INF4170 – Logikk

Forelesning 12: Automatisk bevissøk IV – matriser og koblingskalkyle

Bjarne Holen

Institutt for informatikk, Universitetet i Oslo

11. mai 2010

Dagens plan

1. Automatisk bevissøk IV

Bevissøk med koblinger

- Vi har til nå sett på forskjellige varianter av sekventkalkyle:
  - LK for *klassisk utsagnslogikk*,
  - LJ for *intuisjonistisk utsagnslogikk*,
  - ensidig sekventkalkyle,
  - grunn LK og
  - fri-variabel LK for *klassisk førsteordens logikk*.
- Alle disse kalkylene har to svakheter når de skal implementeres med en automatisk søkealgoritme:
  - Redundans – gjennom formelkopiering kan vi få mange forekomster av én og samme formel.
  - Ingen relevanssjekk – siden det kun tas hensyn til toppkonnektiv ved formelanalyse kan vi risikere å utvide formler som ikke er relevante for å lukke utledningen.
- Vi skal i denne forelesningen se på koblingskalkylen, som ikke har disse problemene.

Redundans i LK-utledninger

$$\dfrac{\dfrac{\times}{Q_2 \to R_1, P_2 \vdash R_2, P_1}}{Q_2 \to R_1 \vdash P_1, P_2 \to R_2} \qquad \dfrac{\times \qquad \dfrac{\times}{Q_1, R_1, P_2 \vdash R_2}}{Q_1 \vdash Q_2, P_2 \to R_2 \qquad Q_1, R_1 \vdash P_2 \to R_2}}{Q_2 \to R_1, Q_1 \vdash P_2 \to R_2}$$

$$P_1 \to Q_1, Q_2 \to R_1 \vdash P_2 \to R_2$$

- I rotsekventen forekommer både P, Q og R to ganger. Vi bruker $_1$ og $_2$ til å skille forekomstene fra hverandre.
- Siden P_2 og P_1 i venstre løvsekvent er forekomster av P, kan vi lukke med disse. Tilsvarende for de andre løvsekventene.
- En LK-utledning kan inneholde mange kopier av en (del)formel i rotsekventen. I utledningen over: 3 kopier av $Q_2 \to R_1$, 4 kopier av $P_2 \to R_2$, 2 kopier av P_1, 6 kopier av P_2, osv.
- Vi har derfor redundans i LK-utledninger.

Ingen relevanssjekk

$$\dfrac{(Q \to R) \vee (R \to Q), P \vdash P \times \qquad (Q \to R) \vee (R \to Q), P \vdash P \times}{(Q \to R) \vee (R \to Q), P \vee P \vdash P}$$

- I rotsekventen over er det to muligheter for utvidelse: $(Q \to R) \vee (R \to Q)$ eller $P \vee P$
  - Hvis vi velger $(Q \to R) \vee (R \to Q)$, vil vi gjøre (en eller flere) utvidelser som ikke bidrar til å lukke løvsekventene.
  - Velger vi derimot $P \vee P$, vil vi kunne lukke direkte.
- Det er de *atomære delformlene* til en formel som er med på å lukke løvsekventene.
- I sekventkalkyle ser vi imidlertid kun på *toppkonnektivene* for å velge hvilken formel vi skal utvide.
- Vi kan derfor risikere å utvide formler som er irrelevante m.h.p. å lukke utledningen.

Matriser og koblingskalkyle

- Bevisbarhet av en formel kan defineres som en egenskap ved formelen direkte, istedenfor via utledninger som i sekventkalkyle.
  - Vi kan representere en formel todimensjonalt ved en matrise bestående av de atomære delformlene.
  - Gyldighet defineres så som en egenskap ved stiene gjennom matrisen.
  - Hver sti må inneholde et komplementært par av atomer – kalt en kobling.
- Koblingskalkyle er basert på matrisekarakteristikken av gyldighet og utnytter at samme kobling kan forekomme på flere stier gjennom en matrise.
- Vi skal begrense oss til å se på koblingskalkyle for utsagnslogikk i disjunktiv normalform.
- Koblingskalkyle er imidlertid ikke begrenset til normalform, og finnes for mange forskjellige logikker – inkludert de vi har sett på i kurset.

