Module 3

XML Processing
(XPath, XQuery, XUpdate)

Part 4: Update, Scripting, Full Text

XML so far

- XML and XML Schema
  - serialization of data (documents + structured data)
  - validity data (constraints on structure)
- XQuery
  - extracting, aggregating, processing (parts of) data
  - constructing new data; transformation of data
- **Next:**
  - **Updates**
  - **Scripting**
  - **Full Text**

XQuery Update Overview

- Use as transformation + DB operation (side-effect)
  - Preserve Ids of affected nodes! (No Node Construction!)
- Updates are expressions!
  - return "()" as result
  - in addition, return a *Pending Update List*
- Updates are fully composable with other expressions
  - however, there are semantic restrictions!
  - e.g., no update in condition of an if-then-else allowed
- Primitive Updates: insert, delete, replace, rename
- Extensions to other expr: FLWOR, TypeSwitch, ...
- Either updates or results, single snapshot per query

Examples

- delete nodes //book[@year lt 1968]
- insert node <author/> into //book[@ISBN eq "34556"]
- for $x in //book
  where $x/year lt 2000 and $x/price gt 100
  return replace value of node $x/price
  with $x/price-0.3*$x/price
- if ($book/price gt 200) then
  rename node $book as "expensive-book"

- Update expressions work on "node" or "nodes"

Language Extensions Overview

- New Expressions:
  - Insert: Insert new XML instances
  - Delete: Delete XML instances
  - Replace, Rename: Replace/Rename XML Instances
  - Transform: modify a copy an existing XDM
  - fn:put(): place an XDM instance into a file/location
- Changed (composition) expressions
  - FLWR: Bulk update
  - If: Conditional update
  - Typeswitch: Type-Based updates
  - Comma Expression: Updates Sequences
  - Function Defintion: Define updating functions

Composability

- *Insert, delete, rename*, *replace, and calls to updating functions* are expressions
- Classify expressions as
  - Simple: all XQuery 1.0 expressions
  - Updating: all new Update expressions
- Updating is *not* fully composable with the rest
  - Semantic, not syntactic restrictions
- Updating only allowed in control-flow expressions (see previous slide) + standalone
- Control-flow expression get class type from their "input", only same type allowed for all inputs (both branches of if updating or simple)

Pending Updates List + Update Conflicts

- Each updating expression produces PUL
- Contains list of update operations (target+data)
- Bulk+control flow expressions need to merge PULs and resolve conflicts:
  - two or more update of the same type on the same node: rename, replaceNode, replaceValue, replaceElementContent
  - Put on the same uri
  - Namespace definitions: insertAttributes, rename, replaceNodes

Snapshot Semantics

- Updates are applied at the very end
  - inserts are not visible during execution
  - avoids Halloween problem
  - allows optimizations (change order of updates)
- Three steps
  - evaluate expr; compose pending update list (PUL)
    - append "primitive" to PUL in every iteration of FOR
  - conformance test of PUL
    - avoid duplicate updates to same node (complicated rule)
    - avoids indeterminism due to optimizations
  - apply PUL (update primitives one at a time)

Halloween Problem

```
for $x in $db/*
return insert node $x into $db
```

- Obviously, not a problem with snapshot semantics.
- (SQL does the same!)

XQuery Scripting

Observation

- Despite of XQuery and XQuery Updates, we still need Java, C#, PHP
  - implement user interfaces
  - write complex applications
- Once you start using Java, you are tempted to do everything in Java
- **Goal: Get rid of Java!!! All XQuery!**
  - XQuery Scripting: Extension of XQuery for script
- W3C published first proposal in March 2008
- Major disagreements on technical/political areas
- Lecture presents 2008 W3C draft

XQuery Scripting Overview

- Relax restrictions on Updating Expressions
  - Updating Expressions are allow to return XDM
  - Allow updating expressions everywhere
- Sequential Expressions:
  - Fine-grained snapshots
  - Expressions with side effects (=> sequential)
  - Define order in which expressions are evaluated
- New "sequential" expressions
  - Apply
  - Assignment
  - Block
  - While
  - Exit

