Vorlesung
– Automatisches Beweisen –

Kap. 4.2: AL-Entscheidungsverfahren
Kap. 4.2.1 Aussagenlogische Resolution

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften

Universität Tübingen

Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)

Wolfgang.Kuechlin@uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de

SR

Kap 4.2

Aussagenlogische Entscheidungsverfahren

SR

Aussagenlogische Entscheidungsverfahren

- Resolution
- Davis-Putnam-Logemann-Loveland (DPLL)
- Ordered Binary Decision Diagrams (OBDDs)
- Semantische Tableaus
- Boolesche Polynome (fehlen noch)

Kalkül

- **Kalkül** (= formales Regelsystem) zum Herleiten neuer Formeln aus gegebener Formelmenge
  - Formeln werden „mechanisch“ erzeugt
  - Daher eignet sich ein Kalkül zur Implementierung
- Schreibweise: $\mathcal{F} \vdash F$

  Formel F wird aus der Formelmenge $\mathcal{F}$ durch Regeln hergeleitet
- Sinnvoll ist das nur, falls die neue Formel auch „gilt“

  d.h. aus $\mathcal{F} \vdash F$ folgt $\mathcal{F} \models F$
  - wir sagen: der Kalkül ist korrekt (sound).
  - Dann gilt: $\mathcal{F} \equiv \mathcal{F} \wedge F$

    Bew.: Jede Interpretation, die $\mathcal{F}$ wahr macht, macht auch $\mathcal{F} \wedge F$ wahr.
- Wenn wir alles Wahre ableiten können, ist der Kalkül vollständig (complete): aus $\mathcal{F} \models F$ folgt $\mathcal{F} \vdash F$
- Widerlegungsvollständig: aus $\mathcal{F} \models \bot$ folgt $\mathcal{F} \vdash \bot$

Resolution

- **Kalkül** zum Herleiten (aussagenlogischer) Formeln in CNF
  - **Deduktionsverfahren:** gegeben $\mathcal{F}$, leite neue Formel F daraus her.
- Entwickelt 1965 von **John Alan Robinson**
  - Ursprünglich für Prädikatenlogik erster Stufe
  - A machine-oriented logic based on the resolution principle. *Journal of the ACM* **12**(1), 23—41.
- Aus Axiomensystem $\mathcal{F}$ (in CNF) können neue gültige Formeln (*Resolventen*) F_i hergeleitet werden, $\mathcal{F} \models F_i$
  - Bem.: $\mathcal{F} \equiv \mathcal{F} \cup F_i$
- Erste Idee: beweise $\mathcal{F} \models F$ durch direkte Herleitung $\mathcal{F} \vdash F$
  - geht nicht immer, da Resolution nur widerlegungsvollständig
- Üblicher Umweg:
  - $\mathcal{F} \models F$ gdw. es gilt nie $\mathcal{F} \wedge \neg F$, also gdw. $\mathcal{F} \wedge \neg F \models \bot$
  - da Resolution widerlegungsvollständig: $\mathcal{F} \wedge \neg F \models \bot$ gdw. $\mathcal{F} \wedge \neg F \vdash \bot$
  - Insgesamt: $\mathcal{F} \models F$ gdw. $\mathcal{F} \wedge \neg F \vdash_{Res} \bot$

Resolution: Kontradiktions-Beweise

- Eine unerfüllbare Formel F heißt **Kontradiktion**.
  - F ist Kontradiktion gdw. $F \models \bot$
- Anwendung: Beweise **Unerfüllbarkeit** einer Klauselmenge $\mathcal{F}$ durch Herleiten der (unerfüllbaren) „leeren“ Klausel $\{\bot\}$, also $\mathcal{F} \vdash^* \bot$
  - Dann ist $\mathcal{F} \equiv \mathcal{F} \cup \{\bot\} \equiv \mathcal{F} \wedge \bot \equiv \bot$
- Methode:
  - Wende **Inferenzregel(n)** zur Herleitung der leeren Klausel an.

    (Leere Klausel { }, gleichbedeutend mit $\{\bot\}$, im Kalkül üblicherweise symbolisiert durch $\square$, ist nicht erfüllbar, da die leere Klausel keine erfüllbaren Literale enthält)
  - **Axiome**: Klauseln der Ausgangsformel
  - Falls sich leere Klausel herleiten lässt, ist ein **Beweis** der Unerfüllbarkeit gefunden.

J. A. Robinson (Quelle: Wikipedia)

- **John Alan Robinson** (* 1930 in Yorkshire, Großbritannien) ist ein englischer Philosoph und Logiker, der wichtige Beiträge zur Logikprogrammierung geleistet hat.
- Nach einem abgeschlossenen Studium der Klassischen Altertums-wissenschaft an der Uni Cambridge ging er 1952 in die USA. Dort studierte er zunächst Philosophie an der U. Oregon und promovierte 1956 in Princeton zum Ph.D. Danach arbeitete er beim Chemiekonzern DuPont, wo er Programmieren und Mathematik lernte. 1961 wechselte er an die Rice University, wo er sich weiter mit Mathematik beschäftigte.
- 1965 veröffentliche er mit „A machine-oriented logic based on the resolution principle“ [J.ACM 12(1)] wichtige Grundlagen zur automatisierbaren Resolution in der Logik. Auf ihn geht ein Algorithmus zur Unifikation von prädikatenlogischen Formeln zurück, der entscheidend beim Nachweis der Unerfüllbarkeit einer prädikatenlogischen Formel ist.

Resolutions-Regel

- (Einzige) Inferenzregel:

$$\frac{C \cup \{l\} \qquad D \cup \{\neg l\}}{C \cup D}\,\text{Res}$$

wobei C U $\{\ell\}$, D U $\{\neg\ell\}$ Klauseln.

