Ambiguität in formalen Sprachen I: Reguläre Sprachen

Christian Wurm (Düsseldorf)

Düsseldorf, 17.12.2019

Formale Sprachen sind nicht per se ambig: jede Turingmaschine kann determinisiert werden.

Formale Sprachen sind ambig

1. Im Hinblick auf eine bestimmte Grammatik/Automat
2. Im Hinblick auf alle Grammatiken/Automaten einer bestimmten Klasse.

Wir werden beide Fälle betrachten:

1. Ambiguität von endlichen Automaten, also für reguläre Sprachen
2. Ambiguität von Sprachen im Hinblick auf alle Kontext-freien Grammatiken

Beide Fälle sind sehr relevant!

Reguläre Sprachen

Zur Erinnerung: reguläre Sprachen sind charakterisiert durch

1. Reguläre Ausdrücke (Kleene!)
2. Reguläre Grammatiken
3. Endliche Automaten

Alle drei sind äquivalent. Die Äquivalenz 2-3 ist trivial; 1-3 ist auch nicht allzu kompliziert.

Wir interessieren uns hier hauptsächlich für endliche Automaten.

Reguläre Sprachen

Auch wenn sie in der Linguistik manchmal stiefmütterlich behandelt werden, sind reguläre Sprachen zentral in der Informatik und Informatik:

- Reguläre Sprachen sind die größte relevante Klasse, für die praktisch alle Entscheidungsprobleme positiv ausfallen
- Und reguläre Sprachen haben eigentlich jede nur denkbare Abschlusseigenschaft.
- Reguläre Sprachen haben eine Vielzahl unabhängiger Charakterisierungen, u.a. auch in Logik (MSO).

Wir betrachten die ersten beiden Punkte etwas detaillierter.

Abschlusseigenschaften

Hier eine Liste relevanter Abschlusseigenschaften.

1. Homomorphismus: $L \in Reg \Rightarrow h[L] \in Reg$
2. Inverser Homomorphismus: $L \in Reg \Rightarrow h^{-1}[L] \in Reg$
3. Schnitt, Vereinigung: $L, L' \in Reg \Rightarrow L \cup L', L \cap L' \in Reg$
4. Komplement: $L \in Reg \Rightarrow \Sigma^* - L \in Reg$
5. Konkatenation: $L, L' \in Reg \Rightarrow L \cdot L' \in Reg$
6. Kleene Stern: $L \in Reg \Rightarrow L^* \in Reg$
7. Quotient: $L, L' \in Reg \Rightarrow L/L', L \backslash L' \in Reg$

Entscheidungsprobleme

Hier eine Liste relevanter Entscheidungsprobleme.

1. Gegeben eine Charakterisierung c von $L \in Reg$, ist $L = \emptyset$?
2. Gegeben eine Charakterisierung c von $L \in Reg$, ist L universell $(= \Sigma^*)$?
3. Gegeben c, c' für L, L', ist $L \subseteq L'$?
4. Gegeben c, c' für L, L', ist $L = L'$?

Entscheidungsprobleme folgen aus Abschlusseigenschaften!

Entscheidungsprobleme – Korrespondenzen

1. Leerheit entspricht Konsistenz einer Spezifikation
2. Universalität entspricht Tautologie, sprich eine Spezifikation ist uninformativ
3. Teilmenge entspricht logischer Implikation zweier Spezifikationen
4. Äquivalenz (mengentheoretisch) entspricht Äquivalenz (zweier Spezifikationen)

Entscheidungsprobleme sind also oftmals verknüpft mit reellen Problemen!

Entscheidungskomplexität

Da alle Charakterisierungen von regulären Sprachen übersetzbar sind, sind Entscheidungsprobleme unabhängig von der Charakterisierung.

