Klauseln und Cut im AK

■ Definitionen

Die Darstellung einer Klausel der Form

$$p_1 \wedge \ldots \wedge p_n \rightarrow q_1 \vee \ldots \vee q_m$$

geschieht hier als ein Paar von Listen, d.h. in der Form

$$\{\{p_1, \ldots, p_n\}, \{q_1, \ldots, q_m\}\}$$

Der Widerspruch hat dann diese Klauseldarstellung:

```
{{}, {}}
```

Damit nimmt die cut Regel folgende Form an: Wenn zwei Klauseln eine Variable enthalten, einmal positiv, d.h. in der ersten Komponente und einmal negativ, d.h. in der zweiten Komponente, dann kann man die Variable dort streichen und die rechten und die linken Seiten zu einer neuen Klausel vereiningen. Dies macht die folgende in *Mathematica* Notation formulierte Regel:

```
generalCutRule =
  cut[{{a1___, x_, e1___}, l1_}, {l2_, {a2___, x_, e2___}}] :>
   {{a1, e1} ~Union~ l2, l1 ~Union~ {a2, e2}};
```

Alle Schnitte für eine gegebene Klauselmenge erhalten wir mit

```
allCuts[klauselMenge_] :=
  ReplaceList[#, generalCutRule] & /@
     (Outer[cut, #, #, 1] &[klauselMenge] // Flatten[#, 1] &) //
    Flatten[#, 1] & // Union[#, klauselMenge] &;
```

Der Code ist nicht besonders effizient (Symmetrie ist zum Beispiel nicht berücksichtigt), aber kurz!

```
cutAbschluß[klauselMenge_] := FixedPoint[allCuts, klauselMenge];
```

Und das war schon alles was wir brauchen.

■ Beispiele

- **Das Geheimnis des 100-jährigen (aus Uwe Schöning's Buch Logik für Informatiker)**
  Aufgabentext siehe auch hier AL-100J.html

Bier - b
Eis - e
Fisch - f

```
klauselMenge100J = {
  {{b, f}, {}},
  {{}, {b, e, f}},
  {{}, {e, f}},
  {{b}, {f}}
 }
```

```
{{{b, f}, {}}, {{}, {b, e, f}}, {{}, {e, f}}, {{b}, {f}}}
```

```
cutAbschluß[klauselMenge100J] // MatrixForm
```

$$\begin{pmatrix} \{\} & \{e, f\} \\ \{\} & \{b, e, f\} \\ \{b\} & \{\} \\ \{b\} & \{e\} \\ \{b\} & \{f\} \\ \{b\} & \{b, e\} \\ \{b\} & \{e, f\} \\ \{f\} & \{e, f\} \\ \{b, f\} & \{\} \\ \{b, f\} & \{e\} \end{pmatrix}$$

```
% // Length
```

```
10
```

Was es damit auf sich hat siehe weiter unten.

```
cutAbschlußRed[klauselMenge100J]
```

```
{{{}, {e, f}}, {{}, {b, e, f}},
 {{b}, {}}, {{b}, {e}}, {{b}, {f}}, {{b, f}, {}}}
```

```
% // Length
```

```
6
```

Was ist die Folgerung? (Gute Aussichten ;-)

- **UA 1 Serie 2 2002 (L)**

```
klauselMengeSerie2UA1L = {
   {{p, r, t}, {}},
   {{}       , {q, s}},
   {{p, s}   , {}},
   {{p, t}   , {}},
   {{}       , {s, t}},
   {{p, r}   , {}},
   {{}       , {p, s}}};
```

Aus der Klauselstruktur folgt, **q** und ¬**r** gehen nicht zu erzeugen. Und wenn ¬**q** bzw. **r** in einer Klausel sind, wird man die durch Schneiden nie wieder los. Also betrachten wir die reduzierte Klauselmenge

```
klauselMengeRedSerie2UA1L = {
   {{p, s}   , {}},
   {{p, t}   , {}},
   {{}       , {s, t}},
   {{}       , {p, s}}};
```

```
caRa = cutAbschluß[klauselMengeRedSerie2UA1L] // MatrixForm
```

{}	{s}
{}	{p, s}
{}	{s, t}
{p}	{}
{p}	{p}
{p}	{s}
{p}	{t}
{s}	{s}
{t}	{s}
{p, s}	{}
{p, t}	{}

```
caRa // Length
```

```
11
```

Die Klauseln die nur aus Literalen bestehen erhalten wir so:

```
Cases[caRa, {{}, {x_}} | {{x_}, {}}]
```

```
{{{}, {s}}, {{p}, {}}}
```

Durch Inspektion sehen wir, auch **¬q** bzw. **r** können mit den erhaltenen Literalen nicht erhalten werden.

