Web data management

Assignments presentation

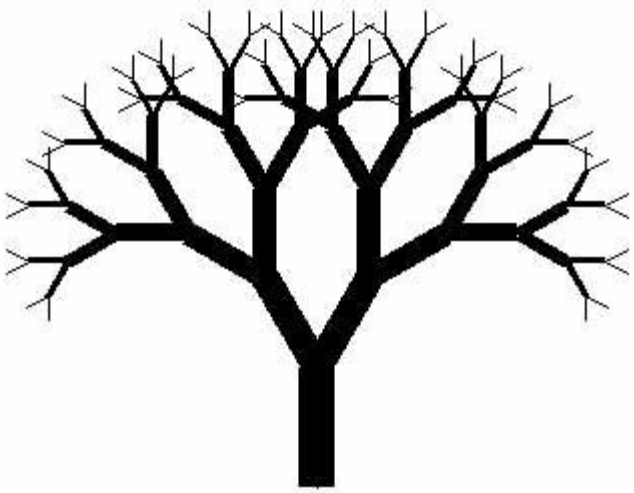
Remco van der Zon

r.w.l.vanderzon@student.tudelft.nl
https://bitbucket.org/remcovdzon/in4331

July 5th, 2012

Assignment 1

Tree Pattern Evaluation



http://www.bfoit.org/Intro_to_Programming/TT_Recursion.html

Assignment 1

Tree Pattern Evaluation

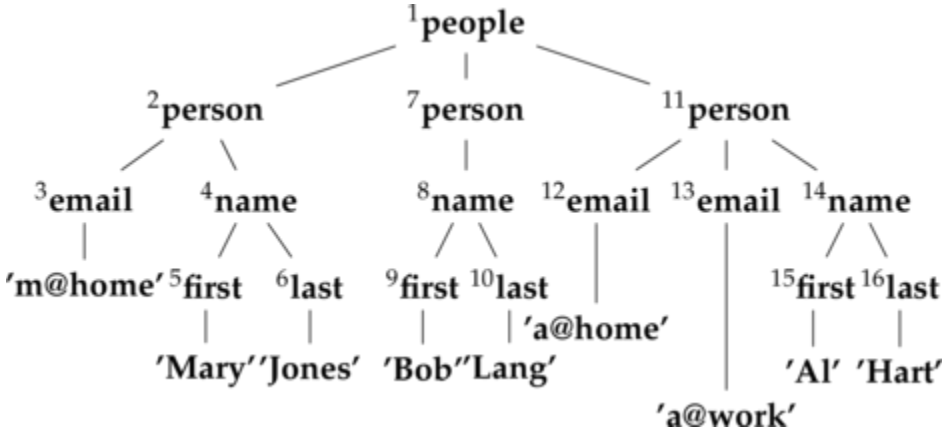
- Goal: query elements from XML file

- Query representation
- Algorithm in a nutshell
- Large XML documents supported

- 16 automated tests

Query representation

- Input dataset (WDM book)

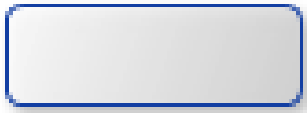


Query representation

- Legenda

Root element

Element

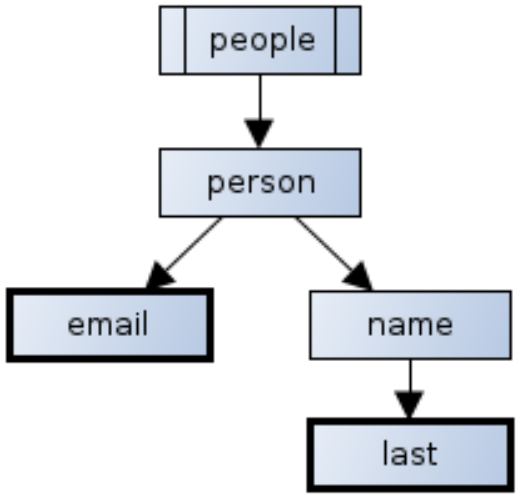


Attribute

Thick border: element selection

Query representation

- Selecting a complete match



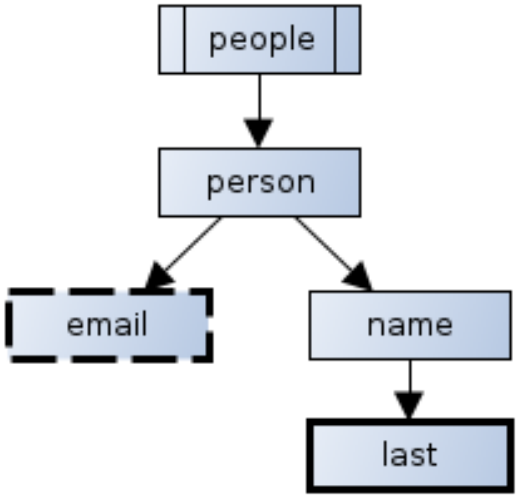

```
TPENode nodeRoot   = new TPENode("people");
TPENode nodePerson = new TPENode("person", nodeRoot);
TPENode nodeEmail  = new TPENode("email", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeLast   = new TPENode("last", nodeName);

nodeEmail.resultvalue = true;
nodeLast.resultvalue = true;
```

Query representation

- Selecting with optional field

  -> Optional nodes result values are set to *null*



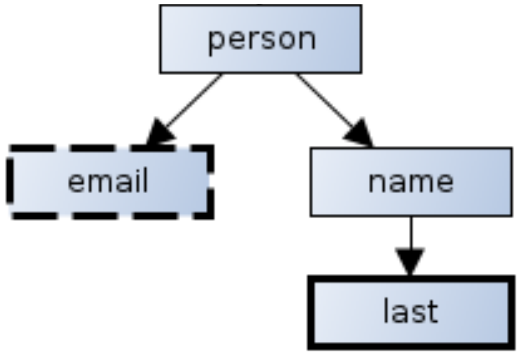

```
TPENode nodeRoot   = new TPENode("people");
TPENode nodePerson = new TPENode("person", nodeRoot);
TPENode nodeEmail  = new TPENode("email", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeLast   = new TPENode("last", nodeName);

nodeEmail.optional(true);

nodeEmail.resultvalue = true;
nodeLast.resultvalue = true;
```

Query representation

- Selecting not from root node



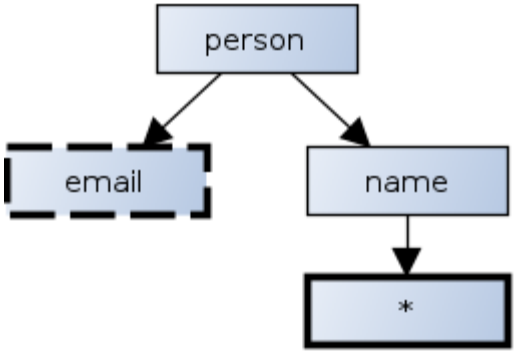

```
TPENode nodePerson = new TPENodeS("person");
TPENode nodeEmail  = new TPENode("email", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeLast   = new TPENode("last", nodeName);

nodeEmail.optional(true);

nodeEmail.resultvalue = true;
nodeLast.resultvalue = true;
```

Query representation

- Select star (*)



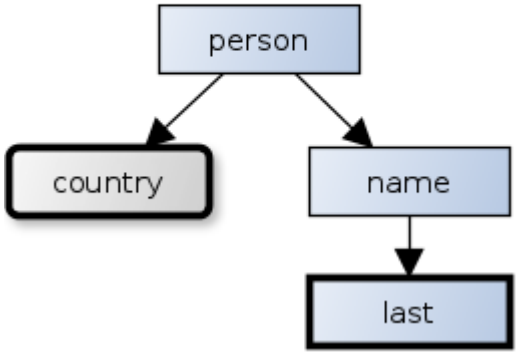

```
TPENode nodePerson = new TPENodeS("person");
TPENode nodeEmail  = new TPENode("email", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeNameS  = new TPENode("last", nodeName);

nodeEmail.optional(true);

nodeEmail.resultvalue = true;
nodeNameS.resultvalue = true;
```

Query representation

- Selection of attributes

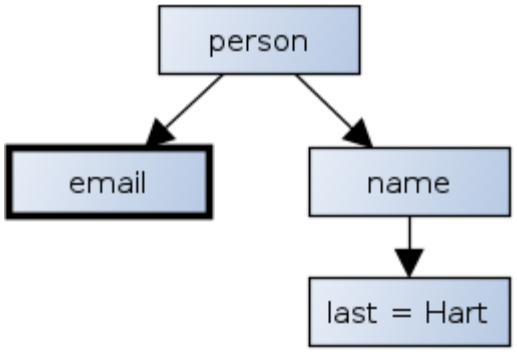



```
TPENode nodePerson = new TPENodeS("person");
TPENode nodeCAttr
       = new TPENodeAttribute("country", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeLast   = new TPENode("last", nodeName);

nodeCAttr.resultvalue = true;
nodeLast.resultvalue = true;
```

