Lernalgorithmen

SoSe 2012 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Lernalgorithmen Gesamtübersicht

0. Einführung

1. Identifikation (aus positiven Beispielen)

2. Zur Identifikation regulärer Sprachen, mit XML-Anwendung

3. HMM — Hidden Markov Models

4. Lernen mittels Anfragen & zur Roboterorientierung

5. Lernen mit negativen Beispielen

6. PAC-Lernen

Informanten-Lernen (Hauptquelle: Das Lehrbuch von C. de la Higuera)

Wiederum: Beschränkung auf reguläre Sprachen / DEAs.

Beispielstrom für L kommt mit positiven oder negativen Markierungen, d.h.: $(w, +)$ ist im Beispielstrom gdw. $w \in L$, und $(w, -)$ ist im Beispielstrom gdw. $w \notin L$. Nach "endlicher Zeit" haben wir daher zwei Sample-Mengen X_+ und X_- gesehen, die zusammen das Sample $X = (X_+, X_-)$ bilden, mit $X_+ \subseteq L$ und $X_- \subseteq \bar{L}$.

Wir zeichnen jetzt bei DEAs eine Menge akzeptierender Zustände F_A und eine Menge *verwerfender* Zustände F_R (mit $F_A \cap F_R = \emptyset$) aus.
$A = (\Sigma, Q, q_0, F_A, F_R, \delta)$ heißt *schwach konsistent* mit $X = (X_+, X_-)$ gdw. $\forall w \in X_+ : \delta^*(q_0, w) \in F_A$ und $\forall w \in X_- : \delta^*(q_0, w) \notin F_A$.
A heißt *stark konsistent* mit $X = (X_+, X_-)$ gdw. $\forall w \in X_+ : \delta^*(q_0, w) \in F_A$ und $\forall w \in X_- : \delta^*(q_0, w) \in F_R$.

Übersicht über die heutige Vorlesung

- Grundkonzepte und Probleme beim Informanten-Lernen

- Vorstellung zweier Algorithmen
  Gold und RPNI

- Was macht Informanten-Lernen schwierig

Grundkonzepte

- Präfixbäume mit positiven und negativen Beispielen für den Algorithmus von Gold

- Allgmeines zu Zustandsverschmelzungsalgorithmen:
  - Kompatibilität
  - Verschmelzen
  - Befördern

- Beobachtungstabelle

Präfixbäume mit positiven und negativen Beispielen

Algorithm 8.1: BUILD-PTA

Data: a sample $\langle X_+, X_- \rangle$

Result: A PTA : $\mathcal{A} = \text{PTA}(\langle X_+, \emptyset \rangle)) = \langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbf{F}_\mathbb{R}, \delta \rangle$

$Q \leftarrow \{q_u : u \in \text{PREF}(X_+ \cup X-)\}$;

for $q_{u \cdot a} \in Q$ **do**

  $\delta(q_u, a) \leftarrow q_{ua}$

end

for $q_u \in Q$ **do**

  if $u \in X_+$ **then**

    $\mathbb{F}_\mathbb{A} \leftarrow \mathbf{F}_\mathbb{A} \cup \{q_u\}$

  end

  if $u \in X_-$ **then**

    $\mathbb{F}_\mathbb{R} \leftarrow \mathbf{F}_\mathbb{R} \cup \{q_u\}$

  end

end

Präfixbäume mit positiven und negativen Beispielen

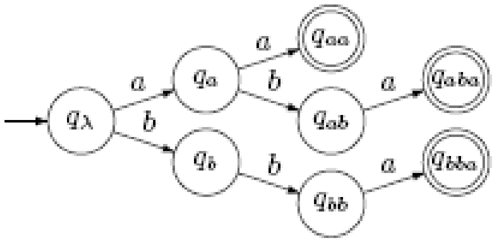


Fig. 8.3. PTA$(\{(aa,+)\,(aba,+)\,(bba,+)\})$.

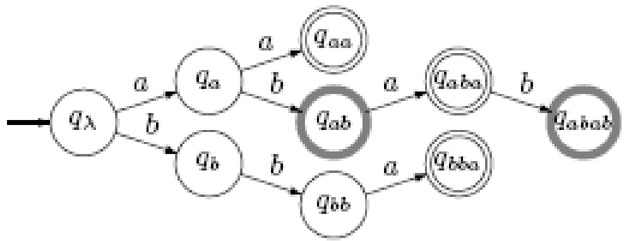


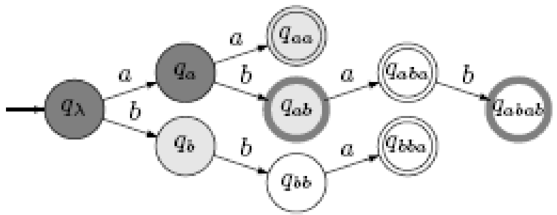
Fig. 8.2. PTA$(\{(aa,+)\,(aba,+)\,(bba,+)\,(ab,-)\,(abab,-)\})$.

Zur Analyse / Angabe von Zustandsverschmelzungsalgorithmen

—*Rote Zustände* (dunkelgrau) sind bereits betrachtet und werden nicht mehr später revidiert.

—*Blaue Zustände* (hellgrau) sind augenblickliche Kandidaten für die Verschmelzung mit roten Zuständen.

—*Weiße Zustände* werden momentan nicht betrachtet.



Zur Analyse / Angabe von Zustandsverschmelzungsalgorithmen

Drei Haupt-Fragen müssen zunächst geklärt werden:

(1) *Kompatibilitätsfrage*: Wann sind zwei Zustände äquivalent und sollten also verschmolzen werden?

Konkret betrachte Paare von roten und blauen Zuständen

(2) *Merge*: Wie ist die Verschmelzung durchzuführen (und wann abzulehnen)?

(3) *Beförderung / Promotion*: Wann kann man von blauen Zuständen sicher sagen, dass man sie zu roten machen kann? Inkompatibilität entdeckt

Kompatibilitätsproblem Betrachte:

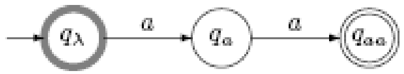


Fig. 8.6. PTA$(\{(aa, +), (\lambda, -)\})$.

Verschmilzt man q_λ und q_a, so werden dadurch keine Wörter akzeptiert, die vorher verworfen wurden und umgekehrt.

Problem: Der neue Automat ist nicht deterministisch, und durch nochmaliges Verschmelzen (mit q_{aa}) entsteht ein Konsistenzproblem: Das leere Wort wird nicht länger verworfen.

$\rightsquigarrow$ Konsistenzprüfungen und Verschmelzungs(tests) müssen "verwoben" entschieden werden.