Overview on semantic changes

- Today update snapshot: entire query
- Change:
  - Apply updates on specific operations (vary snapshot scope)
- Sequential execution order:
  - FLWOR: FLWO clauses are evaluated first; result in a tuple stream; then Return clause is evaluated in order for each tuple. Side-effects made by one row are visible to the subsequent rows.
  - Function Call: Parameters before function
  - COMMA: subexpressions are evaluated in order
  - (UPDATING) FUNCTION CALL: arguments are evaluated first before body gets evaluated
- Semantics is deterministic because of the sequential evaluation order

Example

```
declare sequential function myNs:cumCost($projects) as element( )*
{
    declare $total-cost as xs:decimal :=0;
    for $p in $projects[year eq 2005]
    return
        {set $total-cost := $total-cost+$p/cost;
         <project>
            <name>{$p/name}</name>
            <cost>{$p/cost}</cost>
            <cumCost>{$total-cost}</cumCost>
        <project>
        };
}
```

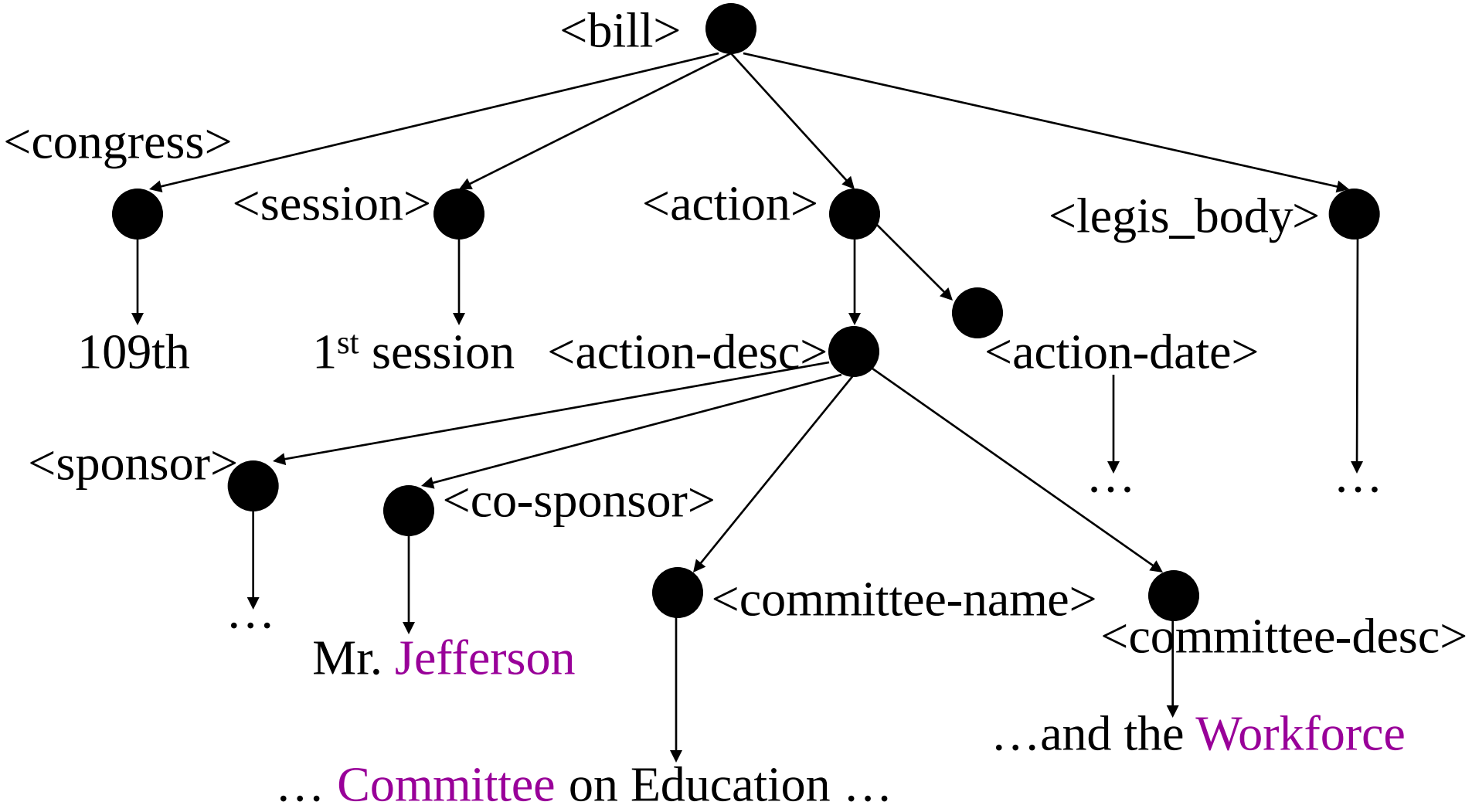
XQuery: self join or recursive function

XQuery Full Text

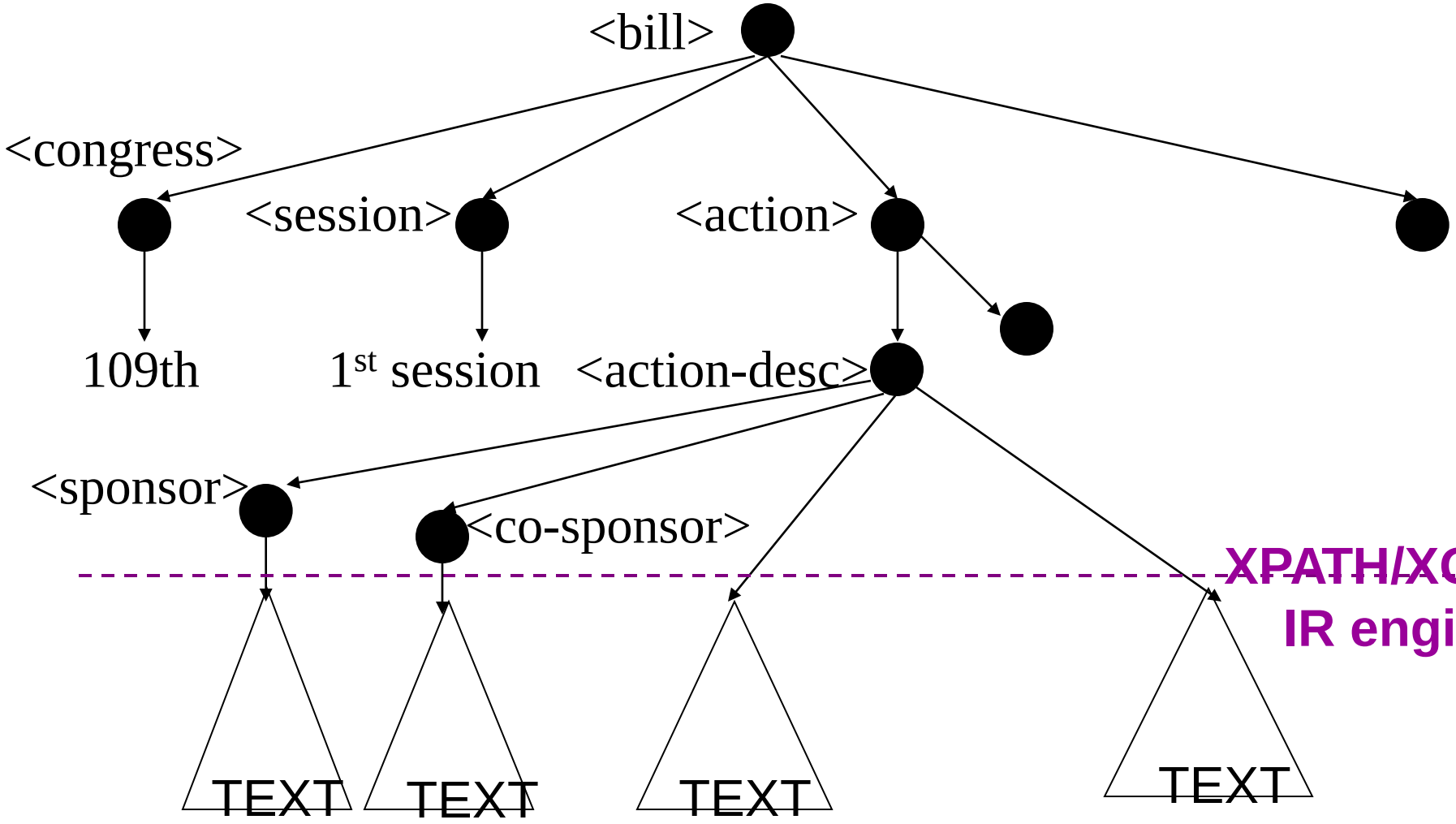
Full Text: Motivation

- XML is able to represent a mix of structured and text information:
  - XML applications: *digital libraries, content management.*
  - XML repositories: *IEEE INEX collection, SIGMOD Record in XML, LexisNexis, the Library of Congress collection, HL7, MPEG7.*
- Need for a language to *search* XML documents

LoC Document Example



Challenges: DB and IR



Challenges

- Searching over Structure+Text
  - *express complex full-text searches and combine them with structural searches*.
