Ausdrucksstärke von monadischem Datalog auf Bäumen

Frank Radmacher

RWTH Aachen

Seminar über Automaten für XML (WS 2005/06)

06. Dezember 2005

Druckversion

Worum es geht . . .

- Aufgabe: Selektieren von Knoten in Bäumen
- Anfragesprache: Monadisches Datalog
- Ausdrucksstärke von monadischem Datalog auf Bäumen

Gliederung

1. Einleitung
2. Monadisches Datalog auf Bäumen
3. Ausdrucksstärke
4. Zusammenfassung

Literatur

Georg Gottlob and Christoph Koch.
Monadic datalog and the expressive power of languages for web information extraction.
JACM: Journal of the ACM, 51, 2004.

Frank Neven and Thomas Schwentick.
Query automata over finite trees.
TCS: Theoretical Computer Science, 275, 2002.

Motivation

- Anfragemechanismen mit der Ausdrucksstärke von MSO-Anfragen:
  - Anfrageautomaten [Neven, Schwentick]
  - Boolesche Attributgrammatiken [Neven, van den Bussche]
- Nachteile:
  - Keine „Programmiersprachen“ für Anfragen
  - Modellierung des gesamten Dokuments (Baums) notwendig

➥ Monadisches Datalog als Anfragesprache

- Arbeitet auf Relationen
- Effiziente Anfrageberechnung
- Core XPath kann effizient auf monadisches Datalog abgebildet werden

Bäume

Annahme: Ein Baum t ist gegeben als relationale Struktur:

- t beschränkt (Baum t_{rk} über der Signatur τ_{rk}):

$$t_{\text{rk}} = \langle \text{dom}, \text{root}, \text{leaf}, (\text{child}_k)_{k \leq K}, (\text{label}_a)_{a \in \Sigma} \rangle$$

- t unbeschränkt (Baum t_{ur} über der Signatur τ_{ur}):

$$t_{\text{rk}} = \langle \text{dom}, \text{root}, \text{leaf}, \text{firstchild}, \text{nextsibling}, \text{lastsibling}, (\text{label}_a)_{a \in \Sigma} \rangle$$

Monadisches Datalog - Syntax (1)

Datalog-Ausdrücke

- Monadisches Datalog arbeitet auf Prädikaten (Relationen)
- Atomare Bausteine sind Datalog-Ausdrücke
- n-stelliger Datalog-Ausdruck: $p(a_1, \ldots, a_n)$
  - p: Prädikat
  - a_i: Variablen und Konstanten über *dom*
    (d. h. Wertebereich ist die Menge aller Baumknoten)
- Datalog-Fakt: variablenfreier Datalog-Ausdruck
- Unterscheide zwei Arten von Prädikaten
  (extensionale und intensionale Prädikate)
- Extensionale Prädikate: Prädikate über der Signatur τ_{rk} bzw. τ_{ur} bereits interpretiert durch gegebenen Baum t

Monadisches Datalog - Syntax (2)

Monadische Datalog-Programme

- Monadisches Datalog-Programm $\mathcal{P}$:
  - Menge von Datalog-Regeln der Form

$$h(x) \leftarrow b_1, \ldots, b_n.$$

    1. h ist kein extensionales Prädikat
    2. b_i ist entweder ein extensionales Prädikat oder es gibt eine Regel der Form $b_i \leftarrow \ldots \in \mathcal{P}$.
    3. Für jede Regel mit Kopf $h(x)$ tritt die Variable x auch im Rumpf in einem b_i auf

  - Regel mit Kopf $h(x)$ definiert ein zusätzliches einstelliges Prädikat (intensionales Prädikat)
  - Einstellige Prädikate entsprechen Markierungen von Baumknoten

Monadisches Datalog - Syntax (3)

Anfragen

- Auszeichung eines intensionalen Prädikats q als Anfrageprädikat:
  - Interpretation von $q(x)$ ist die durch $\mathcal{P}$ bestimmte Auswahl von Baumknoten

Beispiel

Selektiere in einem binären Baum alle Knoten, die...

- mindestens einen mit a beschrifteten Knoten in ihrem linken Teilbaum haben und
- mindestens einen mit b beschrifteten Knoten in ihrem rechten Teilbaum haben.