Query representation

- Selection with predicates




```
TPENode nodePerson = new TPENodeS("person");
TPENode nodeEmail  = new TPENode("email", nodePerson);
TPENode nodeName   = new TPENode("name", nodePerson);
TPENode nodeLast   = new TPENode("last", nodeName);

nodeEmail.optional(true);

nodeEmail.resultvalue = true;
nodeLast.addPredicate(new StringCompare("Hart"));
```

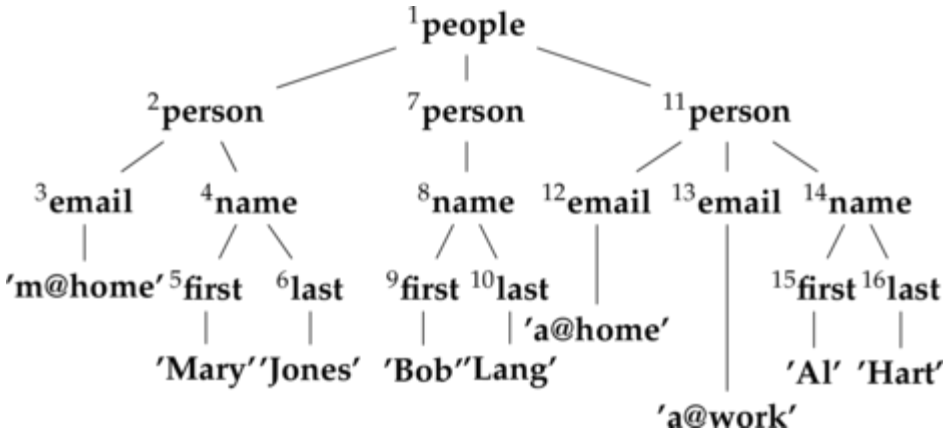
StackEval Algorithm in a nutshell

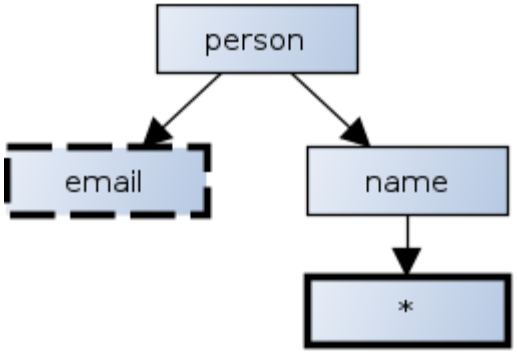
- For each element in query: create a stack
- SAX parser:
  - On tag open: Push match on corresponding element stacks.
    Connect match as child to parent match.
  - On tag close: Remove match from corresponding stack.
    If child requirements are not met: die (remove from parent match).
    If match pushed on root element: solution!
  - On all content: Add content to all matches (building match node content string)
- Each match can calculate its ‘output’ based on its children matches

Tree Pattern Evaluation

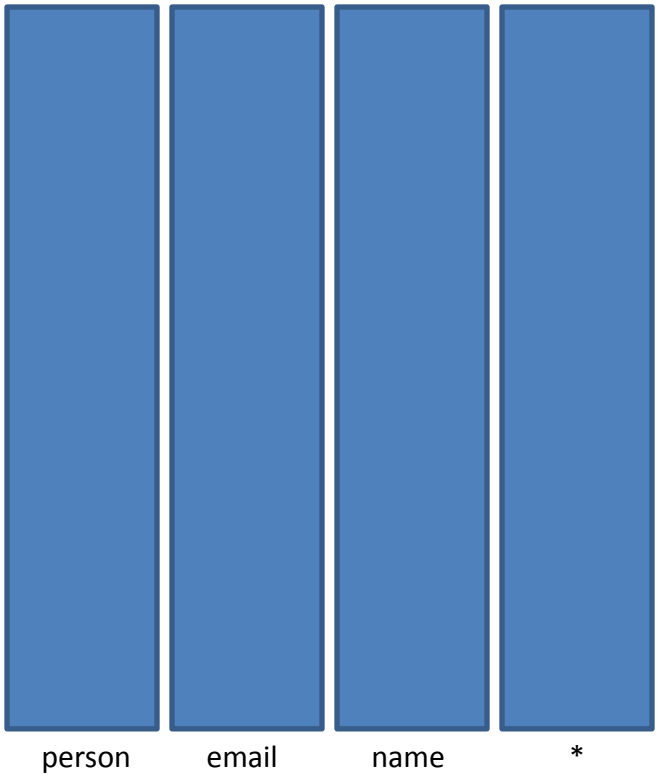
StackEval Algorithm in a nutshell

Example





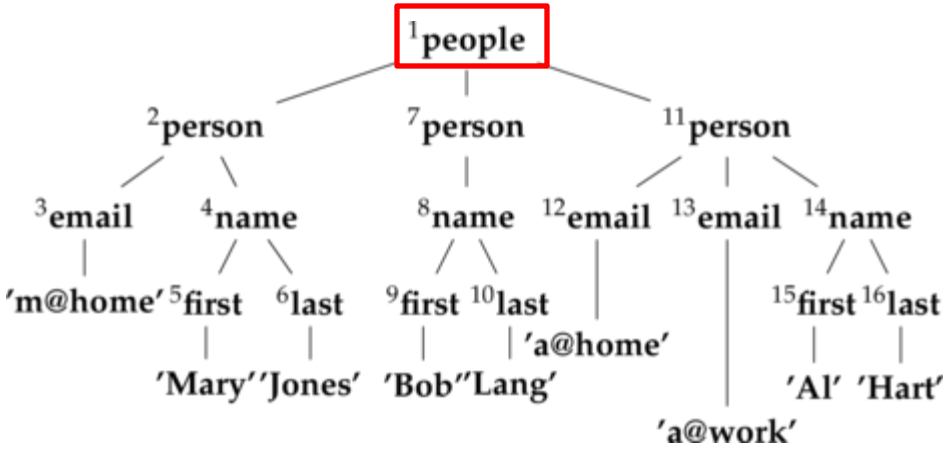
Match:

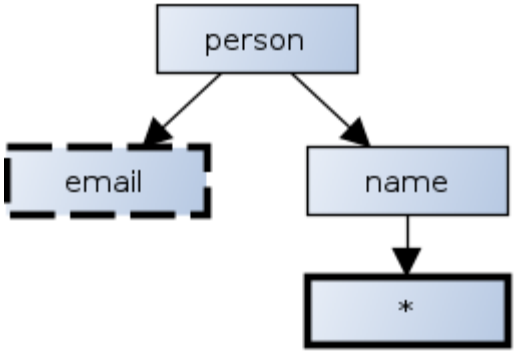


Tree Pattern Evaluation

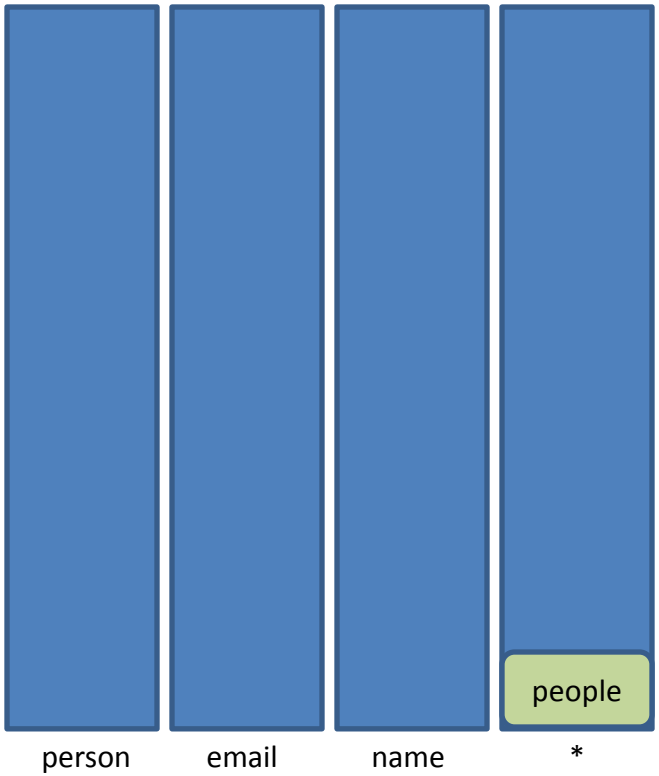
StackEval Algorithm in a nutshell

Example





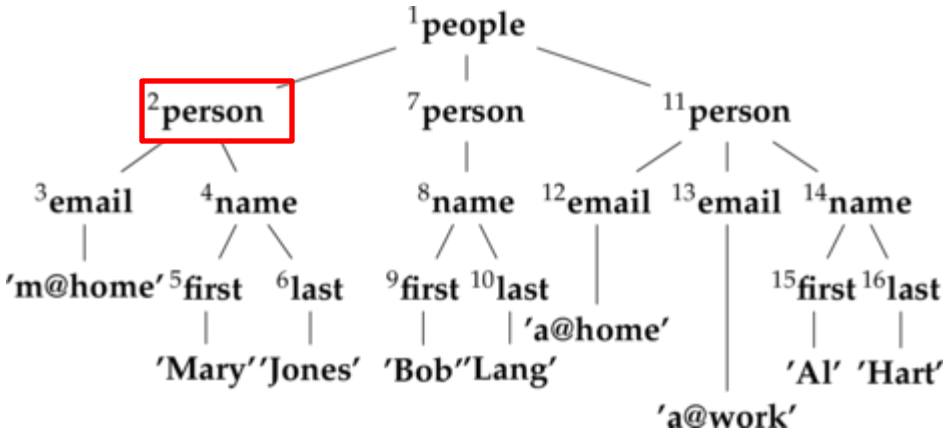
Match:

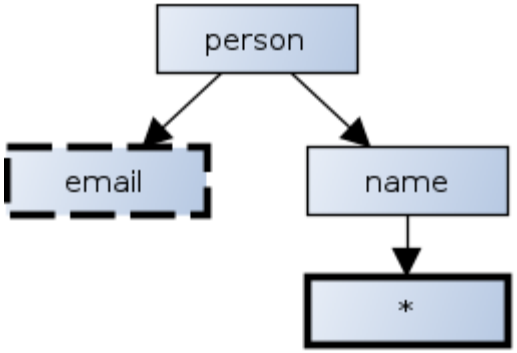


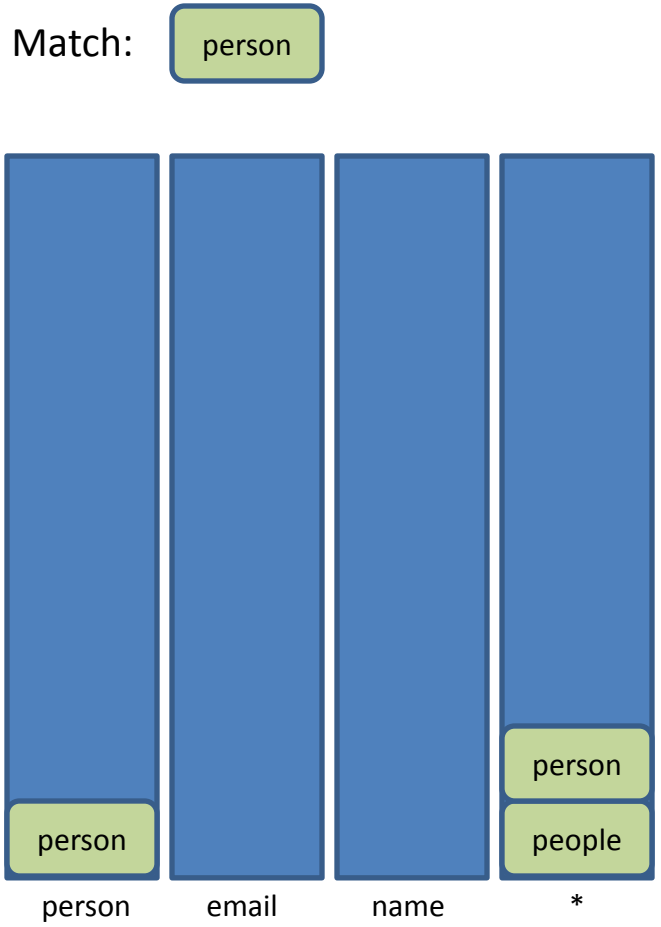
Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example



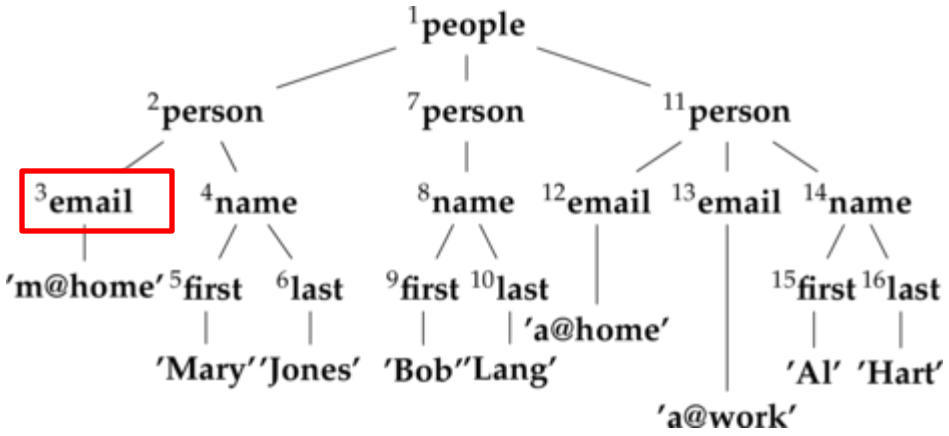


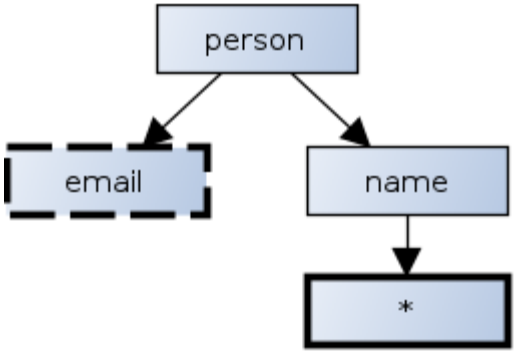


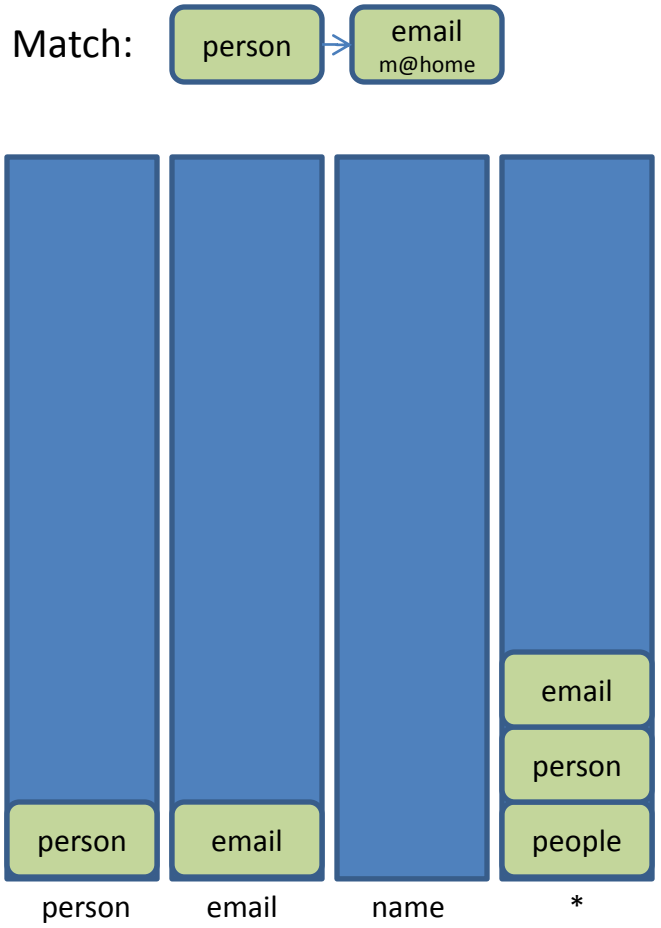
Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example



