkeit: Problem ähnlich zum deterministischen Fall, ein Plan bleibt weiterhin eine Folge von Aktionen, er muss aber für alle möglichen Effektfolgen gelingen
- Volle Beobachtbarkeit: Plan ist Programm, das für jeden Folgezustand eine anderen Folgeaktion anwenden kann (damit variabler als bei Nichtbeobachtbarkeit)
- Partielle Beobachtbarkeit: Kombination beider Lesarten, je nach Beobachtbarkeit der entsprechenden Variablen

Schwierigkeit der Planexistenz-Aufgabe:

- bei Nichtbeobachtbarkeit: EXPSPACE-vollständig
- bei voller Beobachtbarkeit: EXP(TIME)-vollständig
- bei partieller Beobachtbarkeit: 2-EXP(TIME)-vollständig
- Für Pläne polynomieller Länge eingeschränkt:
  z.B. bei Nichtbeobachtbarkeit: Σ_2-vollständig