XML Normalform, Teil 3

Dekompositionsalgorithmus

Sei $D = (E, A, P, R, r)$ eine DTD und Σ eine Menge FA über D.

- Verschieben eines Attributes;
  FD = $q \rightarrow p.@l$, $q \in EPaths(D)$
  Seien $D[p.@l := q.@m]$, $\Sigma[p.@l := q.@m]$ die DTD und die FAs nach Anwendung eines solchen Dekompositionsschrittes.
- Erzeugen eines neuen Element-Typs;
  FD = $\{q, p_1.@l_1, \ldots, p_n.@l_n\} \rightarrow p.@l$, $q \in EPaths(D)$
  Seien $D[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]]$,
  $\Sigma[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]]$ die DTD und die FAs nach Anwendung eines solchen Dekompositionsschrittes.

Verschieben eines Attributes

FD = $q \rightarrow p.@l$, $q \in EPaths(D)$.
$D[p.@l := q.@m] = (E, A', P, R', r)$, wobei

- $A' = A \cup \{@m\}$,
- $R'(last(q)) = R(last(p)) \cup \{@m\}$,
- $R'(last(p)) = R(last(p)) - \{@l\}$,
- $R'(\tau') = R(\tau')$, für alle $\tau' \in E - \{last(q), last(p)\}$.

$\Sigma[p.@l := q.@m] = \{S_1 \rightarrow S_2 \in (D, \Sigma)^+ | S_1 \cup S_2 \subseteq paths(D[p.@l := q.@m])\}$.

- $q \rightarrow p.@l \notin \Sigma[p.@l := q.@m]$,
- $q \rightarrow q.@m \notin \Sigma[p.@l := q.@m]$.

Sei $AP(D, \Sigma)$ die Menge der rechten Seiten der anomalous FDs bzgl. D und Σ.

Satz: Nach "Verschieben eines Attributes" gilt $AP(D', \Sigma') \subset AP(D, \Sigma)$.

Beweis:

(A) Wir zeigen $q.@m \notin AP(D', \Sigma')$.

Angenommen $S' \subseteq paths(D')$ und $S' \rightarrow q.@m \in (D', \Sigma')^+$ ist eine nicht-triviale FA und weiter $S' \subseteq paths(D')$ und $S' \rightarrow q \notin (D', \Sigma')^+$.

Es existiert ein XML-Baum T' mit $T' \models (D', \Sigma')$ und T' enthält Tupel t_1, t_2 so dass $t_1.S' = t_2.S', t_1.S' \neq \perp$ und $t_1.q \neq t_2.q$.

Da $q.@m$ nicht in Σ' auftaucht, kann T' um voneinander verschiedene Werte für $t_1.q.@m, t_2.q.@m$ ergänzt werden.

$T' \models (D', \Sigma')$ und $T' \not\models S' \rightarrow q.@m$, ein Widerspruch. Also, $q.@m \notin AP(D', \Sigma')$.

(B) Sei $S_1 \cup S_2 \subseteq paths(D') - \{q.@m\}$. Wir zeigen $(D, \Sigma) \vdash S_1 \rightarrow S_2$ gdw. $(D', \Sigma') \vdash S_1 \rightarrow S_2$. Es genügt zu zeigen: $(D, \Sigma) \vdash S_1 \rightarrow S_2$ wenn $(D', \Sigma') \vdash S_1 \rightarrow S_2$.

Angenommen $(D, \Sigma) \nvdash S_1 \rightarrow S_2$. Es existiert dann ein XML-Baum $T \models (D, \Sigma)$, $T \not\models S_1 \rightarrow S_2$.

Erweitere T zu T' indem $q.@m$ mit irgendeinem Wert hinzugefügt und $@l$ von $last(p)$ gestrichen wird.

Dann auch $T' \models (D', \Sigma')$, $T' \not\models S_1 \rightarrow S_2$, da alle Pfade aus $\Sigma' \cup \{S_1 \rightarrow S_2\}$ in $paths(D') - \{q.@m\}$ enthalten. q.e.d.