Ihre Komplexität aber nicht! Z.B.:

- Die Frage nach der Äquivalenz minimaler deterministischer Automaten ist lösbar in linearer Zeit (Isomorphie);
- Die Frage nach der Äquivalenz deterministischer Automaten ist lösbar in linearer Zeit (Minimisierung+Isomorphie);
- Die Frage nach der Äquivalenz regulärer Ausdrücke ist PSPACE-vollständig (das ist mehr als NP)!
- Die Frage nach der Äquivalenz nicht-deterministischer Automaten hat exponentielle Komplexität!

Entscheidungskomplexität

Also: auch für Automaten ist die Komplexität sehr unterschiedlich:

- Die Frage nach der Äquivalenz zweier *nichtdeterministischer* Automaten kann **exponentiell** sein (EXPTIME).

Grund ist der notwendige Schritt der Determinisierung (Potenzmengenkonstruktion – exponentiell!).

Exponentielle Probleme sind praktisch meistens nicht lösbar. Deswegen ist der Begriff der **Ambiguität von Automaten** so wichtig.

Automaten: Läufe

Gegeben sei ein Automat $\mathfrak{A} = (Q, \Sigma, \delta, F, q_0)$, ein $w \in \Sigma^*$. Ein **Lauf** l für w ist eine Folge

$$l = (q^0, a_1, q^1, a_2, ..., a_i, q^i)$$

so dass

1. $w = a_1 ... a_i$
2. $q^0 = q_0$
3. für alle $0 \leq n \leq i - 1$, $q^{n+1} \in \delta((q^n, a_{n+1})$

Das ist die normale Definition; man schreibt auch l_w.

Automaten: Läufe

Ein Lauf l für w in $\mathfrak{A}$

$$l = (q^0, a_1, q^1, a_2, ..., a_i, q^i)$$

ist **akzeptierend** falls

1. $w = a_1 ... a_i$
2. $q^0 = q_0$
3. für alle $0 \leq n \leq i - 1$, $(q^n, a_{n+1}, q^{n+1}) \in \delta$
4. $q^i \in F$

Automaten: Ambiguität

Sei $w = a_1 ... a_i$, $\mathfrak{A}$ ein Automat.

Die **Ambiguität** $amb_w(\mathfrak{A})$ von w in $\mathfrak{A}$ ist die Anzahl der **akzeptierenden Läufe**

$$(q^0, a_1, q^1, a_2, ..., a_i, q^i)$$

von w in $\mathfrak{A}$.

Das ergibt Sinn wenn man einen akzeptierenden Lauf als eine Art "Bedeutung" auffasst.

Automaten: Ambiguität

Die n-**Ambiguität eines Automaten** $amb_{\mathfrak{A}}(n)$ ist die maximale Ambiguität eines Wortes in $w \in L(\mathfrak{A})$ mit Länge $\leq n$. Also:

$$amb_{\mathfrak{A}}(n) = \max_{w}(amb_w(\mathfrak{A}) : w \in L(\mathfrak{A}), |w| \leq n\}$$

Die **Ambiguität eines Automaten** quantifiziert man nach dem Wachstum der (monotonen) Funktion $amb_{\mathfrak{A}}(n)$.

Automaten: Ambiguität

Es gibt verschiedene Klassen von Automaten, geordnet nach Teilmenge:

1. deterministisch
2. unambig: $amb_{\mathfrak{A}}(n) \leq 1$ (UAA)
3. konstant begrenzte Ambiguität: $amb_{\mathfrak{A}}(n) \leq c$, c eine Konstante – "endlich ambig" (KAA)
4. $amb_{\mathfrak{A}}(n) \leq L(n)$, L eine lineare Funktion (LAA)
5. $amb_{\mathfrak{A}}(n) \leq P(n)$, P ein Polynom (PAA)
6. $amb_{\mathfrak{A}}(n) \leq E(n)$, E eine exponentielle Funktion (ExpAA)

Automaten: Ambiguität

Es gibt verschiedene Klassen von Automaten, geordnet nach Teilmenge:

1. deterministisch
2. unambig: $amb_{\mathfrak{A}}(n) \leq 1$ (UAA)
3. konstant begrenzte Ambiguität: $amb_{\mathfrak{A}}(n) \leq c$, c eine Konstante (KAA)

Diese gehören zusammen in die Klasse der endlichen Ambiguität.