Disjunktiv normalform

- I ukeoppgavene har vi sett på normalformer.
- En literal er en atomær formel eller negasjonen av en atomær formel:
  - $P, Q, R, \ldots$ er positive literaler og $\neg P, \neg Q, \neg R, \ldots$ er negative literaler.
- En generalisert konjunksjon er en formel på formen $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ der hver φ_i er en formel.
- En generalisert disjunksjon er en formel på formen $(\varphi_1 \vee \ldots \vee \varphi_n)$ der hver φ_i er en formel.

Definisjon (Disjunktiv normalform)

En formel er på disjunktiv normalform (DNF) hvis den er en (generalisert) disjunksjon av en eller flere (generaliserte) konjunksjoner av en eller flere literaler.

Disjunktiv normalform

Hvilke formler er på DNF?

- P ✓
- $P \vee Q$ ✓
- $(P \vee Q) \wedge R$ Nei, venstre konjunkt er en disjunksjon.
- $(\neg P \wedge Q) \vee (\neg Q \wedge P)$ ✓
- $(P \rightarrow Q) \vee Q \vee \neg P$ Nei, venstre konjunkt er en implikasjon.
- $(\neg P \wedge Q) \vee R \vee (\neg R \wedge P) \vee (\neg P \wedge Q \wedge \neg R)$ ✓
- $\neg P \wedge Q \wedge \neg R$ ✓

- Enhver literal er på DNF.
- Enhver disjunksjon av literaler er på DNF.
- Enhver konjunksjon av literaler er på DNF.

Transformasjon til DNF

- Vi har i ukeoppgavene sett at enhver utsagnslogisk formel kan transformeres til en *ekvivalent* formel på DNF.
- Husk: to formler er ekvivalente hvis de oppfylles av nøyaktig de samme valuasjonene/modellene.
- Eksempel:

$$
\begin{aligned}
(P \wedge (P \rightarrow Q)) \rightarrow Q &\Leftrightarrow \neg(P \wedge (P \rightarrow Q)) \vee Q \\
&\Leftrightarrow \neg P \vee \neg(P \rightarrow Q) \vee Q \\
&\Leftrightarrow \neg P \vee \neg(\neg P \vee Q) \vee Q \\
&\Leftrightarrow \neg P \vee (\neg\neg P \wedge \neg Q) \vee Q \\
&\Leftrightarrow \neg P \vee (P \wedge \neg Q) \vee Q
\end{aligned}
$$

Transformasjon fra DNF til KNF

- Vi har sett at alle DNF formler kan transformeres til *ekvivalente* KNF formler ved gjentagende bruk av

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$$

- Hva er problemet med en slik transformasjon?
- Hvorfor vil vi ha formler på KNF?

$$\begin{aligned}(P \wedge (P \rightarrow Q)) \rightarrow Q &\Leftrightarrow \neg P \vee (P \wedge \neg Q) \vee Q \\ &\Leftrightarrow (\neg P \vee (P \wedge \neg Q)) \vee Q \\ &\Leftrightarrow ((\neg P \vee P) \wedge (\neg P \vee \neg Q)) \vee Q \\ &\Leftrightarrow ((\neg P \vee P) \vee Q) \wedge ((\neg P \vee \neg Q) \vee Q) \\ &\Leftrightarrow (\neg P \vee P \vee Q) \wedge (\neg P \vee \neg Q \vee Q)\end{aligned}$$

- Falsifikasjon av 1 klausul falsifiserer formelen

Sammenheng mellom DNF og KNF

- Enkelt å konvertere formel til DNF
- Vi er mest interessert i KNF
- Finnes det en enkel måte å få KNF fra DNF representasjon?
- Anta at vi har en formel på DNF

$$(A_1 \wedge A_2 \ldots \wedge A_n) \vee (B_1 \wedge B_2 \ldots \wedge B_m) \cdots \vee (P_1 \wedge P_2 \ldots \wedge P_s)$$

- Ved gjentatte anvendelser av regelen

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$$

- ser vi at klausulene i KNF representasjonen blir slik

$$(A_i \vee B_j \ldots \vee P_k)$$

$$1 < i < n \quad 1 < j < m \quad \ldots \quad 1 < k < s$$

- Merk at KNF klausulene får 1 element fra hver DNF klausul!