Fixpunktsemantik (1)

- Gegeben: monadisches Datalog-Programm $\mathcal{P}$ mit intensionalen Prädikaten $p_1, \ldots, p_n$
- $T^i_{p_j}$: Menge der Baumknoten, die nach der i-ten Ausführung von $\mathcal{P}$ mit p_j markiert sind
- $T^0_{p_1}, \ldots, T^0_{p_n} := \emptyset$
- $\mathcal{T}^i_{\mathcal{P}} := \{p_j(v) \mid p_j \text{ intensionales Präd.}, v \in T^i_{p_j}\}$
  $\cup \{p(a_1, \ldots, a_n) \in t_{\text{rk/ur}}\}$
- $T^{i+1}_{p_j} := \mathit{trans}^i_{p_j}(\mathcal{T}^i_{\mathcal{P}})$

Fixpunktsemantik (2)

- Auswertung: Funktion ϕ: Belegung der Variablen
  erweitert $\varphi(a) = a$ für alle Konstanten a
  Auf Datalog-Ausdrücken:
  $\phi(p(a_1, \ldots, a_m)) := p(\phi(a_1), \ldots, \phi(a_m))$.
- Transformationen: $\mathit{trans}^i_{p_j} : 2^{\text{Prim}} \to 2^{\text{dom}}$

$$\mathit{trans}^i_{p_j}(X) = T^i_{p_j} \cup \{\phi(x) \in \text{dom}_t \mid \text{es gibt eine Regel } p_j(x) \leftarrow b_1, \ldots, b_n. \in \mathcal{P}, \text{ so dass } \phi(b_1), \ldots, \phi(b_n) \in X\}$$

- Folgen $T^0_{p_j}, T^1_{p_j}, T^2_{p_j}, \ldots$ Fixpunkte $T^n_{p_j} = T^{n+1}_{p_j} =: T^\omega_{p_j}$
- Folge $\mathcal{T}^0_\mathcal{P}, \mathcal{T}^1_\mathcal{P}, \mathcal{T}^2_\mathcal{P}, \ldots$ Fixpunkt $\mathcal{T}^n_\mathcal{P} = \mathcal{T}^{n+1}_\mathcal{P} =: \mathcal{T}^\omega_\mathcal{P}$
- Es existiert stets kleinster Fixpunkt $\mathcal{T}^\omega_\mathcal{P}$

Zusammenfassung: Monadisches Datalog

- Monadisches Datalog: Praktische Anfragesprache über Bäumen
- Fixpunkt $\mathcal{T}_{\mathcal{P}}^{\omega}$ existiert und lässt sich effizient berechnen
- Auswertung möglich in $O(|\mathcal{P}| \cdot |\text{dom}|)$
- Ausdrucksstärke von MSO-definierbaren Anfragen

Satz (Monadisches Datalog $\iff$ MSO)

Eine einstellige Anfrage über Bäumen ist MSO-definierbar gdw. ein monadisches Datalog-Programm über τ_{rk} bzw. τ_{ur} existiert, das die selbe Anfrage definiert.

Monadisches Datalog $\Rightarrow$ MSO

Satz (Monadisches Datalog $\Rightarrow$ Π_1-MSO)

Jede monadische Datalog Anfrage über Bäumen ist Π_1-MSO-definierbar.

Beweis.

Gegeben: monadisches Datalog-Programm $\mathcal{P}$, intensionale Präd. $P_1, \ldots, P_n$, Anfrageprädikat P_1.

Konstruiere Π_1-MSO-Formel

$$\varphi(x) := \forall P_1 \ldots \forall P_n \, (\, \mathsf{SAT}(P_1, \ldots, P_n) \rightarrow x \in P_1 \,)$$

$\mathsf{SAT}(P_1, \ldots, P_n)$: Konjunktion aller Formeln, die Regeln aus $\mathcal{P}$ entsprechen.

$h \leftarrow b_1, \ldots, b_m.$ wird zu $\forall z_1 \ldots \forall z_k \, (b_1 \wedge \ldots \wedge b_m \rightarrow h)$ mit FO-Variablen $z_1, \ldots, z_k$. $\square$

MSO $\Rightarrow$ Monadisches Datalog

- Wir wissen bereits:

Satz

Eine einstellige Anfrage über Bäumen ist MSO-definierbar genau dann, wenn ein QA^r bzw. SQA^u existiert, der diese Anfrage berechnet.