- Beispiele:

$$\frac{\{x, y, \neg z\} \quad \{u, \neg v, z\}}{\{x, y, u, \neg v\}} \qquad \frac{\{x, u, \neg v\} \quad \{\neg u\}}{\{x, \neg v\}}$$

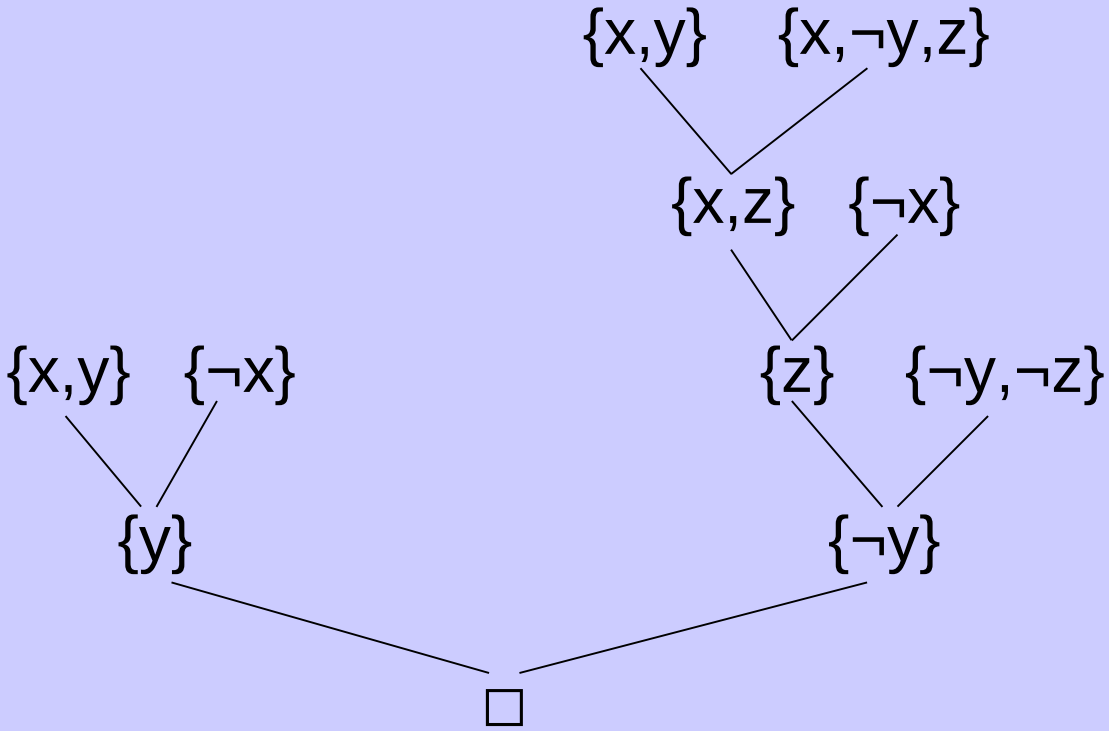
Resolutionskalkül

$$\frac{K_1 \quad K_2}{R}\text{Res}$$

- Begriffe allgemein bei Kalkül:
  - K_1, K_2: Prämissen
  - R: Konklusion
- Begriffe bei Resolutionsregel:
  - R: Resolvente *(resolvent)*
  - K_1, K_2: Eltern-Klauseln *(parent clauses)*
  - „R entsteht durch Resolution über ℓ aus K_1 und K_2“
  - Leere Klausel □ ist nicht erfüllbar, steht für {⊥}

Resolutionsbeweis: Beispiel

F3={{x,y},{x,¬y,z},{¬x},{¬y,¬z}}



Korrektheit der Resolution

- Der Resolutionskalkül ist korrekt (sound). D.h. für alle Klauselmengen $\mathcal{F}$ und alle Klauseln D gilt:

  $\mathcal{F} \vdash^*_{Res} D$ impliziert $\mathcal{F} \models D$

- Damit gilt auch:

  $\mathcal{F} \vdash^*_{Res} D$ impliziert $\mathcal{F} \equiv \mathcal{F} \cup D$

- Korrektheit:
  - Die Konklusion $C \cup D$ wird von den Prämissen $C \cup \{\ell\}$, $D \cup \{\neg\ell\}$ impliziert.
    - Jede Interpretation β, die beide Prämissen wahr macht, muss entweder ℓ oder $\neg\ell$ wahr machen.
    - Falls $\beta(\ell)=1$, so $\beta(D)=1$; falls $\beta(\neg\ell)=1$, so $\beta(C)=1$.
    - Also in jedem Fall $\beta(C \cup D)=1$
  - Damit ist ein einzelner Beweisschritt korrekt. Korrektheit einer Ableitungskette folgt per Induktion

Resolutionsabschluss

- Herleitungsbegriff: $\mathcal{F} \vdash^* C$
  - Aus der Klauselmenge $\mathcal{F}$ lässt sich durch eine endliche Anzahl von Anwendungen der Resolutionsregel die Klausel C herleiten
- Resolutionsabschluss:
  - $Res^0(\mathcal{F}) = \mathcal{F}$
  - $Res^1(\mathcal{F}) = \mathcal{F} \cup \{R \mid R$ (nicht-tautologische) Resolvente zweier Klauseln K_1, K_2 aus $\mathcal{F}\}$
  - $Res^{k+1}(\mathcal{F}) = Res^1(Res^k(\mathcal{F}))$ für $k \geq 1$
  - $Res^*(\mathcal{F}) = \bigcup_{k \geq 0} Res^k(F)$
- Wiederum gilt: $\mathcal{F} \equiv Res^0(\mathcal{F}) \equiv Res^k(\mathcal{F}) \equiv Res^*(\mathcal{F})$

Subsumtion / Subsumption

- Eine Klausel C subsumiert *(subsumes)* eine Klausel D, genau dann wenn $C \subseteq D$.
  - subsumieren: unterordnen, unter etwas einordnen
  - Constraint D ordnet sich unter, denn Constraint C ist strenger als D
- Falls Klausel C die Klausel D subsumiert,

  dann gilt $C \models D$.
  - Also gilt auch wieder: $C \wedge D \equiv C$
- Bei Erfüllbarkeitsprüfung und im Resolutionsbeweis können subsumierte Klauseln gelöscht werden.
  - sobald C erfüllt ist, ist auch D erfüllt
  - falls D nicht erfüllbar ist, so ist auch C nicht erfüllbar
  - Alternativer Beweis: $\mathcal{F} \cup C \equiv \mathcal{F} \cup C \cup D$, also kann D gelöscht werden.

Resolutionsverfahren

- Sei eine Menge von Axiomen (bzw. Constraints) C gegeben. Um einen Satz D zu beweisen, $C \models D$, verfahre wie folgt:

1. Negiere D. Bringe $C \cup \neg D$ in CNF. Nenne das Resultat F.
2. Wiederhole
   i. Bilde alle nicht-tautologischen Resolventen R von F
   ii. Falls $\square \in R$, return „proof“.
   iii. Lösche in R alle subsumierten Klauseln.
   iv. Falls $R = \{\}$, return „disproof“.
   v. Lösche in F alle durch R subsumierten Klauseln.
   vi. $F := F \cup R$

- Falls $C \models D$, so stoppt das Verfahren in (ii) wegen der Widerlegungs-Vollständigkeit (s.u.)
- Andernfalls stoppt das Verfahren in (iv), da es maximal 3^n subsumptionsfreie Klauseln in n Variablen (x, ¬x, don´t care x) gibt.