Overblikk over Matrise-søk

- For å vise at en sekvent er gyldig

$$P, (P \rightarrow Q) \vdash Q$$

- Bytt ut meta-symbol med objekt-symbol

$$(P \wedge (P \rightarrow Q)) \rightarrow Q$$

- Konverter til DNF

$$\neg P \vee (P \wedge \neg Q) \vee Q$$

- Elementene i DNF klausulene utgjør søylene i matrisen

$$\left| \begin{array}{cc} & \\ & P \quad Q \\ \neg P & \neg Q \\ & \end{array} \right|$$

Overblikk over Matrise-søk

- Vi har sett at ved å plukke 1 element fra hver DNF klausul
- (søylene i matrisen) får vi KNF klausulene.
- Dersom enhver KNF klausul har en kobling ($\neg$ A, A)
- vil matrisen representere enn IKKE-falsifiserbar formel (gyldig)
- En matrise er en kompakt representasjon av formelen
- Vi bruker denne til å søke igjennom alle KNF klausuler
- Vi søker målrettet og leter etter koblinger

Matriser

Definisjon (Matrise)

- *En klausul er en endelig mengde literaler. (metasymbol , $:= \wedge$)*
- *En matrise er en endelig mengde klausuler. (metasymbol , $:= \vee$)*

Eksempel (klausuler):

- $\{P\}$
- $\{P, \neg P, Q\}$
- $\{\neg Q, R, P\}$
- $\{\}$

Eksempel (matriser):

- $\{\{P\}, \{\neg P\}\}$
- $\{\{Q\}, \{\neg P, R\}, \{\neg R, P, \neg Q\}\}$
- $\{\}$
- $\{\{\}\}$

- En klausul er
  - positiv hvis den bare inneholder positive literaler, og
  - negativ hvis den bare inneholder negative literaler.

Matriser

Definisjon (Semantikk for matriser)

La v være en boolsk valuasjon.

- *For klausuler: $v \models \{L_1, \ldots, L_n\}$ hvis og bare hvis $v \models L_i$ for alle L_i.*
- *For matriser: $v \models \{K_1, \ldots, K_n\}$ hvis og bare hvis $v \models K_i$ for en K_i.*

Eksempel

La v være slik at $v \models P$, $v \not\models Q$ og $v \models R$.

- $v \models \{\{P\}, \{\neg P\}\}$? $\surd$
- $v \models \{\{Q\}, \{\neg P, R\}, \{\neg R, P, \neg Q\}\}$? *Nei, v oppfyller ingen klausuler.*
- $v \models \{\}$ *der* $\{\}$ *er en tom matrise?* *Nei, v oppfyller ingen klausuler i $\{\}$.*
- $v \models \{\{\}\}$ *(matrisen som kun inneholder en tom klausul)?* $\surd$
  ($v \models \{\} \in \{\{\}\}$ siden alle valuasjon oppfyller en tom klausul.)

Falsifiserbarhet av matriser

v	$M = \{K_1, \ldots, K_n\}$	$K = \{L_1, \ldots, L_m\}$
oppfyller	$v \models K_i$ for en K_i	$v \models L_i$ for alle L_i
falsifiserer	$v \not\models K_i$ for alle K_i	$v \not\models L_i$ for en L_i

- En boolsk valuasjon v falsifiserer
  - en klausul i M hvis v falsifiserer en av literalene i klausulen
  - alle klausulene i M hvis v falsifiserer en literal i hver klausul
- For hver klausul K_i har vi $|K_i|$ valg av literaler å falsifisere.
- Vi får maksimalt $|K_1| \times \ldots \times |K_n|$ måter å falsifisere M på.