Verschmelzen

Algorithm 8.2: MERGE

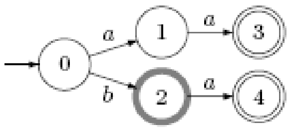
Input: NFA : $\mathcal{A} = \langle \Sigma, Q, \mathbb{I}, \mathbb{F}_{\mathbb{A}}, \mathbb{F}_{\mathbb{R}}, \delta_N \rangle$, 2 states q_1 and q_2

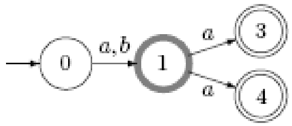
Output: NFA : $\mathcal{A} = \langle \Sigma, Q', q_0, \mathbb{F}_{\mathbb{A}}, \mathbb{F}_{\mathbb{R}}, \delta'_N \rangle$ in which q_1 and q_2 have been merged into q_1

for $q \in Q$ **do**
  for $a \in \Sigma$ **do**
    if $q_2 \in \delta_N(q, a)$ **then**
      $\delta_N(q, a) \leftarrow \delta_N(q, a) \cup \{q_1\}$
    end
    if $q \in \delta_N(q_2, a)$ **then**
      $\delta_N(q_1, a) \leftarrow \delta_N(q_1, a) \cup \{q\}$
    end
  end
end
if $q_2 \in \mathbb{I}$ **then**
  $\mathbb{I} \leftarrow \mathbb{I} \cup \{q_1\}$
end
if $q_2 \in \mathbb{F}_{\mathbb{A}}$ **then**
  $\mathbb{F}_{\mathbb{A}} \leftarrow \mathbb{F}_{\mathbb{A}} \cup \{q_1\}$
end
if $q_2 \in \mathbb{F}_{\mathbb{R}}$ **then**
  $\mathbb{F}_{\mathbb{R}} \leftarrow \mathbb{F}_{\mathbb{R}} \cup \{q_1\}$
end

Problem:

Nichtdeterminismus kann entstehen

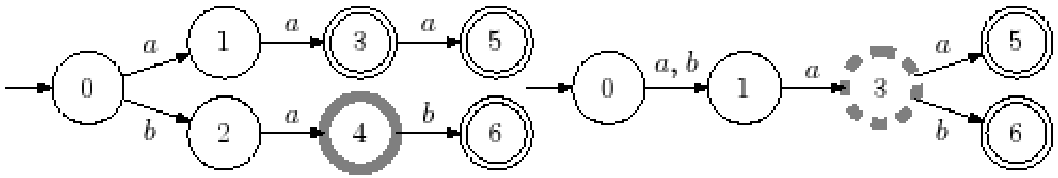




Verschmelzen

Das Auflösung von Nichtdeterminismus-Konflikten durch (weiteres) Zustands-verschmelzen kann zu Inkonsistenzen führen:

Ist im Beispiel (rechts) Zustand 3 akzeptierend oder verwerfend?



Hier wurde zunächst (versuchsweise) Zustand 1 und 2 verschmolzen; dann erst tritt der aufzulösende Nichtdeterminismus-Konflikt auf.

Befördern (Promotion)

Ist q_u von blau nach rot zu befördern, so werden die direkt von q_u aus erreichbaren Nachbarn von weiß nach blau befördert (Kandidaten).

Algorithm 8.3: PROMOTE

Input: DFA : $\mathcal{A} = \langle \Sigma, Q, q_0, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$, a BLUE state q_u, sets RED, BLUE, WHITE

Output: DFA : $\mathcal{A} = \langle \Sigma, Q, q_0, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$, sets RED, BLUE, WHITE

RED $\leftarrow$ RED $\cup \{q_u\}$;

for $a \in \Sigma$: q_{ua} *is* WHITE **do**

| add q_{ua} to BLUE

end

Beobachtungstabelle im Algorithmus von Gold, Terminologie de la Higuera

Ähnlich wie beim MAT-Lernen, nur dass jetzt ausdrücklich “Nichtwissen” aufgeführt werden muss, da Wissen nicht erfragt werden kann.

$O : STA \times EXP \to \{0, 1, *\}$. $*$ bezeichnet sog. *Löcher*.

STA: Zustandsmenge, indiziert durch präfixabgeschlossene Wortmenge.

STA wird in $BLUE$ und RED unterteilt. $BLUE$ enthält genau solche Zustände, die nicht selbst Präfix eines anderen Zustands sind.

EXP: suffixabgeschlossene Menge der *Experimente*

Eine Tabelle heißt *vollständig* gdw. sie enthält keine Löcher.

Schreibweise: $OT[u][e]$ für $O(u, e)$.

Somit bezeichnet $OT[u]$ eine Zeile.

Zeilen $OT[u]$ und $OT[v]$ heißen *offenkundig verschieden* (kurz: OD) gdw. es gibt Experiment $e \in EXP$, sodass $\{OT[u][e], OT[v][e]\} = \{0, 1\}$.

Nicht offenkundig verschiedene Zeilen heißen *konsistent*.

Beobachtungstabelle im Algorithmus von Gold, Terminologie de la Higuera

Ein Beispiel:

	λ	a	b
λ	0	$*$	1
a	$*$	0	0
b	1	1	$*$
aa	0	0	$*$
ab	0	$*$	$*$

Die Tabelle ist nicht vollständig.

Wie hinfort üblich, werden die roten (oben stehenden) Zustände von den blauen durch einen Strich getrennt.

$\mathrm{OT}[a]$ und $\mathrm{OT}[\lambda]$ sind inkonsistent.

$\mathrm{OT}[aa]$ ist sowohl mit $\mathrm{OT}[a]$ als auch mit $\mathrm{OT}[\lambda]$ konsistent.

Eine vollständige Tabelle heißt *geschlossen*, wenn es zu jedem blauen Zustand u einen roten v gibt mit $\mathrm{OT}[u] = \mathrm{OT}[v]$.

	λ	a
λ	0	1
a	1	0
b	1	0
aa	0	0
ab	1	0

	λ	a
λ	0	1
a	1	0
b	0	1
aa	0	1
ab	1	0

Die linke Tabelle ist nicht geschlossen (wegen aa), wohl aber die rechte.