Widerlegungsvollständigkeit der Resolution

- Der Resolutionskalkül ist widerlegungsvollständig *(refutation complete)*. D.h. für alle Formeln F gilt:

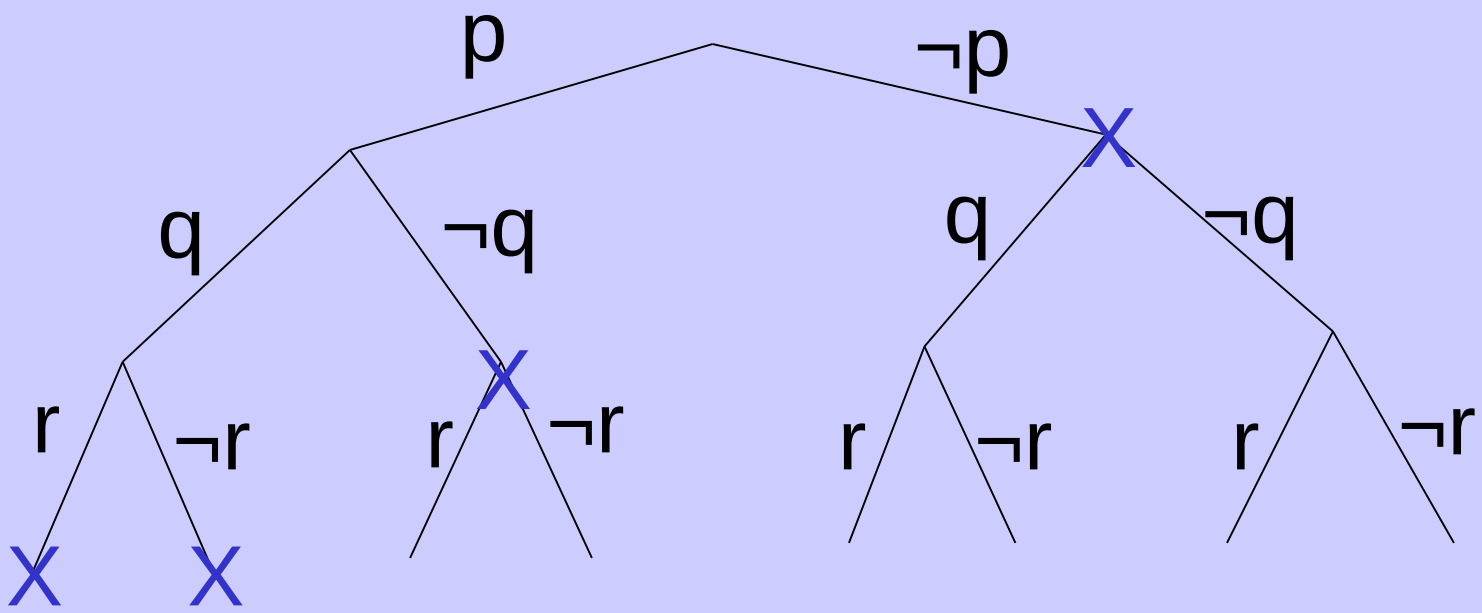
  F unerfüllbar impliziert $F \vdash^*_{Res} \square$

  ($F \models \bot$ impliziert $F \vdash^*_{Res} \square$)

- Der Resolutionskalkül ist nicht vollständig.
  - Bsp: $x \wedge \neg x \models y$, aber y ist nicht durch Resolution aus $x \wedge \neg x$ ableitbar, denn es gibt nur eine einzige Resolvente $\square$.
  - wohl aber gilt $\{\{x\}, \{\neg x\}, \{\neg y\}\} \vdash^*_{Res} \square$
- Beweis der Widerlegungs-Vollständigkeit
  - semantische Bäume: Repräsentation aller Belegungen als Baum
  - Auswirkungen eines Resolutionsschritts auf diesen Baum

Semantische Bäume

S = {{p}, {¬p,q}, {¬r}, {¬p,¬q,r}}



Belegung: Pfad im Baum
Fehlerknoten (X): Hier wird eine Klausel falsifiziert

Semantische Bäume (2)

- Jeder Knoten k im Baum $\mathcal{T}$ definiert eine partielle Interpretation β_k, gegeben durch die Belegungen auf dem Pfad von der Wurzel zum Knoten.
- Jedes Blatt b in $\mathcal{T}$ definiert eine vollständige Interpretation β_b. Falls β_b (F)=0 gilt, dann wird F durch b falsifiziert und b heißt geschlossen, sonst offen. Jedes Blatt repräsentiert auch einen Zweig.

- $\mathcal{T}$ ist geschlossen, wenn alle Zweige (Blätter) geschlossen sind.

- $\mathcal{T}$ ist geschlossen, genau dann wenn F unerfüllbar.

Semantische Bäume (3)

- Ein innerer Knoten k falsifiziert F, wenn es eine Klausel C von F gibt, die falsifiziert wird, d.h. wenn $\beta_k(C)=0$.
- Ein Knoten eines Zweiges, der F falsifiziert und der Wurzel am nächsten ist, heißt Fehlerknoten.
- Eine Klausel, die durch einen Fehlerknoten falsifiziert wird, ist die mit diesem Fehlerknoten assoziierte (Fehler-)Klausel.
- Ein Inferenzknoten n ist ein Knoten, dessen Kinder beide Fehlerknoten sind.
  - Zu jedem Kind n_1, n_2 gibt es jeweils eine Klausel C_1, C_2, die am Kindknoten falsifiziert wird.
  - Unter der Belegung ν am Inferenzknoten sind $\nu(C_1)$, $\nu(C_2)$ jeweils unit Klauseln (denn jeweils mit einer einzigen weiteren Variablenbelegung werden sie falsifiziert). $\nu(C_1)$, $\nu(C_2)$ sind komplementäre Literale, denn sie werden durch zwei komplementäre Belegungen derselben Variable an den Kindknoten falsifiziert.