DNF-formler som matriser

- En formel på DNF kan sees på som en matrise der klausulene tilsvarer disjunktene i formelen.
- Eksempel: formelen

$$\neg P \vee (P \wedge \neg Q) \vee Q$$

tilsvarer matrisen

$$\{\{\neg P\}, \{P, \neg Q\}, \{Q\}\}$$

tilsvarer KNF representasjonen

$$(\neg P \vee P \vee Q) \wedge (\neg P \vee \neg Q \vee Q)$$

Stier

Definisjon (Sti)

La M være en matrise.

- *En sti gjennom M er en mengde som inneholder nøyaktig én literal fra hver klausul i M.*
- *En sti gjennom M er partiell hvis den mangler literaler fra én eller flere klausuler i M.*

Eksempel

	P	Q
$\neg P$	$\neg Q$	

- *Stier:* $\{\neg P, P, Q\}$ *og* $\{\neg P, \neg Q, Q\}$.
- *Partieller stier:* $\{\neg P\}$, $\{\neg P, P\}$, $\{\neg P, \neg Q\}$, $\{P\}$, $\{P, Q\}$, $\{Q\}$, $\{Q, \neg Q\}$ *og* $\{Q, \neg P\}$.

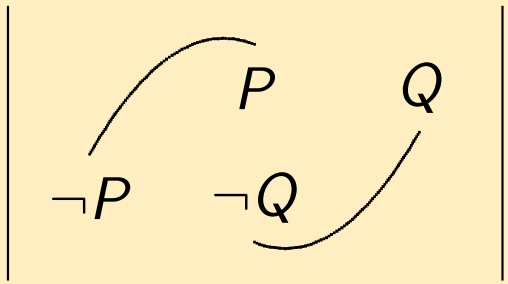
Hver sti gjennom en matrise representerer en mulig falsifikasjon!

Koblinger

Definisjon (Koblinger)

La M være en matrise. En kobling i M er en partiell sti gjennom M på formen $\{A, \neg A\}$ der A er en atomær formel.

Eksempel



- *Koblinger: $\{\neg P, P\}$ og $\{\neg Q, Q\}$.*
- *Vi markerer sammenkoblede literaler med en bue i den grafiske matrisenotasjonen.*

Åpne og lukkede stier

- En kobling $\{P, \neg P\}$ er ikke falsifiserbar:
  - en boolsk valuasjon kan ikke falsifisere både P og $\neg P$ samtidig!
- Derfor vil en sti som inneholder en kobling, ikke være falsifiserbar.
- Dersom alle stiene gjennom en matrise inneholder en kobling, vil matrisen ikke være falsifiserbar – og dermed må formelen den representerer være gyldig.
- Vi sier at en (partiell) sti i en matrise er
  - åpen hvis den *ikke* inneholder noen kobling, og
  - lukket hvis den inneholder en kobling.

Matriser vs. LK-utledninger

- Stiene gjennom matrisen til en formel F tilsvarer løvsekventene vi får hvis vi gjør en *maksimal* LK-utledning for sekventen $\vdash F$:
  - Negative literaler er antecedentformler, og
  - positive literaler er succedentformler.

$$\left|\begin{array}{cc} & P \quad Q \\ \neg P & \neg Q \end{array}\right|$$

$$\dfrac{\dfrac{P \vdash P, Q \qquad \dfrac{P, Q \vdash Q}{P \vdash \neg Q, Q}}{P \vdash P \wedge \neg Q, Q}}{\vdash \neg P, P \wedge \neg Q, Q}$$

- Lukkede stier gjennom matriser tilsvarer aksiomer i LK-utledninger.