Gold baut einen Automaten (fast wie im MAT-Fall)

Algorithm 8.4: GOLD-BUILDAUTOMATON

Input: A closed and complete observation table $(\textsc{Sta}, \textsc{Exp}, \textsc{Ot})$

Output: A DFA $\mathcal{A}(\langle \textsc{Sta}, \textsc{Exp}, \textsc{Ot} \rangle) = \langle \Sigma, Q, q_\lambda, \mathbb{F}_{\mathbb{A}}, \mathbb{F}_{\mathbb{R}}, \delta \rangle$

$Q \leftarrow \textsc{Red}$;

$\mathbb{F}_{\mathbb{A}} \leftarrow \{q_{we} \in \textsc{Red} : \textsc{Ot}[w][e] = 1\}$;

$\mathbb{F}_{\mathbb{R}} \leftarrow \{q_{we} \in \textsc{Red} : \textsc{Ot}[w][e] = 0\}$;

for $q_w \in Q$ **do**

  for $a \in \Sigma$ **do**

    $\delta(q_w, a) \leftarrow q_u : q_u \in \textsc{Red} \wedge \textsc{Ot}[u] = \textsc{Ot}[wa]$

  end

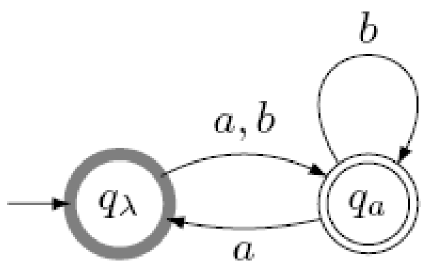
end

Gold baut einen Automaten (fast wie im MAT-Fall)

Ein Beispiel

	λ	a
λ	0	1
a	1	0
b	1	0
aa	0	1
ab	1	0

	a	b
q_λ	q_a	q_a
q_a	q_λ	q_a



Problem: Wir werden in der Regel nur unvollständige Tafeln aus einem vorliegenden Sample bekommen, und zwar mit folgendem Verfahren:

Algorithm 8.5: $\textsc{Gold-BuildTable}$

Input: A sample X, a set $\textsc{Red}$ prefix-closed
Output: A table $\langle \textsc{Red}, \textsc{Exp}, \textsc{Ot} \rangle$
$\textsc{Exp} \leftarrow \textsc{Suff}(X)$;
$\textsc{Blue} \leftarrow \textsc{Red} \cdot \Sigma \setminus \textsc{Red}$;
for $p \in \textsc{Red} \cup \textsc{Blue}$ **do**
  for $e \in \textsc{Exp}$ **do**
    if $p.e \in X_+$ **then**
      $\textsc{Ot}[p][e] \leftarrow 1$
    else
      if $p.e \in X_-$ **then**
        $\textsc{Ot}[p][e] \leftarrow 0$
      else
        $\textsc{Ot}[p][e] \leftarrow *$
      end
    end
  end
end

Ein Beispiel für $X = \{(aa, +), (bbaa, +), (aba, -)\}$
Die Sterne sind der Übersichtlichkeit wegen nicht enthalten.

	λ	a	aa	ba	aba	baa	$bbaa$
λ			1		0		1
a		1		0			
b						1	
aa	1						
ab		0					

Lemma: Gibt es einen blauen Zustand, der offenkundig verschieden von jedem roten ist, so kann die Tabelle nicht geschlossen werden, auf welche Weise auch immer die Löcher gefüllt werden.
Daher befördert der Algorithmus von Gold solche blauen Zustände.

Der allgemeine Algorithmus nach Gold

Die Initialisierung baut Tafel unter der Annahme, nur λ wäre in RED.

Algorithm 8.6: Algorithm GOLD for DFA identification.

Input: a Sample X

Output: a DFA $\mathcal{A}$ consistent with the sample

$\langle \text{RED}, \text{EXP}, \text{OT} \rangle \leftarrow$ GOLD-INITIALISE(X);

while $\exists q_x \in$ BLUE *such that* OT[x] *is* OD **do**

  RED $\leftarrow$ RED $\cup \{q_x\}$;

  BLUE $\leftarrow$ BLUE $\cup \{q_{xa}\}$;

  update OT

end

GOLD-FILLHOLES(OT);

$\mathcal{A} \leftarrow$ GOLD-BUILDAUTOMATON($\langle \text{RED}, \text{EXP}, \text{OT} \rangle$);

if CONSISTENT($\mathcal{A}$,X) **then**

  return $\mathcal{A}$

else

  return PTA(X)

end

Wie füllt Gold die Löcher?

In Golds Originalarbeit (1978) gar nicht!

Gold “rät einfach”, mit welchem roten Zustand ein blauer verschmolzen wird, sollte es Unsicherheiten geben.
Es wird also ein DEA aus einer unvollständigen Beobachtungstabelle abgeleitet.
Durch “falsches Raten” können Inkonsistenzen entstehen.

De la Higuera schlägt in seinem Buch ein konkreteres Verfahren vor.

Example 8.3.5 *We provide an example run of Algorithm* Gold *(8.6).*

Let $X_+ = \{bb, abb, bba, bbb\}$ *and* $X_- = \{a, b, aa, bab\}$.

We first build the observation table corresponding to $\text{Red} = \{q_\lambda\}$.

	λ	a	b	aa	ab	ba	bb	abb	bab	bba	bbb
λ		0	0	0			1	1	0	1	1
a	0	0					1				
b	0		1		0	1	1				

Table 8.2. *The table for* $X_+ = \{bb, abb, bba, bbb\}$ $X_- = \{a, b, aa, bab\}$ *and* $\text{Red} = \{q_\lambda\}$

Now, Table 8.2 is not closed because of row $\textsc{Ot}[b]$. *So we promote* q_b *and update the table obtaining Table 8.3. But Table 8.3 is not closed because of*

	λ	a	b	aa	ab	ba	bb	abb	bab	bba	bbb
λ		0	0	0			1	1	0	1	1
b	0		1		0	1	1				
a	0	0					1				
ba			0								
bb	1	1	1								

Table 8.3. *The table for* $X_+ = \{bb, abb, bba, bbb\}$ $X_- = \{a, b, aa, bab\}$ *and* $\textsc{Red} = \{q_\lambda, q_b\}$.