- Konstruiere zu einem Anfrageautomaten ein monadisches Datalog-Programm, das die gleiche Anfrage berechnet
- Hier: Nur Beweis für beschränkte Bäume

Anfrageautomaten $\Rightarrow$ Monadisches Datalog

Beschränkte Anfrageautomaten

- Erinnerung: Ein beschränkter Anfrageautomat (QA^r) (engl. ranked query automaton) ist ein Tupel

$$\mathcal{A} = \langle Q, \Sigma, F, s, \delta_{\uparrow}, \delta_{\downarrow}, \delta_{\text{root}}, \delta_{\text{leaf}}, \lambda \rangle$$

- Anfrageautomaten arbeiten deterministisch

Satz (QA^r $\Rightarrow$ Monadisches Datalog)

Zu jedem QA^r $\mathcal{A}$ existiert ein monadisches Datalog-Programm $\mathcal{P}$, das eine zu $\mathcal{A}$ äquivalente Anfrage definiert.

$QA^r \Rightarrow$ Monadisches Datalog

Idee

- Modellierung von Konfigurationen zu aufwändig
- Lauf deterministisch: Codierung der Zustandszuweisungen im Lauf von $\mathcal{A}$
- Zustandszuweisung: Paar (q, v) (im Lauf von $\mathcal{A}$ besuchter Knoten v bekommt Zustand q zugewiesen)
- Menge dieser Zustandszuweisungen („History“):

$$H = \{(q, v) \mid v \in C_i \text{ und } c_i(v) = q \text{ für ein } i\},$$

- Charakterisierung eines deterministischen Laufs in $\mathcal{A}$
- Wenn Zustandszuweisung (q, v) im Lauf auftritt, soll Prädikat $q(v)$ beweisbar sein

$QA^r \Rightarrow$ Monadisches Datalog

Simulation Übersicht

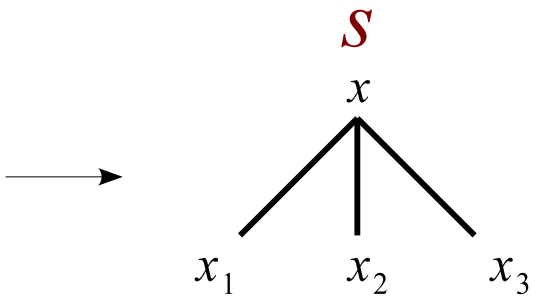
1. Anfangszustand
2. Aufwärtstransition
3. Abwärtstransition
4. Wurzeltransition
5. Blatttransition
6. Akzeptanzbedingung
7. Auswahlfunktion

$QA^r \Rightarrow$ Monadisches Datalog

1. Anfangszustand

Wenn s der Anfangszustand von $\mathcal{A}$ ist, enthält $\mathcal{P}$ die Regel

$$s(x) \leftarrow \mathsf{root}(x).$$

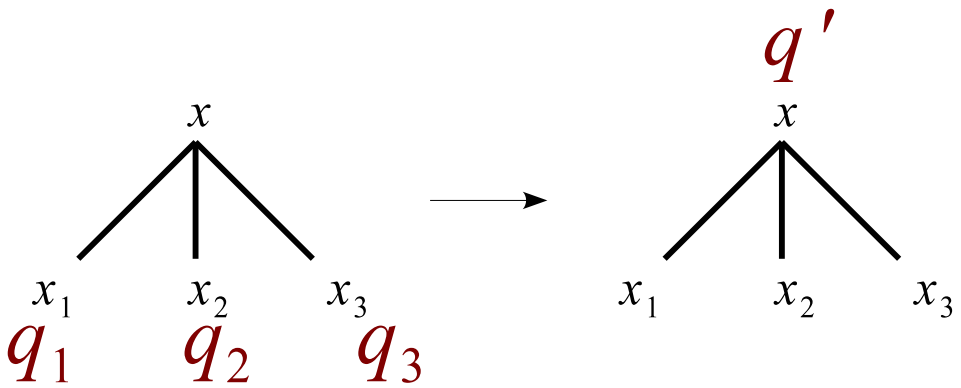


$QA^r \Rightarrow$ Monadisches Datalog

2. Aufwärtstransition

Wenn $\delta_\uparrow(\langle q_1, a_1\rangle, \ldots, \langle q_n, a_n\rangle) = q'$, enthält $\mathcal{P}$ die Regel

$$\begin{aligned} q'(x) \leftarrow\ & \mathsf{child}_1(x, x_1), \ldots, \mathsf{child}_n(x, x_n), \\ & q_1(x_1), \ldots, q_n(x_n), \\ & \mathsf{label}_{a_1}(x_1), \ldots, \mathsf{label}_{a_n}(x_n). \end{aligned}$$