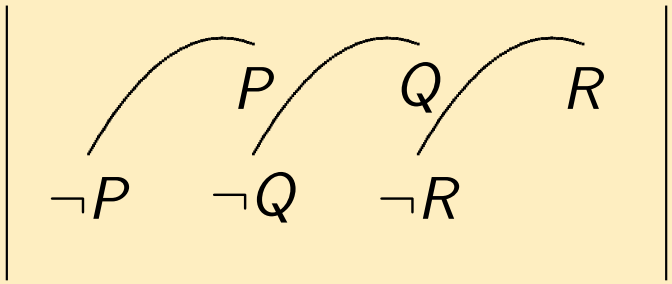
Matrisekarakterisering av gyldighet

Teorem

En formel F på DNF er gyldig hvis og bare hvis enhver sti gjennom matrisen til F inneholder en kobling.

Eksempel

- *Formelen* $(P \wedge (P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow R$ *er gyldig.*
- *På DNF får vi formelen* $\neg P \vee (P \wedge \neg Q) \vee (Q \wedge \neg R) \vee R$.



Koblinger: $\{\neg P, P\}$, $\{\neg Q, Q\}$ *og* $\{\neg R, R\}$.

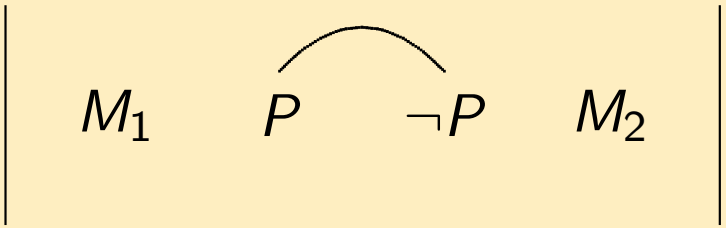
Stier: $\{\neg P, P, Q, R\}$, $\{\neg P, P, \neg R, R\}$, $\{\neg P, \neg Q, Q, R\}$ *og* $\{\neg P, \neg Q, \neg R, R\}$.

- *Alle stiene inneholder en kobling.*

Koblingskalkyle

- Matrisekarakteriseringen av gyldighet gir oss muligheten til å avgjøre bevisbarhet ved å sjekke at alle stier inneholder en kobling.
- En første tilnærming vil være å liste opp alle stiene gjennom en matrise og sjekke hver av dem for koblinger.
- Det er imidlertid slik at én kobling kan forekomme på flere stier gjennom en matrise.

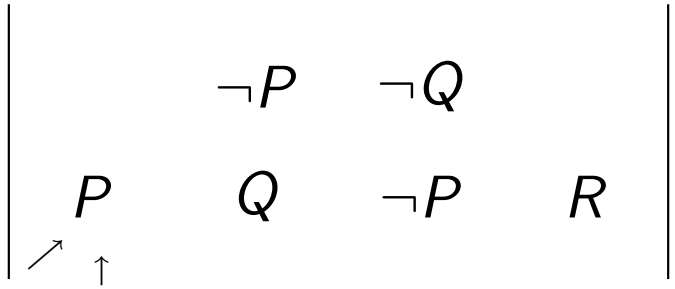
Eksempel



Uansett hvordan M_1 og M_2 ser ut vil enhver sti gå gjennom koblingen $\{P, \neg P\}$.

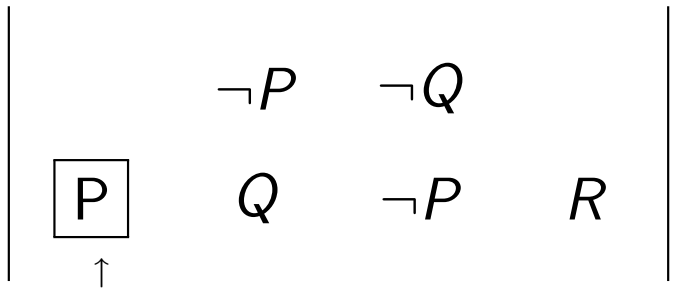
- Det er derfor en god idé å fokusere på *koblinger* istedenfor *stier*.
- Vi skal vise grunnidéene i koblingskalkylen ved et eksempel.