Widerlegungsvollständigkeit der Resolution

- F unerfüllbar genau dann, wenn $\mathcal{T}$ geschlossen.
- In einem (nicht-trivialen) geschlossenen $\mathcal{T}$ gibt es mindestens einen Inferenzknoten n (mit Kindern n_1, n_2) und part. Interpret. ν.
- Seien C_1 und C_2 die mit den Fehlerknoten n_1 und n_2 assoziierten Klauseln. Diese unterscheiden sich dann (mindestens) in dem unterhalb des Fehlerknotens belegten Literal. Wenn C_1 nach x=1 falsifiziert wird, dann ist $C_1 = C \cup \{\neg x\}$, entsprechend $C_2 = D \cup \{x\}$.
  - In C und D können nur Literale vorkommen, die am Knoten n falsifiziert werden, also $\nu(C) = 0$ und $\nu(D)=0$. Es kann aber $C \subset D$ oder $D \subset C$ sein.
- Dann können C_1 und C_2 zu R resolviert werden. Wegen $\nu(C) = 0$ und $\nu(D)=0$ gilt auch $\nu(R) = \nu(C \cup D) = 0$.
- $F \cup \{R\}$ hat also einen Fehlerknoten, der entweder n ist, oder ein Vorgänger von n (falls in $C \cup D$ nicht alle in ν belegten Variablen vorkommen), und R ist die assoziierte (neue) Fehlerklausel.

Varianten der Resolution: Unit-Resolution

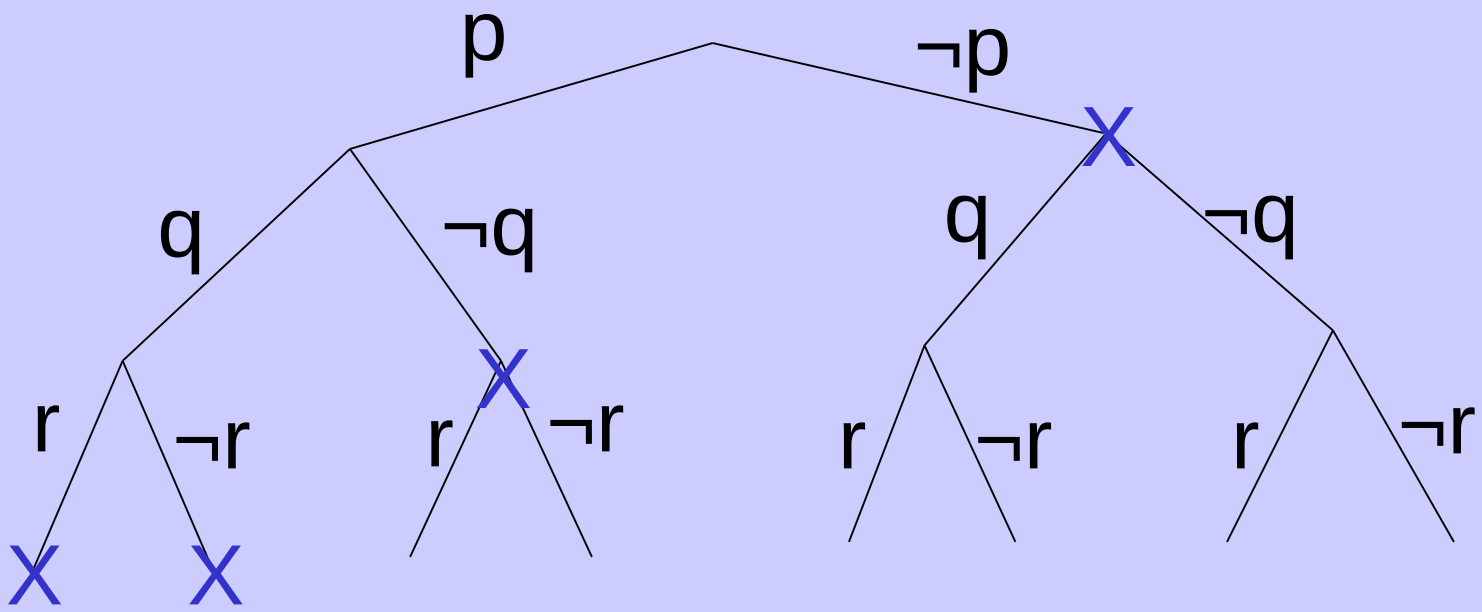
- Ziel: schränke Bildung von Resolventen ein ohne die Vollständigkeit zu verlieren
- Unit-Resolution
  - Mindestens eine Elternklausel ist Unit-Klausel
    - Aus dem Resolutionspartner wird ein Literal gelöscht
  - Unit-Resolution ist widerlegungsvollständig für Hornklauseln
  - $F \vdash_{\text{Unitres}} \square$ ist in linearer Zeit entscheidbar
  - Unit-Resolution ist nicht mehr widerlegungsvollständig
    - $\{\{a, b\}, \{a, \neg b\}, \{\neg a, b\}, \{\neg a, \neg b\}\} \models \square$

Varianten der Resolution: Ordered Resolution

- Geordnete Resolution
  - Voraussetzung: strikte, totale Ordnung $\prec$ auf Variablen
  - Literal, über das resolviert wird, muss in jeder Elternklausel maximal sein.
  - Geordnete Resolution ist widerlegungs-vollständig
    - Baue semantischen Baum mit kleinster Variable an der Wurzel, größter Variable an den Blättern
    - Falls $\mathcal{F}$ unerfüllbar gibt es einen Inferenzknoten n.
    - Die entsprechende Resolution an n ist zulässig, da sie über ein maximales Literal resolviert. (Die beteiligten Klauseln können kein größeres Literal beinhalten, da sie sonst nicht unmittelbar unterhalb n schon falsifiziert würden. Denn größere Literale werden weiter abwärts im Baum belegt und dort sind für beide möglichen Belegungen Zweige enthalten.)