$\textsc{Ot}[bb]$. *Since* q_{bb} *is obviously different from both* q_λ *(because of experiment* λ*) and* q_b *(because of experiment* b*), we promote* q_{bb} *and update the table to Table 8.4:*

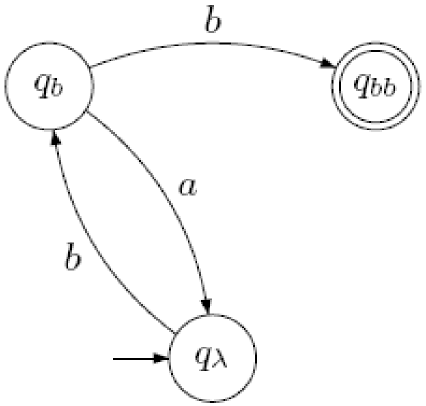
	λ	a	b	aa	ab	ba	bb	abb	bab	bba	bbb
λ		0	0	0			1	1	0	1	1
b	0		1		0	1	1				
bb	1	1	1								
a	0	0					1				
ba			0								
bba	1										
bbb	1										

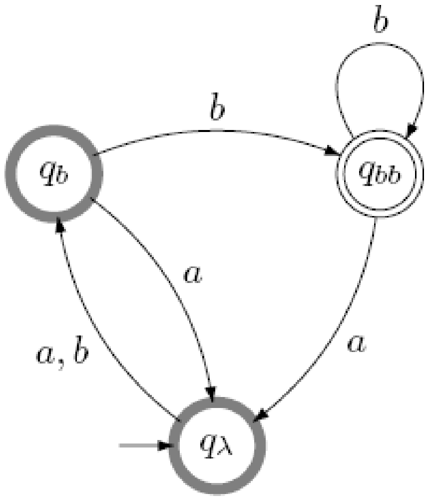
Table 8.4. *The table for* $X_+ = \{bb, abb, bba, bbb\}$ $X_- = \{a, b, aa, bab\}$ *and* $\text{RED} = \{q_\lambda, q_b, q_{bb}\}$.

At this point there are no BLUE rows that are obviously different from the RED rows. This yields a certain number of safe decisions about the automaton. These are depicted in Figure 8.11(a). The others have to be guessed:

- for a equivalent lines could be λ and b (so q_a could be either q_λ or q_b),
- for bba possible candidates are λ and bb (so q_{bba} could be either q_λ or q_{bb}),
- for bbb possible candidates are λ and bb (so q_{bbb} could be either q_λ or q_{bb}).

Raten kann fehlschlagen. Links steht die “sichere Info”.

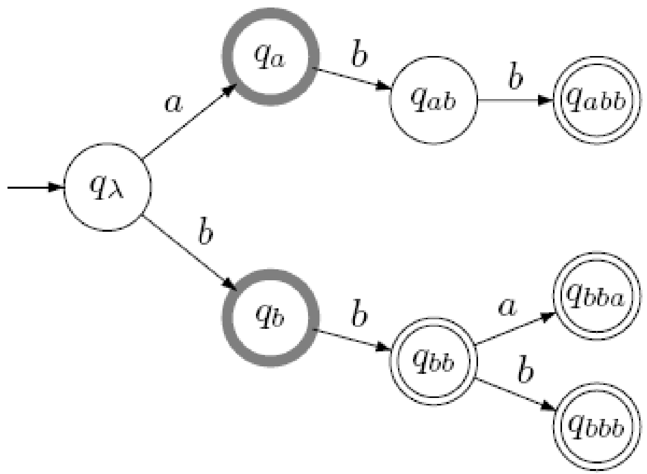


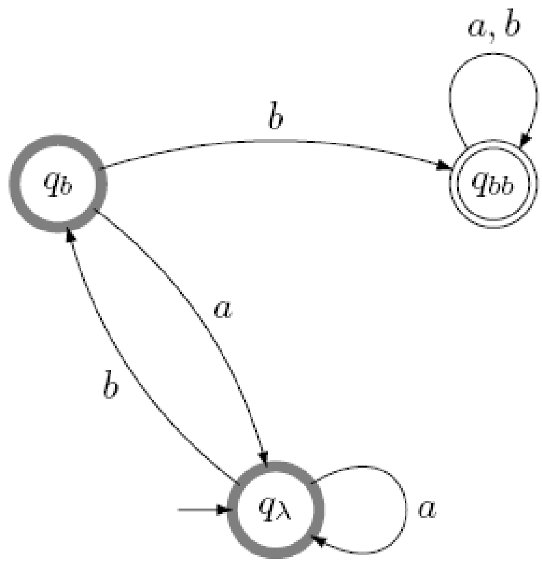


Im Beispiel wird $\mathtt{bba}$ verworfen, was aber erst nach der (z.B. Greedy-) Konstruktion des DEA ersichtlich wird.
Im Falle solch eines falschen Ratens wird einfach der PTA zurückgeliefert.

Raten kann fehlschlagen, muss aber nicht. . .

Hier werden zwei mögliche Ausgaben gezeigt, eine wegen falschen und eine durch richtiges Raten(s).





Satz: Für ein Sample X liefert der Algorithmus von Gold stets einen DEA, der konsistent mit X ist. Im Limes wird jedwede reguläre Sprache identifiziert.

Um letzteres einzusehen, konstruiere man zu vorgelegtem DEA ein Sample, sodass alle Zustände als offenkundig verschieden erkannt werden.
Für einen DEA mit n Zuständen gibt es so ein Sample der Höchstgröße n^3 (mit Zeichenketten der Länge maximal n^2).

RPNI—eine algorithmische Alternative

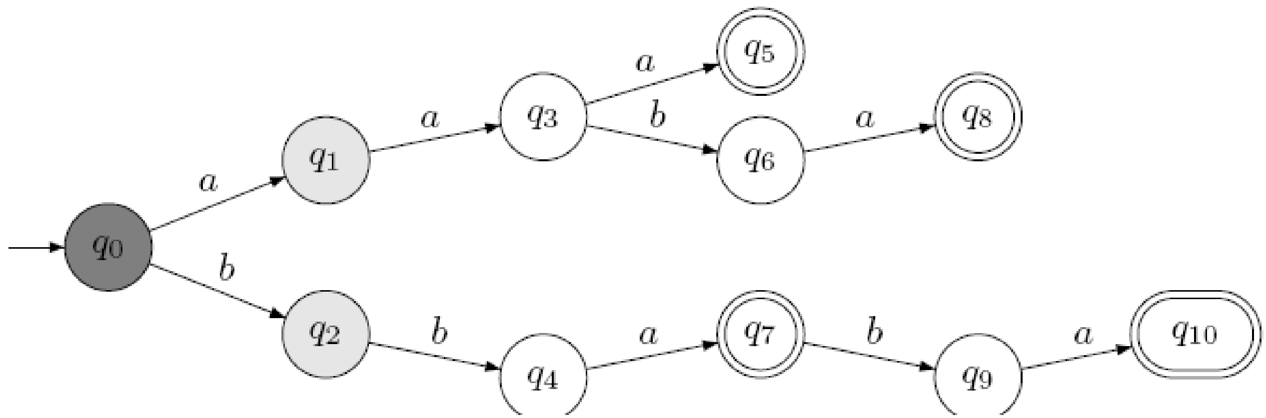
RPNI (regular positive and negative grammatical inference) verschmilzt sehr viel schneller Zustände als der Algorithmus von Gold.
Er startet mit einem Präfixakzeptor zu X_+ und verschmilzt nach und nach Zustände (wenn nicht Inkompatibilitäten dagegen stehen).

Es gibt zahlreiche Varianten, und wir wollen im Folgenden nur eine dieser "bei der Arbeit beobachten".