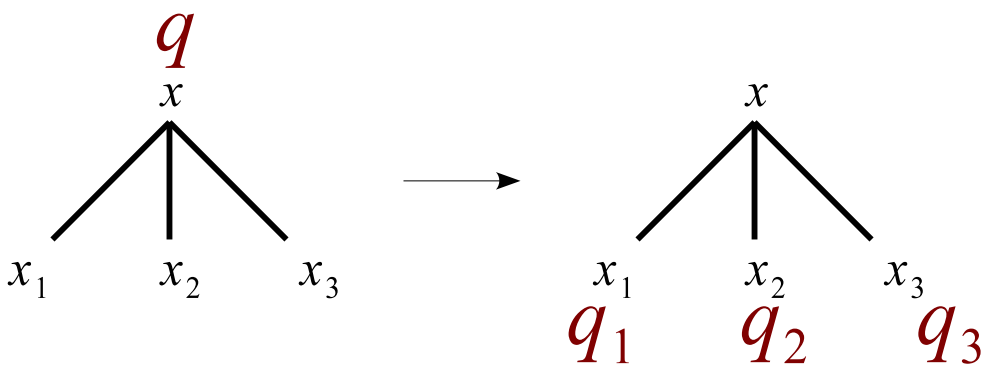


$QA^r \Rightarrow$ Monadisches Datalog

3. Abwärtstransition

Wenn $\delta_\downarrow(q, a, n) = q_1 \dots q_n$, enthält $\mathcal{P}$ die Regeln