Ordered Resolution: Vollständigkeit

S = {{p}, {¬p,q}, {¬r}, {¬p,¬q,r}}



Im ersten Schritt einzig zulässig: R({¬p,¬q,r},{¬r})={¬p,¬q}
Im zweiten Schritt einzig zulässig: R({¬p,¬q},{¬p,q})={¬p}
Im dritten Schritt einzig zulässig: R({¬p},{p})={}

Tautologie-Beweise: backward dual (BD) resolution

- Backward dual resolution (BD Resolution) arbeitet „rückwärts“ und auf Termen statt Klauseln (dual)
- Seien 2 **Terme** $T_1 = A \wedge x$ und $T_2 = B \wedge \neg x$ mit komplementären Literalen x und ¬x gegeben. Dann ist $D = A \wedge B$ die ***BD-Resolvente*** von T_1 und T_2 und es gilt:
  1. $D \models T_1 \vee T_2$.
  2. Also gilt $T_1 \vee T_2 \equiv T_1 \vee T_2 \vee D$
  3. Die BD-Resolvente von $T_1 = x$ und $T_2 = \neg x$ ist $\top$, denn $\top \models x \vee \neg x$.
- Beweis: Sei b eine Belegung mit b(D)=1.
  1. falls b(x)=1, dann ist $b(T_1)=1$, falls b(x)=0, dann ist $b(T_2)=1$
  2. „⇒“ ist klar. „⇐“: falls b(D)=1, so auch $b(T_1 \vee T_2)=1$.
- Satz: falls $A \vdash^*_{BDres} B$, dann $B \models A$
  - Beweis: Induktion.
- Korollar: falls $B \vdash^*_{BDres} \top$, dann $\models B$

BD-Resolution: Vollständigkeit

- Eine Formel F heißt **Tautologie** gdw. $\models F$.
- Satz: BD Resolution ist tautologie-vollständig
  - Falls $\models D$, dann $D \vdash^*_{BDres} T$
- Beweis über (duale) semantische Bäume
  - Sei D eine aussagenlogische Formel in DNF und $\models D$.
  - Bilde einen semant. Baum. An jedem Ast gibt es einen (höchsten) Knoten, an dem ein Term von D wahr wird. Dieser Term schließt den Ast ab.
    - Andernfalls gäbe es einen Ast (= Belegung), die keinen Term von D wahr macht, also D falsifiziert.
  - Entweder ist die Wurzel schon mit T markiert, fertig.
  - Oder es muss mindestens einen Knoten K mit 2 Kindern geben, die mit Termen K_1 und K_2 markiert sind. Zu diesen gibt es eine BD-Resolvente, die K oder einen Knoten oberhalb von K wahr macht.
  - Per Induktion ist nach endl. vielen Schritten die Wurzel wahr (mit Term T).

BD-Resolution: Vollständigkeit

- BD-Resolution ist nicht vollständig.
  - Bsp: $x \wedge \neg x \models y$, aber $x \wedge \neg x$ ist nicht durch BD-Resolution aus y ableitbar, denn es gibt keine einzige Resolvente.
  - wohl aber gilt $\{\neg x \vee x \vee y\} \vdash^*_{Res} \top$

BD-Resolution: Prim-Implikanten

- Def.: Ein **Implikant** einer Formel F ist ein **Term** D mit $D \models F$
- Def. (W. Quine): Ein **Prim-Implikant** P ist ein Implikant, in dem keine überflüssigen Literale enthalten sind. Streicht man in einem Prim-Implikanten P irgend ein Literal, so ist P kein Implikant mehr.
  - Jeder Implikant $D \models F$ ist entweder prim oder er hat einen Prim-Implikanten $P \models F$ als Teil-Term: $D = P \wedge Q$.
    - Streiche in D alle überflüssigen Literale solange noch $D' \models F$. Das Resultat $P \models F$ ist prim.
  - Erinnerung: Falls $D \models F$, dann $F \equiv F \vee D$
- Prim-Implikanten sind nützlich, weil sie kurz sind
  - sie helfen, eine Formel zu minimieren (boolsche Fkt. kompakt darzustellen)
  - Erinnerung: Absorptionsgesetz: $(A \wedge B) \vee A \equiv A$
  - Erzeuge einen Prim-Implikanten A und lösche alle Terme, die A enthalten
  - Bsp.: $F = (A \wedge B) \vee (A \wedge \neg B) \vee C$

    A und C sind (die einzigen) Prim-Implikanten. $A \equiv (A \wedge B \vee A \wedge \neg B)$, $F \equiv A \vee C$.

BD-Resolution: Vollständigkeit

- Satz: BD Resolution ist vollständig bezüglich Prim-Implikanten
  - Für jeden Prim-Implikanten D von F gilt $F \vdash^*_{BDres} D$
- Beweis über (duale) semantische Bäume
  - Sei D ein Prim-Implikant von F. Seien Var(D) die Variablen in D.
  - Bilde einen semantischen Baum S für F, in dem die Variablen in Var(D) höher stehen als die Variablen in Var(F) \ Var(D). Dann gibt es genau einen höchsten Knoten K in S, an dem D wahr wird.
  - Jede Belegung, die D wahr macht, ist ein Ast, der durch den Knoten K geht. Da $D \models F$, gibt es auf jedem solchen Ast einen (höchsten) Knoten, an dem ein Term von DNF(F) wahr wird.
    - Falls D Element von DNF(F), so ist dies der Knoten K selbst, es sind 0 Resolutionen nötig.
    - Andernfalls muss es im Teilbaum mit Wurzel K mindestens einen Knoten K´ mit 2 Kindern geben, die mit Termen T_1 und T_2 markiert sind. Zu diesen gibt es eine BD-Resolvente, die K´ oder einen Knoten oberhalb von K´ wahr macht.
  - Per Induktion ist nach endlich vielen Schritten die Markierung D der Wurzel K erreicht. Eine kürzere Markierung als D gibt es nicht, da D schon prim ist. (K kann nicht „übersprungen“ werden.)

DNF-Minimierung mit BD-Resolution

- Da man mit BD-Resolution alle Prim-Implikanten einer Formel in DNF bilden kann, erhält man mit BD-Resolution ein **Minimierungsverfahren** analog zu Quine-McCluskey.
  1. Bilde alle Prim-Implikanten von F durch BD-Res. Ausgehend von DNF(F).
  2. Suche eine (möglichst minimale) Teilmenge der Prim-Implikanten so aus, dass ihre Disjunktion äquivalent zu F ist, also $(P_1 \vee P_2 \vee \ldots \vee P_n) \equiv F$
- Bemerkungen
  - Im Gegensatz zu Quine-McCluskey kann man mit einer beliebigen DNF von F starten; man benötigt keine KDNF.
  - Die minimale Mengenauswahl in 2. ist NP-vollständig („set cover“)
  - Es gibt verschiedene Verfahren für den Äquivalenztest in 2.
    - Quine-McCluskey nutzt üblicherweise die Wertetabellen, also KDNF(F) und KDNF(P)
    - Äquivalenzbeweis z.B. auch über Absorptionsgesetz möglich. Man sucht eine Teilmenge der Primimplikanten, sodass jeder Term von DNF(F) durch ein P_i absorbiert wird. Da die P_i kurz sind, absorbiert ein P_i oft mehrere Terme der DNF(F).
  - Erinnerung: Absorptionsgesetz: $(A \wedge B) \vee A \equiv A$

DNF-Minimierung mit BD-Resolution

- Analog zur Subsumption bei Resolution, setzt man bei BD-Resolution die Absorption sofort ein, um überflüssige Terme sofort zu eliminieren
- Bildung der Prim-Implikanten durch BD-Resolution in Verbindung mit Absorption, nach den Gleichungen
  - BD-Resolventenbildung: $(A \wedge \neg x) \vee (B \wedge x) \equiv (A \wedge \neg x) \vee (B \wedge x) \vee (A \wedge B)$
  - Absorption: $(A \wedge B) \vee A \equiv A$
  - Wiederhole: {Bilde Resolventen; Absorbiere;}

    bis keine neuen Terme mehr erzeugt werden.
- Minimierung von F
  - Bilde die Prim-Implikanten durch BD-Resolution mit Absorption
  - Eliminiere überflüssige Implikanten durch Beschränkung auf eine minimale Teilmenge der Implikanten, die die DNF(F) absorbieren.

