Übung zur Vorlesung

Inferenzmethoden

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2010/11

Blatt 2 — Abgabetermin: (18. November 2010)

Aufgabe 2.1 (Aussagenlogische Konnektionsmethode)

Prüfen Sie mit Hilfe der (allgemeinen) Konnektionsmethode, ob die folgenden Matrizen gültig sind. Geben Sie entweder einen Beweis oder ein Gegenbeispiel an.

$$\begin{bmatrix} A^T & A^F & A^T \\ B^T & B^F & B^F \end{bmatrix} \qquad \begin{bmatrix} A^T & A^F & C^T & B^F \\ B^T & C^F & & \end{bmatrix} \qquad \begin{bmatrix} A^T & B^F & C^F & D^T & A^F \\ B^T & C^T & D^F & B^F & \end{bmatrix}$$

$$\begin{bmatrix} A^T & A^F & A^F & B^F & C^F \\ B^T & B^F & B^T & A^T & A^F \\ C^T & & & & \end{bmatrix} \qquad \begin{bmatrix} A^F & C^T & A^T & B^F & C^F \\ B^T & D^F & & D^T & \\ & B^T & & & \end{bmatrix} \qquad \begin{bmatrix} A^T & A^T & A^F & A^F & B^F & C^F \\ B^F & B^T & B^T & B^T & & \\ C^T & C^T & C^T & C^F & & \end{bmatrix}$$

Zur Zeit- und Platzersparnis sollten Sie mehrere Schritte in einer Matrix durchführen. Numerieren Sie dazu die Konnektionen in der verwendeten Reihenfolge durch, und machen Sie die Richtung der Konnektionen deutlich.

Aufgabe 2.2 (Komplexität des Extensionsverfahrens)

2.2–a Zeigen Sie durch Angabe eines Beispiels, daß das Extensionsverfahren im schlimmsten Fall exponentiellen Zeitaufwand (gemessen an der Anzahl der Variablen und Klauseln) benötigt.

2.2–b Wie ist die Komplexität des Verfahrens, wenn man sich auf Matrizen mit maximal 2 Literalen pro Klausel beschränkt?
Welche Konsequenz hat dieses Ergebnis für die Komplexität des 2SAT Problems?

2.2–c Wie stark würde ein Beweisverfahren beschleunigt werden, wenn man anstelle des im schlimmsten Fall exponentiellen Robinson Unifikationsalgorithmus ein Unifikationsverfahren programmieren könnte, das in jeder Situation in genau einem Schritt den allgemeinsten Unifikator bestimmt oder Nichtunifizierbarkeit feststellt?

Aufgabe 2.3 (Unifikation)

Seien f, g, h Funktionszeichen, a, c Konstanten und x, y, z, u Variablen.

Versuchen Sie, die folgenden Termpaare mit Hilfe des Herbrand-Robinson-Algorithmus zu unifizieren. Stellen Sie diesen Vorgang tabellarisch dar.

2.3–a $g(f(x), a, x)$ und $g(y, z, z)$

2.3–b $g(y, h(x), h(a))$ und $g(c, h(h(y)), x)$

2.3–c $g(g(h(x), y), z)$ und $g(z, g(y, h(a)))$

2.3–d $g(g(g(z, a), z), z)$ und $g(g(x, y), x)$

Versuchen Sie, die folgenden Termpaare mit Hilfe des Martelli-Montanari-Algorithmus zu unifizieren. Stellen Sie den Vorgang schrittweise unter Angabe der verwendeten Umformungsregeln dar.

2.3–e $g(h(a), h(h(y)))$ und $g(x, h(x))$

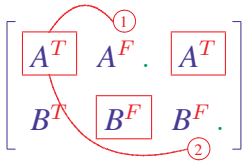
2.3–f $f(h(y), g(y, a), h(x))$ und $f(x, g(z, a), h(z))$

2.3–g $f(g(x, y), h(a), z)$ und $f(z, y, g(c, a))$

Lösung 2.1

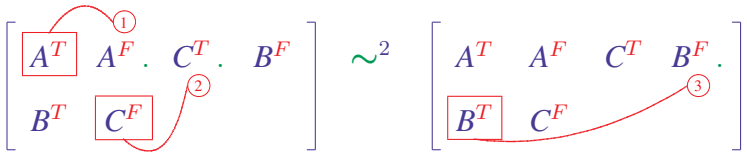
Das Verfahren wird jeweils bis zum nächsten Bereinigungs- oder Rücksetzungsschritt innerhalb von einer Matrix beschrieben. Indizes an den Konnektionen numerieren die Reihenfolge ihrer Anwendung.