$$q_i(x_i) \leftarrow q(x), \mathsf{child}_i(x, x_i), \mathsf{label}_a(x).$$

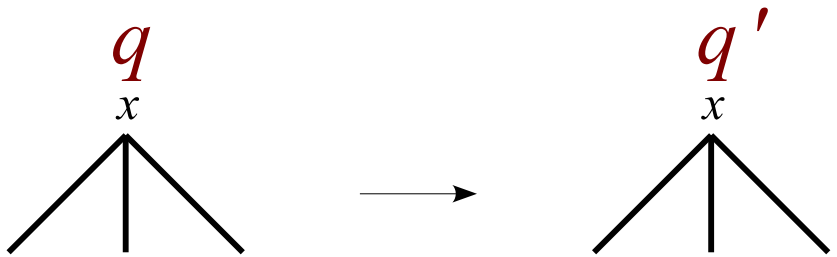


$QA^r \Rightarrow$ Monadisches Datalog

4. Wurzeltransition

Wenn $\delta_{\mathsf{root}}(q, a) = q'$, enthält $\mathcal{P}$ die Regel

$$q'(x) \leftarrow q(x), \mathsf{label}_a(x), \mathsf{root}(x).$$

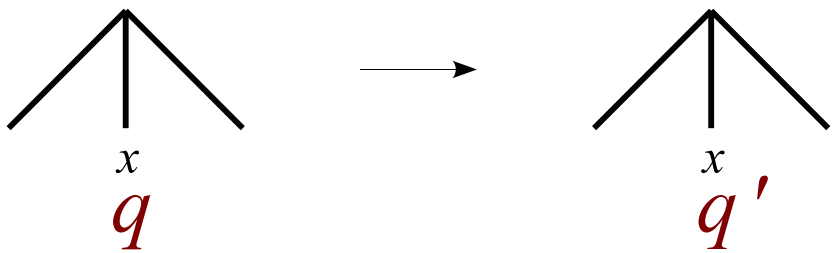


$QA^r \Rightarrow$ Monadisches Datalog

5. Blatttransition

Wenn $\delta_{\mathsf{leaf}}(q, a) = q'$, enthält $\mathcal{P}$ die Regel

$$q'(x) \leftarrow q(x), \mathsf{label}_a(x), \mathsf{leaf}(x).$$



$QA^r \Rightarrow$ Monadisches Datalog

Akzeptanz und Anfragemarkierungen

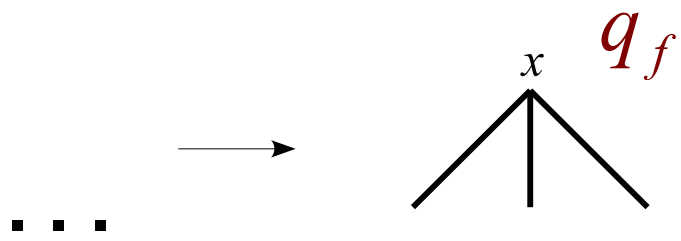
- Bisher:
  - Simulation von Transitionen
  - Markierungen für auftretende Zustandszuweisungen
- Zu tun:
  - Anfragemarkierungen query(x) entsprechend Auswahlfunktion $\lambda : Q \times \Sigma \to \{\bot, \top\}$.
  - Dazu notwendig: Prädikat accept für die Akzeptanz des Automaten

$QA^r \Rightarrow$ Monadisches Datalog

6. Akzeptanzbedingung

Wenn $q_f \in F$, enthält $\mathcal{P}$ die Regel

$$\text{accept}(x) \leftarrow \text{root}(x), q_f(x).$$

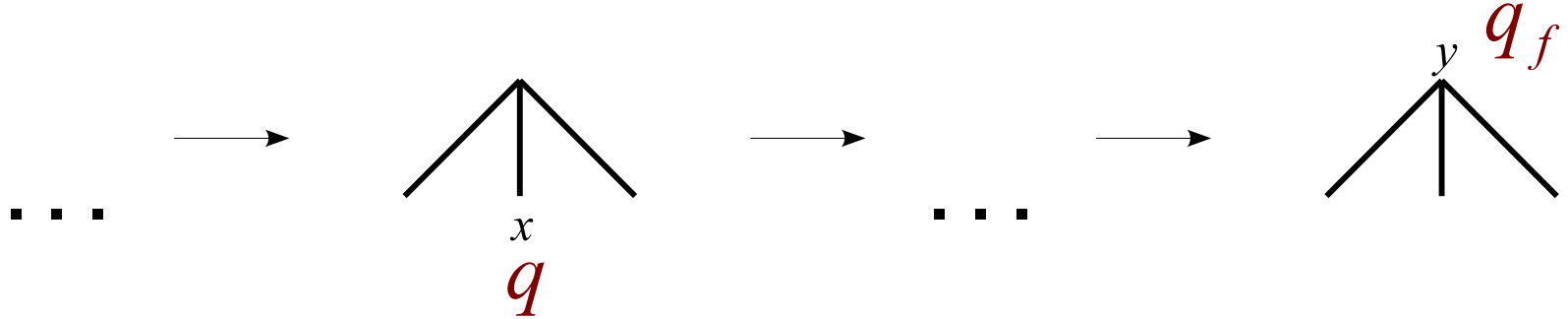


$QA^r \Rightarrow$ Monadisches Datalog

7. Auswahlfunktion

Wenn $\lambda(q, a) = \top$, enthält $\mathcal{P}$ die Regel

$$\mathsf{query}(x) \leftarrow q(x), \mathsf{label}_a(x), \mathsf{accept}(y).$$



$QA^r \Rightarrow$ Monadisches Datalog

Beispiel (1)

Beispiel

- Betrachte QA^r mit $Q = \{q_0, q_1, q_2, q_3, q_f, q_+\}$, $F = \{q_f\}$ und folgenden Transitionen:
  1. $\delta_\downarrow(q_0, *, 2) = \langle q_1, q_1 \rangle$
  2. $\delta_\uparrow(\langle q_1, * \rangle, \langle q_1, * \rangle) = q_2$
  3. $\delta_\downarrow(q_2, *, 2) = \langle q_3, q_3 \rangle$
  4. $\delta_\uparrow(\langle q_3, * \rangle, \langle q_3, * \rangle) = q_f$
  5. für alle anderen $(q_*, *) \in Q \times \Sigma : \delta_\uparrow(\langle q_*, * \rangle, \langle q_*, * \rangle) = q_+$
- Auswahlfunktion: $\lambda(q, *) = \begin{cases} \top & \text{falls } q = q_f \\ \bot & \text{sonst} \end{cases}$

$QA^r \Rightarrow$ Monadisches Datalog

Beispiel (2)

