WS 2015/16

Diskrete Strukturen

Kapitel 2: Aussagenlogik (2)

Hans-Joachim Bungartz

Lehrstuhl für wissenschaftliches Rechnen
Fakultät für Informatik
Technische Universität München

http://www5.in.tum.de/wiki/index.php/Diskrete_Strukturen_-_Winter_15

Kapitel 2: Grundlagen (Aussagenlogik 2)

- Mathematische und notationelle Grundlagen
  - Mengen
  - Relationen und Abbildungen
  - **Aussagen- und Prädikatenlogik**
  - Beweismethoden
  - Wachstum von Funktionen

- Normalformen:
  - Ein Literal ist eine Aussagenvariable oder die Negation einer Aussagenvariable.
  - Eine Klausel ist eine Disjunktion von Literalen oder die Formel **false** (Disjunktion von 0 Literalen).
  - Eine Formel in konjunktiver Normalform (KNF) ist
    - eine Konjunktion von Klauseln

      (d.h. eine Formel der Form $(\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m_i} L_{i,j}))$

      mit $L_{i,j} \in \{A_1, A_2, \ldots\} \cup \{\neg A_1, \neg A_2, \ldots\}$).
    - oder die Formel $\mathbf{true}$ (Konjunktion von 0 Klauseln).

Kapitel 2: Grundlagen (Aussagenlogik 2)

- Normalformen:
  - Eine Formel ist in KNF, falls in allen Pfaden ihres Syntaxbaums Konjunktionen vor Disjunktionen vor Negationen vorkommen.

- Normalformen: Umformungsmethode
  - Formeln aus der Praxis sind oft in KNF.
  - Außerdem: Das folgende Verfahren konstruiert für eine beliebige Formel F eine äquivalente KNF-Formel:

1. Ersetze in F jedes Vorkommen einer Teilformel der Bauart

$$\neg\neg G \quad \text{durch} \quad G,$$
$$\neg(G \wedge H) \quad \text{durch} \quad (\neg G \vee \neg H),$$
$$\neg(G \vee H) \quad \text{durch} \quad (\neg G \wedge \neg H),$$

bis keine derartige Teilformel mehr vorkommt.

2. Ersetze jedes Vorkommen einer Teilformel der Bauart

   $(F \vee (G \wedge H))$ durch $((F \vee G) \wedge (F \vee H))$,

   $((F \wedge G) \vee H)$ durch $((F \vee H) \wedge (G \vee H))$,

   bis keine derartige Teilformel mehr vorkommt.

Alle Schritte erhalten Äquivalenz.

Nach Schritt 1 kommen im Syntaxbaum Konjunktionen und Disjunktionen vor Negationen vor.

Nach Schritt 2 kommen im Syntaxbaum Konjunktionen vor Disjunktionen vor.

- Algorithmen für KNF-SAT:
  - Wir kennen bisher nur den folgenden Algorithmus:
    - Erzeuge alle minimalen Belegungen, die zur Formel passen.
    - Für jede solche Belegung prüfe, ob sie die Formel erfüllt.
  - Zwei Probleme:
    - Es müssen 2^n Belegungen geprüft werden!!
    - Wenn die Formel unerfüllbar ist, dann werden mit Sicherheit **alle** Belegungen geprüft.

- Algorithmen für KNF-SAT:
  - Wir präsentieren zwei Algorithmen:
    - DPLL (Davis-Putnam-Logemann-Loveland):
      - Versucht, die Anzahl der zu prüfenden Belegungen zu reduzieren.
      - Auf Erfüllbarkeit gerichtet: kann früh terminieren, wenn die Formel erfüllbar ist.
      - Wir präsentieren nur eine vereinfachte Version.
    - Resolution (Robinson):
      - Auf Nicht-Erfüllbarkeit gerichtet: Kann früh terminieren, wenn die Formel nicht erfüllbar ist.

- DPLL (Davis-Putnam-Logemann-Loveland):

  Definition: Es seien F eine KNF-Formel und p eine Variable von F.

  $F[p\backslash\mathbf{true}]$ bezeichnet die Formel, die entsteht, indem jedes Vorkommnis von p in F durch $\mathbf{true}$ ersetzt wird und das Ergebnis mit Hilfe der Regeln

  $F \wedge \mathbf{true} \equiv F$ $\quad F \vee \mathbf{true} \equiv \mathbf{true}$ $\quad \neg\mathbf{true} \equiv \mathbf{false}$

  $F \wedge \mathbf{false} \equiv \mathbf{false}$ $\quad F \vee \mathbf{false} \equiv F$ $\quad \neg\mathbf{false} \equiv \mathbf{true}$

  vereinfacht wird.

  $F[p\backslash\mathbf{false}]$ wird analog definiert.

- DPLL (Davis-Putnam-Logemann-Loveland):

  Beispiel:

$$F = (p \vee \neg q \vee r) \wedge (\neg p \vee \neg r) \wedge (q \vee \neg r) \wedge p.$$

$$F[p \backslash \mathbf{true}] = (\mathbf{true} \vee \neg q \vee r) \wedge (\neg \mathbf{true} \vee \neg r) \wedge (q \vee \neg r) \wedge \mathbf{true}$$

$$\equiv (\mathbf{true} \vee \neg q \vee r) \wedge (\mathbf{false} \vee \neg r) \wedge (q \vee \neg r)$$

$$\equiv \mathbf{true} \wedge \neg r \wedge (q \vee \neg r)$$

$$\equiv \neg r \wedge (q \vee \neg r).$$

- DPLL (Davis-Putnam-Logemann-Loveland):

  Beispiel:

$$F = (p \vee \neg q \vee r) \wedge (\neg p \vee \neg r) \wedge (q \vee \neg r) \wedge p$$

$$F[p \backslash \mathbf{false}] =$$

$$= (\mathbf{false} \vee \neg q \vee r) \wedge (\neg \mathbf{false} \vee \neg r) \wedge (q \vee \neg r) \wedge \mathbf{false}$$

$$\equiv \mathbf{false}.$$

- DPLL (Davis-Putnam-Logemann-Loveland):

  Fakt: F ist erfüllbar gdw. $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ erfüllbar sind.

  Begründung.

  Sei β Belegung mit $[F](\beta) = 1$.

  Wenn $\beta(p) = 1$, dann gilt $[F](\beta) = \left[F[p\backslash\mathbf{true}]\right](\beta) = 1$.

  Wenn $\beta(p) = 0$, dann gilt $[F](\beta) = \left[F[p\backslash\mathbf{false}]\right](\beta) = 1$.

  Sei β Belegung mit $[F[\mathrm{p}\backslash\mathbf{true}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 1$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $[F](\beta') = 1$.

  Sei β Belegung mit $[F[\mathrm{p}\backslash\mathbf{false}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 0$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $[F](\beta') = 1$.

- DPLL (Davis-Putnam-Logemann-Loveland):

  Algorithmus 1:

  Wenn $F = \mathbf{true}$, dann antworte „erfüllbar“.

  Wenn $F = \mathbf{false}$, dann antworte „unerfüllbar“.

  Wenn $\mathbf{true} \neq F \neq \mathbf{false}$, dann

  wähle eine Variable p, die in F vorkommt;

  prüfe, ob $F[p \backslash \mathbf{true}]$ oder $F[p \backslash \mathbf{false}]$ erfüllbar sind;

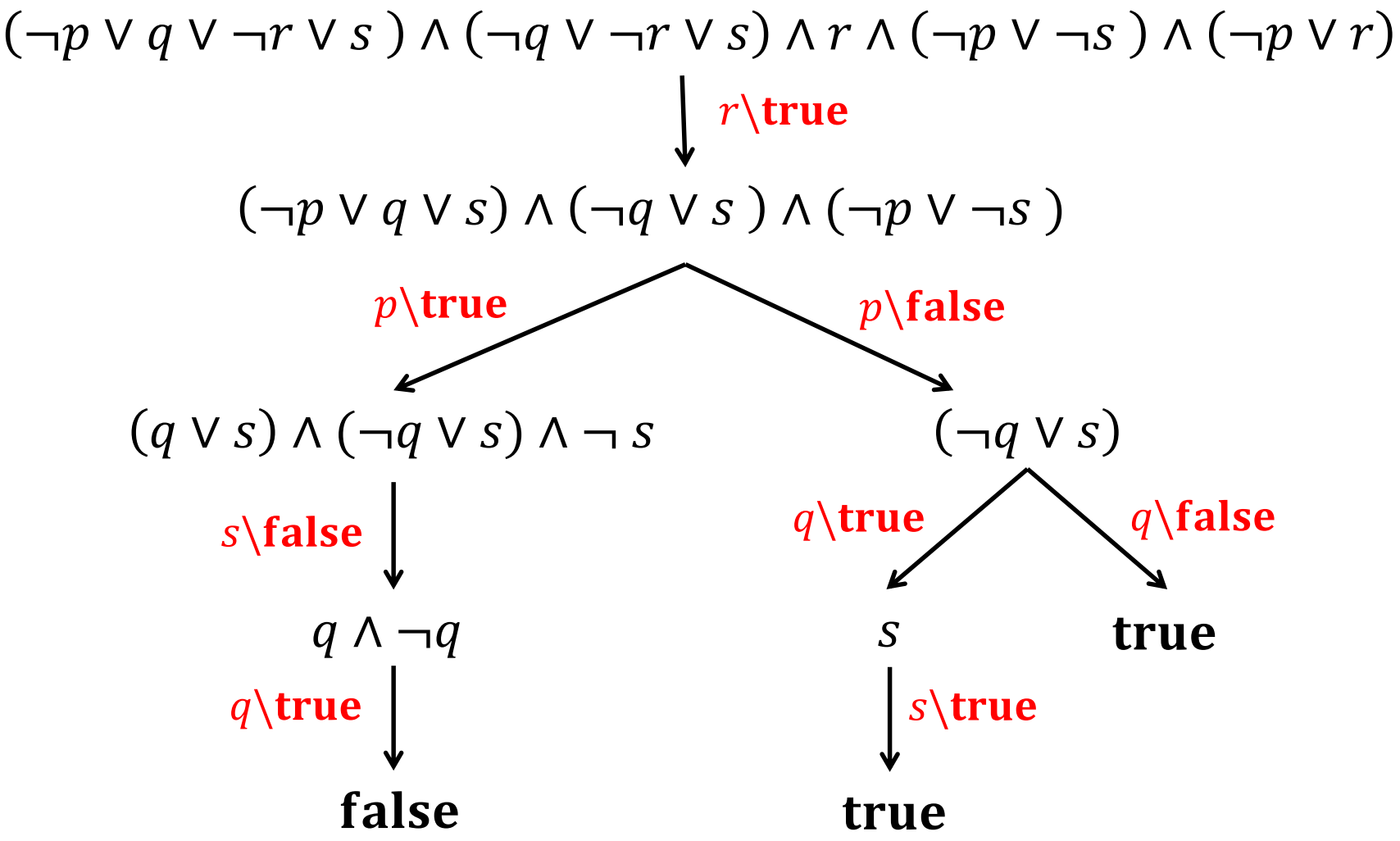
  wenn mindestens eine von den beiden erfüllbar ist, antworte „erfüllbar“, sonst „unerfüllbar“.