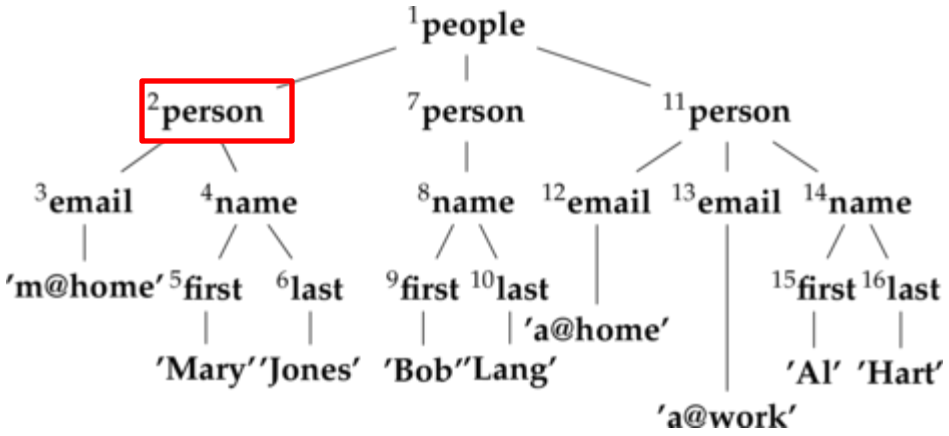


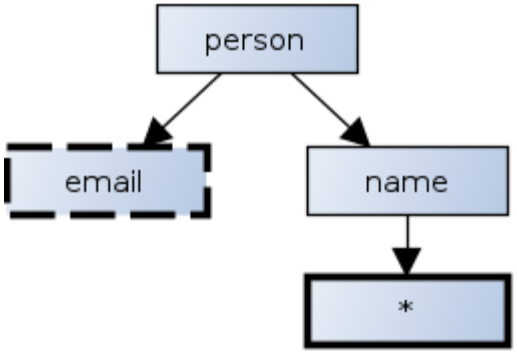


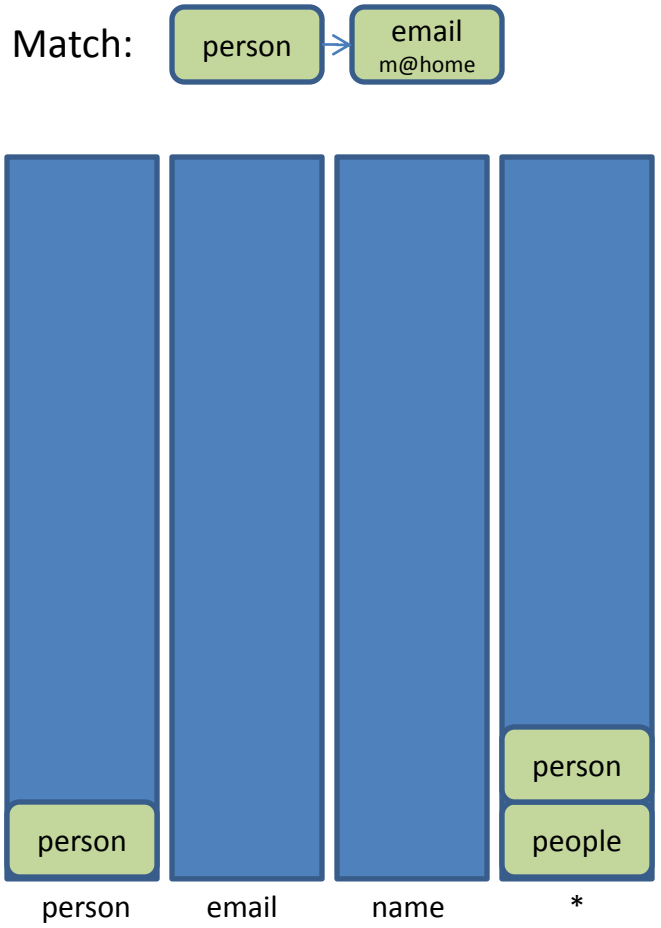
Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example