Oder vollständig ohne Tricks:

```
caa = cutAbschluß[klauselMengeSerie2UA1L]
```

```
{{{}, {s}}, {{}, {p, s}}, {{}, {q, s}}, {{}, {s, t}},
 {{p}, {}}, {{p}, {p}}, {{p}, {q}}, {{p}, {s}}, {{p}, {t}},
 {{r}, {s}}, {{s}, {s}}, {{t}, {s}}, {{p, r}, {}}, {{p, r}, {s}},
 {{p, s}, {}}, {{p, t}, {}}, {{r, t}, {s}}, {{p, r, t}, {}}}
```

```
caa // Length
```

```
18
```

```
Cases[caa, {{}, {x_}} | {{x_}, {}}]
```

```
{{{}, {s}}, {{p}, {}}}
```

Schlußfolgerung:
Solingen: Ja,
Passau: Nein,
Rest: Vielleicht

- **UA 2 Serie 2 2002 (L)**

```
klauselMengeSerie2UA2L = {
   {{p, q}   , {}},
   {{q, s}   , {}},
   {{s}      , {t}},
   {{p}      , {s, t}},
   {{q}      , {s, t}},
   {{r}      , {s, t}},
   {{r}      , {p, q}},
   {{p, t}   , {q}},
   {{p}      , {s, q}}
  };
```

Wir überprüfen zuerst, ob der Widerspruch und welche Literale ableitbar sind.
Aus der Klauselstruktur folgt, ¬**r** geht nicht zu erzeugen. Und wenn **r** in einer Klausel ist, wird man **r** durch Schneiden nie wieder los. Also betrachten wir die reduzierte Klauselmenge:

```
klauselMengeRedSerie2UA2L = {
   {{p, q}   , {}},
   {{q, s}   , {}},
   {{s}      , {t}},
   {{p}      , {s, t}},
   {{q}      , {s, t}},
   {{p, t}   , {q}},
   {{p}      , {s, q}}
  };
```

Den vollständigen Cutabschluß brauchen wir nicht zu berechnen, denn Fälle in denen nur Tautologien entstehen, bringen keine neuen Erkenntnisse. Also Paare in denen eine Variable links und rechts vorkommt, brauchen nur bei dieser Variable geschnitten werden, wenn es nicht noch ein zweites solches Variablenpaar gibt.

```
(caRa = cutAbschluß[klauselMengeRedSerie2UA2L]) // MatrixForm
```

{p}	{}
{p}	{q}
{p}	{s}
{p}	{t}
{p}	{q, s}
{p}	{q, t}
{p}	{s, q}
{p}	{s, t}
{p}	{q, s, t}
{q}	{t}
{q}	{s, t}
{s}	{t}
{p, q}	{}
{p, q}	{q}
{p, q}	{s}
{p, q}	{t}
{p, q}	{q, s}
{p, q}	{q, t}
{p, q}	{s, t}
{p, q}	{q, s, t}

{p, s}	{}
{p, s}	{q}
{p, s}	{s}
{p, s}	{t}
{p, s}	{q, s}
{p, s}	{q, t}
{p, s}	{s, t}
{p, s}	{q, s, t}
{p, t}	{}
{p, t}	{q}
{p, t}	{s}
{p, t}	{t}
{p, t}	{q, s}
{p, t}	{q, t}
{p, t}	{s, t}
{p, t}	{q, s, t}
{q, s}	{}
{p, q, s}	{}
{p, q, s}	{q}
{p, q, s}	{s}
{p, q, s}	{t}
{p, q, s}	{q, s}
{p, q, s}	{q, t}
{p, q, s}	{s, t}
{p, q, s}	{q, s, t}
{p, q, t}	{}
{p, q, t}	{q}
{p, q, t}	{s}
{p, q, t}	{t}
{p, q, t}	{q, s}
{p, q, t}	{q, t}
{p, q, t}	{s, t}
{p, q, t}	{q, s, t}
{p, s, t}	{}
{p, s, t}	{q}
{p, s, t}	{s}
{p, s, t}	{t}
{p, s, t}	{q, s}
{p, s, t}	{q, t}
{p, s, t}	{s, t}
{p, s, t}	{q, s, t}
{p, q, s, t}	{}
{p, q, s, t}	{q}
{p, q, s, t}	{s}
{p, q, s, t}	{t}
{p, q, s, t}	{q, s}
{p, q, s, t}	{q, t}
{p, q, s, t}	{s, t}
{p, q, s, t}	{q, s, t}

