University of Washington
Department of Computer Science & Engineering

Managing XML and Semistructured Data

Lecture 4: Path Expressions

Prof. Dan Suciu

Spring 2001

In this lecture

- Path expressions
- Regular path expressions
- Evaluation techniques

Resources:

Data on the Web Abiteboul, Buneman, Suciu : section 4.1

Path Expressions

Examples:

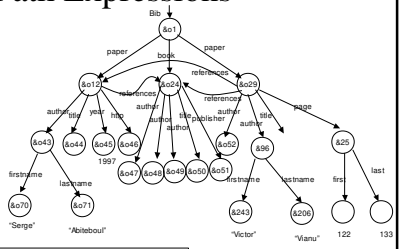
- Bib.paper
- Bib.book.publisher
- Bib.paper.author.lastname

Given an OEM instance, the *answer* of a path expression *p* is a set of objects

Path Expressions

Examples:

DB =



Bib.paper={&o12,&o29}
Bib.book.publisher={&o51}
Bib.paper.author.lastname={&o71,&206}

Answer of a Path Expression

Simple evaluation algorithms for Answer(P,DB):

Answer(P, DB) = f(P, root(DB))

Where:

f(ε, x) = {x}

f(L.P, x) = ∪{f(P,y) | ∃(x,L,y) ∈ edges(DB)}

Runs in PTIME in size(P), size(db):

- PTIME complexity

Regular Path Expressions

R ::= label | _ | R.R | (R|R) | R* | R+ | R?

Examples:

- Bib.(paper|book).author
- Bib.book.author.lastname?
- Bib.book.(references)*.author
- Bib.(_)*.zip

Applications of Regular Path Expressions

- Navigating uncertain structure:
  - Bib.book.author.lastname?
- Syntactic substitute for inheritance:
  - Bib.(paper|book).author
  - Better: Bib.publication.author, but we don't have inheritance

Applications of Regular Path Expressions

- Computing transitive closure:
  - Bib.(_)*.zip
    = everything accessible
  - Bib.book.(references)*.author
    = everything accessible via references
- Some regular expressions of doubtful practical use:
  - (references.references)*
    = a path with an even number of references
  - (_._)* = paths of even length
  - (_._._.(_)?)* = paths of length (3m + 4n) for some m,n
- But make great examples for illustration ☺

Answer of a Regular Path Expression

Recall:

- Lang(R) = the set of words P generated by R

Answer of regular path expressions:

- Answer(R,DB) = $\cup${Answer(P,DB) | P ∈ Lang(R)}

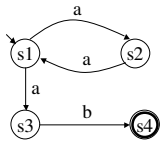
Need an evaluation algorithm that copes with cycles

Regular Path Expressions

Recall: each regular expression ➔ NDFA

Example: R = (a.a)*.a.b

A =



states(A) = {s1,s2,s3,s4}
initial(A) = s1
terminal(A) = {s4}

Regular Path Expressions

Canonical Evaluation Algorithm

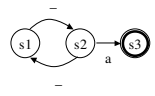
Answer(R,DB):

1. construct A from R
2. construct *product automaton* G = A x DB:
   - nodes(G) = states(A) x nodes(db)
   - edges(G) = {((s,x),L,(s',x') | (s,L,s') ∈ edges(A), (x,L,x') ∈ edges(DB)}
   - root(G) = (initial(A), root(DB))
3. compute G^{acc} = set of nodes accessible from root(G)
4. return {x | s ∈ terminal(A) s.t. (s,x) ∈ G^{acc}}

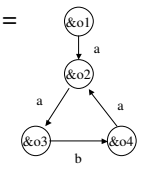
Regular Path Expressions

Example: R = _.(_._)*.a

A =

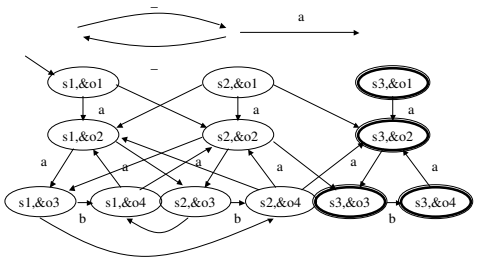


DB =

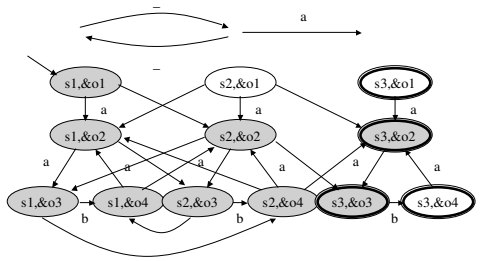


Answer of R on DB = { &o2, &o3}

Compute Product Automaton G



Compute Accessible Part G^{acc}



Answer(R,DB) = {&o2, &o3}

Complexity of Regular Path Expressions

- The evaluation algorithm runs in PTIME in size(R), size(DB)
- Even when there are cycles in DB