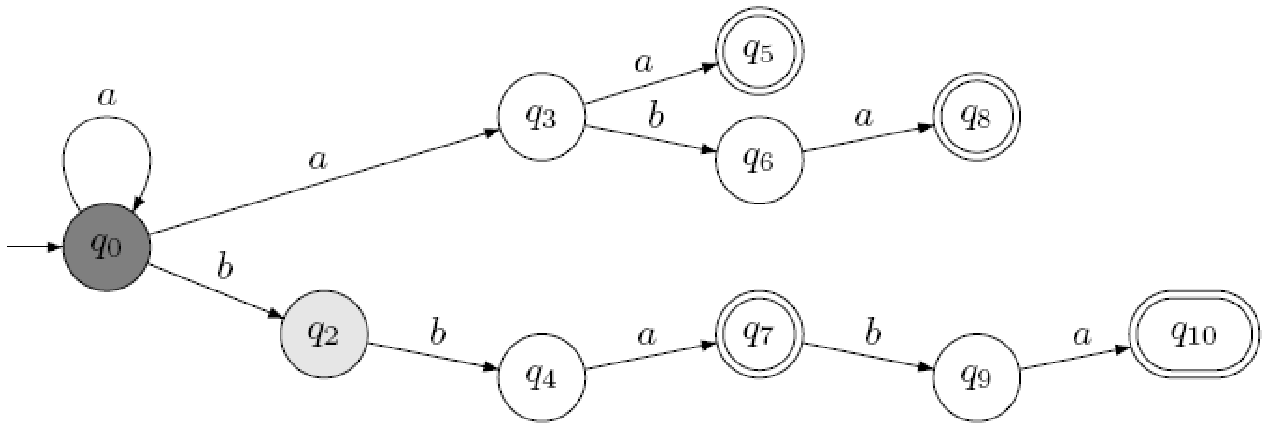
Wir betrachten als Datenmenge:

$X_+ = \{aaa, aaba, bba, bbaba\}$, $X_- = \{a, bb, aab, aba\}$

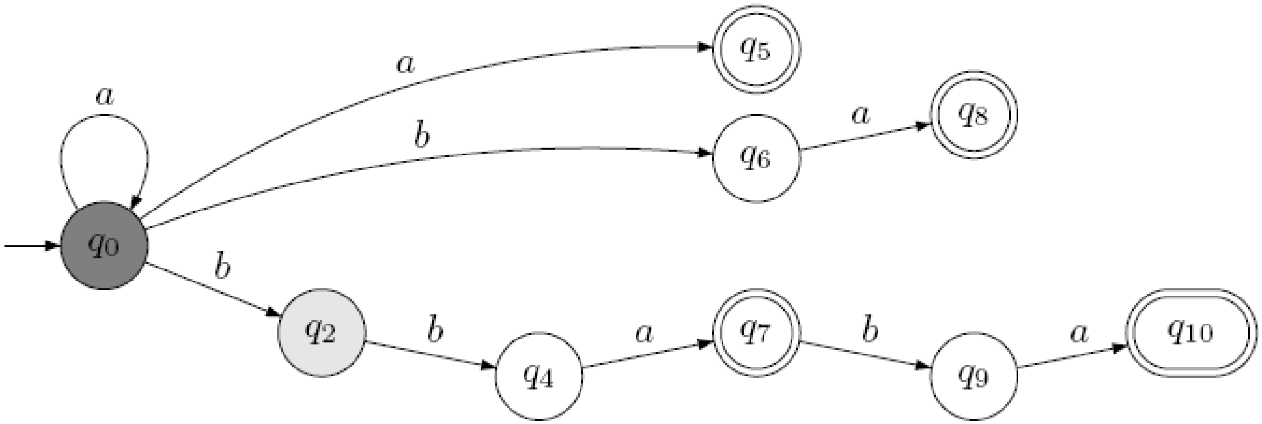
Daraus bauen wir zunächst den PTA zu X_+:



Wir versuchen nun, q_0 und q_1 zu verschmelzen.

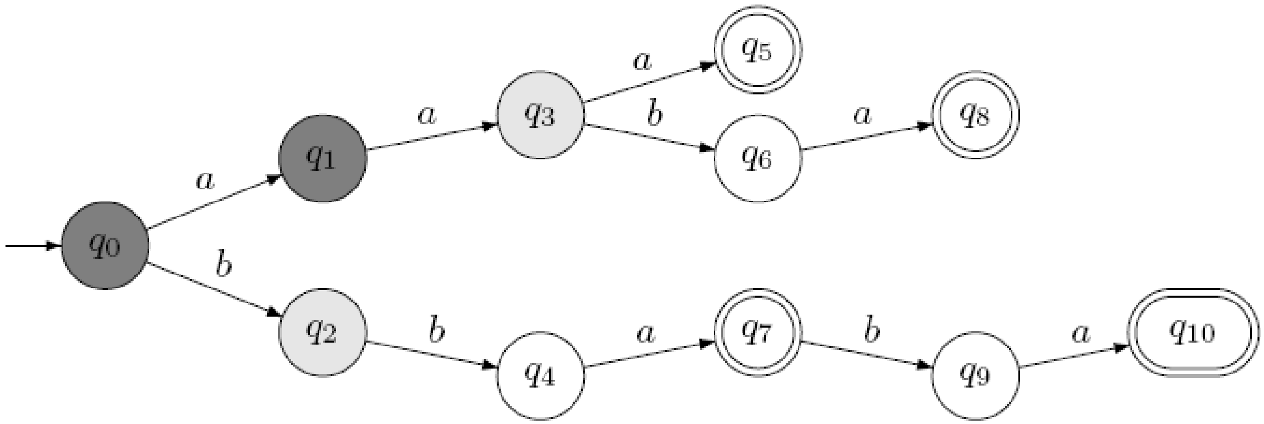


Um Nichtdeterminismus aufzulösen, müssen wir weiter verschmelzen mit q_3.