Startsteget



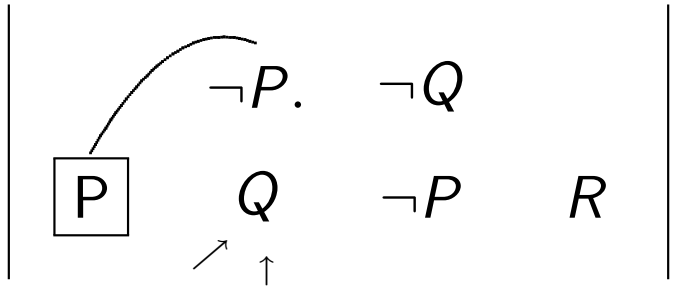
- Vi starter med å velge en startklausul.
- Vi velger $\{P\}$ og markerer denne med $\uparrow$ under klausulen, og markerer alle literalene i startklausulen med $\nearrow$.

Utvidelsessteget I



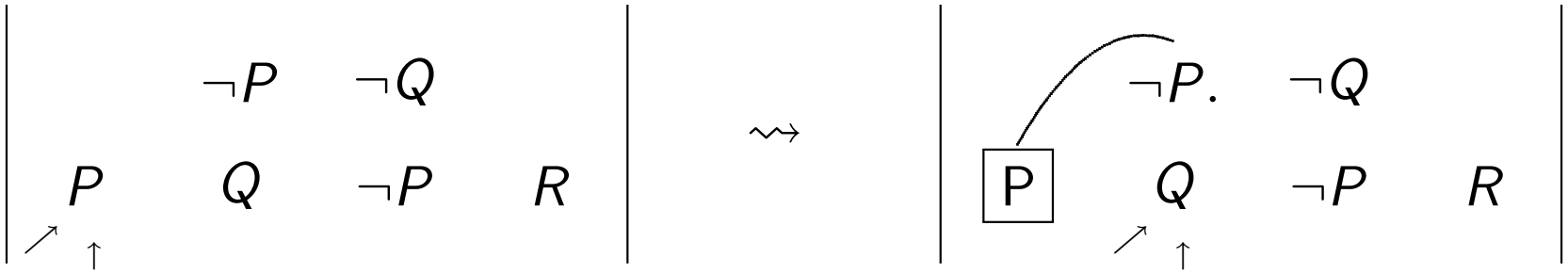
- Vi kaller klausulen som er markert med $\uparrow$, for aktiv klausul.
- Hvis en literal i aktiv klausul er markert med $\nearrow$ må vi sjekke alle stier som inneholder literalen.
- I dette tilfellet har vi bare ett valg: P.
- Vi markerer P med en ramme for å indikere at literalen er en del av den stien vi for øyeblikket undersøker – den aktive stien.
- Samtidig fjerner vi $\nearrow$-symbolet fra P.

Utvidelsessteget II



- Vi utvider den aktive stien ved å koble P med en komplementær literal i en av de andre klausulene.
- Vi har tre valg: $\{\neg P, Q\}$, $\{\neg Q, \neg P\}$ og $\{R\}$
- Vi velger den første klausulen og markerer de sammenkoblede literalene med en bue.
- Alle stier som springer ut fra den sammenkoblede $\neg P$ vil være lukkede på grunn av koblingen $\{P, \neg P\}$. Dette markeres med '.' etter $\neg P$.
- Den sammenkoblede klausulen settes aktiv og de resterende literalene på den markeres med $\nearrow$.

Utvidelsessteget – oppsummering



↑ markerer aktiv klausul

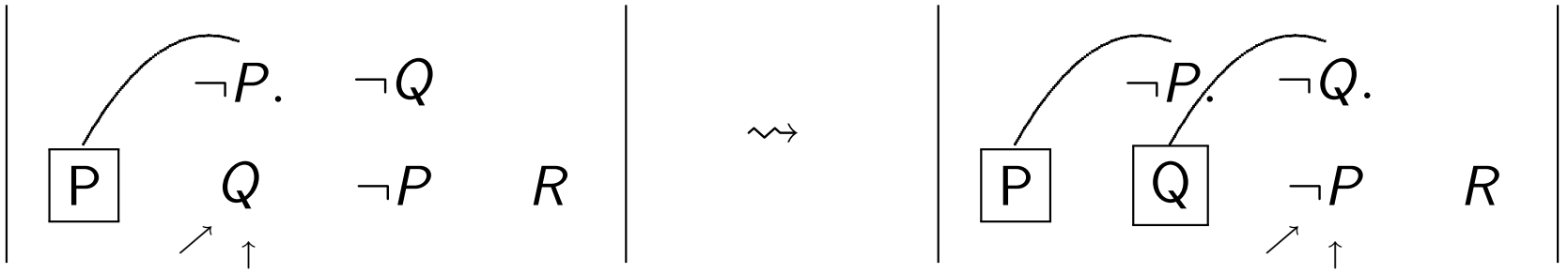
$\boxed{L}$ markerer literaler på den aktive stien