DNF-Minimierung mit BD-Resolution

- Im Allgemeinen sind nicht alle Prim-Implikanten nötig, um alle Terme der DNF(F) zu absorbieren.
  - Bsp.: Sei F = ab + a´c. Prim-Implikanten sind ab, a´c, bc.

    Es ist F ≡ ab + a´c, der Implikant bc wird nicht benötigt.
- Gesucht ist eine (absolut) minimale Teilmenge der Prim-Implikanten
  - dieses Problem ist NP-vollständig, da es i.A. mehrere Minima gibt
- Greedy-Lösung mit Auswahl-Heuristik
  - wähle den Prim-Implikanten, der die meisten Terme von F absorbiert und streiche diese weg (siehe Bsp. oben).
  - wiederhole dies, bis keine Terme mehr übrig sind
    - Es können keine Terme übrig bleiben, denn diese wären ansonsten selbst Prim-Implikanten von F
  - Dann ist F als Disjunktion der benutzten Prim-Implikanten darstellbar.
    - die Lösung ist nicht notwendigerweise minimal (s.u.)

Beispiel 1: DNF-Minimierung mit BD-Resolution

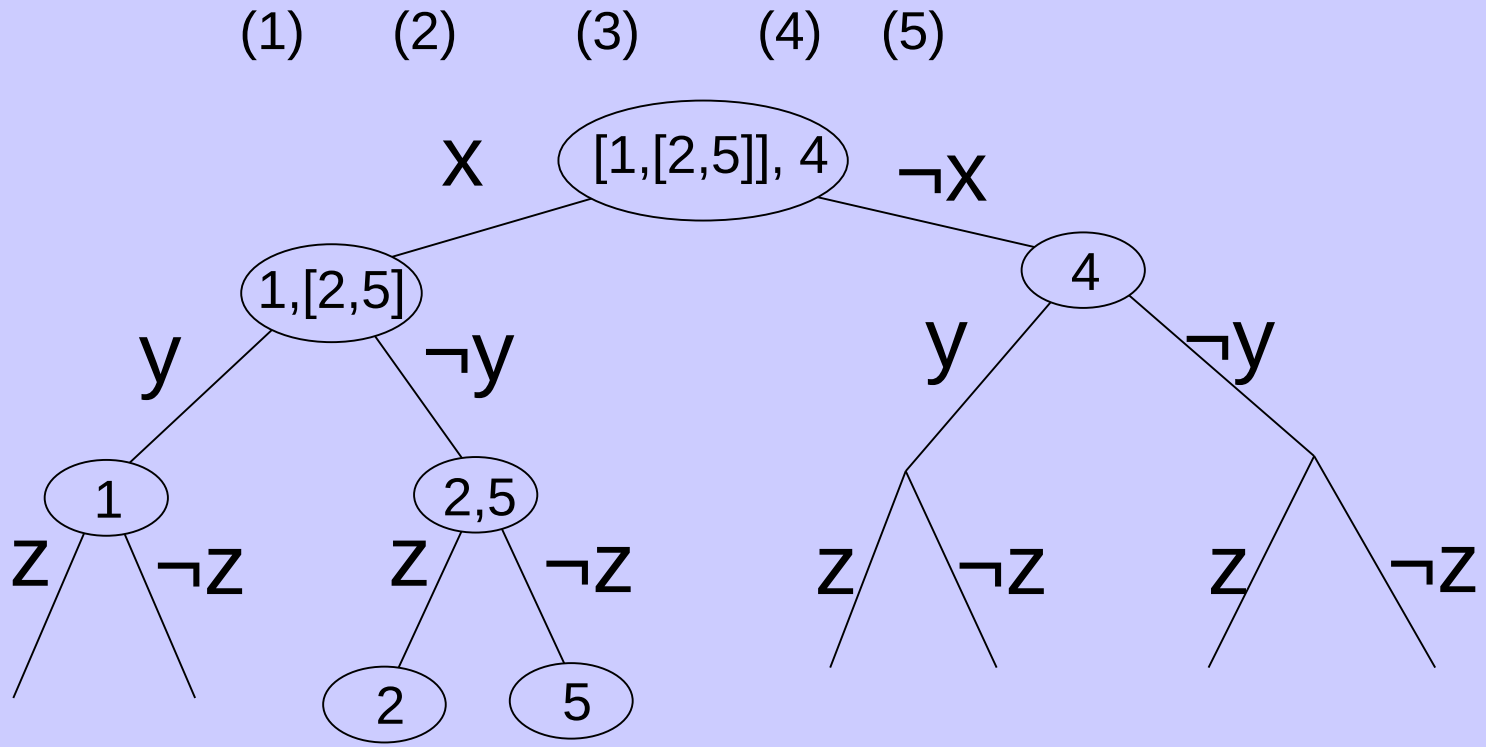
- Gegeben F = xy + xy´z + x´yz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xy; 2) xy´z; 3) x´yz´;
  - F_1: 1,2) xz; 1,3) yz´; 2,3) xx´zz´;

Beispiel 1: DNF-Minimierung mit BD-Resolution

- Gegeben F = xy + xy´z + x´yz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xy; 2) xy´z; 3) x´yz´;
  - F_1: 1,2) xz; 1,3) yz´;
  - F_2: 2,[1,3]) xzz´´; [1,2],[1,3]) xy;
- Keine weiteren Resolventen möglich
  - Prim-Implikanten sind xy, xz, yz´.
  - Also F ≡ xy + xz + yz´.
- Aber xy ist unnötig, es ist F ≡ xz + yz´
  - Denn KDNF(F) = xyz + xyz´ + xy´z + x´yz´ wird vollständig absorbiert durch xz und yz´
  - Alternativ: Denn xy ist BD-Resultante von xz und yz´, und somit xy + xz + yz´ ≡ xz + yz´