Erzeugen eines neuen Element-Typs

FD = $\{q, p_1.@l_1, \ldots, p_n.@l_n\} \rightarrow p.@l$, $q \in EPaths(D)$.
$D[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]] = (E', A, P', R', r)$, wobei

- $E' = E \cup \{\tau, \tau_1, \ldots, \tau_n\}$,
- Sofern $P(last(q))$ ein regulärer Ausdruck s, dann $P'(last(q))$ ist die Konkatenation von s und τ.
- $P'(\tau)$ ist die Konkatenation von $\tau_1, \ldots, \tau_n$.
- $P'(\tau_i) = \epsilon$, $1 \leq i \leq n$,
- $P'(\tau') = P(\tau')$, $\tau' \in E - \{last(q)\}$,
- $R'(\tau) = \{@l\}$, $R'(\tau_i) = \{@l_i\}$, $1 \leq i \leq n$,
- $R'(last(p)) = R(last(p)) - \{@l\}$ und $R'(\tau') = R(\tau')$ für $\tau' \in E - \{last(p)\}$.

$\Sigma[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]]$ enthält die folgenden FAs

- $S_1 \rightarrow S_2 \in (D, \Sigma)^+$, wobei $S_1 \cup S_2 \subseteq paths(D')$,
- Falls $S_1 \cup S_2 \subseteq \{q, p_1, \ldots, p_n, p_1.@l_1, \ldots, p_n.@l_n, p.@l\}$ und $S_1 \rightarrow S_2 \in (D, \Sigma)^+$, dann ändere p_i zu $q.\tau.\tau_i$, $p_i.@l_i$ zu $q.\tau.\tau_i.@l_i$ und $p.@l$ nach $q.\tau.@l$.
- $\{q, q.\tau.\tau_1.@l_1, \ldots, q.\tau.\tau_n.@l_n\} \rightarrow q.\tau$ und $\{q.\tau, q.\tau.\tau_i.@l_i\} \rightarrow q.\tau.\tau_i$, $1 \leq i \leq n$.

Eine FA $\{q, p_1.@l, \ldots, p_n.@l_n\} \rightarrow p_0.@l_0$ ist (D, Σ)-*minimal*, wenn keine anomalous FA $S' \rightarrow P_i.@l_i \in (D, \Sigma)^+$ existiert, so dass $0 \leq i \leq n$ und $S' \subseteq \{q, p_1, \ldots, p_n, p_0.@l_0, \ldots, p_n.@l_n\}$ und $|S'| \leq n$ und S' enthält maximal einen Element-Pfad.

Satz: Nach ”Erzeugen eines neuen Elementtyps” bzgl. einer minimalen FA gilt $AP(D', \Sigma') \subset AP(D, \Sigma)$.

Proposition: (D, Σ) ist in XNF, gdw. für jede nicht-triviale FA in Σ der Form $S \to p.@l$ oder $S \to p.\mathtt{S}$ gerade $S \to p \in (D, \Sigma)^+$.

Algorithmus:

1. If (D, Σ) is in XNF, then return (D, Σ); otherwise go to step (2).
2. If there is an anomalous FD $X \to p.@l$ and $q \in EPaths(D)$ such that $q \in X$ and $q \to X \in (D, \Sigma)^+$, then
   (a) Choose a fresh attribute $@m$
   (b) $D := D[p.@l := q.@m]$
   (c) $\Sigma := \Sigma[p.@l := q.@m]$
   (d) Go to step (1)
3. Choose a (D, Σ)-minimal anomalous FD $X \to p.@l$, where $X = \{q, p_1.@l_1, \ldots, p_n.@l_n\}$
   (a) Create fresh element types $\tau, \tau_1, \ldots, \tau_n$
   (b) $D := D[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]]$
   (c) $\Sigma := \Sigma[p.@l := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n, @l]]$
   (d) Go to step (1)

Implikationsproblem funktionaler Abhängigkeiten

Sei A ein Alphabet.