. markerer lukkede partielle stier

↗ markerer literaler i åpne partielle stier

1. Velg en literal L markert med ↗ i den aktive klausulen.
2. Bytt ut ↗ med en boks rundt L. Velg en L-kobling.
   - Hvis det er flere alternativer, ta vare på dem.
3. Marker den koblede literalen med '.'
4. Marker de resterende literalene i den koblede klausulen med ↗.
5. Flytt ↑ til den sammenkoblede klausulen.

Vi foretar nok et utvidelsessteg



↑ markerer aktiv klausul

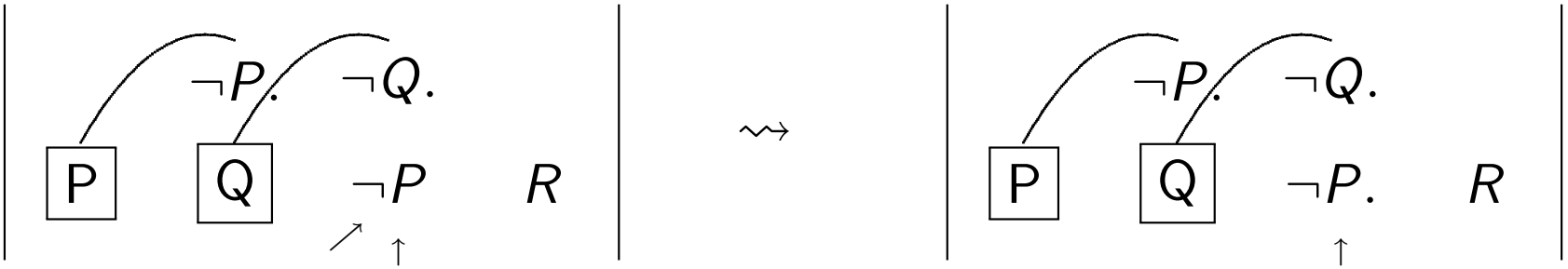
. markerer lukkede partielle stier

$\boxed{L}$ markerer literaler på den aktive stien

↗ markerer literaler i åpne partielle stier

1. Vi velger literalen Q i den aktive klausulen.
2. Vi bytter ut ↗ med en boks rundt Q. Vi har bare ett alternativ til kobling: $\neg Q$.
3. Markerer den koblede literalen med '.'
4. Markerer de resterende literalene i den koblede klausulen med ↗.
5. Flytt ↑ til den sammenkoblede klausulen.

Reduksjonssteget



↑ markerer aktiv klausul

. markerer lukkede partielle stier

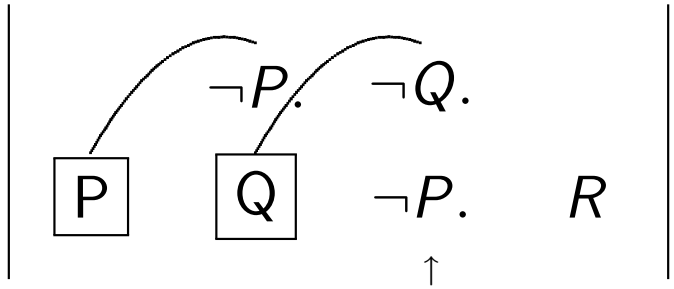
$\boxed{L}$ markerer literaler på den aktive stien

↗ markerer literaler i åpne partielle stier

- I situasjonen til venstre finnes det ingen literaler å koble $\neg P$ med.
- Det finnes imidlertid en komplementær literal i den aktive stien: P.
- Vi kan derfor foreta et reduksjonssteg:

1. Blant literalene som er merket med ↗ i den aktive klausulen, velg en L som er komplementær med en literal på den aktive stien.
2. Fjern ↗ fra L og marker den med '.'