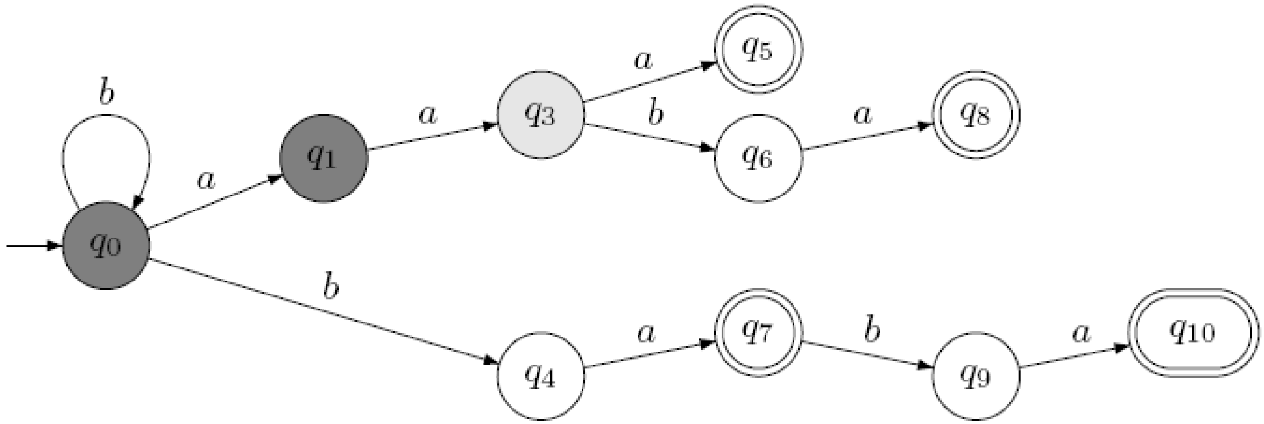
Betrachte die negativen Beispiele: $\rightsquigarrow$ a wird fälschlicherweise akzeptiert.
Daher dürfen wir q_0 und q_1 nicht verschmelzen.
Das hätten wir schon früher merken können: Schon der nichtdeterministische Automat der vorigen Folie hat fälschlicherweise aba akzeptiert.

Jedenfalls kann q_1 jetzt befördert werden, und sein Nachfolger wird somit blau.

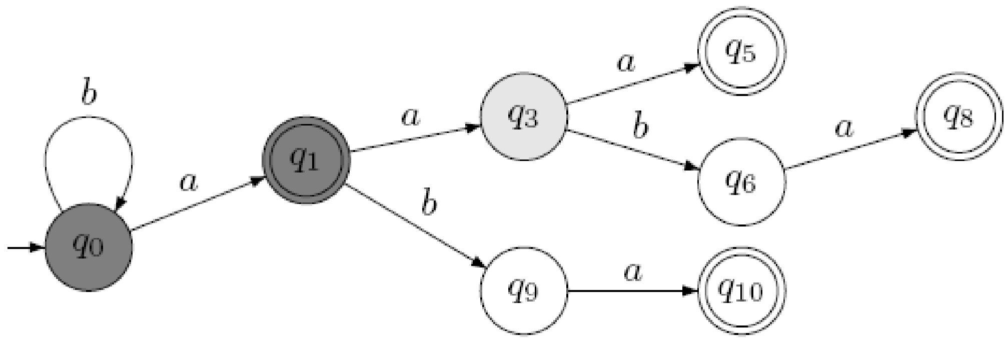


Nächster Versuch: Verschmilz q_0 und q_2:

8.4 RPNI

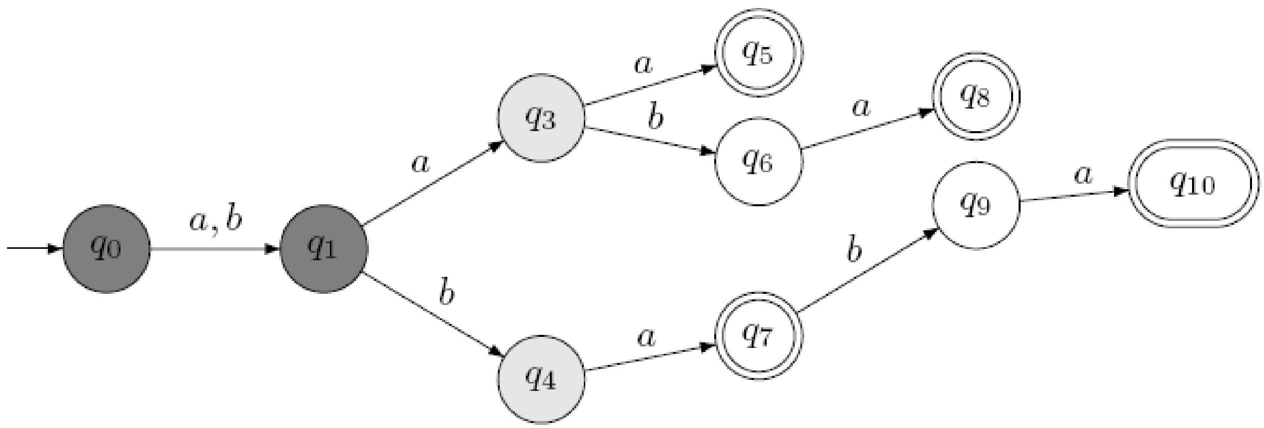


Nächster Versuch: Verschmilz q_0 und q_2:



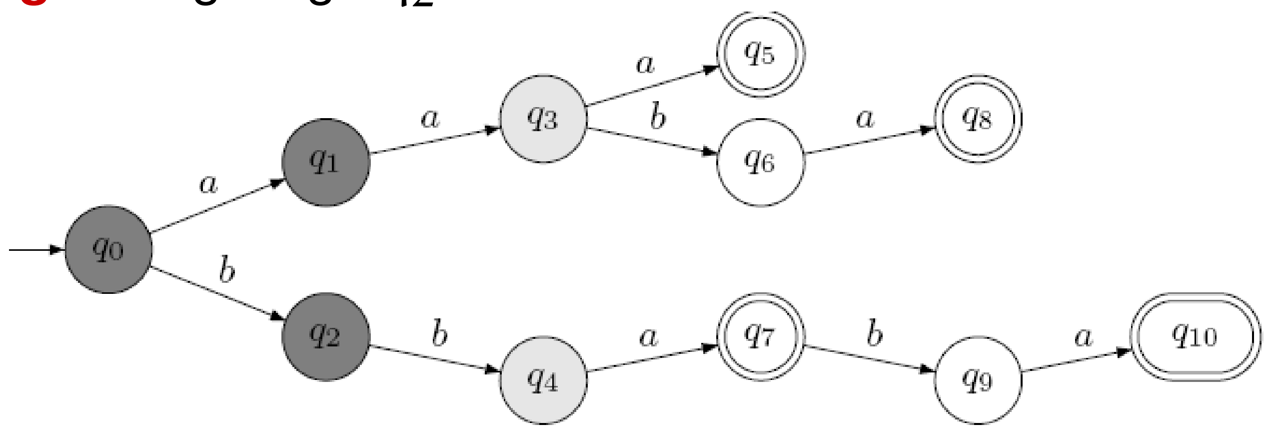
Scheitert ebenso: Der durch Determinisierung entstehende Automat akzeptiert wiederum a.