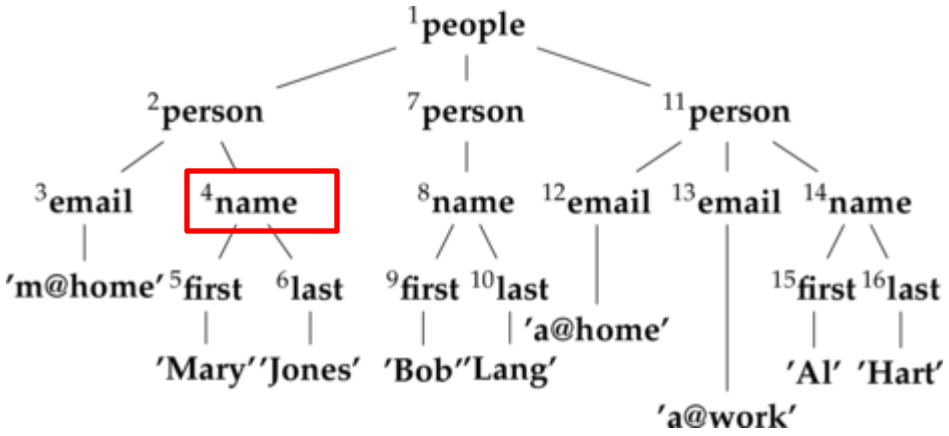


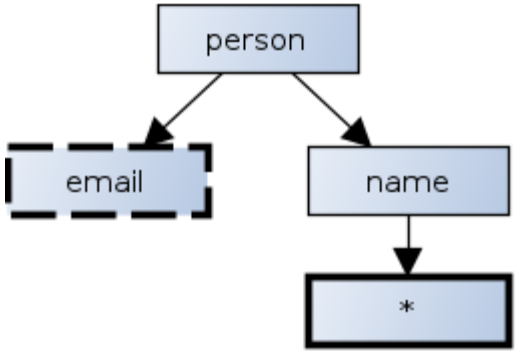


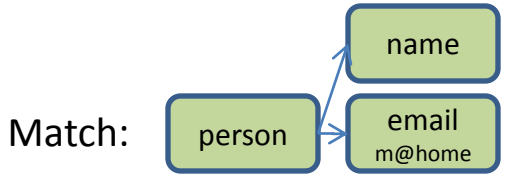
Tree Pattern Evaluation

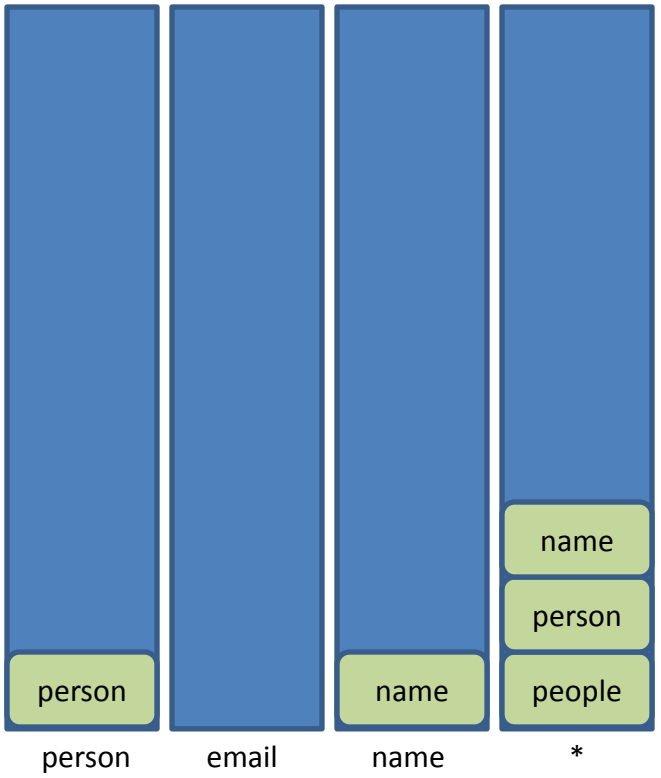
StackEval Algorithm in a nutshell

Example





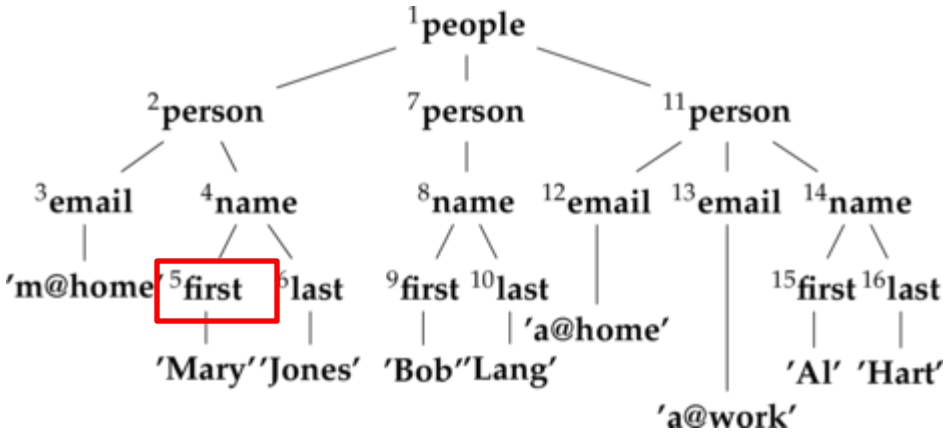


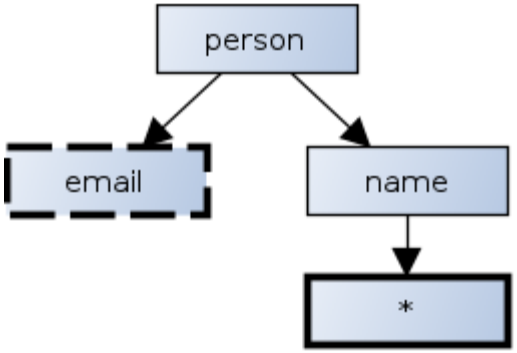


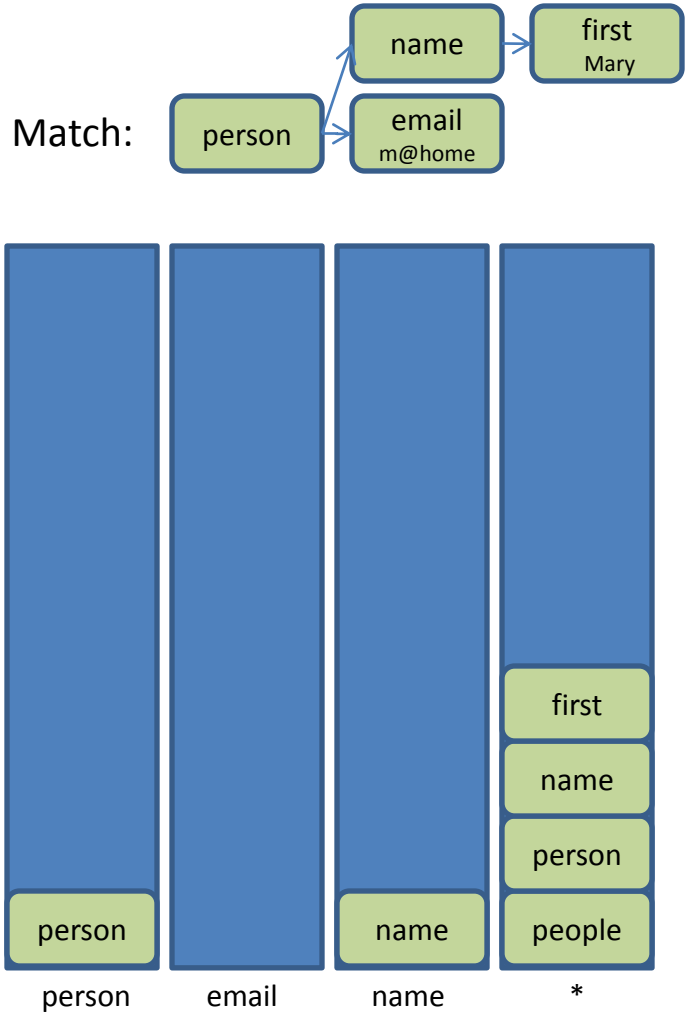
Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example