Automaten: Ambiguität

4. $amb_{\mathfrak{A}}(n) \leq L(n)$, L eine lineare Funktion. (LAA)
5. $amb_{\mathfrak{A}}(n) \leq P(n)$, P ein Polynom. (PAA)
6. $amb_{\mathfrak{A}}(n) \leq E(n)$, E eine exponentielle Funktion. (ExpAA)

Wir unterscheiden also den endlichen und den unendlichen Fall; im letzteren parametrisieren wir nach Wachstum.

Falls die Ambiguität unbegrenzt ist, wächst sie mindestens linear!

Ambiguität: Beispiele

1. Der minimale nichtdeterministische Automat für $(a+b)^*aaa$ hat keine Ambiguität.
2. Derselbe Automat mit a-Übergang auf dem finalen Zustand hat lineare Ambiguität $(n-2)$
3. Derselbe Automat mit totalen Übergängen hat unendliche Ambiguität, und zwar exponentielle! $(\geq 2^{\frac{n-3}{2}})$
4. (q_0, q_1, q_2), a geht im durch alle Zustände im Kreis und invariant auf q_0; b geht invariant auf q_1, q_2. $F = \{q_0\}$.
   Dieser Automat lässt sich nicht so einfach disambiguieren!

Ambiguität: Beispiele

Aufgabe

- Finden Sie einen Automaten mit polynomialer Ambiguität!

Ambiguität und Nichtdeterminismus

Ambiguität ist nur indirekt verwandt mit Nichtdeterminismus:

- Es gibt Automaten, die sind hochgradig nichtdeterministisch, haben jedoch keine Ambiguität!
- Nichtdeterminismus hängt ab von den Läufen zu einem Wort,
- Ambiguität von den **akzeptierenden** Läufen

Dazu später mehr!

Zustandskomplexität

Beide Begriffe hängen aber ab von der Zustandskomplexität einer Sprache:

Zustandskomplexität einer Sprache

Die Zustandskomplexität einer Sprache L (kurz: $ZK(L)$) ist die Anzahl der Zustände im minimalen DEA $\mathfrak{A}$ so dass $L = L(\mathfrak{A})$.

$ZK(L)$ hat, nach Myhill-Nerode, eine intuitive Bedeutung:

- für $w, v \in \Sigma^*$ wird definiert: $w \sim_L v$ gdw. $wx \in L \Leftrightarrow vx \in L$.
- $ZK(L) = |\Sigma^* / \sim_L|$, wobei $\Sigma^* / \sim_L$ die Menge der $\sim_L$ Äquivalenzklassen ist.

Nichtdeterministische Zustandskomplexität

Nichtdeterministische Zustandskomplexität einer Sprache

Die nxichtdeterministische Zustandskomplexität einer Sprache L (kurz: $NZK(L)$) ist die Anzahl der Zustände im minimalen NEA $\mathfrak{A}$ so dass $L = L(\mathfrak{A})$.

$NZK(L)$ hat nach meinem Wissen *keine* intuitive Bedeutung; außerdem ist es sehr hart zu errechnen (brachial ausprobieren).

Nichtdeterministische Automaten sind also durchaus mysteriöse Wesen!

Wichtige Ergebnisse

Deswegen sind die folgenden Ergebnisse so interessant:

Theorem (Weber and Seidl)

Es ist in polynomieller Zeit entscheidbar, ob ein NEA in KAA oder PAA oder ExpAA ist.

Konstante Ambiguität ist natürlich beschränkt durch die Anzahl der Zustände; kann aber sehr groß werden:

Theorem (Weber and Seidl)

Die Ambiguität eines KAA mit n Zuständen ist begrenzt durch $5^{\frac{n}{2}} \cdot n^n$.