- DPLL (Davis-Putnam-Logemann-Loveland):
  - Algorithmus 1 kann durch eine geschickte Wahl der Variablen p beschleunigt werden.
  - Eine Ein-Literal-Klausel ist eine Klausel, die nur ein Literal enthält, d.h., eine Klausel der Gestalt $\{p\}$ oder der Gestalt $\{\neg p\}$.

- DPLL (Davis-Putnam-Logemann-Loveland):

  Algorithmus 2:

  Wenn $F = \mathbf{true}$, dann antworte „erfüllbar“.
  Wenn $F = \mathbf{false}$, dann antworte „unerfüllbar“.
  Wenn $\mathbf{true} \neq F \neq \mathbf{false}$, dann
  wenn F eine Ein-Literal-Klausel $\{p\}$ enthält,
  dann prüfe, ob $F[p \backslash \mathbf{true}]$ erfüllbar ist;
  antworte „erfüllbar“ gdw. $F[p \backslash \mathbf{true}]$ erfüllbar.
  wenn F eine Ein-Literal-Klausel $\{\neg p\}$ enthält,
  dann prüfe, ob $F[p \backslash \mathbf{false}]$ erfüllbar ist;
  antworte „erfüllbar“ gdw. $F[p \backslash \mathbf{false}]$ erfüllbar.
  wenn F keine Ein-Literal-Klausel enthält,
  dann wähle eine Variable p, die in F vorkommt;
  prüfe, ob $F[p \backslash \mathbf{true}]$ oder $F[p \backslash \mathbf{false}]$ erfüllbar sind;
  wenn mindestens eine von den beiden erfüllbar ist,
  dann antworte „erfüllbar“, sonst „unerfüllbar“.


$(\neg p \vee q \vee \neg r \vee s) \wedge (\neg q \vee \neg r \vee s) \wedge r \wedge (\neg p \vee \neg s) \wedge (\neg p \vee r)$
$r \backslash \mathbf{true}$
$(\neg p \vee q \vee s) \wedge (\neg q \vee s) \wedge (\neg p \vee \neg s)$
$p \backslash \mathbf{true}$
$p \backslash \mathbf{false}$
$(q \vee s) \wedge (\neg q \vee s) \wedge \neg s$
$(\neg q \vee s)$
$s \backslash \mathbf{false}$
$q \backslash \mathbf{true}$
$q \backslash \mathbf{false}$
$q \wedge \neg q$
s
$\mathbf{true}$
$q \backslash \mathbf{true}$
$s \backslash \mathbf{true}$
$\mathbf{false}$
$\mathbf{true}$

- DPLL (Davis-Putnam-Logemann-Loveland):
  - Korrektheit des DPLL-Verfahrens:
    1. Das Verfahren terminiert für jede Eingabeformel.

       Folgt daraus, dass $F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ mindestens eine Variable weniger als F enthalten. Wenn das Verfahren nicht früher terminiert, dann kommen wir bei Formeln mit 0 Variablen an. Die einzigen solchen Formeln sind $\mathbf{true}$ und $\mathbf{false}$.
    2. Wenn die Formel erfüllbar ist, dann antwortet das Verfahren „erfüllbar“. Folgt aus Fakt.
    3. Wenn die Formel unerfüllbar ist, dann antwortet das Verfahren „unerfüllbar“. Folgt aus Fakt.
  - Ausführlicher Beweis später in der Vorlesung.

- Mengendarstellung von KNF-Formeln:
  - Eine Klausel wird durch die Menge ihrer Literale dargestellt:
    - Die Menge $\{p, \neg q, s\}$ stellt die Klausel $(p \vee \neg q \vee s)$ dar.
    - Die leere Menge stellt die Formel **false** dar.
  - Eine KNF-Formel wird durch die Menge ihrer Klauseln beschrieben (genauer: die Menge der Darstellungen ihrer Klauseln)
    - Die Klauselmenge $\{\{\neg p\}, \{p, \neg q, s\}\}$ stellt die Formel $\neg p \wedge (p \vee \neg q \vee s)$ dar.
    - Die leere Klauselmenge stellt die Formel **true** dar.

- DPLL mit Mengendarstellung:

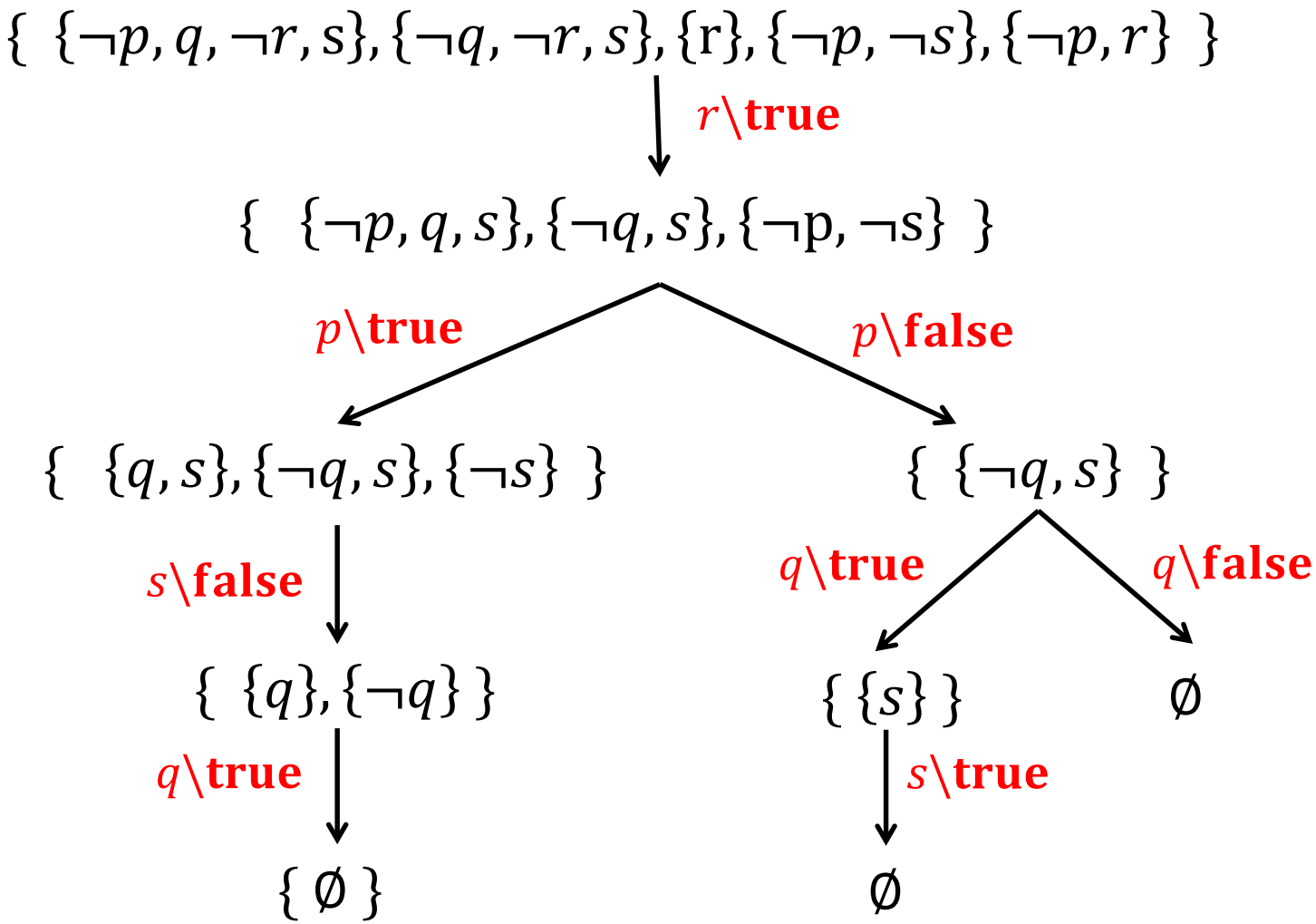
  Algorithmen 1 und 2 werden aufgrund der folgenden Eigenschaft mit Hilfe der Mengendarstellung implementiert:

  Sei F eine KNF-Formel und sei p eine Variable von F. Die Mengendarstellung von $F[p \backslash \mathbf{true}]$ entsteht, indem:

  – alle Klauseln von F, die das Literal p enthalten, gestrichen werden, und

  – in allen Klauseln, die das Literal $\neg p$ enthalten, dieses gestrichen wird.

  Die Mengendarstellung von $F[p \backslash \mathbf{false}]$ erhält man analog.