Beispiel

$$q_0(x) \leftarrow \mathsf{root}(x). \qquad \Rightarrow q_0(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$$q_1(x_1) \leftarrow q_0(x), \mathsf{child}_1(x, x_1), \mathsf{label}_a(x). \qquad \Rightarrow q_1(v_1) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$$q_1(x_2) \leftarrow q_0(x), \mathsf{child}_2(x, x_2), \mathsf{label}_a(x). \qquad \Rightarrow q_1(v_2) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$$q_2(x) \leftarrow \mathsf{child}_1(x, x_1), \mathsf{child}_2(x, x_2), q_1(x_1), q_1(x_2), \mathsf{label}_a(x_1), \mathsf{label}_a(x_2). \qquad \Rightarrow q_2(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$$q_3(x_1) \leftarrow q_2(x), \mathsf{child}_1(x, x_1), \mathsf{label}_a(x). \qquad \Rightarrow q_3(v_1) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$$q_3(x_2) \leftarrow q_2(x), \mathsf{child}_2(x, x_2), \mathsf{label}_a(x). \qquad \Rightarrow q_3(v_2) \in \mathcal{T}_{\mathcal{P}}^{\omega}$$

$QA^r \Rightarrow$ Monadisches Datalog

Beispiel (3)

Beispiel

$$
\begin{aligned}
q_f(x) \leftarrow\ & \mathsf{child}_1(x, x_1), \mathsf{child}_2(x, x_2), \\
& q_3(x_1), q_3(x_2), \\
& \mathsf{label}_a(x_1), \mathsf{label}_a(x_2). && \Rightarrow q_f(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega} \\
\mathsf{accept}(x) \leftarrow\ & \mathsf{root}(x), q_f(x). && \Rightarrow \mathsf{accept}(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega} \\
\mathsf{query}(x) \leftarrow\ & q_f(x), \mathsf{label}_a(x), \mathsf{accept}(y). && \Rightarrow \mathsf{query}(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega} \\
\langle q_+ \rangle(x) \leftarrow\ & \mathsf{child}_1(x, x_1), \mathsf{child}_2(x, x_2), \\
& \langle q_1 \rangle(x_1), \langle q_3 \rangle(x_2), \\
& \mathsf{label}_a(x_1), \mathsf{label}_a(x_2). && \Rightarrow q_+(v_0) \in \mathcal{T}_{\mathcal{P}}^{\omega}
\end{aligned}
$$

$QA^r \Rightarrow$ Monadisches Datalog

Problem des naiven Ansatzes

- Problem: Aufwärtstransitionen hängen von mehreren Knoten und deren Zuständen ab
- Sicherstellen, dass Zustandszuweisungen der Kinder auch wirklich in in der gleichen Konfiguration auftreten
- Trick: Speichere zusätzlich die letzte Zustandszuweisung des Elternknotens

$QA^r \Rightarrow$ Monadisches Datalog

Erweiterte Zustandszuweisungen

- Erweitere Zustandszuweisungen (q, v) zu Tupeln (q_0, q, v) (q_0 letzter Zustand des Elternknotens von v)

Lemma

Seien $q_0 \in Q$, $v \in$ dom. Falls $\langle q, \mathit{label}(v)\rangle \in U$, dann gibt es höchstens einen Zustand q, so dass (q_0, q, v) eine erweiterte Zustandszuweisung ist.

- $(q, x) \rightsquigarrow q(x)$
- $(q_0, q, x) \rightsquigarrow \langle q_0, q\rangle(x)$
- Zusätzlicher Dummy-Zustand: $\triangledown$

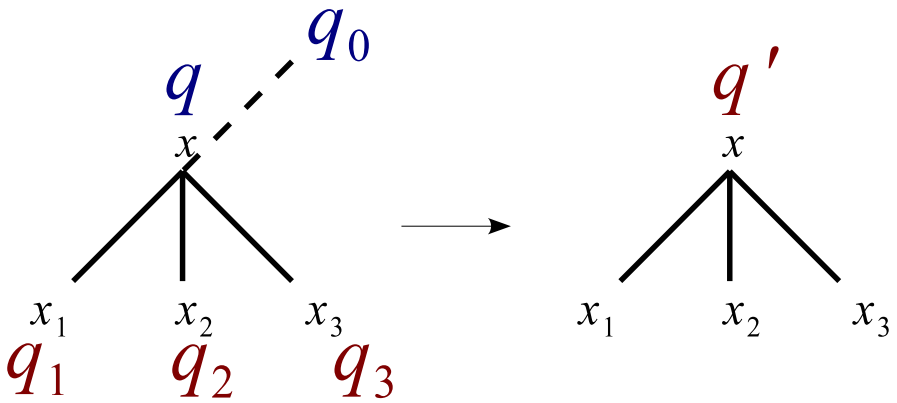
$QA^r \Rightarrow$ Monadisches Datalog

Aufwärtstransition (angepasste Simulation)