caRa // Length

69

Davon sind aber noch eine Reihe Klauseln Tautologien, d.h. eine Variable kommt sowohl links als auch in der rechts vor, nämlich diese:

```
caReq = Cases[caRa, {{a1___, x_, e1___}, {a2___, x_, e2___}}]
```

```
{{{p, q}, {q}}, {{p, q}, {q, s}}, {{p, q}, {q, t}},
 {{p, q}, {q, s, t}}, {{p, s}, {s}}, {{p, s}, {q, s}},
 {{p, s}, {s, t}}, {{p, s}, {q, s, t}}, {{p, t}, {t}}, {{p, t}, {q, t}},
 {{p, t}, {s, t}}, {{p, t}, {q, s, t}}, {{p, q, s}, {q}},
 {{p, q, s}, {s}}, {{p, q, s}, {q, s}}, {{p, q, s}, {q, t}},
 {{p, q, s}, {s, t}}, {{p, q, s}, {q, s, t}}, {{p, q, t}, {q}},
 {{p, q, t}, {t}}, {{p, q, t}, {q, s}}, {{p, q, t}, {q, t}},
 {{p, q, t}, {s, t}}, {{p, q, t}, {q, s, t}}, {{p, s, t}, {s}},
 {{p, s, t}, {t}}, {{p, s, t}, {q, s}}, {{p, s, t}, {q, t}},
 {{p, s, t}, {s, t}}, {{p, s, t}, {q, s, t}}, {{p, q, s, t}, {q}},
 {{p, q, s, t}, {s}}, {{p, q, s, t}, {t}}, {{p, q, s, t}, {q, s}},
 {{p, q, s, t}, {q, t}}, {{p, q, s, t}, {s, t}}, {{p, q, s, t}, {q, s, t}}}
```

```
caReq // Length
```

```
37
```

Also brauchen wir eigentlich nur die folgenden Klauseln wirklich zu berechnen:

```
Complement[caRa, caReq]
```

```
{{{p}, {}}, {{p}, {q}}, {{p}, {s}}, {{p}, {t}},
 {{p}, {q, s}}, {{p}, {q, t}}, {{p}, {s, q}}, {{p}, {s, t}},
 {{p}, {q, s, t}}, {{q}, {t}}, {{q}, {s, t}}, {{s}, {t}},
 {{p, q}, {}}, {{p, q}, {s}}, {{p, q}, {t}}, {{p, q}, {s, t}},
 {{p, s}, {}}, {{p, s}, {q}}, {{p, s}, {t}}, {{p, s}, {q, t}},
 {{p, t}, {}}, {{p, t}, {q}}, {{p, t}, {s}}, {{p, t}, {q, s}},
 {{q, s}, {}}, {{p, q, s}, {}}, {{p, q, s}, {t}}, {{p, q, t}, {}},
 {{p, q, t}, {s}}, {{p, s, t}, {}}, {{p, s, t}, {q}}, {{p, q, s, t}, {}}}
```

```
% // Length
```

```
32
```

Nun schauen wir nach, welche Literale im Cutabschluß sind und ob evtl. der Widerspruch abgeleitet wurde:

```
Cases[caRa, {{}, {x_}} | {{x_}, {}} | {{}, {}}]
```

```
{{{p}, {}}}
```

Also ist nur **p,** nicht jedoch ¬**p** ableitbar.

Vollständig ohne Tricks:

```
caa = cutAbschluß[klauselMengeSerie2UA2L]
```

```
caa // Length
```

```
309
```

```
Cases[caa, {{}, {x_}} | {{x_}, {}} | {{}, {}}]
```

```
{{{p}, {}}}
```

Für die restlichen Ausdrücke müssen wir jeweils den zu prüfenden Ausdruck negieren, in Klauselform C_2 (evtl mehrere Klauseln!) umformen und überprüfen, ob sich aus einer zu X äquivalenten Klauselmenge C_1 und der Vereinigung mit C_2 der Widerspruch ableiten läßt. Nur dann, folgt *H* aus **X**.

Wir bezeichnen die zu überprüfenden Ausdrücke von links nach rechts mit H_1, ..., H_6.

Für $H_1 = \mathbf{p}$ und $H_2 = \neg\mathbf{p}$ haben wir die Antwort schon gefunden.