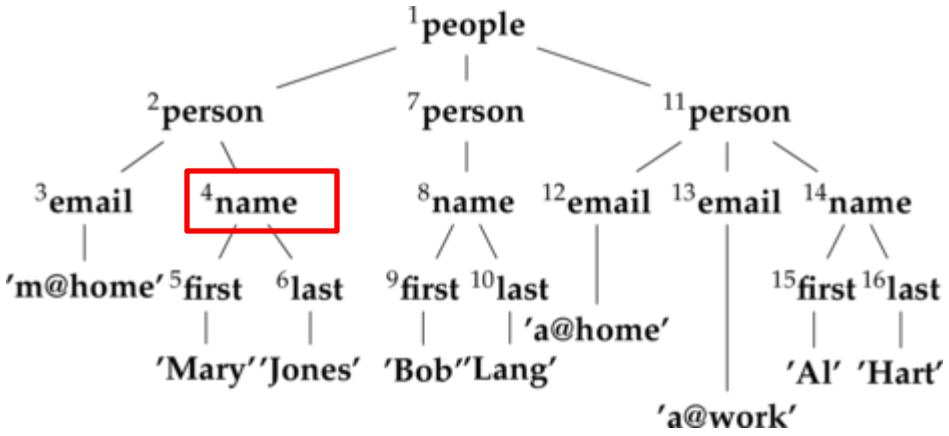


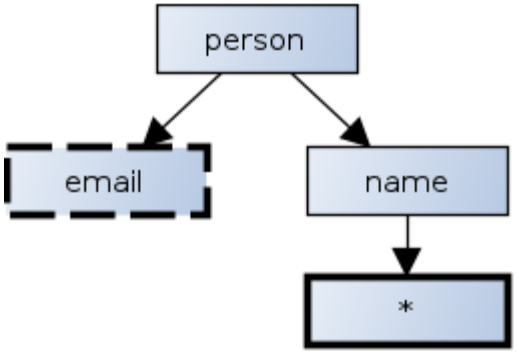


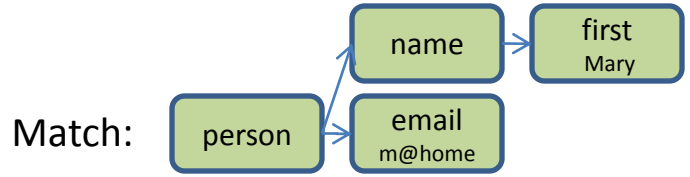
Tree Pattern Evaluation

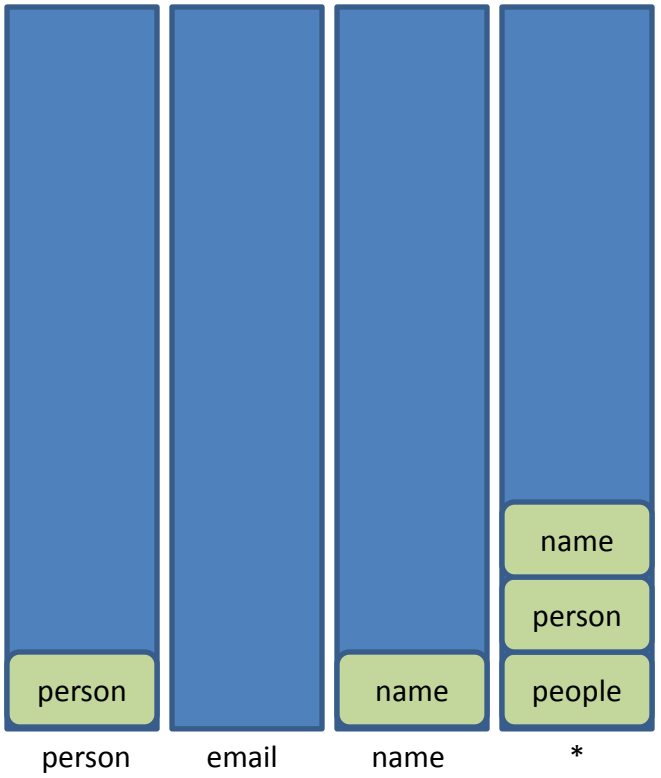
StackEval Algorithm in a nutshell

Example





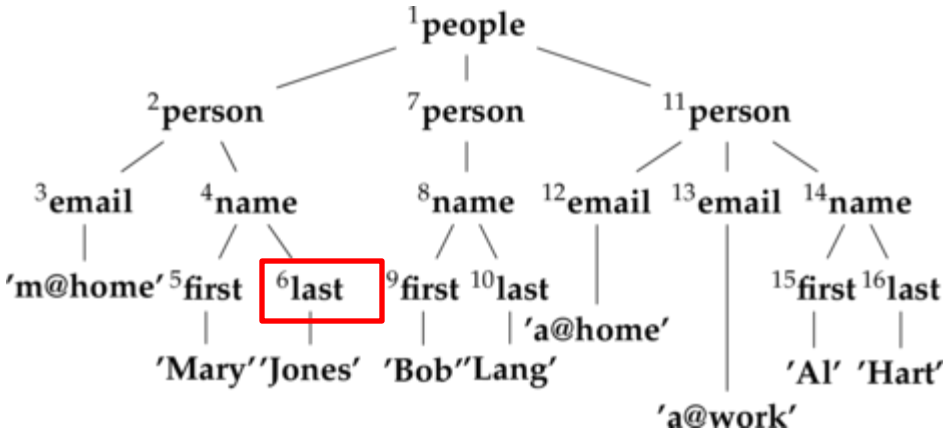


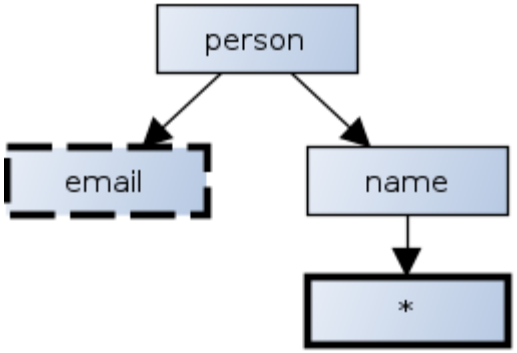


Tree Pattern Evaluation

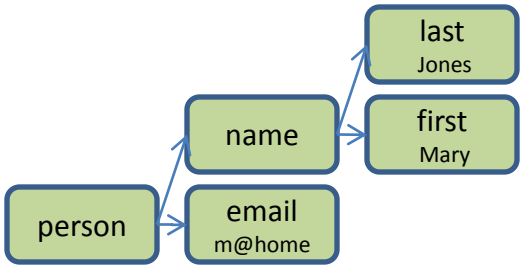
StackEval Algorithm in a nutshell

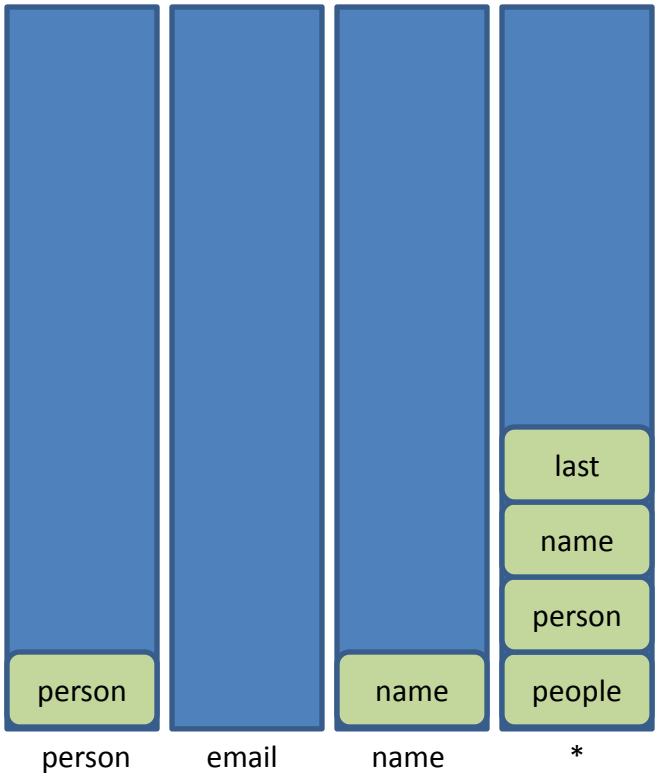
Example





Match:

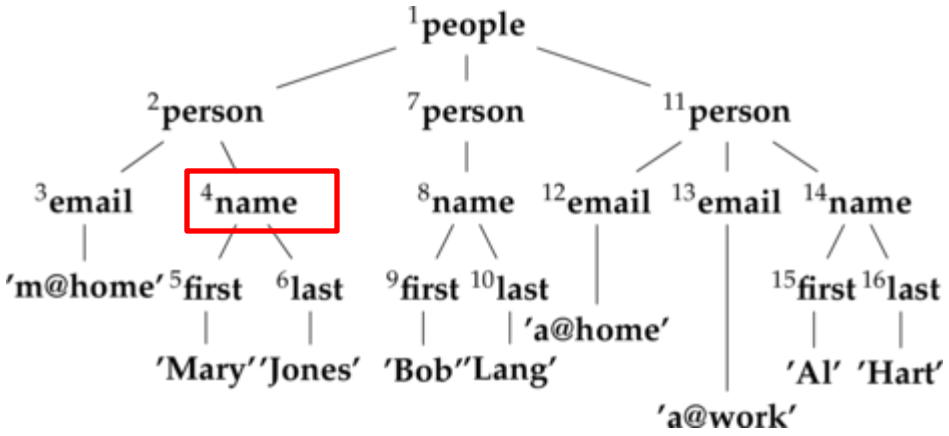


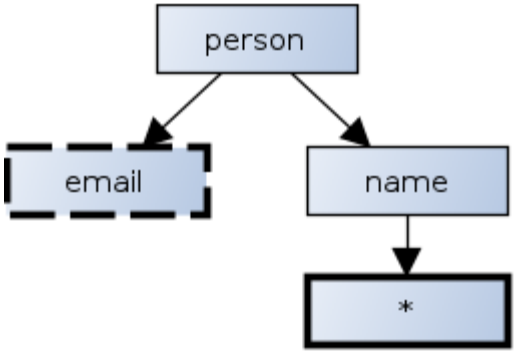


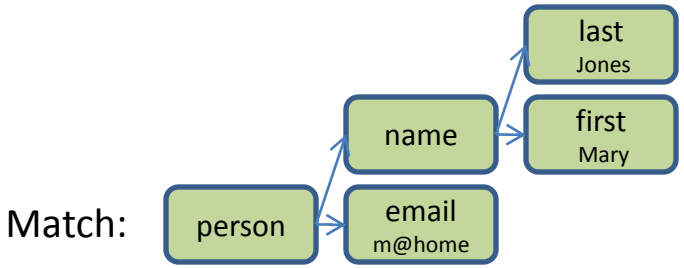
Tree Pattern Evaluation

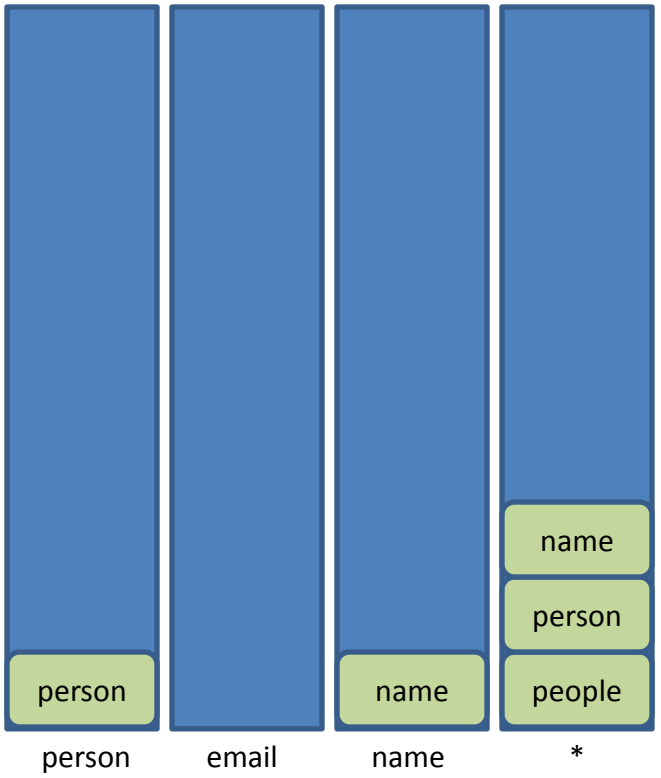
StackEval Algorithm in a nutshell

Example





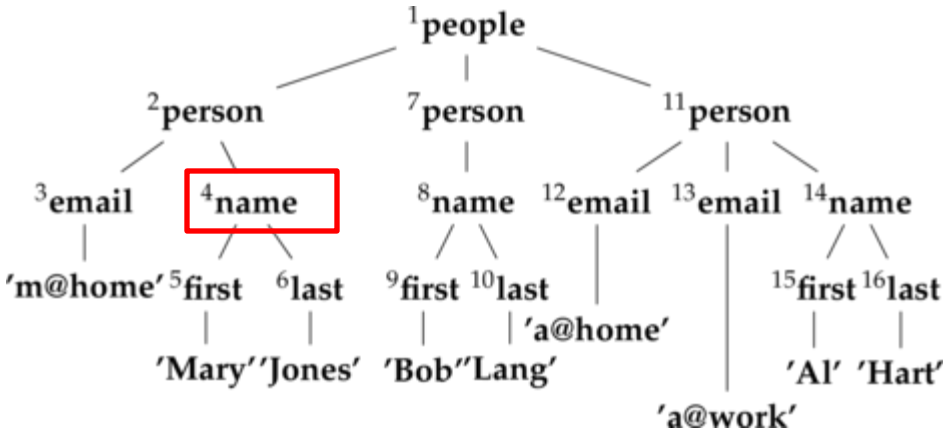


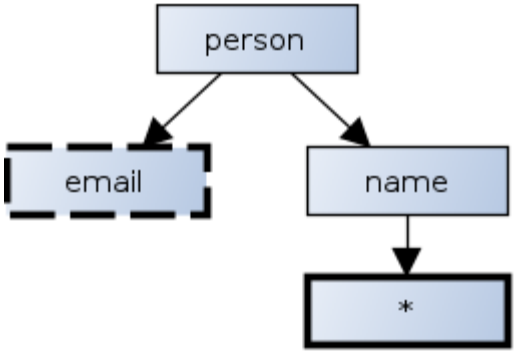


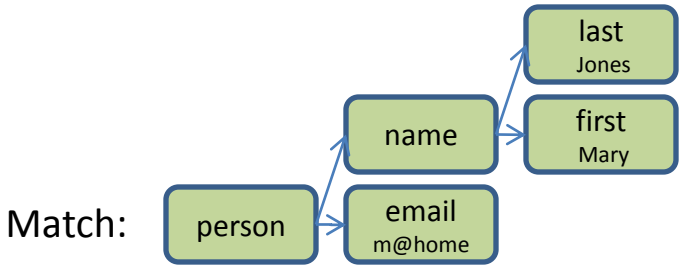
Tree Pattern Evaluation

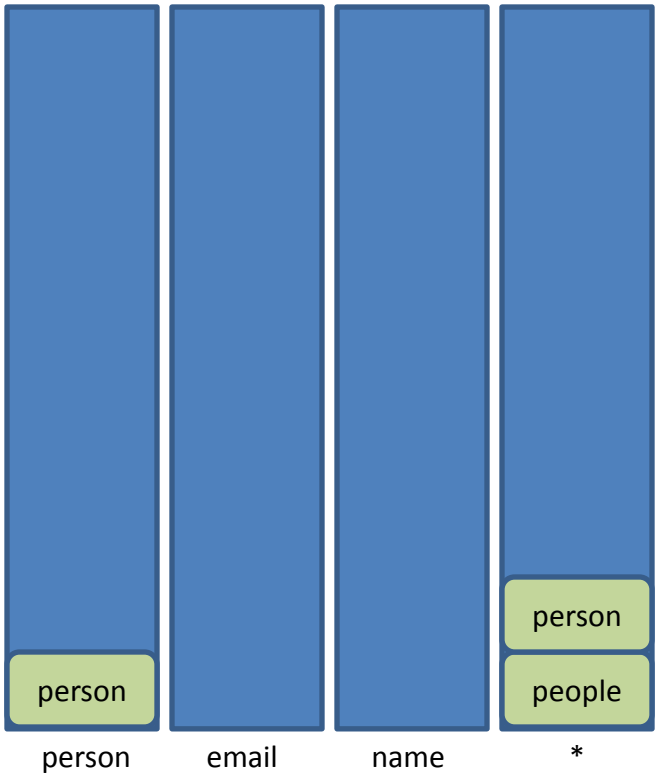
StackEval Algorithm in a nutshell

Example





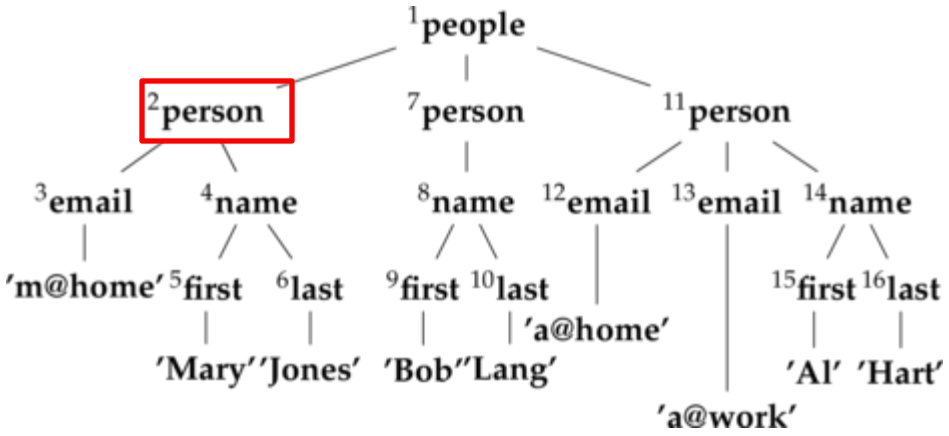


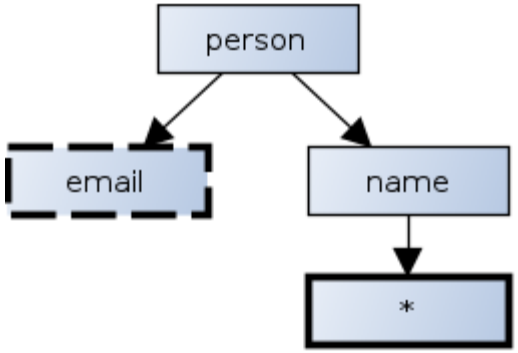


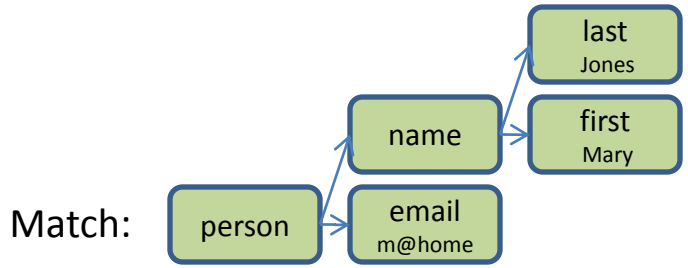
Tree Pattern Evaluation

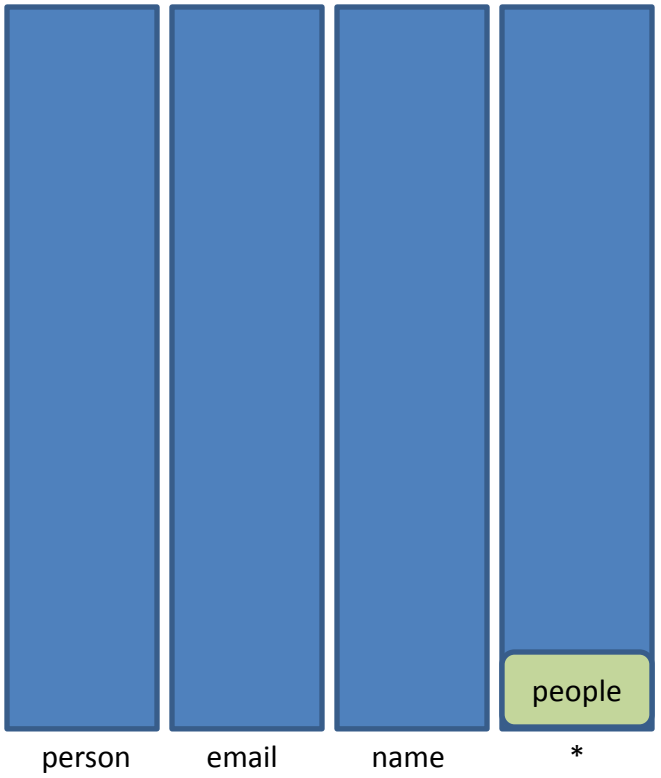
StackEval Algorithm in a nutshell

Example







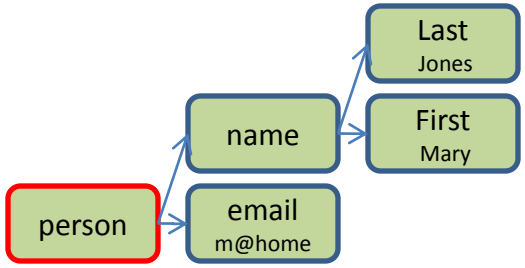


Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example – From match to solution

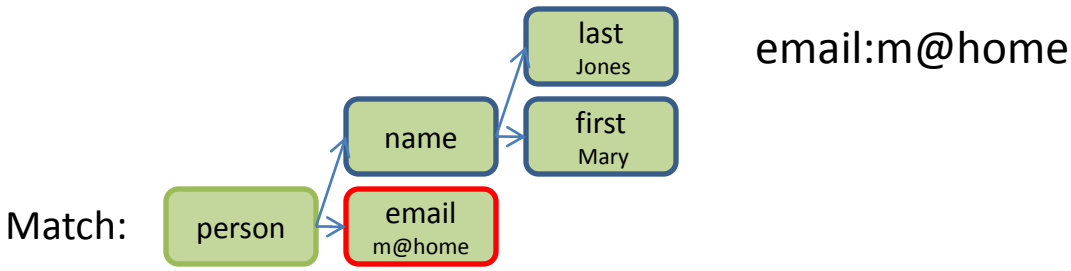
Match:



Tree Pattern Evaluation

StackEval Algorithm in a nutshell

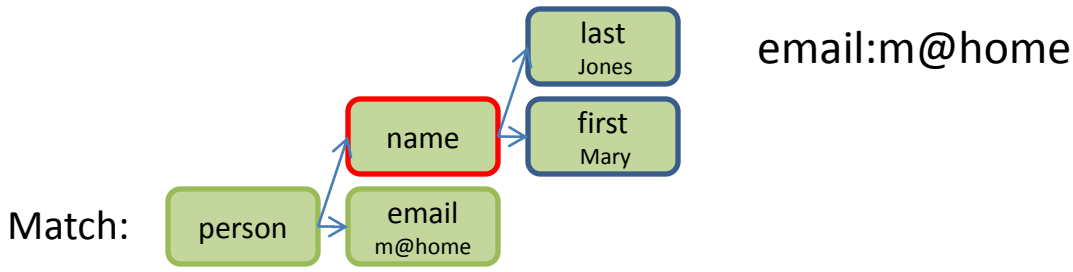
Example – From match to solution



Tree Pattern Evaluation

StackEval Algorithm in a nutshell

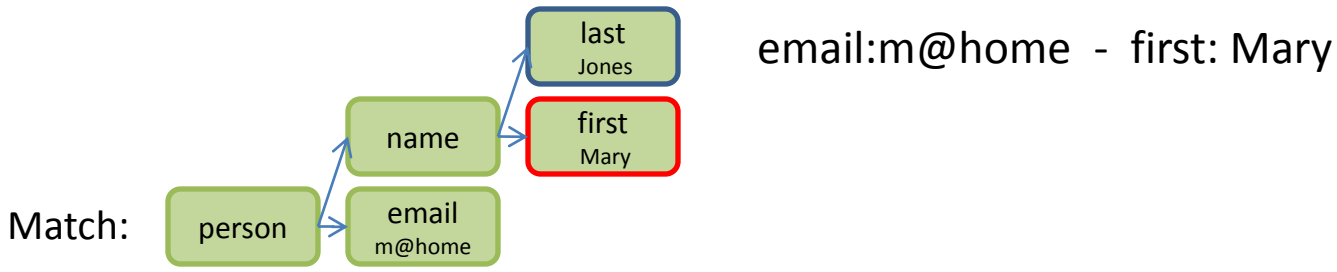
Example – From match to solution



Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example – From match to solution