{ {¬p, q, ¬r, s}, {¬q, ¬r, s}, {r}, {¬p, ¬s}, {¬p, r} }
r\true
{ {¬p, q, s}, {¬q, s}, {¬p, ¬s} }
p\true
p\false
{ {q, s}, {¬q, s}, {¬s} }
{ {¬q, s} }
s\false
q\true
q\false
{ {q}, {¬q} }
{ {s} }
∅
q\true
s\true
{ ∅ }
∅

- Resolution:
  - Grundidee: Füge iterativ neue Klauseln zur Formel hinzu, die logische Konsequenzen der ursprünglichen Klauseln sind.
  - Beispiel: Wenn die Formel Klauseln $(\neg p \vee q \vee r)$ und $(\neg r \vee s \vee t)$ enthält, dann kann die Klausel $(\neg p \vee q \vee s \vee t)$ hinzugefügt werden.

    Die Klauseln sind äquivalent zu

    $p \rightarrow (q \vee r) \quad r \rightarrow (s \vee t) \quad p \rightarrow (q \vee s \vee t)$.

- Resolution mit Mengendarstellung:
  - Beispiel: Wenn die Formel Klauseln p und $\neg p$ enthält, dann kann die Klausel false hinzugefügt werden.

    Die Klauseln sind äquivalent zu

    $$p \qquad p \rightarrow \mathbf{false} \qquad \mathbf{false}$$

    Die neue Klausel zeigt, dass die Formel nicht erfüllt werden kann.

- Resolution mit Mengendarstellung:

  Definition: Seien K_1, K_2 und R Klauseln in Mengendarstellung. Dann heißt R resolvent von K_1 und K_2, wenn es ein Literal L gibt mit $L \in K_1$ und $\bar{L} \in K_2$ und

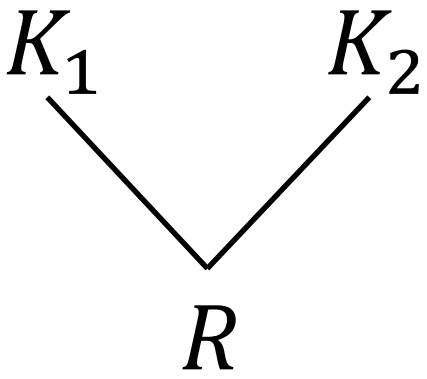
$$R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\bar{L}\})$$

  Hierbei ist $\bar{L}$ definiert als

$$\bar{L} = \begin{cases} \neg p, \text{falls } L = p; \\ p, \text{falls } L = \neg p. \end{cases}$$

- Resolution:

  Wir stellen diesen Sachverhalt durch folgendes Diagramm dar:



Die leere Klausel wird mit dem Symbol „□“ bezeichnet.

- Das Resolutionsverfahren:

while F die leere Klausel nicht enthält

{ if F zwei Klauseln K_1, K_2 enthält
mit einem Resolventen R, der $R \notin F$
erfüllt (d.h. R ist nicht Klausel von F)

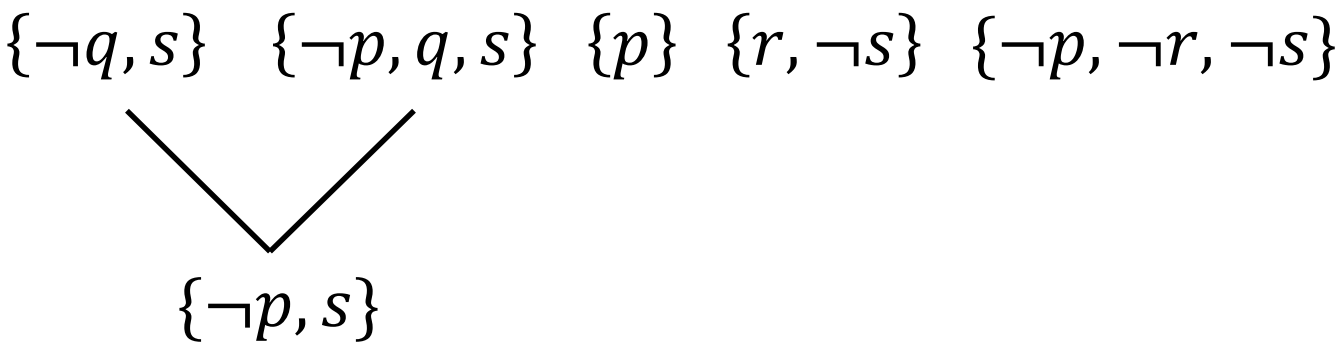
then füge R als neue Klausel zu F hinzu;

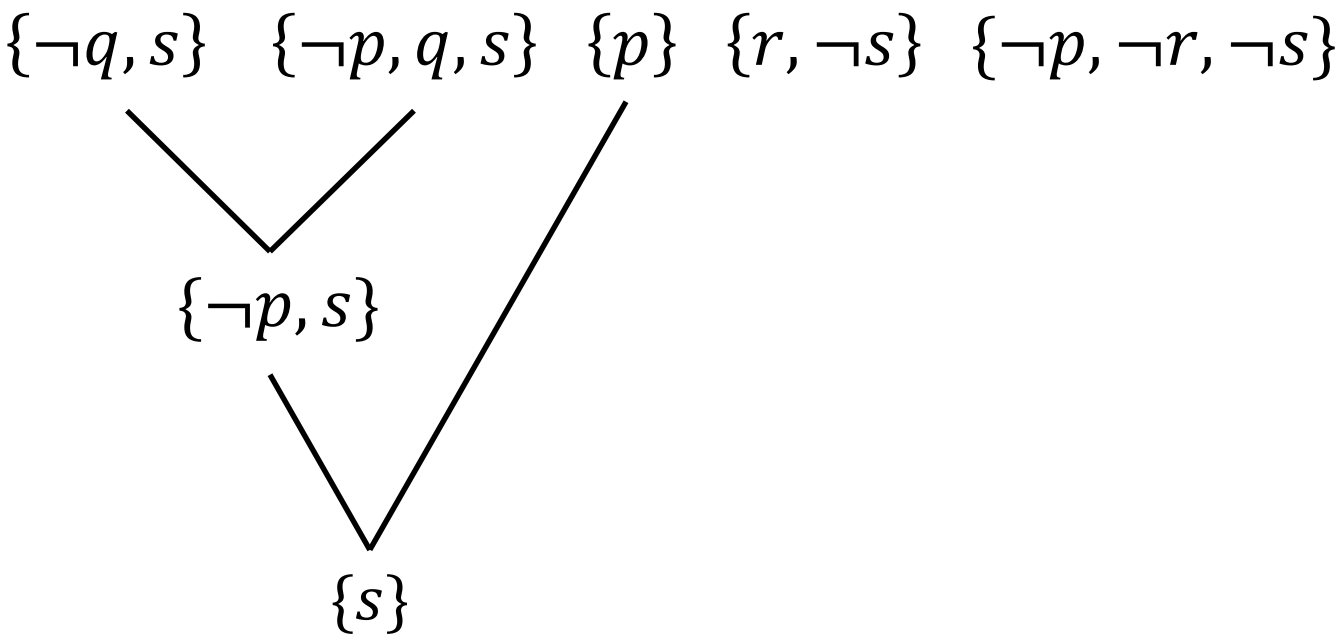
else antworte „erfüllbar“ und halte;