Für $H_3 = (\mathbf{p} \vee \mathbf{r})$ gilt $\neg H_3 =_{sem} \neg\mathbf{p} \wedge \neg\mathbf{r}$ und mit dieser Klausel kann man den Widerspruch nicht herleiten, wenn er schon aus C_1 nicht herleitbar war, weil wieder $\neg\mathbf{r}$ nicht wegzukriegen ist.

```
klauselMengeSerie2UA2L = {
   {{p, q}   , {}},
   {{q, s}   , {}},
   {{s}      , {t}},
   {{p}      , {s, t}},
   {{q}      , {s, t}},
   {{r}      , {s, t}},
   {{r}      , {p, q}},
   {{p, t}   , {q}},
   {{p}      , {s, q}}
  };
```

Jetzt folgt reine Hand- bzw. Kopfrechnung. Für H_4 müssen wir die Klauseln **p** und **r** zu C_1 hinzunehmen und reduzieren damit die Klauselmenge **klauselMengeSerie2UA2L** sofort zu

```
{
  {{}}, {{p}},
  {{}}, {{r}},
  {{q}   , {}},
  {{q, s}   , {}},
  {{s}      , {t}},
  {{p}      , {s, t}},
  {{q}      , {s, t}},
  {{}      , {s, t}},
  {{}      , {p, q}},
  {{t}   , {q}},
  {{}      , {s, q}}
 };
```

Im nächsten Schritt schneiden wir mit $\neg\mathbf{q}$ und erhalten

```
{
  {{}}, {{p}},
  {{}}, {{r}},
  {{q}   , {}},
  {{q, s}   , {}},
  {{s}      , {t}},
  {{p}      , {s, t}},
  {{q}      , {s, t}},
  {{}      , {s, t}},
  {{}      , {p}},
  {{t}   , {}},
  {{}      , {s}}
 };
```

Im nächsten Schritt schneiden wir mit $\neg\mathbf{t}$ und erhalten

```
{
  {{}}, {{p}},
  {{}}, {{r}},
  {{q}   , {}},
  {{q, s}   , {}},
  {{s}      , {}},
  {{p}      , {s}},
  {{q}      , {s}},
  {{}      , {s}},
  {{}      , {p}},
  {{t}   , {}},
  {{}      , {s}}
 };
```

Und nun sehen wir schon, daß wir mit **s** und ¬**s** den Widerspruch nachweisen können. Also H_4 ist ableitbar.

Als nächstes zu H_5 :

```
klauselnH5 = {
   {{}      , {p, q, r}},
   {{p, q}   , {}},
   {{q, s}   , {}},
   {{s}      , {t}},
   {{p}      , {s, t}},
   {{q}      , {s, t}},
   {{r}      , {s, t}},
   {{r}      , {p, q}},
   {{p, t}   , {q}},
   {{p}      , {s, q}}
  };
cutAbschluß[klauselnH5]


% // Length
961
```

Dies hat wirklich etwas zu lange gedauert, also lassen wir das Erzeugen von Tautologien und definieren eine neue reduzierte cut-Regel!

```
generalCutRuleRed =
  cut[{{a1___, x_, e1___}, l1_}, {l2_, {a2___, x_, e2___}}] :>
   {{a1, e1} ~Union~ l2, l1 ~Union~ {a2, e2}} /;
    Intersection[{a1, e1} ~Union~ l2, l1 ~Union~ {a2, e2}] == {};
```

```
allCutsRed[klauselMenge_] :=
  ReplaceList[#, generalCutRuleRed] & /@
     (Outer[cut, #, #, 1] &[klauselMenge] // Flatten[#, 1] &) //
    Flatten[#, 1] & // Union[#, klauselMenge] &;
```

```
cutAbschlußRed[klauselMenge_] :=
  FixedPoint[allCutsRed, klauselMenge];
```

Das reduziert Zeit und Speicherplatz doch schon deutlich wie wir nachfolgend sehen ;-).

```
caKlauselnH5 = cutAbschlußRed[klauselnH5];
```

```
caKlauselnH5 // Length
```

```
166
```

```
Cases[caKlauselnH5, {{}, {x_}} | {{x_}, {}} | {{}, {}}]
```

```
{{{}, {q}}, {{}, {t}}, {{p}, {}}, {{s}, {}}}
```

Kein Widerspruch ableitbar. Sorry, ja, für eine Handrechnung ist dieser Fall wohl immer noch zu aufwendig. :-(.

```
klauselnH6 = {
   {{q}      , {r}},
   {{p, q}   , {}},
   {{q, s}   , {}},
   {{s}      , {t}},
   {{p}      , {s, t}},
   {{q}      , {s, t}},
   {{r}      , {s, t}},
   {{r}      , {p, q}},
   {{p, t}   , {q}},
   {{p}      , {s, q}}
  };
```

```
caKlauselnH6 = cutAbschlußRed[klauselnH6];
```

```
% // Length
```

```
116
```

```
Cases[caKlauselnH6, {{}, {x_}} | {{x_}, {}} | {{}, {}}]
```

```
{{{p}, {}}}
```

Also H_6 auch nicht ableitbar!