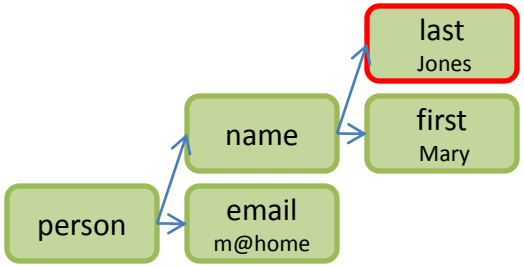
email:m@home - first: Mary

Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example – From match to solution

Match:



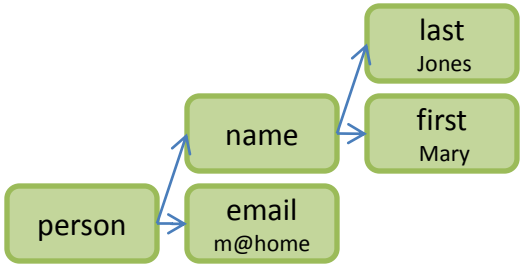
email:m@home - first: Mary
email:m@home - last: Jones

Tree Pattern Evaluation

StackEval Algorithm in a nutshell

Example – From match to solution

Match:



email:m@home - first: Mary
email:m@home - last: Jones

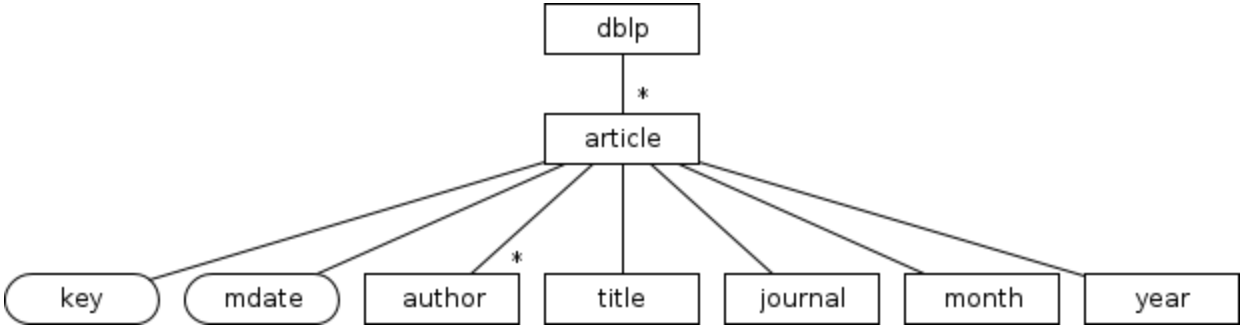
Large XML documents supported

- Results can be printed or piped to file or other application during run-time.
  -> Information removed from stacks when a match has been found
- DEMO
  -> Also demonstrates how to print results as XML

http://www.teknocrat.com/how-to-send-large-files-in-email.html

Large XML documents supported

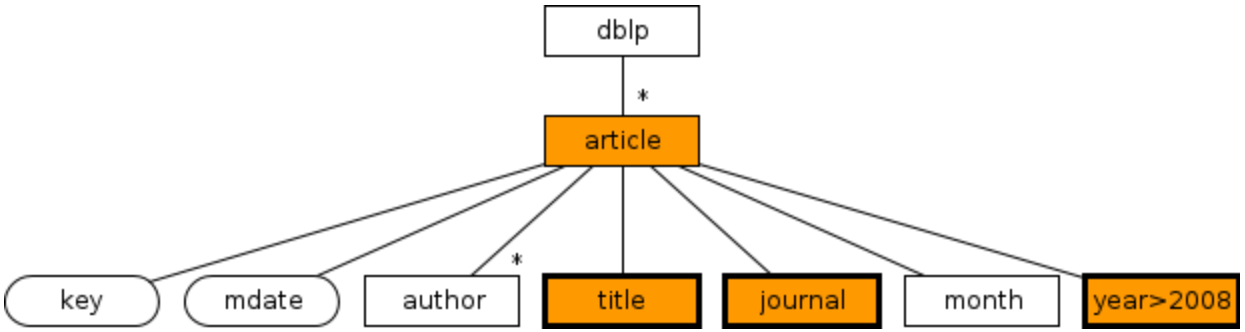
- Input set: DBLP (1018M)



http://www.teknocrat.com/how-to-send-large-files-in-email.html

Large XML documents supported

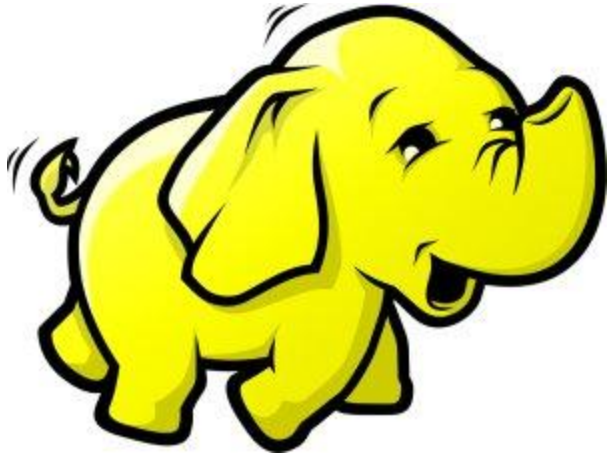
- Input set: DBLP (1018M)
- Select title, journal and year of articles with year > 2008



http://www.teknocrat.com/how-to-send-large-files-in-email.html

Assignment 2

Large-Scale datamanagement with Hadoop



Setting up Hadoop

- Following WDM-book steps:

```
hadoop namenode -format
start-dfs.sh
hadoop fs -put movies/ /movies
```

- Use hdfs-uri as input paths:

```
hdfs://localhost:9000/movies
```

Inverted files project

- Goal: For each distinct word in all documents generate a list of occuring documents, and calculate the IDF-value

Inverted files project

- IDF: Inverse Document Frequency

$$IDF_t = \log_{10}(N / df_t)$$

t: term

N: total documents count

df_t: number of documents with this term

Inverted files project

- Input dataset: movies from WDM book files.
- Output snippet (DEMO):

```
(...)
hunters     0.8450980400142568  "Heat":1
ideal       0.8450980400142568  "Lost in Translation":1
if          0.3010299956639812  "Lost in Translation":1 "Heat":1 "Spider-Man":1
illfated    0.8450980400142568  "Marie Antoinette":1
imperial    0.8450980400142568  "Marie Antoinette":1
improbable  0.8450980400142568  "Lost in Translation":1
in          0.0                 "Lost in Translation":3 "Marie Antoinette":3
                                "Heat":7 "Spider-Man":1 "A History of Violence":2
                                "Unforgiven":1 "Match Point":1
including   0.8450980400142568  "Heat":1
(...)
```

Reading XML files

- Getting the right XML in the Mapper

```
<movie><!-- all content here --></movie>
```

- Implemented with Mahout XMLInput parser

```
Configuration conf = new Configuration();
conf.set("xmlinput.start", "<movie>");
conf.set("xmlinput.end", "</movie>");

Job job = new Job(conf, "Movies analyzer");
job.setInputFormatClass(XMLInput.class);
```

- XML Analyzed with assignment 1 code

Inverted files project

- Getting the number of documents
  - Delivered implementation: hardcoded
  - Other solution: run a Mapper first, which ‘calculates’ the number of documents, then run the inverted files mapper.
  - Adjusted implementation: use HDFS: it can count!

```
hadoop fs -count [-q] <paths>
```

Large-Scale datamanagement with Hadoop

Inverted files project

- One map-combine-reduce job needed.
- Mapper

  Normalize text

  Tokenize -> emit for each word a pair (document, 1) with *word* as key
- Combiner

  for each *word*, collect all (document,1) pairs,
  -> merge them to (docuemnt, *n*) pairs.
- Reducer

  For each *word*, calculate the IDF-value and write this along with a list of al documents

Web data management

Assignments presentation

Remco van der Zon

r.w.l.vanderzon@student.tudelft.nl
https://bitbucket.org/remcovdzon/in4331

July 5th, 2012