Nächster Versuch: Verschmilz q_1 und q_2:

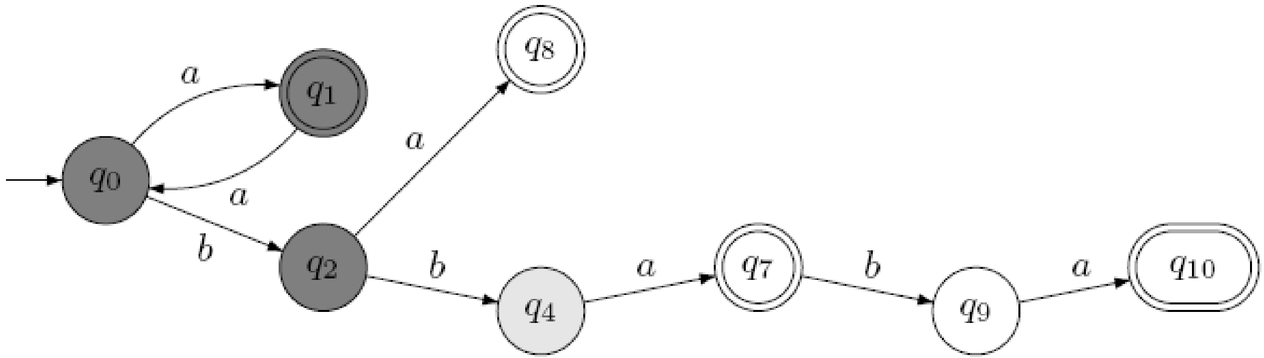


Scheitert ebenso: Der durch Determinisierung entstehende Automat akzeptiert aba.

Beförderung ist angesagt: q_2 ist ebenfalls rot:

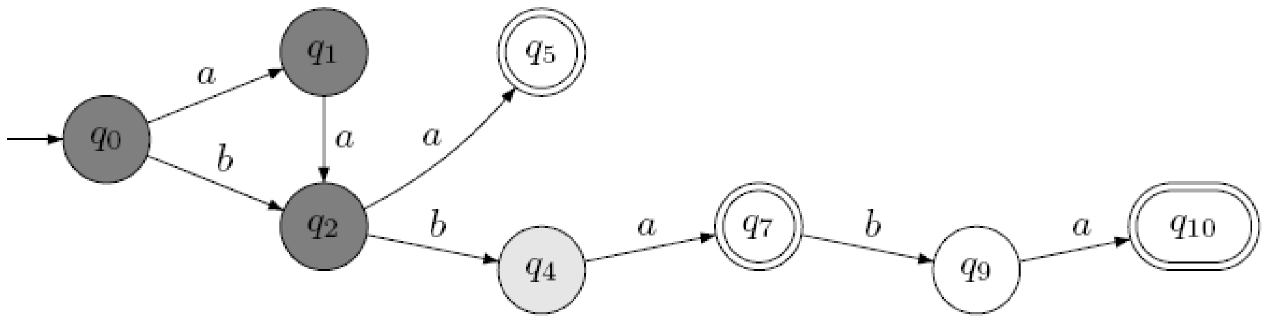


Nächster Versuch: Verschmilz q_0 und q_3:
Scheitert ebenso: Der durch Determinisierung entstehende Automat akzeptiert a.



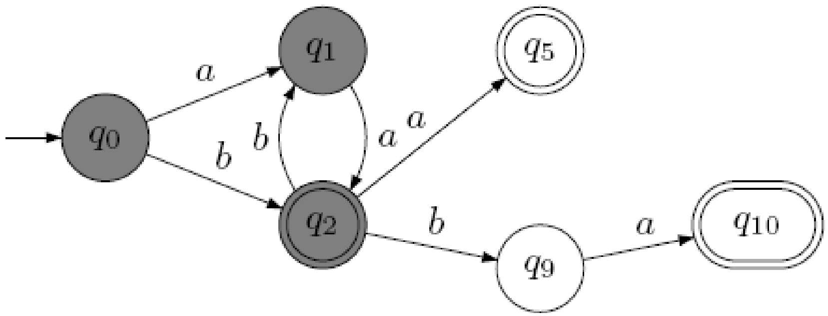
Genauso scheitert (diesmal wegen aba) das Verschmelzen von q_1 und q_3.

Nächster Versuch: Verschmilz q_2 und q_3:



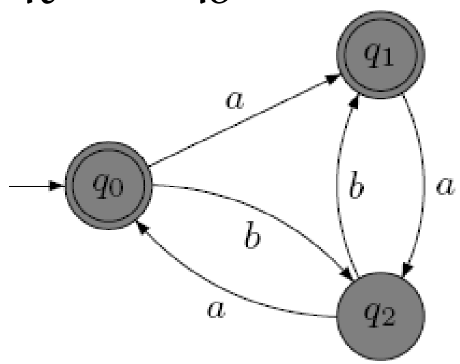
Dieses Verschmelzen verspricht Erfolg.
Mit Blick auf den PTA wird als nächstes q_4 untersucht (nicht q_5).

Übernächster Versuch: Verschmilz q_1 und q_4:



Dieses Verschmelzen verspricht Erfolg.
Mit Blick auf den PTA wird als nächstes q_5 untersucht.

Nächster Versuch: Verschmilz q_0 und q_5:



Wir haben jetzt alle Zustände als rot bestätigt und können den so erhaltenen Automaten “ruhigen Gewissens” als Hypothese äußern.

Vergleich Gold und RPNI

Ausgangspunkt unterschiedliche PTAs.

Unterschiedliches Verhalten, wenn kein blauer Zustand mehr gefunden wird, der offenkundig von allen roten verschieden ist.
1) Gold rät; im Fehlerfall wird PTA ausgegeben.
2) RPNI untersucht mögliche Verschmelzungen. Das kann zu weiteren Beförderungen führen oder auch (durch Backtracking) zum (nachträglichen) Verwerfen solcher Verschmelzungs- und Beförderungsentscheidungen.

Warum Heuristiken?

Golds Algorithmus und auch RPNI enthalten viele heuristische Entscheidungen. Ist das wirklich nötig?
Das “richtige Verfahren” würde doch, ausgehend von $X = (X_+, X_-)$, einen kleinstmöglichen DEA D suchen, der konsistent mit X ist.

Satz: Die Entscheidungsvariante des beschriebenen Konsistenzproblems ist NP-vollständig.

Dennoch konvergieren sowohl der Goldsche Algorithmus als auch RPNI (prinzipiell unabhängig von den Heuristiken).