2.1–a



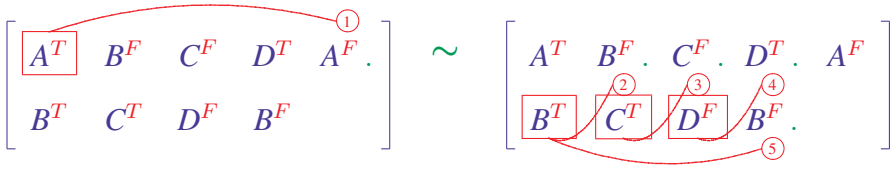
Keine weitere Extension, Bereinigung oder Rücksetzung möglich.
Nicht gültig, Gegenbeispiel $\neg A \wedge B$

2.1–b



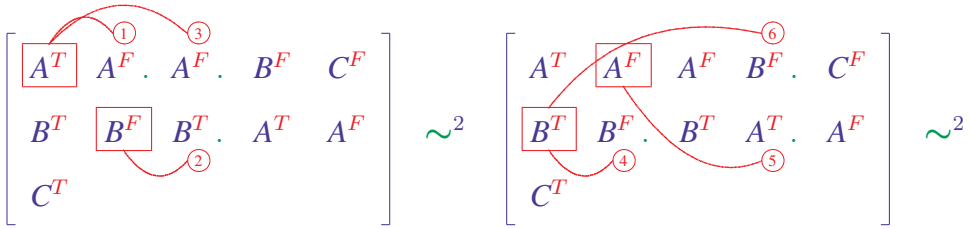
Gültig

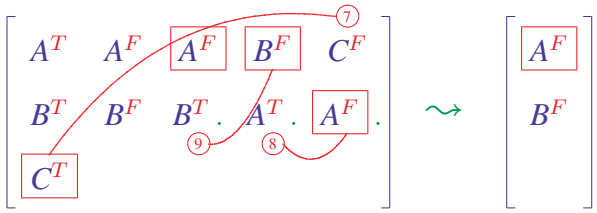
2.1–c



Gültig

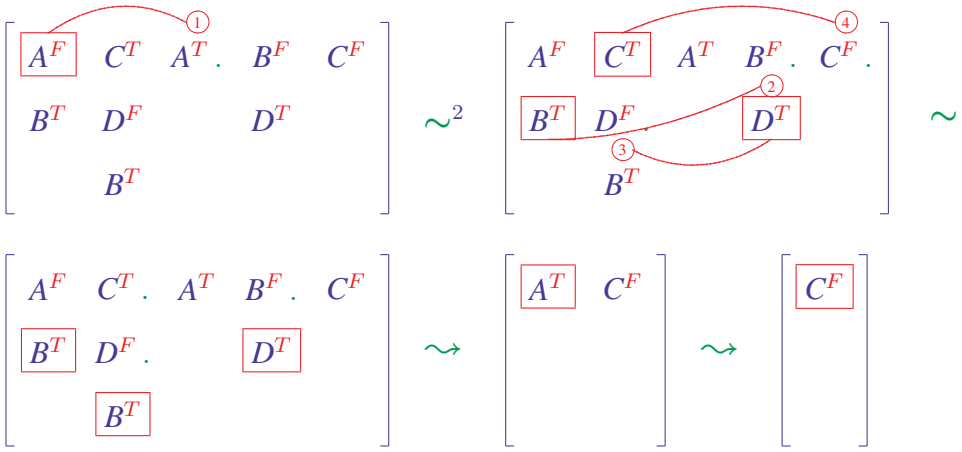
2.1–d





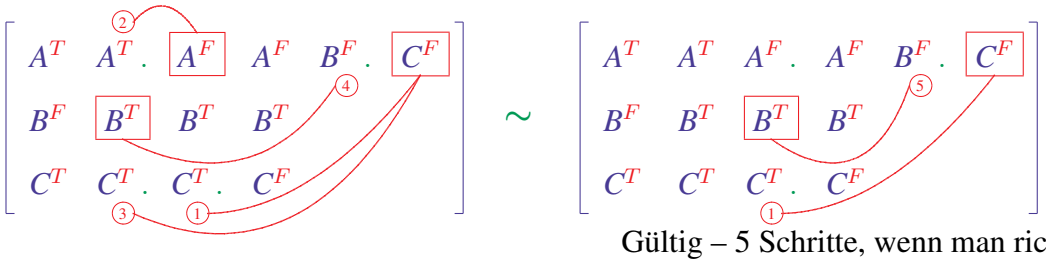
Keine Extension, Bereinigung oder Rücksetzung möglich.
Nicht gültig, Gegenbeispiel $A \wedge B \wedge \neg D$
Die Matrix wäre gültig, wenn in Klausel 3 ein A^T statt dem A^F stehen würde.