- Ein regulärer Ausdruck s über A heißt *trivial*, wenn
  - er die Form $(s_1, \ldots, s_n)$ hat,
  - zu jedem s_i ein $a_i \in A$ existiert, so dass s_i von einer der folgenden Formen:

$$a_i,\ a_i?,\ a_i+,\ a_i*$$

    wobei $a_i \neq a_j$, $i \neq j$.
- Ein regulärer Ausdruck s über A heißt *einfach*, wenn ein trivialer regulärer Ausdruck s' existiert, so dass jedes Wort $w \in L(s)$ sich als Permutation eines Wortes $w' \in L(s')$ ergibt, und umgekehrt.
- Eine DTD heißt *einfach*, wenn alle Produktionen einfache reguläre Ausdrücke über $E \cup \mathsf{S}$ verwenden.

Beispiele:

- $(a*, b*, c*)$ ist trivial; $(a|b|c)*$ ist einfach; $(a|b)$ ist nicht einfach.

Beispiel ebXML Process Specification Schema (Ausschnitt):

```
<!E ProcessSpecification (Documentation*, SubstitutionSet*,
    (Include | BusinessDocument | ProcessSpecification | Package |
     BinaryCollaboration | BusinessTransaction | MultiPartCollaboration)*)>
<!E Include (Documentation*)>
<!E BusinessDocument (ConditionExpression?, Documentation*)>
<!E SubstitutionSet (DocumentSubstitution | AttributSubstitution | Documentation)*>
...
```

Beispiel University:

```
<!E university (course*)>
<!E course (number, student*)>
<!E number (#PCDATA)>
<!E student ((name | FLName), grade)>
<!E name (#PCDATA)>
<!E FLName (first_name, last_name)>
<!E first_name (#PCDATA)>
<!E last_name (#PCDATA)>
<!E grade (#PCDATA)>
```

Frequently Asked Questions:

```
<!E section (logo*, title,
     (qna+ | q+ | (p | div | section)+))>
...
```

Sei A ein Alphabet.

- Ein regulärer Ausdruck s über A heißt *einfache Disjunktion*, wenn gilt
  - $s = \epsilon$, oder
  - $s = a$, $a \in A$, oder
  - $s = s_1|s_2$, wobei s_1, s_2 einfach Disjunktionen über Alphabeten A_1, A_2 und $A_1 \cap A_2 = \emptyset$.
- Eine DTD heißt *disjunktiv*, wenn für $\tau \in E$, $P(\tau) = s_1, \ldots, s_m$, wobei jedes s_i entweder
  - ein einfacher regulärer Ausdruck, oder
  - eine einfache Disjunktion

  über einem Alphabet A_i, wobei $A_i \cap A_j = \emptyset$, $i \neq j$, $1 \leq i, j \leq m$.
- Eine DTD D heißt *relational*, wenn für jeden XML-Baum T mit $T \models D$, gilt:

$$X \subset tuples_D(T), X \neq \emptyset \Longrightarrow trees_D(X) \models D.$$

Beispiel: Die DTD `<!ELEMENT A (B, B)>` ist nicht relational.

Satz: Jede disjunktive DTD ist relational.

Beweis:

Sei D eine disjunktive DTD, T ein XML-Baum, $T \models D$ und $\emptyset \neq X \subset tuples_D(T)$.
Angenommen, $trees_D(X) \not\models D$, d.h. es existiert ein XML-Baum T' mit $T' \not\models D$.
Dann existiert ein Pfad p zu einem Knoten v in T', so dass $lab(v) = \tau$ und $ele(v)$ ist nicht gemäß $P(\tau)$.

Sei $P(\tau) = s$, wobei s eine einfache Disjunktion über einem Alphabet A.
Dann existiert ein $t' \in X$, so dass $t'.p = v$ und $t'.p.a = \perp$ für jedes $a \in A$.
Aber da $T \models D$ existiert ein Tupel $t \in tuples_D(T)$, so dass $t.p.b \neq \perp$, $b \in A$. Weiter, $t'.w = t.w$ für alle $w \in paths(D)$, wobei $p.b$ kein Präfix von w.
Damit $t' \subset t$. Dies ist ein Widerspruch zu $t, t' \in tuples_D(T)$.

Der Fall s einfach regulär oder einfach disjunktiv ist analog.

Das Implikationsproblem für funktionale Abhängigkeiten in XML

- ist lösbar in quadratischer Zeit für einfache DTDs,
- ist coNP-vollständig für disjunktive, bzw. relationale DTDs,
- nicht endlich axiomatisierbar, im Allgemeinen.