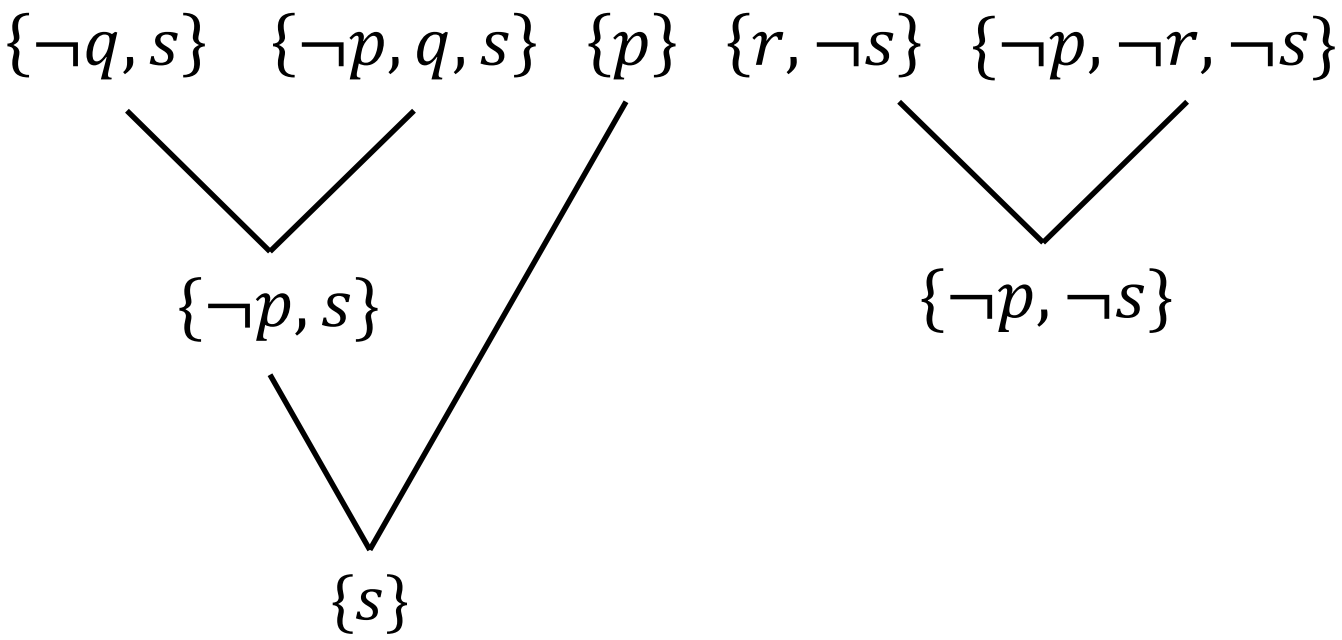
}

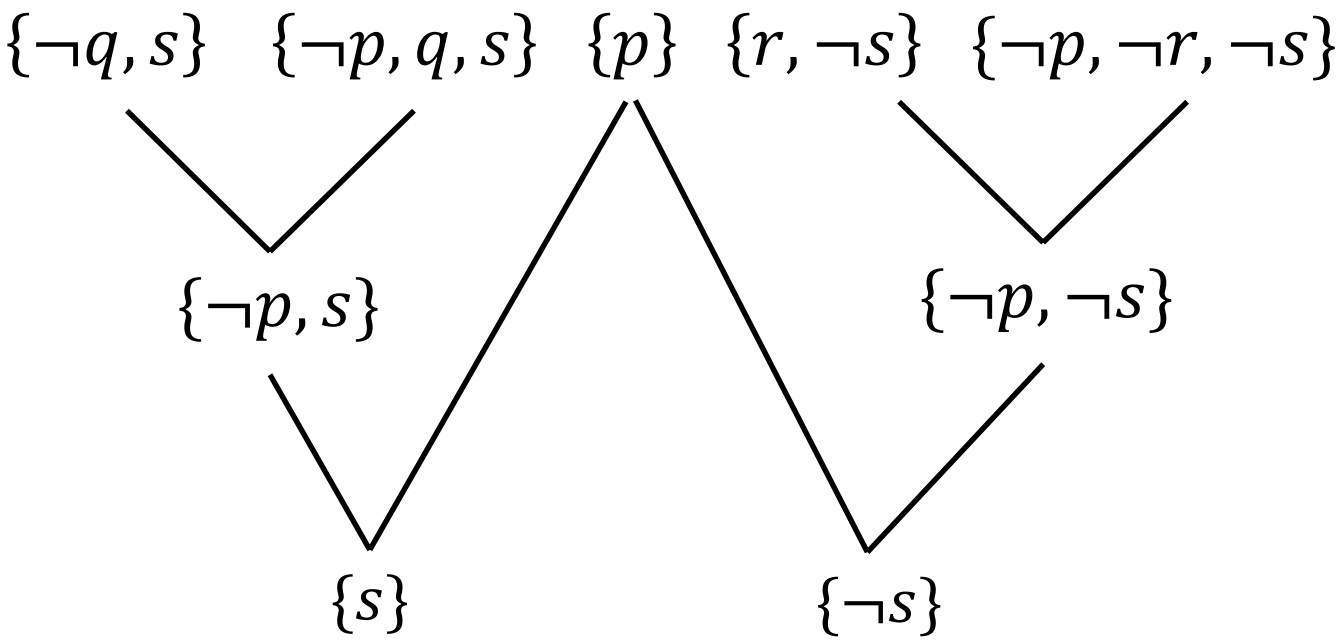
antworte „unerfüllbar“ und halte;

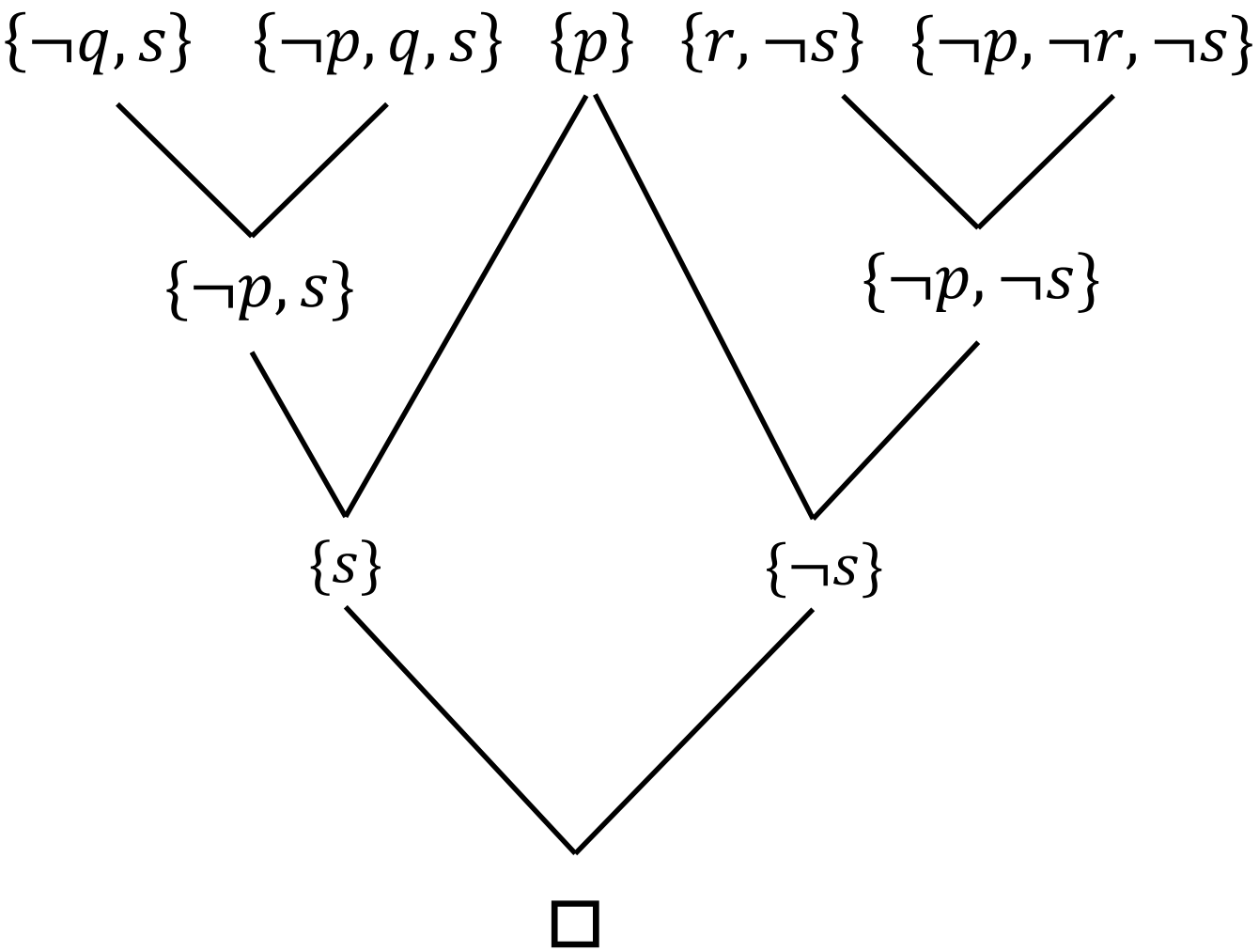
$$\{\neg q, s\} \quad \{\neg p, q, s\} \quad \{p\} \quad \{r, \neg s\} \quad \{\neg p, \neg r, \neg s\}$$


{¬q, s}
{¬p, q, s}
{p}
{r, ¬s}
{¬p, ¬r, ¬s}
{¬p, s}

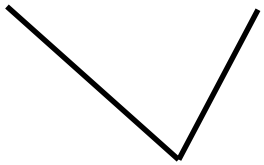

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{s\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{\neg p, \neg s\}$
$\{s\}$