2.1–e



Keine Extension, Bereinigung oder Rücksetzung möglich.
Nicht gültig, Gegenbeispiel $\neg A \wedge \neg B \wedge C \wedge \neg D$
Die Matrix wäre gültig, wenn in Klausel 2 ein B^F statt dem B^T stehen würde.

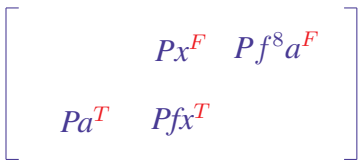
2.1–f



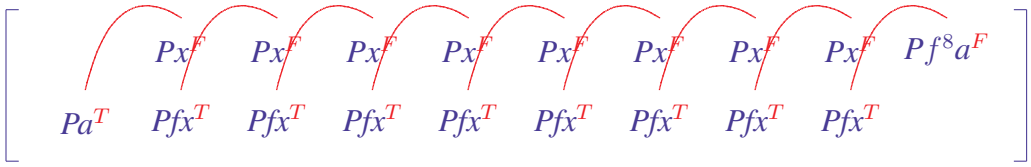
Gültig – 5 Schritte, wenn man richtig anfängt

Lösung 2.2

2.2–a Als erstes Beispiel, das gerne benutzt wurde um zu zeigen, daß Resolution schneller als die Konnetktionsmethode ist, nehmen wir folgende prädikatenlogische Formel



Das Extensionsverfahren (ohne Reduktionstechniken) würde von der mittleren Klausel insgesamt 8 Kopien ziehen müssen um den Beweis zu führen



Ersetzt man 8 durch 2^n, dann sieht man schnell, daß 2^n Kopien gezogen werden müssen, währen der Resolutionsbeweis zuerst $\{\mathrm{P}f^2\mathrm{a}$, dann Pf^4a^T, ... und schließlich $Pf^{2^n}a^T$ ableitet und hieraus die leere Klausel erzeugt.

Dies ist aber kein echtes Beispiel, da auch der Text des letzten Literals $Pf^{2^n}a^F$ exponentiell wächst und somit die Laufzeit linear bleibt. Ein echtes Beispiel muß rein aussagenlogisch sein und bis auf wenige Ausnahmen mindestens 3 Literale pro Klausel verwenden

Eine aussagenlogische Matrix, die exponentielle Laufzeit erzeugt, müsste dafür sorgen, daß in n Klauseln mindestens 2^n Pfade durchlaufen werden müssen. Wenn man pro Klausel minde-

stens 3 Literale verwendet, dann gibt es in jedem Extensionsschritt mindestens 2 Alternative für die weitere Verarbeitung. Durch die Gestaltung der Matrix muß nun sichergestellt werden, daß keine dieser Alternativen vor Erreichen der vollen Pfadlänge n terminiert.

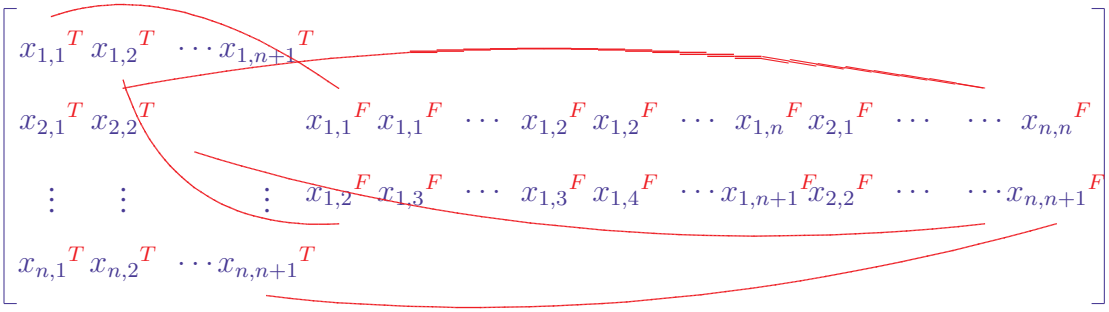
Auch wenn die Grundidee relativ klar ist, ist es nicht leicht eine konkrete Matrix dieser Art zu gestalten. Es wird ziemlich schnell ersichtlich, daß man neben den n "Hauptklauseln" eine Reihe von Zwischenklauseln benötigt, die dafür sorgen, daß Pfade nicht frühzeitig abgeschlossen werden können. Wenn die Anzahl dieser Klauseln polynomiell in n ist, bleibt die Laufzeit dann insgesamt exponentiell in der Größe der Matrix.