Wichtige Ergebnisse

Theorem (Leung)

Für jedes $n \in \mathbb{N}$ existiert ein UAA mit n Zuständen so dass der minimale äquivalente DEA 2^{n-1} Zustände hat.

Theorem (Leung)

Für jedes $n \in \mathbb{N}$ existiert ein KAA mit n Zuständen so dass jeder äquivalente UAA mindestens 2^{n-1} Zustände hat.

Ambiguität zu reduzieren kann also sehr teuer sein!

NB: wir erlauben undefinierte Übergänge, also partielle Funktion δ! Sonst ändern sich die Zahlen.

Wichtige Ergebnisse

Theorem

Für $\mathfrak{A}_1, \mathfrak{A}_2 \in KAA$ kann in polynomieller Zeit bestimmt werden, ob $L(\mathfrak{A}_1) = L(\mathfrak{A}_2)$.

Für $\mathfrak{A}_1, \mathfrak{A}_2 \in KAA$ kann in polynomieller Zeit bestimmt werden, ob $L(\mathfrak{A}_1) \subseteq L(\mathfrak{A}_2)$.

Eine begrenzte Ambiguität macht also gewisse Entscheidungsprobleme erst handhabbar.

Nichtdeterminismus

Es gibt drei Maße für Nichtdeterminismus in einem Automaten:

1. Anzahl nichtdeterministischer Schritte im besten Lauf (über ein Wort)
2. Anzahl nichtdeterministischer Schritte im schlechtesten Lauf (über ein Wort)
3. Anzahl der Läufe (für ein Wort), akzeptierend oder nicht

Raten (wie in erraten)

Raten (wie in erraten) ist ein Mass dafür, wieviel wir richtig raten müssten um am Ende richtig zu liegen. Die Definition ist wie folgt:

Sei l ein Lauf für w, m die Länge von l, n die Länge von w, wobei $m \leq n$ (wegen undefinierten Übergängen)

Das **Raten** $\gamma(l)$ in $\mathfrak{A}$, $w = a_1 ... a_n$, ist definiert als

$$\gamma_{\mathfrak{A}}(l) = log_2(|\delta(q_0, a_1)|) + \sum_{i=1}^{m-1} log_2|\delta(q_i, a_{i+1})| \tag{1}$$

Der Logarithmus sagt uns, wieviel Information (Bits) wir brauchen, um den richtigen Lauf zu erraten. $log(1) = 0$!

Verzweigung

Die Verzweigung eines Übergangs ist definiert als:

$$|\delta(q, a)|.$$

Offensichtlich hat das Raten mit der Verzweigung zu tun; die Verzweigung ist das Produkt der Einzelverzweigungen, also:

$$\beta_{\mathfrak{A}}(I) = 2^{\gamma_{\mathfrak{A}}(I)} \tag{2}$$

Raten

Wir haben nun auf Läufen geraten. Als nächstes raten wir auf Worten;

Das **Raten** eines Automaten auf einem Wort $w = a_1 ... a_n$, ist definiert einmal **maximal**, einmal **minimal**: (m ist die Länge von l, $m \leq n$ wegen undefinierten Übergängen)

$$\gamma_{\mathfrak{A}}^{max}(w) = \max_{l \text{ erkennt } w} \gamma_{\mathfrak{A}}(l) \tag{3}$$

$$\gamma_{\mathfrak{A}}^{min}(w) = \min_{l \text{ erkennt } w} \gamma_{\mathfrak{A}}(l) \tag{4}$$

Baumweite

Die Baumweite (wegen Berechnungsbäumen) von $\mathfrak{A}$ für w ist wie folgt definiert:

$$bw_{\mathfrak{A}}(w) = |\{l : l \text{ erkennt } w \text{ in } \mathfrak{A}\}| \tag{5}$$

Parametrisierung

γ und *bw* sind bislang nur für Worte definiert. Wie üblich erweitern wir das Konzept auf Automaten:

- $\gamma(\mathfrak{A})$ ist konstant, falls $max_w\{\gamma_w(\mathfrak{A})\} \leq k$
- $\gamma(\mathfrak{A})$ ist linear, falls $max_{|w|\leq n}\{\gamma_w(\mathfrak{A})\} \leq l(n)$, l eine lineare Funktion
- $\gamma(\mathfrak{A})$ ist polynomiell, falls $max_{|w|\leq n}\{\gamma_w(\mathfrak{A})\} \leq P(n)$, P ein Polynom
- $\gamma(\mathfrak{A})$ ist exponentiell, falls $max_{|w|\leq n}\{\gamma_w(\mathfrak{A})\} \leq E(n)$, E eine exponentielle Funktion

Parametrisierung

γ und bw sind bislang nur für Worte definiert. Wie üblich erweitern wir das Konzept auf Automaten:

- $bw(\mathfrak{A})$ ist konstant, falls $max_w\{bw_w(\mathfrak{A})\} \leq k$
- $bw(\mathfrak{A})$ ist linear, falls $max_{|w|\leq n}\{bw_w(\mathfrak{A})\} \leq l(n)$, l eine lineare Funktion
- $bw(\mathfrak{A})$ ist polynomiell, falls $max_{|w|\leq n}\{bw_w(\mathfrak{A})\} \leq P(n)$, P ein Polynom
- $bw(\mathfrak{A})$ ist exponentiell, falls $max_{|w|\leq n}\{bw_w(\mathfrak{A})\} \leq E(n)$, E eine exponentielle Funktion

Ergebnisse

Theorem

$bw(\mathfrak{A})$ ist beschränkt entweder durch k, zwischen linear und polynomiell, oder beschränkt durch 2^m.

Natürlich gibt es eine exponentielle Relation zwischen $\gamma(\mathfrak{A})$ und $bw(\mathfrak{A})$:

$$2^{\gamma(\mathfrak{A})} \propto bw(\mathfrak{A}) \tag{6}$$

Aber wegen dem Unterschied min/max gibt es keine eindeutige Relation!

Ambiguität – Nichtdeterminismus

Theorem

Nimm an, $\mathfrak{A}$ ist ein minimaler *NEA* (das ist ein sehr schwieriges Konzept!) Für alle $m \in \mathbb{N}$ gilt:

$$max(\gamma_{\mathfrak{A}}^{max}(m),\ amb_{\mathfrak{A}}(m)) \leq bw(m) = O(amb_{\mathfrak{A}}(m) \cdot \gamma_{\mathfrak{A}}^{max}(m))$$

Das ist natürlich ein Hammer; es gibt meines Wissens keinen Algorithmus, der den minimalen NEA berechnet!

Reguläre Ausdrücke

Ambiguität ist auch für reguläre Ausdrücke ein Problem; wir definieren unambige Ausdrücke induktiv.

Unambige reguläre Ausdrücke

- a ist unambig.
- Falls e_1, e_2 unambig sind, $\|e_1\| \cap \|e_2\| = \emptyset$, dann ist $e_1 + e_2$ unambig.
- Falls e_1, e_2 unambig sind, und falls $wv = w'v' \in \|e_1 \cdot e_2\|$, $w \in \|e_1\|$, $w' \in \|e_1\|$, impliziert $w = w'$ – dann ist $e_1 \cdot e_2$ unambig.
- Falls e unambig ist, und für jedes $w \in \|e^*\|$ gibt es ein n so dass $w = v^n$, $v \in \|e\|$, dann ist e^* unambig.

Reguläre Ausdrücke

Da aber reguläre Ausdrücke allgemein weniger wichtig sind als Automaten (für computationelle Zwecke), spielt das Konzept keine so große Rolle (aber in XML z.B. schon)

Literatur

- Weber, A., Seidl, H. On the degree of ambiguity of finite automata.
- Leung, H. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata.
- Yo-Sub Hana, Arto Salomaa, Kai Salomaa: Ambiguity, Nondeterminism and State Complexityof Finite Automata