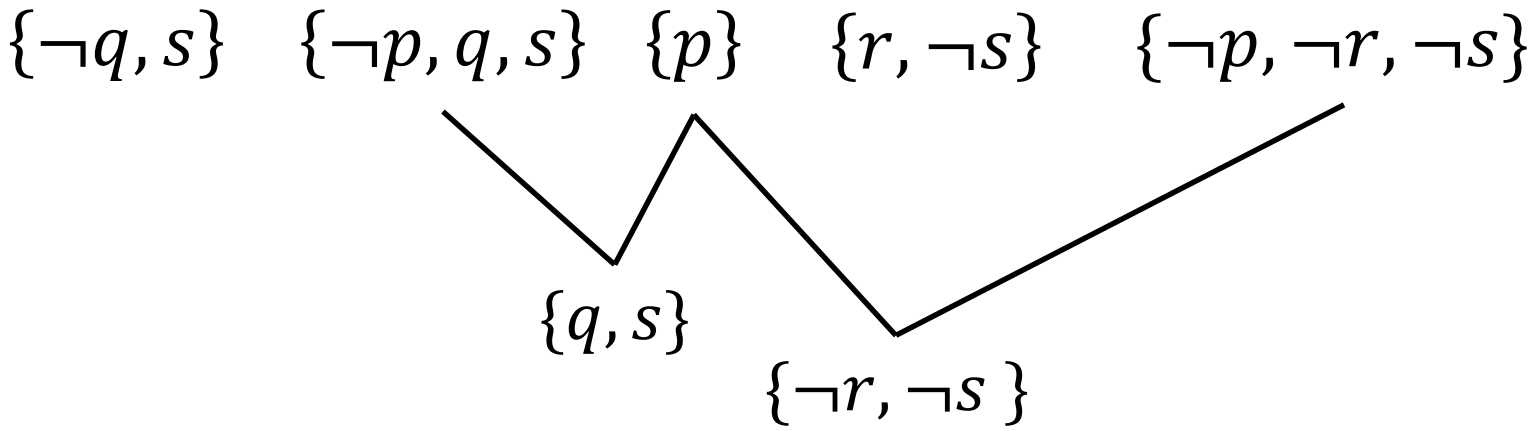

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{\neg s\}$

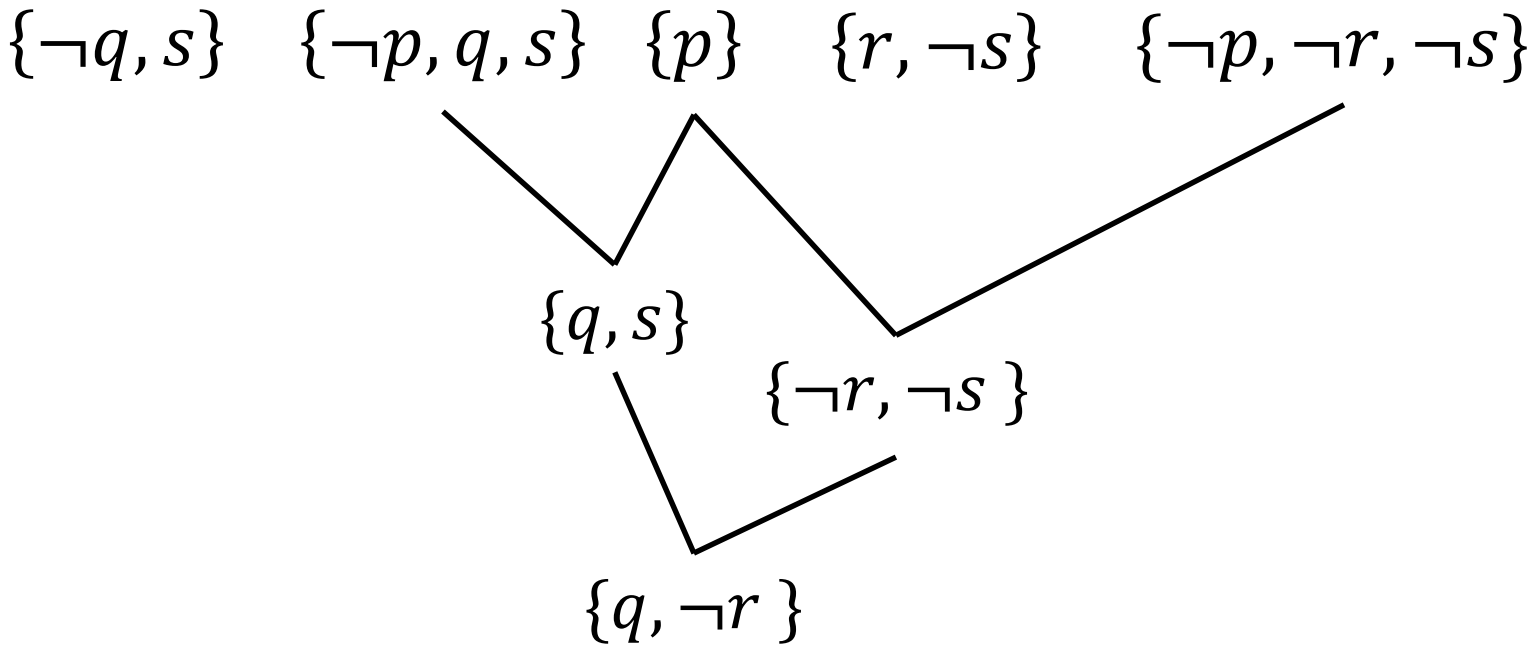

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{\neg s\}$
$\square$

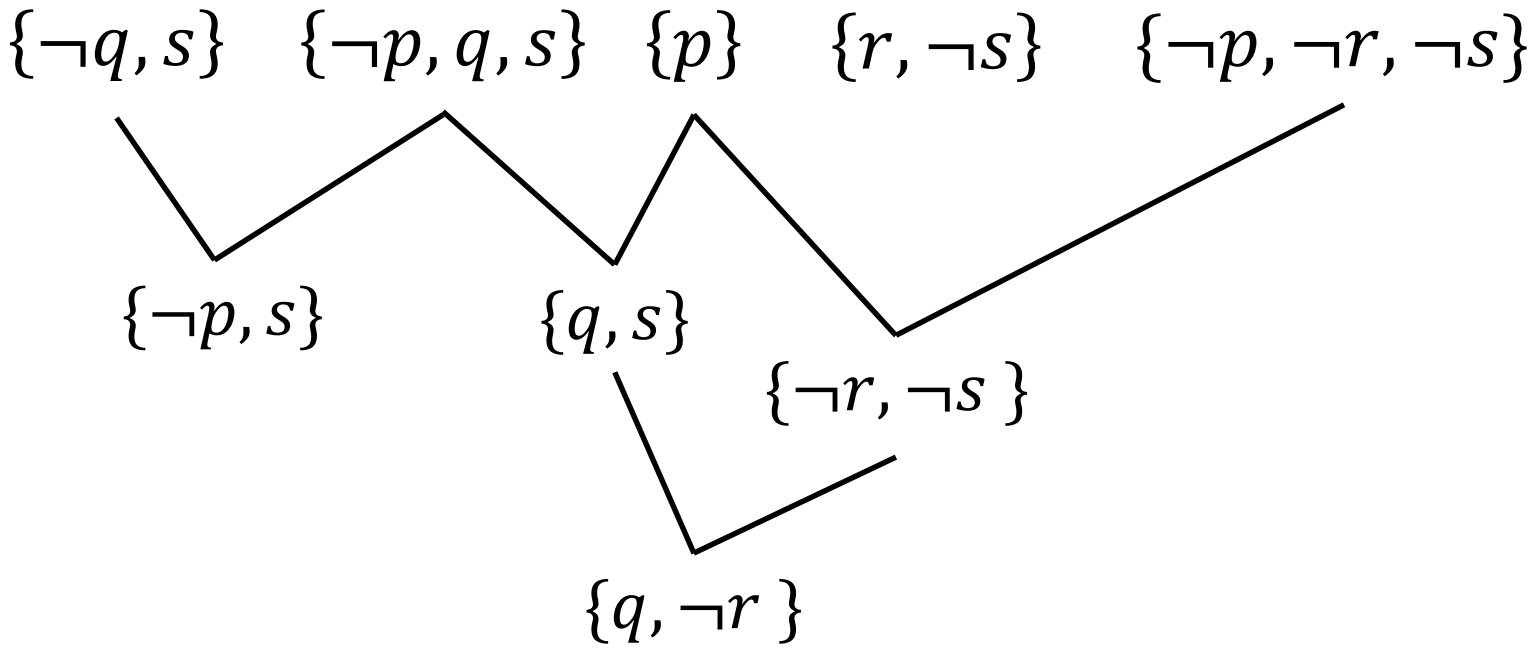
$\{\neg q, s\}$ $\{\neg p, q, s\}$ $\{p\}$ $\{r, \neg s\}$ $\{\neg p, \neg r, \neg s\}$