Fullført søk – suksess

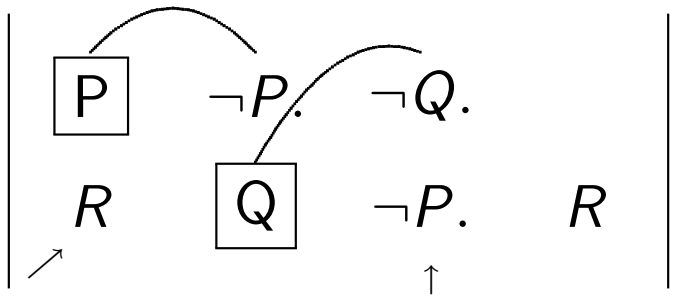


- I situasjonen over er alle literaler i aktiv klausul markert med '.', dvs. at alle stier som fortsetter ut fra denne klausulen inneholder koblinger.
- I tillegg er ingen literaler i klausuler på den aktive stien merket med $\nearrow$, dvs. at vi ikke har noen partielle åpne stier igjen å sjekke.
- Tilstanden er et vitne på at søket er fullført med suksess – alle stier gjennom matrisen inneholder koblinger.

Kalkyle vs. søkealgoritme

- Koblingskalkylen består av reglene start, utvidelse og reduksjon.
- Reglene definerer et sett stisjekkingstilstander.
- I tillegg har vi en beskrivelse av hvilke tilstander som representerer suksess i stisjekkingen.
- En søkealgoritme for koblingskalkylen må spesifisere en *rekkefølge* å gjøre reglene i.
- Vi skal nøye oss med å presentere noen viktige poenger m.h.p. implementasjon av en søkealgoritme.
- Til slutt skal vi ta en titt på en Prolog-implementasjon av koblingskalkylen.

Sjekke alle åpne partielle stier



- Vi ser på suksesstilstanden med matrisen fra eksempelet, men legger til R i første klausul. Den nye matrisen inneholder flere stier uten koblinger, f.eks. $\{R, Q, \neg P, R\}$.
- Tilstanden over er ikke en suksesstilstand, siden vi har en partiell åpen sti: den nye literalen R er merket med $\nearrow$.
- Vi må gå tilbake og se hva som skjer hvis vi velger R som del av den aktive stien istedenfor P i venstre klausul.
- Vi får en låst tilstand, siden R ikke kan kobles med noen literaler i matrisen.
- Vi må altså sjekke alle åpne partielle stier (literaler merket med $\nearrow$) før vi kan konludere med suksess.

Implementasjon i Prolog – leanCoP

```
prove(Mat) :-
  append(MatA,[Cla|MatB],Mat), append(MatA,MatB,Mat1),
  \+member(-_,Cla), prove(Cla,Mat1,[]).
prove([],_,_).
prove([Lit|Cla],Mat,Path) :-
  (-NegLit=Lit;-NegLit\=Lit,-Lit=NegLit),
   ( member(NegLit,Path); append(MatA,[Cla1|MatB],Mat),
     append(MatA,MatB,Mat1), append(ClaA,[NegLit|ClaB],Cla1),
     append(ClaA,ClaB,Cla3), prove(Cla3,Mat1,[Lit|Path])
   ), prove(Cla,Mat,Path).
```

- leanCoP er en elegant Prolog-implementasjon av koblingskalkylen for klassisk logikk i normalform.
- Utviklet av Jens Otten ved Universitetet i Potsdam utenfor Berlin.
- Utnytter Prologs innebygde unifikasjon og backtracking.
- For mer info: `http://www.leancop.de/`