In der Literatur gelten die sogenannten Pigeonhole Formeln als Standardbeispiel für ein schweres aussagenlogisches Beweisproblem. Sie sagen aus, daß man $n+1$ Tauben nicht in n Taubenschlägen unterbringen kann, weil ansonsten 2 Tauben im gleichen Taubenschlag sein müssen. Im der deutschen Literatur spricht man auch vom "Schubfachprinzip".

In Logik erster Stufe mit Zahlen ist dies ein sehr einfach zu lösendes Problem. Der Beweis wird mittels Induktion geführt und ist sehr schnell durchzuführen. In der Aussagenlogik ist es jedoch ein komplexes Problem und führt zu einer Schar von Formeln $P_n, n \in \mathbb{N}$, die für ein konkretes n besagen:

> *Wenn Taube 1 sich im Taubenschlag 1 befindet oder ... Taube 1 sich im Taubenschlag n befindet, und wenn Taube 2 sich im Taubenschlag 1 befindet oder ... Taube $n+1$ sich im Taubenschlag n befindet,*
> *dann ist Taube 1 im Taubenschlag 1 und Taube 2 im Taubenschlag 1,*
> *oder Taube 1 im Taubenschlag 1 und Taube 3 im Taubenschlag 1,*
> *... oder Taube n im Taubenschlag n und Taube $n+1$ im Taubenschlag n.*

Mit $x_{i,j}$ drücken wir aus "Taubenschlag i ist durch Taube j belegt". Nach Normalisieren der obigen Aussage ergibt sich dann die folgende Matrix



Die Matrix hat $n(n+1)$ Variablen und $n+1+\frac{n^3+n^2}{2}$ Klauseln.

Um den in ${x_{1,1}}^T$ beginnenden Pfad zu schließen, müssen in jedem Schritt eine der Zweierklauseln und aus diesem eine neue (!) n-Klausel angesteuert werden. In der Zweierklausel gibt es genau eine Möglichkeit, aber in der konnektierten n-Klausel fallen jeweils $n-1$ Alternativen an. Da alle mit T markierten Klauseln durchlaufen werden müssen, fallen insgesamt $\mathcal{O}(n^n)$ Konnektionen an, die das Extensionsverfahren untersucht. Damit ist die Laufzeit insgesamt exponentiell.

2.2–b Wenn die Matrix maximal 2 Literalen pro Klausel besitzt, dann wird das Extensionsverfahren in jedem Schritt (mindestens) ein Literal einer neuen Klausel konnektieren und das ggf. verbleibende Literal zum aktuellen Pfad hinzufügen. Weitere Alternativen gibt es nicht. Der erste Bereinigungsschritt wird somit (iterativ) den gesamten aktuellen Pfad bis zur Startklausel zurückbauen.

Bei insgesamt n Klauseln wird das Verfahren in maximal n Schritten den im Startliteral beginnenden Pfad abgeschlossen oder ein Gegenbeispiel gefunden haben. Bei maximal 2 Startliteralen fallen also (einschließlich Bereinigungsschritten) maximal $4n$ Schritte an, d.h. die Gültigkeit von M kann in linearer Zeit (relativ zur Anzahl der Klauseln) bewiesen werden.

Im Gegensatz zum $\mathcal{NP}$-vollständigen 3SAT liegt 2SAT in $\mathcal{P}$ oder genauer in $\mathcal{O}(n)$.

2.2–c Ein Beweisverfahren führt in nahezu jedem Schritt Unifikationen durch und dabei sind die konnektierten Literale unter der bisherigen Substitution entwerder identisch oder nicht unifizierbar. Beides stellt der Robinson Unifikationsalgorithmus in wenigen Schritten fest und deshalb hat er im Mittelwert über alle Unifikationsversuche des Beweisverfahrens konstante Laufzeit.

Eine Reduktion auf genau einen Schritt wird also nur eine Beschleunigung des Verfahrens um einen (kleinen) konstanten Faktor mit sich bringen. Im Verhältnis zu Reduktionsverfahren und guten Suchstrategien, die Beschleunigungen um Faktoren wie 10^{10} erzielen (durch frühzeitige Begrenzung des Suchraums) ist dieser Effekt vernachlässigbar.