  - *specify a search context and return context.*
- Scores and Ranking
  - *Goal: find the most relevant results (remember how Google won over Altavista)*
  - *Typically assign a score value to each item of the result set, order by this value*
  - *In FT*
    - *specify a scoring condition,*
    - *possibly over both full-text and structured predicates*
    - *obtain k best results based on query relevance scores*

“FT Search” in XQuery 1.0

- Provides very rudimentary text/IR support
  - fn:contains(e, keywords)
  - Returns true iff element e contains keywords
- No support for complex IR queries
  - Distance predicates, stemming, …
- No scoring

Example Query

- From XQuery Full-Text Use Cases Document
  - Find the titles of the books that contain the phrases “Usability” and “Web site” in this order, in the same paragraph, using stemming if necessary to match the tokens
  - Find the titles of the books that contain “Usability” and “testing” within a window of 3 words, and return them in score order
- Such queries are used, e.g. in legal applications

XML FT Search Definition

- *Context expression*: XML elements searched:
  - pre-defined XML elements.
  - XPath/XQuery queries.
- *Return expression*: XML fragments returned:
  - pre-defined meaningful XML fragments.
  - XPath/XQuery to build answers.
- *Search expression*: FT search conditions:
  - Boolean keyword search.
  - proximity distance, scoping, thesaurus, stop words, stemming.
- *Score expression*:
  - system-defined scoring function.
  - user-defined scoring function.
  - query-dependent keyword weights.

Syntax Overview

One new XQuery construct, two extensions

1) FTContainsExpr
   - Expresses “Boolean” full-text search predicates