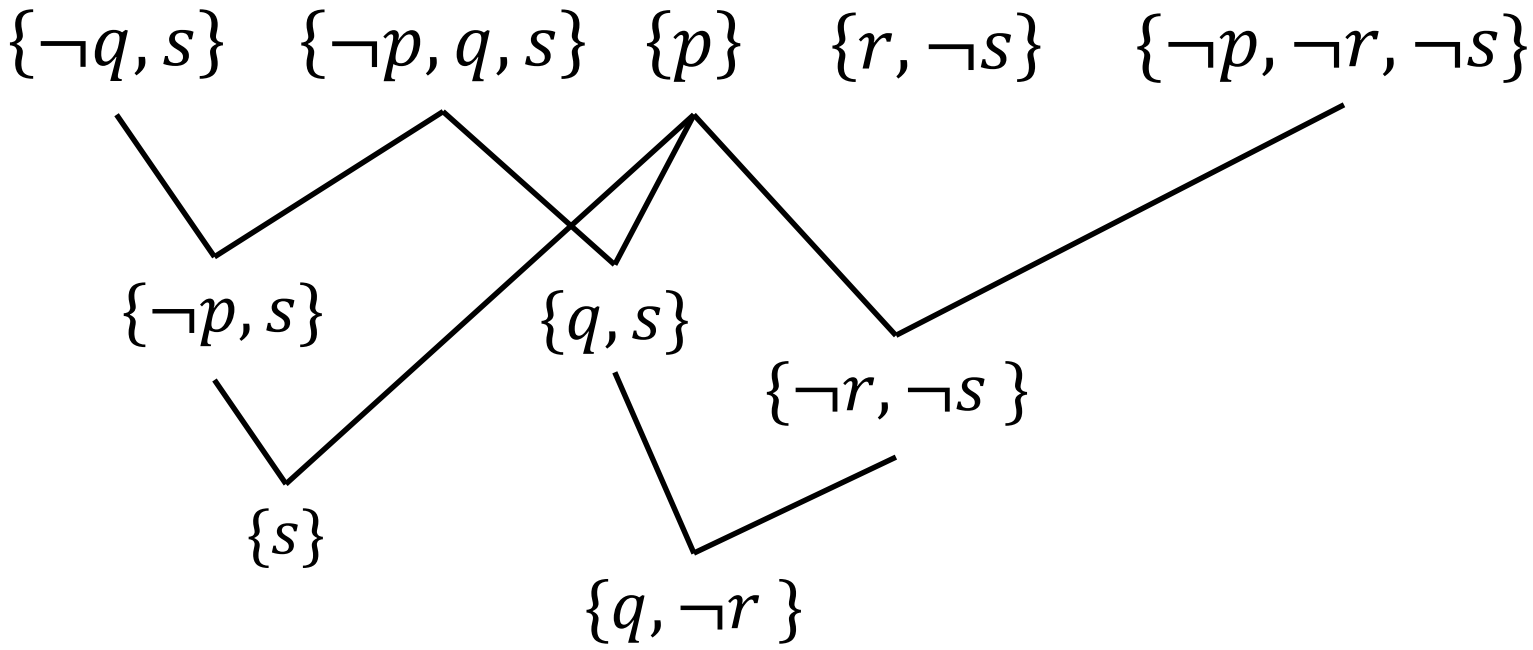


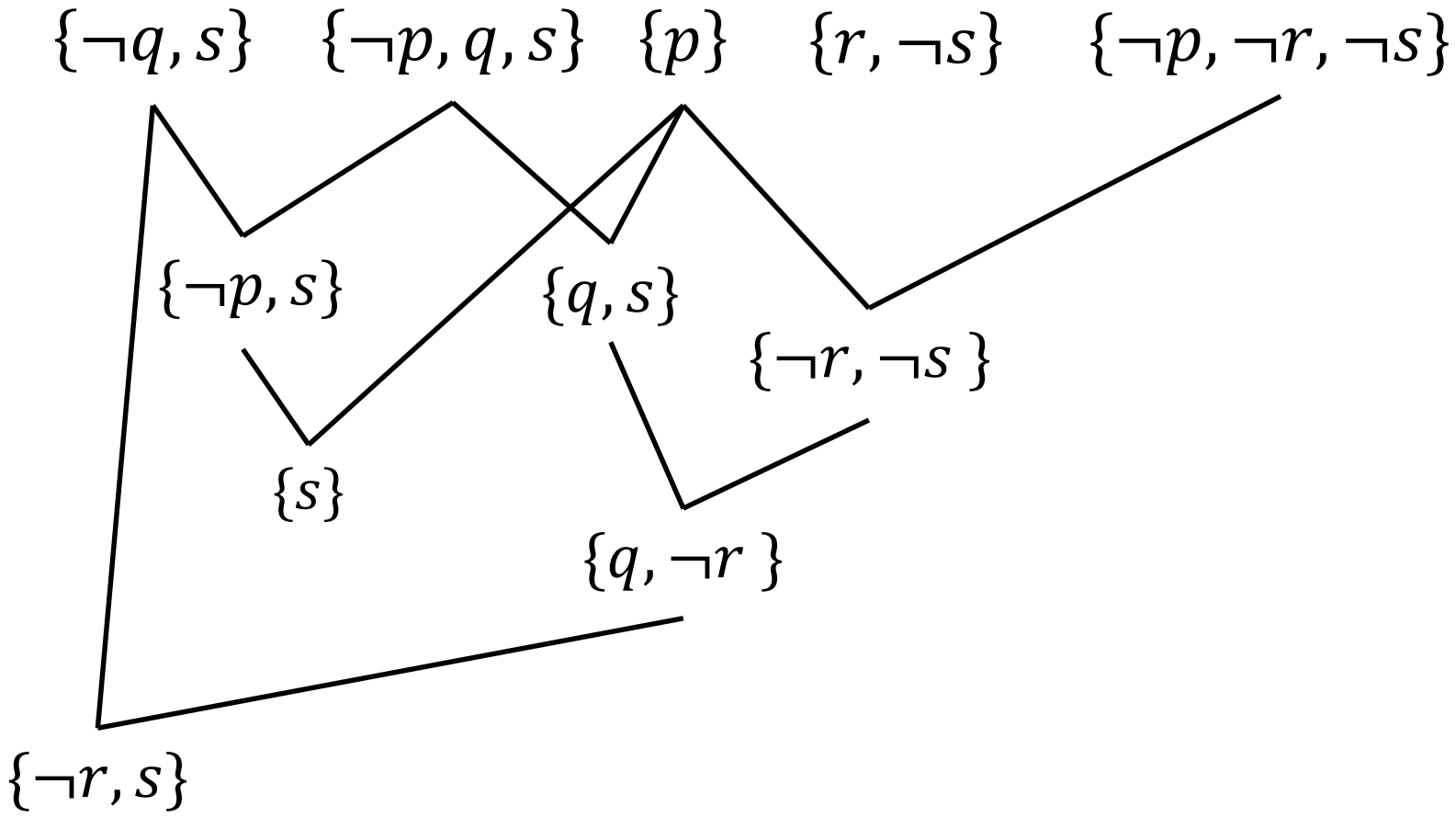
$\{q, s\}$

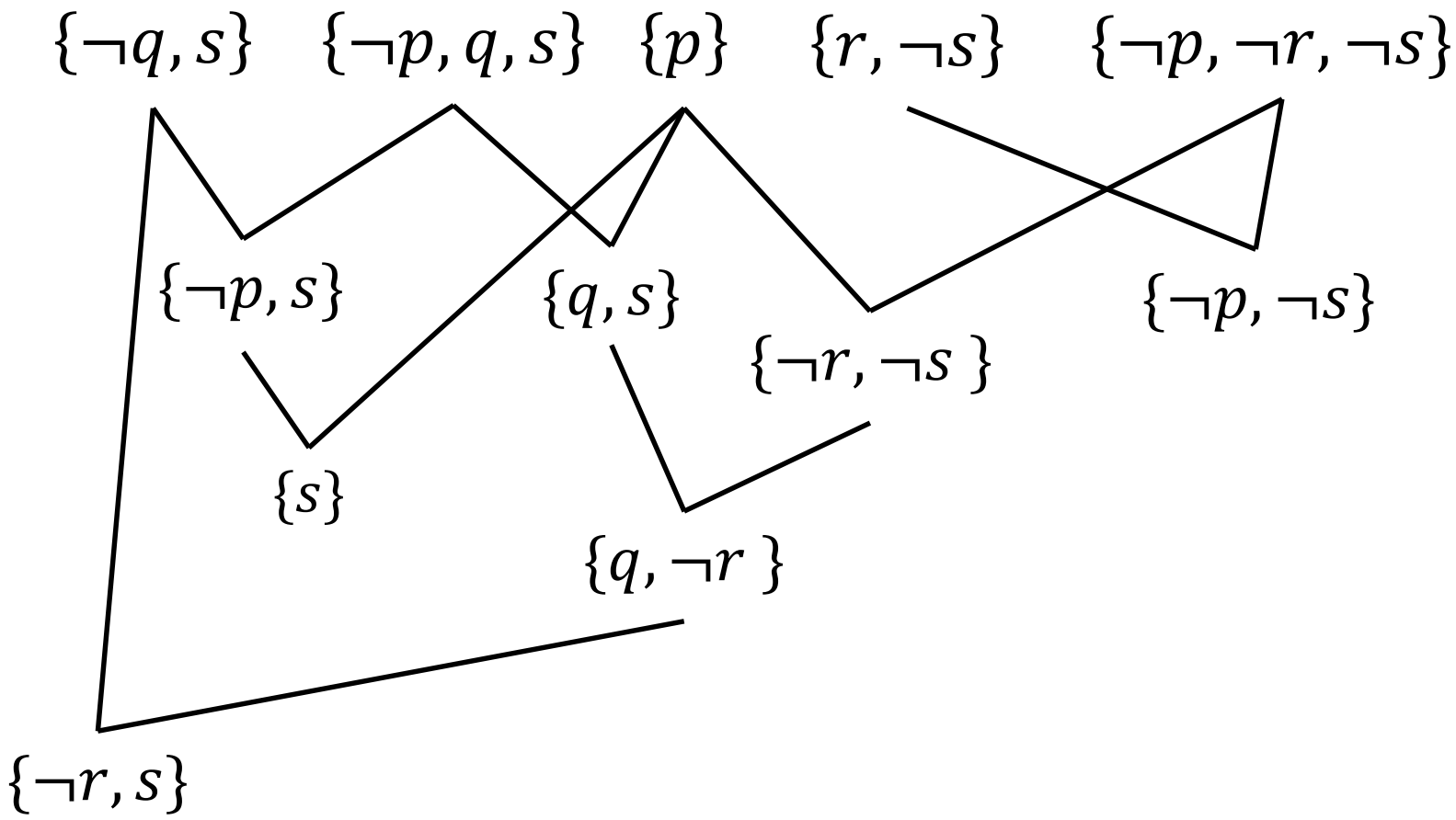

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$

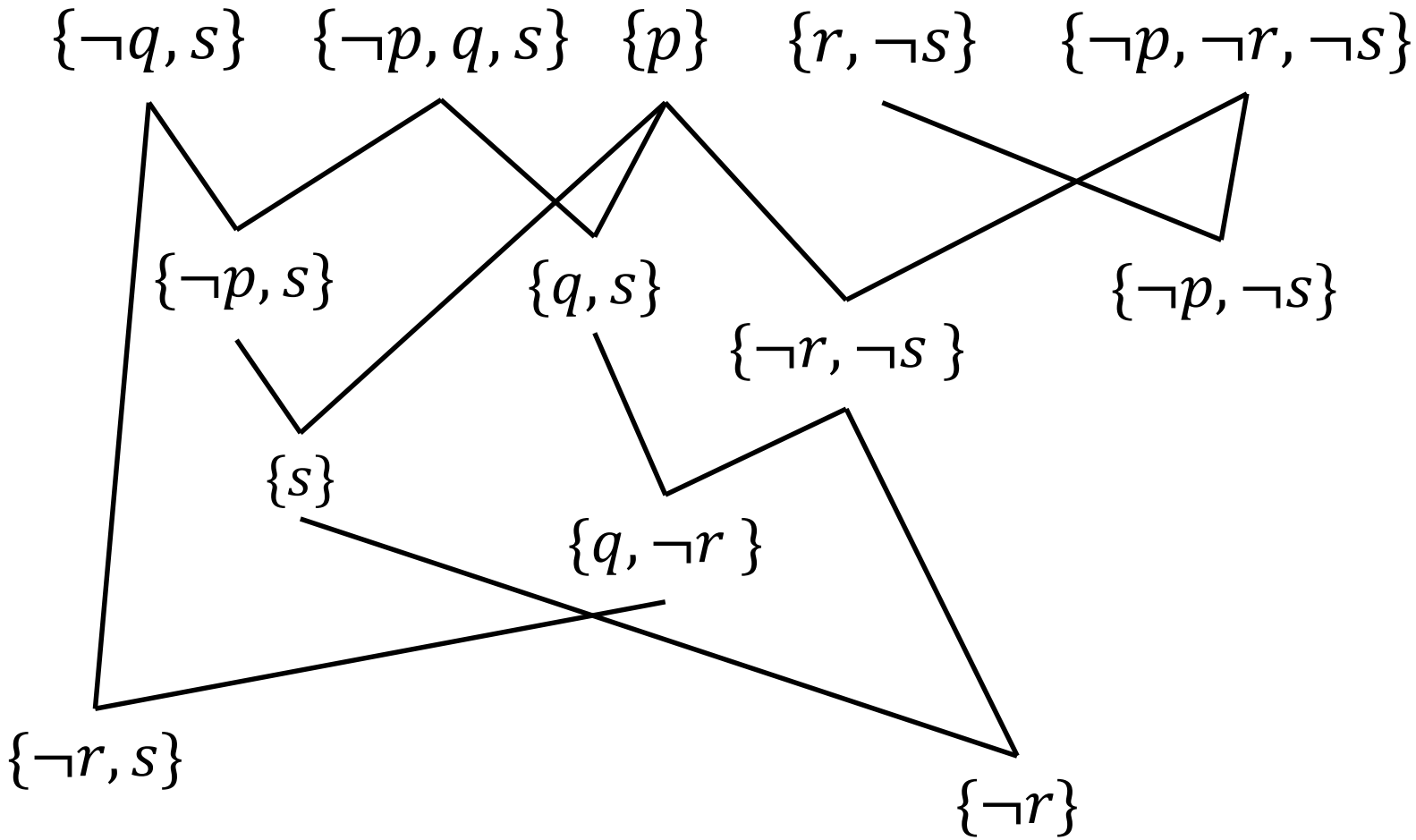

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{q, \neg r\}$

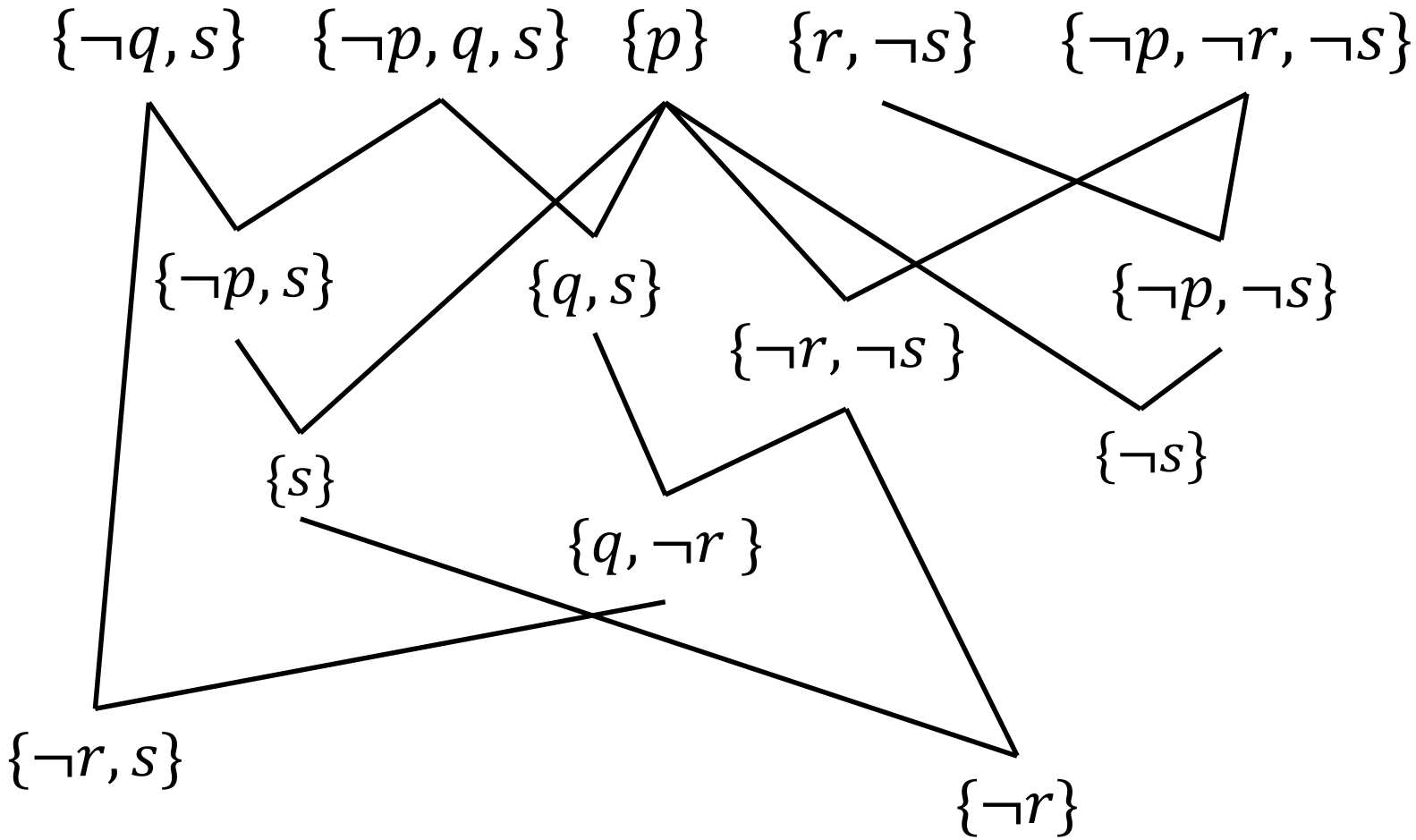

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{q, \neg r\}$

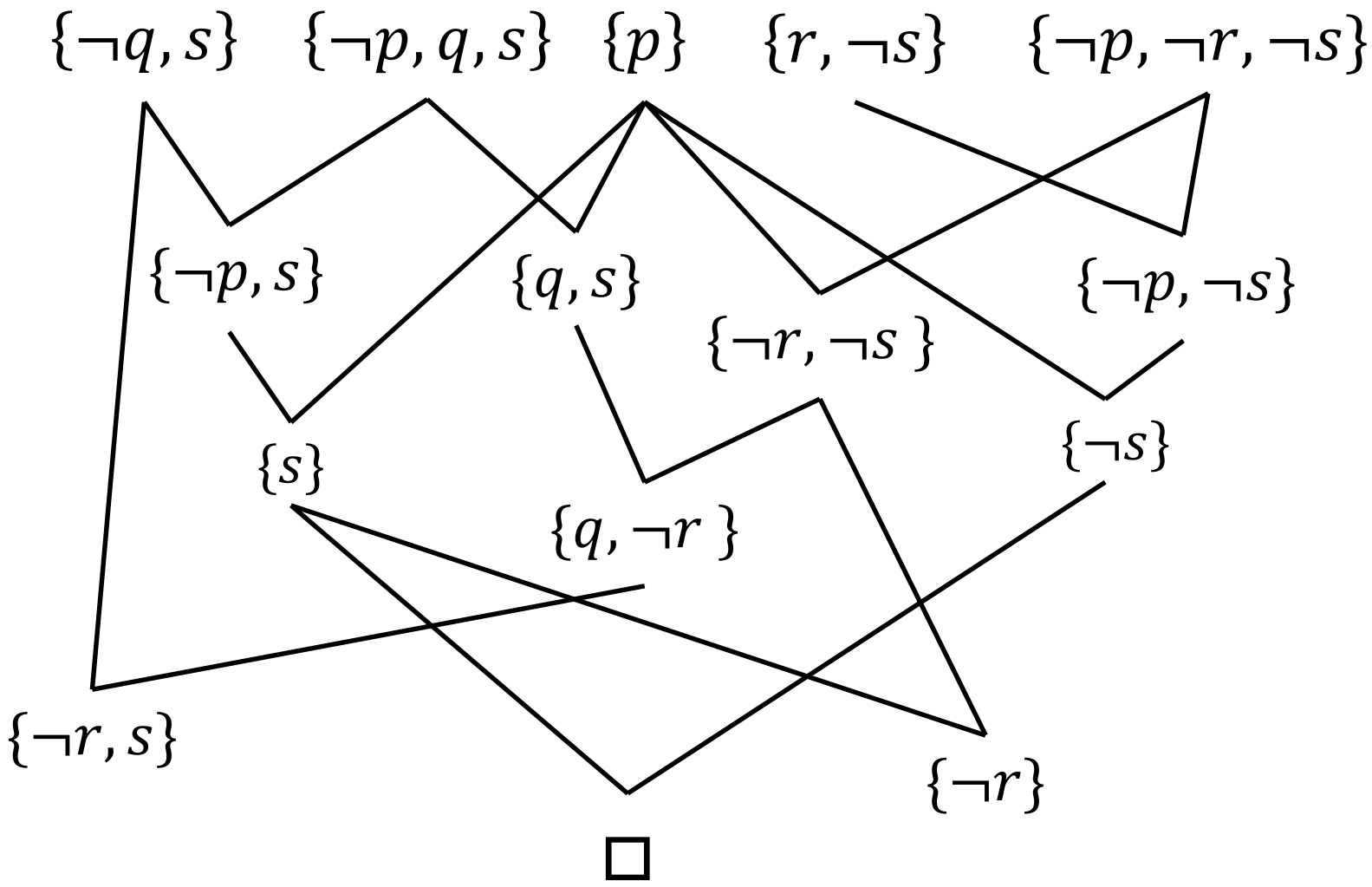

$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{s\}$
$\{q, \neg r\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{s\}$
$\{q, \neg r\}$
$\{\neg r, s\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{q, \neg r\}$
$\{\neg r, s\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{q, \neg r\}$
$\{\neg r, s\}$
$\{\neg r\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{\neg s\}$
$\{q, \neg r\}$
$\{\neg r, s\}$
$\{\neg r\}$


$\{\neg q, s\}$
$\{\neg p, q, s\}$
$\{p\}$
$\{r, \neg s\}$