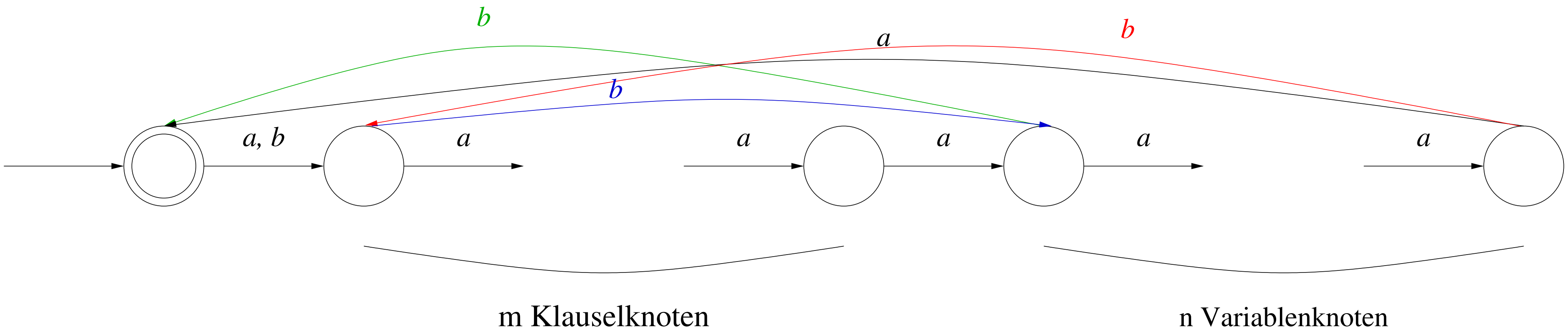
Eine Beweisskizze für die NP-Härte

Betrachte (o.E.; Warum?) SAT-Instanz, bei der Klauseln entweder nur positive oder nur negative Literale enthalten;
also Variablen $\{x_1, \ldots, x_n\}$ und Klauseln $C = C_P \cup C_N$; $C = \{c_1, \ldots, c_m\}$.

Konstruiere Konsistenzinstanz mit Alphabet $\Sigma = \{a, b\}$, Parameter $n + m + 1$.
1. $\lambda, a^{n+m+1} \in X_+$, $b \in X_-$.
2. Für $1 \leq k < n + m + 1$, $a^k \in X_-$, $a^{n+m+k+1} \in X_-$.
3. Für $c_i \in C_P$: $a^i bb \in X_+$, sowie $a^i b a^{n-j+1} \in X_- \iff x_j \notin c_i$.
4. Für $c_i \in C_N$: $a^i bb \in X_-$, sowie $a^i b a^{n-j+1} \in X_- \iff \bar{x}_j \notin c_i$.
5. Für $i, j = 1, \ldots, m$: $a^i b a^{n+j} \in X_+$.

Wegen 1. und 2. hat jeder konsistente DEA wenigstens $n + m + 1$ Zustände $q_a, \ldots, q_{a^{n+m+1}}$.
Außerdem gilt: $q_{a^k} = q_{a^{n+m+k+1}}$ für $k = 0, \ldots, n + m$, d.h. q_λ ist akz.
Zu zeigen: SAT-Instanz erfüllbar gdw. $\exists$ konsistenter DEA mit $n+m+1$ Zuständen

Eine Beweisskizze für die NP-Härte



Blaue $\mathfrak{b}$-Bögen modellieren die Literalauswahl pro Klausel
(Bedingung 5 schließt $\mathfrak{b}$-Bögen zwischen Klauselknoten aus.)
Grüne $\mathfrak{b}$-Bögen zeigen auf 1 gesetzte Variablen an.
Rote $\mathfrak{b}$-Bögen zeigen auf 0 gesetzte Variablen an.

Beweis für die NP-Härte

Sei ϕ eine erfüllende Belegung.
Füge b-Kante von $q_{a^{m+j}}$ nach q_λ ein gdw. $\phi(x_j) = 1$.
Füge b-Kante von $q_{a^{m+j}}$ nach q_b ein gdw. $\phi(x_j) = 0$.
Es werde $c_i \in C$ "wegen" Literal $\ell_j \in \{x_j, \bar{x}_j\}$ erfüllt.
Füge b-Kante von q_{a^i} ein nach $q_{a^{m+j}}$.
$\rightsquigarrow$ $q_\lambda = q_{a^{n+m+1}} = q_{a^{m+j}a^{n-j+1}} = q_{a^iba^{n-j+1}} = q_{a^ibb}$ akz., falls $c_i \in C_P$, d.h.: $\ell_j = x_j$
sowie $q_\lambda = q_{a^{n+m+1}} = q_{a^{m+j}a^{n-j+1}} = q_{a^iba^{n-j+1}}$ akz., aber $a^ibb \in X_-$, denn $q_{a^ibb} = q_b$ verwirft, falls $c_i \in C_N$, d.h.: $\ell_j = \bar{x}_j$
$\rightsquigarrow$ Es gibt konsistenten DEA mit $n + m + 1$ Zuständen.

Ist A umgekehrt konsistenter DEA mit $n + m + 1$ Zuständen, so lies Belegung $\phi(x_j)$ von den b-Ausgängen von $q_{a^{m+j}}$ ab:
Wird akz. Zustand erreicht, so setze $\phi(x_j) = 1$, sonst $\phi(x_j) = 0$.
Betrachte Klausel $c_i \in C$:
$c_i \in C_P$ wird durch $x_j \in c_i$ erfüllt, falls $q_{a^ib} = q_{a^{m+j}}$ wegen $a^ibb,\ a^iba^{n-j+1} \notin X_-$.
$c_i \in C_N$ wird durch $\bar{x}_j \in c_i$ erfüllt, falls $q_{a^ib} = q_{a^{m+j}}$ wegen $a^ibb,\ a^iba^{n-j+1} \in X_-$.
Determinismus garantiert, dass keine Widersprüche bei Belegung möglich sind.

Ein letzter Rückblick auf Gold

Das eigentliche Problem beim Goldschen Algorithmus ist das “Löcherfüllen”. Hierzu passt das folgende Entscheidungsproblem:

Gegeben: Sample $X = (X_+, X_-)$ (für Sprache $L \subseteq \{a, b\}^*$) und dazu konsistente Beobachtungstabelle $O : STA \times EXP \to \{0, 1, *\}$ mit offenkundig verschiedenen roten Zuständen $RED \subseteq STA$, sodass $STA = RED \cup RED\{a, b\}$ und RED präfix-abgeschlossen.
Frage: Gibt es einen vollständigen zu X konsistenten DEA mit Zustandsmenge RED, der den aus (RED, X) abgeleiteten unvollständigen aber “sicheren” DEA als Teilautomaten enthält?

Gold konnte mit einer ähnlichen Reduktion wie der obigen zeigen, dass dieses Problem NP-hart ist. (Mitgliedschaft in NP klar.)