   - Seamlessly composes with other XQuery expressions
   - Integrates into grammar as comparison
2) Scoring Extensions
   - Extension to FLWOR expression
   - Possible at for and let
   - Can score FTContainsExpr *and* other expressions

FTContainsExpr and Scoring

- FTContainsExpr := RangeExpr (“contains text" FTSelection FTIgnoreOption?)?

```
//books//section [ . contains text ("usability" occurs exactly 4 times
    using stemming ftand "Software" using case sensitive) using
    stop words default window 4 words ordered]
```

- Scoring

```
for $b score $s in
      //books [ ./title contains text "XML" weight 0.4 and .//section
                  contains text ("indexing" using stemming ftand
                  "ranking" using thesaurus default)
                  distance exactly 5 words and ./price < 50 ]
order by  $s
return <result score="{$s}">  {$b/title, $b//authors}  </result>
```

FTSelection – IR operations

- FTSelection := FTOr FTPosFilter* ("weight" RangeExpr)?
- FTOr := FTAnd ("ftor" FTAnd)*
- ...
- FTPrimaryWithOptions ::= FTPrimary FTMatchOptions?
- FTPosFilter ::= FTOrder | FTWindow | FTDistance | FTScope | FTContent
- FTMatchOption ::= FTLanguageOption
  | FTWildCardOption
  | FTThesaurusOption
  | FTStemOption
  | FTCaseOption
  | FTDiacriticsOption
  | FTStopWordOption
  | FTExtensionOption

Summary

- Updates: Side-effects
  - change data without re-creating the data
  - data keeps its identity (stays the "same")
- Scripting: Programmability
  - assignments, control flow
  - visibility of updates
- Fulltext: Search
  - Combination of text search and structural queries
  - Rich (text)-IR operations
  - Scoring