$\{\neg p, \neg r, \neg s\}$
$\{\neg p, s\}$
$\{q, s\}$
$\{\neg r, \neg s\}$
$\{\neg p, \neg s\}$
$\{s\}$
$\{\neg s\}$
$\{q, \neg r\}$
$\{\neg r, s\}$
$\{\neg r\}$
$\square$

- Resolution:

Fakt. Sei F eine Formel in KNF-Form. Wenn R Resolvent zweier Klauseln von F ist, dann gilt

$$F \equiv F \cup \{R\}.$$

Begründung:

Seien K_1, K_2 die durch R resolvierten Klauseln mit $K_1 = K_1' \vee A$ und $K_2 = K_2' \vee \neg A$. Es gibt F' mit

$$\begin{aligned} F &= F' \wedge (K_1' \vee A) \wedge (K_2' \vee \neg A) \\ &\equiv F' \wedge (K_1' \vee A) \wedge (K_2' \vee \neg A) \wedge (K_1' \vee K_2') \\ &\equiv F' \wedge (K_1' \vee A) \wedge (K_2' \vee \neg A) \wedge R \\ &\equiv F \wedge R. \end{aligned}$$

- Resolution:

  Aus dem Lemma folgt: Wenn irgendwann die leere Klausel hinzugefügt wird (d.h. wenn das Verfahren „unerfüllbar“ antwortet), dann gilt

  $$F \equiv F \wedge \mathbf{false} \equiv \mathbf{false}$$

  und somit ist die Formel F unerfüllbar.