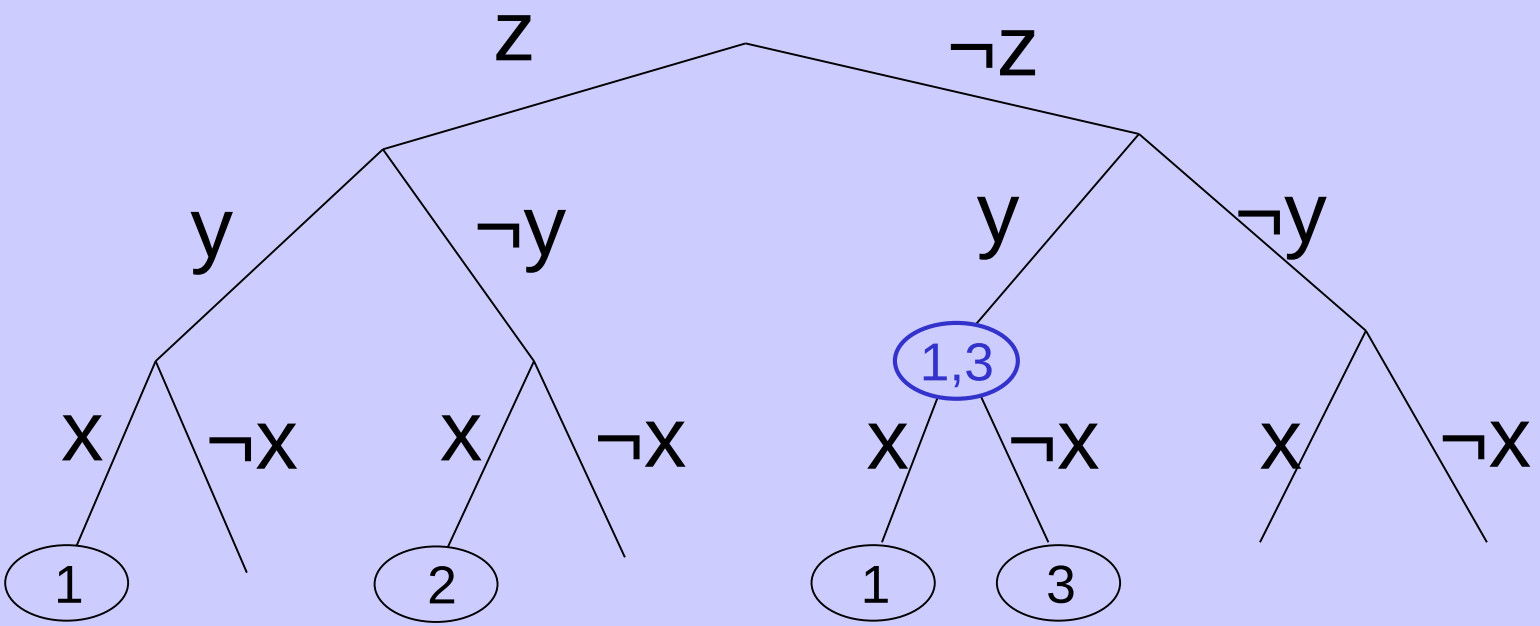
Beispiel 1: Semantische Bäume und BD-Resolution

- xz ist Prim-Implikant von F, d.h. xz ⊨ F bzw. ⊨ F + x´ + z´
- Betrachte xy + xy´z + x´yz´ + x´ + z´



Beispiel 2: Semantische Bäume und BD-Resolution

- yz´ ist Prim-Implikant von F = xy + xy´z + x´yz´
  (1) (2) (3)
- yz´ kann also durch BD-Resolution abgeleitet werden
- Betrachte Ordnung: z ≺ y ≺ x



Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
    - Absorption: 1,4) absorbiert 4); 2,3) absorbiert 3); 2,4 absorbiert 4); 3,4 absorbiert 4)

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 1,[2,3]) wxz; 1,[2,4]) wxy; 1,[3,4]) wxy;
    2,[1,3]) wz; 2,[1,4]) wy; 2,[3,4]) wz´;
    [1,2],[2,3]) wz; [1,2],[2,4]) wy; [1,2],[3,4]) wxy;
    [1,3],[2,4]) wxy; [1,3],[3,4]) wx; [1,4],[2,3]) wx;
    [2,3],[2,4]) wz´;

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 1,[2,3]) wxz; 1,[2,4]) wxy; 1,[3,4]) wxy;
    2,[1,3]) wz; 2,[1,4]) wy; 2,[3,4]) wz´;
    [1,2],[2,3]) wz; [1,2],[2,4]) wy; [1,2],[3,4]) wxy;
    [1,3],[2,4]) wxy; [1,3],[3,4]) wx; [1,4],[2,3]) wx;
    [2,3],[2,4]) wz´;
    - Absorption: 1,3) absorbiert 1,[2,3]);
      1,4) absorbiert 1,[2,4]) und 1,[3,4]) und [1,2],[3,4]) und [1,3],[2,4])
      2,[1,3]) wz absorbiert 1,2) und 1,3) und [1,2],[2,3]);
      2,[1,4]) wy absorbiert 1,4) und 2,4) und [1,2],[2,4]); 2,[3,4]) wz´ absorbiert [2,3],[2,4]);
      [1,3],[3,4]) wx absorbiert 3,4) und [1,4],[2,3])

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 1,[2,3]) wxz; 1,[2,4]) wxy; 1,[3,4]) wxy;
    2,[1,3]) wz; 2,[1,4]) wy; 2,[3,4]) wz´;
    [1,2],[2,3]) wz; [1,2],[2,4]) wy; [1,2],[3,4]) wxy;
    [1,3],[2,4]) wxy; [1,3],[3,4]) wx; [1,4],[2,3]) wx;
    [2,3],[2,4]) wz´;
  - F_3: 1,[2,[3,4]]) wxz; 2,[[1,3],[3,4]]) w; ….

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 1,[2,3]) wxz; 1,[2,4]) wxy; 1,[3,4]) wxy;
    2,[1,3]) wz; 2,[1,4]) wy; 2,[3,4]) wz´;
    [1,2],[2,3]) wz; [1,2],[2,4]) wy; [1,2],[3,4]) wxy;
    [1,3],[2,4]) wxy; [1,3],[3,4]) wx; [1,4],[2,3]) wx;
    [2,3],[2,4]) wz´;
  - F_3: 1,[2,[3,4]]) wxz; 2,[[1,3],[3,4]]) w; ….
    - Absorption: w absorbiert jeden Term mit Variable w. Keine weiteren Resolventen möglich.