Lösung 2.3

2.3–a

Nr.	$DIFF(s\sigma, t\sigma)$	σ
0	$\{\{f(x), y\},\ \{a, z\},\ \{x, z\}\}$	[]
1	$\{\{a, z\},\ \{x, z\}\}$	$[f(x)/y]$
2	$\{\{x, a\}\}$	$[f(x)/y, a/z]$
3	$\{\}$	$[f(a)/y, a/z, a/x]$

2.3–b

Nr.	$DIFF(s\sigma, t\sigma)$	σ
0	$\{\{x, h(y)\},\ \{y, c\},\ \{h(a), x\}\}$	[]
1	$\{\{y, c\},\ \{a, y\}\}$	$[h(y)/x]$
2	$\{\{a, c\}\}$	$[h(c)/x, c/y]$

Die vorhandene Differenz ist nicht verhandelbar 2 Konstanten).

2.3–c

Nr.	$DIFF(s\sigma, t\sigma)$	σ
0	$\{\{g(h(x), y), z\},\ \{z, g(y, h(a))\}\}$	[]
1	$\{\{h(x), y\},\ \{y, h(a)\}\}$	$[g(h(x), y)/z]\}$
2	$\{\{x, a\}\}$	$[g(h(x), h(x))/z, h(x)/y]$
3	$\{\}$	$[g(h(a), h(a))/z, h(a)/y, a/x]$

2.3–d

Nr.	$DIFF(s\sigma, t\sigma)$	σ
0	$\{\{g(z, a), x\},\ \{z, y\},\ \{z, x\}\}$	[]
1	$\{\{z, y\},\ \{z, g(z, a)\}\}$	$[g(z, a)/x]$
2	$\{\{y, g(y, a)\}\}$	$[g(y, a)/x, y/z]$

Die vorhandene Differenz ist nicht verhandelbar (Occurs check).

2.3–e $\{g(h(a), h(h(y)))\doteq g(x, h(x))\}$
$\rightsquigarrow \{h(a)\doteq x,\ h(h(y))\doteq h(x)\}$
$\rightsquigarrow \{x\doteq h(a),\ h(h(y))\doteq h(x)\}$
$\rightsquigarrow \{x\doteq h(a),\ h(h(y))\doteq h(h(a))\}$
$\rightsquigarrow \{x\doteq h(a),\ h(y)\doteq h(a)\}$
$\rightsquigarrow \{x\doteq h(a),\ y\doteq a\}$

Die Menge ist im Zielformat. Als Unifikator ergibt sich: $[h(a)/x, a/y]$.

2.3–f $\{f(h(y), g(y, a), h(x))\doteq f(x, g(z, a), h(z))\}$
$\rightsquigarrow \{h(y)\doteq x,\ g(y, a)\doteq g(z, a),\ h(x)\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ g(y, a)\doteq g(z, a),\ h(x)\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ g(y, a)\doteq g(z, a),\ h(h(y))\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ y\doteq z,\ a\doteq a,\ h(h(y))\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ y\doteq z,\ a\doteq a,\ h(h(z))\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ y\doteq z,\ h(h(z))\doteq h(z)\}$
$\rightsquigarrow \{x\doteq h(y),\ y\doteq z,\ h(z)\doteq z\}$
$\rightsquigarrow \{x\doteq h(y),\ y\doteq z,\ z\doteq h(z)\}$

Die beiden Terme sind nicht unifizierbar ("$z \in h(z)$").

2.3–g $\{f(g(x, y), h(a), z)\doteq f(z, y, g(c, a))\}$
$\rightsquigarrow \{g(x, y)\doteq z,\ h(a)\doteq y,\ z\doteq g(c, a)\}$
$\rightsquigarrow \{z\doteq g(x, y),\ h(a)\doteq y,\ z\doteq g(c, a)\}$
$\rightsquigarrow \{z\doteq g(x, y),\ h(a)\doteq y,\ g(x, y)\doteq g(c, a)\}$
$\rightsquigarrow \{z\doteq g(x, y),\ y\doteq h(a),\ g(x, y)\doteq g(c, a)\}$
$\rightsquigarrow \{z\doteq g(x, y),\ y\doteq h(a),\ g(x, h(a))\doteq g(c, a)\}$
$\rightsquigarrow \{z\doteq g(x, y),\ y\doteq h(a),\ x\doteq c,\ h(a)\doteq a\}$

Die beiden Terme sind nicht unifizierbar (keine Variable eliminierbar).