  Man muss noch zeigen:

  - Wenn das Verfahren „erfüllbar“ antwortet, dann ist die Formel erfüllbar. (Kommt später in der Vorlesung)
  - Das Verfahren terminiert. (Folgt aus der Tatsache, dass es endlich viele Klauseln mit n Variablen gibt.)

- Resolution:
  - Korrektheit der Resolution:
    1. Das Verfahren terminiert für jede Eingabeformel.

       Folgt daraus, dass nur endlich viele Klauseln aus einer endlichen Menge von Variablen konstruiert werden können.
    2. Wenn die Formel erfüllbar ist, dann antwortet das Verfahren „erfüllbar“.

       Folgt aus dem Fakt.
    3. Wenn die Formel unerfüllbar ist, dann antwortet das Verfahren „unerfüllbar“.

       Schwieriger. Wird später in der Vorlesung gezeigt.

- Ein Kalkül für logische Inferenzen:
  - Wir beschreiben eine Menge von Inferenzregeln (ein Kalkül).

    Mit den Inferenzregeln können Ausdrücke der Gestalt

$$A \vdash F$$

    mechanisch erzeugt werden, wobei A eine Konjunktion von Annahmen und F eine Formel ist.

- Ein Kalkül für logische Inferenzen.
  - Es wird gelten:

$$A \vdash F \text{ gdw. } A \models F.$$

  - Das Kalkül bietet damit eine Alternative zum Folgerungstest.
  - Die Regeln sind ähnlich zu den Formationsregeln:
    - Basisregeln: „$A \vdash F$ für die Konjunktion A und die Formel F“.
    - Induktive Regeln: „Wenn $A \vdash F_1, \ldots, A \vdash F_n$, dann $B \vdash F$“.

- Ein Kalkül für logische Inferenzen:
  - Grafische Darstellung der Basisregeln:

$$\frac{}{A \vdash F}$$

  - Grafische Darstellung der induktiven Regeln:

$$\frac{A_1 \vdash F_1, \ldots, A_n \vdash F_n}{B \vdash F}$$

Intuition: Um $B \vdash F$ zu beweisen, reicht es zu zeigen, dass $A \vdash F_1$ und $A_2 \vdash F_2$ und … und $A_n \vdash F_n$ gelten.

- Ein Kalkül für logische Inferenzen:
  - Annahmeregel: Für jedes A, für jede Annahme F in A:

$$\frac{}{A \vdash F}$$

  - Ausgeschlossener Dritte: Für jedes A, für jedes F:

$$\frac{}{A \vdash F \vee \neg F}$$

- Ein Kalkül für logische Inferenzen:
  - Regel für **true**: Für jedes A:

$$\frac{}{A \vdash \mathbf{true}}$$

  - Regel für **false**: Für jedes A, für jede Formel F:

$$\frac{A \vdash F \qquad A \vdash \neg F}{A \vdash \mathbf{false}}$$

- Ein Kalkül für logische Inferenzen:
  - Konjunktionseinführung: Für alle A, für alle F, G:

$$\frac{A \vdash F \qquad A \vdash G}{A \vdash F \wedge G}$$

  - Konjunktionsbeseitigung: Für alle A, für alle F, G:

$$\frac{A \vdash F \wedge G}{A \vdash F} \qquad \frac{A \vdash F \wedge G}{A \vdash G}$$

- Ein Kalkül für logische Inferenzen:
  - Disjunktionseinführung: Für alle A, für alle F, G :

$$\frac{A \vdash F}{A \vdash F \vee G} \qquad \frac{A \vdash F}{A \vdash G \vee F}$$

  - Disjunktionsbeseitigung: Für alle A, für alle F, G, H :

$$\frac{A \vdash F \vee G \quad A, F \vdash H \quad A, G \vdash H}{A \vdash H}$$

- Ein Kalkül für logische Inferenzen:
  - Negationseinführung: Für alle A, für alle F:

$$\frac{A, F \vdash \mathbf{false}}{A \vdash \neg F}$$

  - Negationsbeseitigung: Für alle A, für alle F:

$$\frac{A, \neg F \vdash \mathbf{false}}{A \vdash F}$$

- Ein Kalkül für logische Inferenzen:
  - Implikationseinführung: Für alle A, für alle F, G:

$$\frac{A, F \vdash G}{A \vdash F \to G}$$

  - Implikationsbeseitigung: Für alle A, für alle F, G:

$$\frac{A \vdash F \to G \qquad A \vdash F}{A \vdash G}$$

- Ein Kalkül für logische Inferenzen:
  - Theorem: Die Regelmenge hat die folgenden zwei Eigenschaften:
    - Wenn $F \vdash G$, dann $F \models G$. (Korrektheit, soundness)

      „Aus der Regelmenge werden nur gültige Inferenzen hergeleitet.“
    - Wenn $F \models G$, dann $F \vdash G$. (Vollständigkeit, completeness)

      „Jede gültige Inferenz kann mit der Regelmenge hergeleitet werden.“

- Beispiel eines formalen Beweises:
  - Wir betrachten nochmal die Inferenz

    Wenn
    es heute bewölkt ist, und
    wir nur dann schwimmen gehen, wenn es sonnig ist, und
    wir immer dann rudern, wenn wir nicht schwimmen, und
    wir immer früh nach Hause kommen wenn wir rudern, dann
    werden wir heute früh nach Hause kommen.
  - Wir formalisieren sie in der Aussagenlogik und zeigen ihre Korrektheit mit Hilfe der Inferenzregeln.

- Beispiel eines formalen Beweises
  - Wir führen Variablen für die Aussagen ein

    sonnig = ***„es ist sonnig“***

    schwim = ***„wir werden schwimmen“***

    rudern = ***„wir werden rudern“***

    früh = ***„wir werden früh nach Hause kommen“***.
  - Annahmemenge A:

    (a) ¬sonnig

    (b) schwim → sonnig

    (c) ¬schwim → rudern

    (d) rudern → früh

- Beispiel eines formalen Beweises:
  - Eine Herleitung von $A \vdash F$ ist eine endliche Sequenz $A_1 \vdash F_1,\ \ A_2 \vdash\ F_2, \ldots, A_n \vdash F_n$ mit:
    - $A_n = A$ und $F_n = F$
    - Für alle i, $1 \leq i \leq n$, kann der Ausdruck $A_i \vdash F_i$ aus einer Teilmenge der Ausdrücke $A_1 \vdash F_1, \ldots, A_{i-1} \vdash\ F_{i-1}$ durch Anwendung einer Regel gewonnen werden.
  - Ein formaler Beweis der Aussage „F folgt aus den Annahmen $A_1, \ldots, A_n$“ ist eine Herleitung von $A_1, A_2, \ldots, A_n \vdash F$.

Schritt				Bewiesen durch
1.	A, schwim	$\vdash$	schwim	An.
2.	A, schwim	$\vdash$	schwim $\rightarrow$ sonnig	An. (b)
3.	A, schwim	$\vdash$	$\neg$ sonnig	An. (a)
4.	A, schwim	$\vdash$	sonnig	Imp.Bes. 1,2
5.	A, schwim	$\vdash$	**false**	**false**.Ein. 3,4
6.	A	$\vdash$	schwim $\rightarrow$ **false**	Imp.Ein. 5
7.	A	$\vdash$	$\neg$ schwim	Neg.Ein. 6
8.	A	$\vdash$	$\neg$ schwim $\rightarrow$ rudern	An. (c)
9.	A	$\vdash$	rudern	Imp.Bes. 7,8
10.	A	$\vdash$	rudern $\rightarrow$ früh	An. (d)
11.	A	$\vdash$	früh	Imp.Bes. 9,10

Kapitel 2: Grundlagen (Aussagenlogik 2)

- Zusammenfassung Aussagenlogik:
  - Syntax und Semantik:
    - Operatoren und Wahrheitstabellen.
  - Logische Begriffe:
    - Tautologie, Widerspruch, Erfüllbarkeit, Äquivalenz, Folgerung.
  - Äquivalenzregeln.
  - Vollständige Operatorenmengen.
  - Algorithmen für (Nicht-)Erfüllbarkeit:
    - DPLL, Resolution
  - Folgerungsregeln:
    - Formale Beweise

Praktische Anwendungen in der Informatik:

- Schaltungen (NAND-Gatter als Grundbaustein)
- Programmierung (z.B. if-then-else)
- SAT-Problem als Repräsentant eines „schwierigen“ Problems (NP-Vollständigkeit)