Beispiel 2: DNF-Minimierung mit BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der BD-Resolventen
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 1,[2,3]) wxz; 1,[2,4]) wxy; 1,[3,4]) wxy;
    2,[1,3]) wz; 2,[1,4]) wy; 2,[3,4]) wz´;
    [1,2],[2,3]) wz; [1,2],[2,4]) wy; [1,2],[3,4]) wxy;
    [1,3],[2,4]) wxy; [1,3],[3,4]) wx; [1,4],[2,3]) wx;
    [2,3],[2,4]) wz´;
  - F_3: 1,[2,[3,4]]) wxz; 2,[[1,3],[3,4]]) w; ….
    - Absorption: w absorbiert jeden Term mit Variable w. Keine weiteren Resolventen möglich.
- Prim-Implikanten sind xyz und w. In diesem Fall werden beide gebraucht, um alle Terme von F zu absorbieren (abzudecken).
- Insgesamt ist F ≡ xyz + w.

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
    - Absorption: 1,4) absorbiert 4);

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 3,[1,4]) wx

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 3,[1,4]) wx
    - 3,[1,4]) absorbiert 3) und 1,4)

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 3,[1,4]) wx
  - F_3: 2,[3,[1,4]]) w

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 3,[1,4]) wx
  - F_3: 2,[3,[1,4]]) w
    - 2,[3,[1,4]]) w absorbiert 2) und 3,[1,4])

Beispiel 3: DNF-Minimierung mit ordered BD-Resolution

- Gegeben F = xyz + wx´ + wxy´ + wxyz´. („TI-Schreibweise“)
- Berechnung der geordneten BD-Resolventen mit w<x<y<z
  - F_0: 1) xyz; 2) wx´; 3) wxy´; 4) wxyz´
  - F_1: 1,2) wyz; 1,3) wxz; 1,4) wxy; 2,3) wy´; 2,4) wyz´ 3,4) wxz´;
  - F_2: 3,[1,4]) wx
  - F_3: 2,[3,[1,4]]) w
- w ist offensichtlich Prim-Implikant.
- Geordnete BD-Resolution ist nicht vollständig bezgl. Prim-Implikanten-Bildung
  - Bsp.: Sei F = ab + a´c, sei a<b<c. Prim-Implikanten sind ab, a´c, bc; aber bc ist nicht durch geordnete Resolution herleitbar.
- Da aber in Bsp.2 auch keine allgemeinen BD-Resolventen mehr möglich sind ist auch xyz ein Prim-Implikant
- Insgesamt ist F ≡ xyz + w.

Bsp 4 zu BD-Resolution: Vollständigkeit eines Regelsystems

- Expertensysteme sind regelbasierte Systeme (RBS) zur Entscheidungsfindung
- Boolesche RBS haben die Form $\{<B_i \rightarrow A_i>\}$
  - die B_i sind aussagenlog. Bedingungen, die A_i sind irgendwelche Aktionen
- Grundfrage: deckt das RBS den Entscheidungsraum vollständig ab? (Wird jedes Problem durch das RBS gelöst?)
- Beispiel: Finden der Teile zu einem bestellten Fahrzeug durch eine regelbasierte Stückliste
  - zu jeder geometrischen Einbauposition im Fahrzeug wird eine Regelmenge definiert, die die alternativen Teile (z.B. Radios) und deren Einbaubedingungen beschreibt (z.B.: falls Großbritannien, dann Digitalradio)
  - Jede Regel hat die Form <Einbaubedingung → Teil>
  - Die Frage ist: wird für jedes Fahrzeug ein Teil gefunden?

Bsp 4 zu BD-Resolution: Vollständigkeit eines Regelsystems

- Mit *jeder* Variablenbelegung (Auftrag) wird ein Teil getroffen:
  - $T \equiv B_1 \vee \ldots \vee B_m$
- Jeder Auftrag außer dem „Null“-Auftrag bekommt ein Teil:
  - $\models x_1 \vee \ldots \vee x_n \rightarrow B_1 \vee \ldots \vee B_m$
- Baubare (zulässige) Aufträge sind Lösungen der Formel PÜF. Jeder baubare Auftrag muss ein Teil treffen:
  - $\models PÜF \rightarrow B_1 \vee \ldots \vee B_m$
- Einschränkung: jeder baubare Auftrag mit Motor M1 oder M2 muss ein Teil treffen:
  - $\models PÜF \wedge (M1 \vee M2) \rightarrow B_1 \vee \ldots \vee B_m$
- Ein *over release* ist eine Bedingung, die öfter als nötig zutrifft. Vermeidung von *over releases* bezüglich M1:
  - $\models PÜF \wedge M1 \leftrightarrow B_1 \vee \ldots \vee B_m$ bzw. $PÜF \wedge M1 \equiv B_1 \vee \ldots \vee B_m$

Zusammenfassung

- Sei F ≡ (A ∨ p) ∧ (B ∨ ¬p) ∧ R (wobei p in A, B, R nicht enthalten)
  - diese Darstellung ist immer erreichbar
- Davis-Putnam (*elimination rule*): F ≅ (A ∨ B) ∧ R
  - p ist in (A ∨ B) ∧ R nicht mehr enthalten (eliminiert)
  - F inkonsistent gdw. (A ∨ B) ∧ R inkonsistent
    - Formel wird (sehr viel) größer, aber Fortschritt, weil p eliminiert ist.
- Davis-Logemann-Loveland *(splitting rule)*:
  - F inkonsistent gdw. sowohl (A ∧ R) als auch (B ∧ R) inkonsistent
    - kein erneutes Herstellen von CNF nötig, Formel wird einfacher, divide&conquer
- SAT-Solving *(splitting rule)*:
  - F inkonsistent gdw. sowohl $F|_{p=0}$ als auch $F_{|p=1}$ inkonsistent
- Resolution: (A ∨ p)∧(B ∨ ¬p)∧R ≡ (A ∨ p)∧(B ∨ ¬p)∧(A ∨ B)∧R
- BD-Resolution: (A∧p) ∨ (B∧¬p) ∨ R ≡ (A∧p) ∨ (B∧¬p) ∨ (A∧B) ∨ R
  - F tautologisch gdw. (A∧B) ∨ R tautologisch.