Wenn $\delta_\uparrow(\langle q_1, a_1\rangle, \ldots, \langle q_n, a_n\rangle) = q'$, enthält $\mathcal{P}$ die Regeln

$$\begin{aligned}\langle q_0, q'\rangle(x) \leftarrow\ & \langle q_0, q\rangle(x),\\ & \mathsf{child}_1(x, x_1), \ldots, \mathsf{child}_n(x, x_n),\\ & \langle q, q_1\rangle(x_1), \ldots, \langle q, q_n\rangle(x_n),\\ & \mathsf{label}_{a_1}(x_1), \ldots, \mathsf{label}_{a_n}(x_n).\end{aligned}$$

für alle $q \in Q, q_0 \in (Q \cup \{\nabla\})$.



$QA^r \Rightarrow$ Monadisches Datalog

Korrektheit und Vollständigkeit der Konstruktion

- Erinnerung („History“):
  $H = \{(q, v) \mid v \in C_i \text{ und } c_i(v) = q \text{ für ein } i\}$
- Für eine Menge $X \subseteq \mathcal{T}_{\mathcal{P}}^{\omega}$ definieren wir
  $\pi(X) := \{(q, v) \mid \exists q_0 \langle q_0, q\rangle(v) \in X\}$
- Zu zeigen ist $\pi(\mathcal{T}_{\mathcal{P}}^{\omega}) = H$

Skip

- Vollständigkeit ($\pi(\mathcal{T}_{\mathcal{P}}^{\omega}) \supseteq H$):
  Klar per Konstruktion, denn alle Transitionen in $\mathcal{A}$ werden auch in $\mathcal{P}$ kodiert
- Korrektheit ($\pi(\mathcal{T}_{\mathcal{P}}^{\omega}) \subseteq H$):
  Induktion über die Berechnung des Fixpunktes $\mathcal{T}_{\mathcal{P}}^{\omega}$

$QA^r \Rightarrow$ Monadisches Datalog

Identität der selektierten Knotenmengen

Noch zu zeigen:

- Durch $\mathcal{P}$ definierte Anfrage ist zu der durch $\mathcal{A}$ definierten Anfrage äquivalent

$$
\begin{aligned}
& \{v \mid \mathcal{A} \text{ selektiert } v \text{ in } t\} \\
\equiv\ & \{v \mid \mathcal{A} \text{ akzeptiert } t,\ (q, v) \in H \text{ und } \lambda(q, \mathsf{label}(v)) = \top\} \\
\equiv\ & \{v \mid \mathit{query}(v) \in \mathcal{T}^{\omega}_{\mathcal{P}}\}
\end{aligned}
$$

$QA^r \Rightarrow$ Monadisches Datalog

Mögliche Vereinfachung des Beweises

- Ist die Erweiterung der Zustandszuweisungen von (q, v) zu (q_0, q, v) immer notwendig?
- Wir wissen: $\mathcal{L}(\uparrow\text{DTA}) = \mathcal{L}(\text{2DTA})$
- Für Berechnung ist ein Bottom-Up Durchlauf ausreichend
- Mit dieser Einschränkung für Anfrageautomaten ist die naive Simulation möglich.
- Ausdrucksstärke bleibt wegen obiger Äquivalenz erhalten

Zusammenfassung

- Monadisches Datalog als Anfragesprache über Bäumen.
  - Syntax
  - Fixpunktsemantik
  - Effektive Auswertung
- Äquivalenz von monadischen Datalog-Anfragen und MSO-Anfragen
  - Monadisches Datalog $\Rightarrow$ Π_1-MSO-Anfragen
  - MSO-Anfragen $\Rightarrow$ Anfrageautomaten $\Rightarrow$ Monadisches Datalog
  - Effektive Auswertung von Anfrageautomaten
  - Nach Anpassung der Anfrageautomaten ist eine Vereinfachung der Simulation möglich

Vielen Dank

für die Aufmerksamkeit