View-Based Tree-Language Rewritings

Laks Lakshmanan, Alex Thomo

University of British Columbia, Canada

University of Victoria, Canada

Importance of trees – XML

- Semi-structured textual formats are very popular.

```
<movie>
        <title>House of cards</title>
        <year>2013</year>

        <character>
                <name>Francis</name>
                <actor>Kevin Spacey</actor>
        </character>

        <character>
                <name>Claire</name>
                <actor>Robin Wright</actor>
        </character>
</movie>
```

XML (Multi TB) success stories:

1. Elsevier
   - Papers and books
2. JPMorgan Chase & Co
   - Stock research data
3. JetBlue Airways
   - Document management

Source: MarkLogic
XML Impacting the Enterprise Tapping into the Power of XML: Five Success Stories

Importance of trees – JSON

- Semi-structured textual formats are very popular.

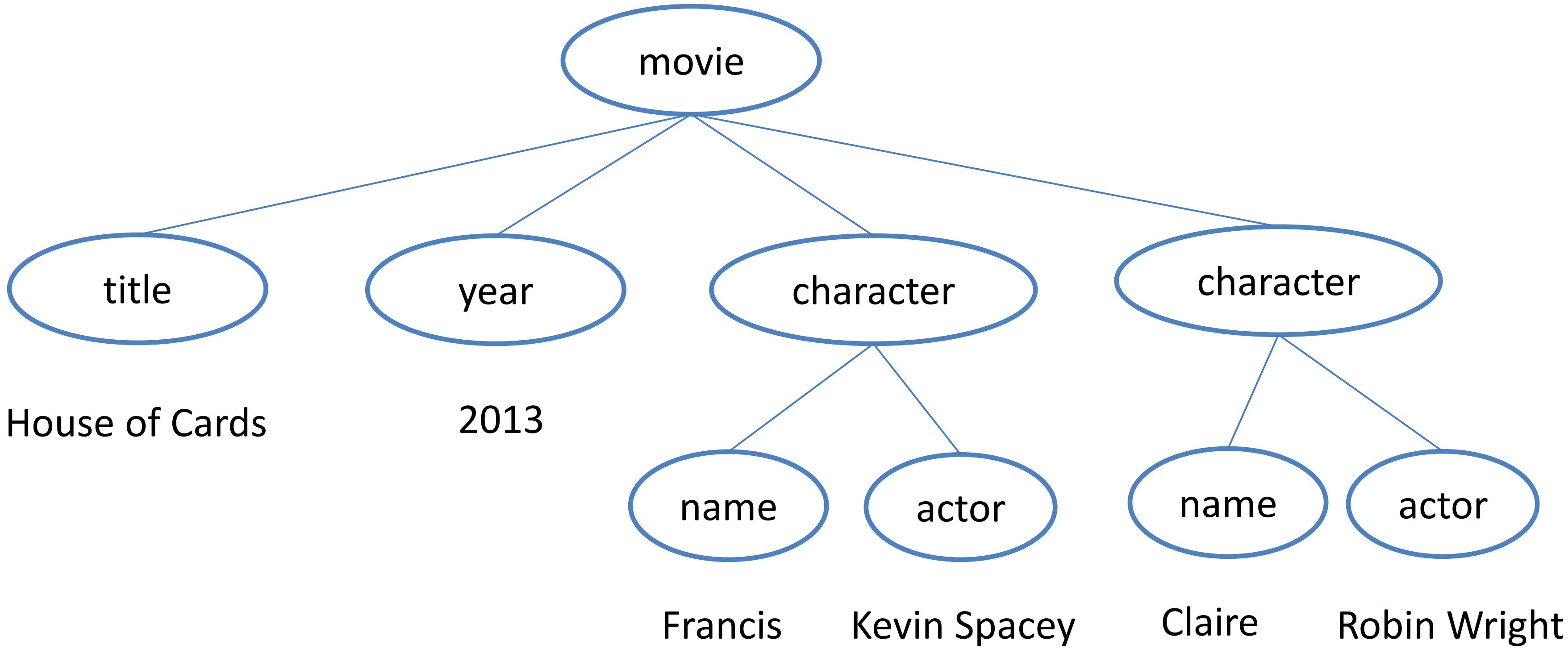
```
"movie": {
        "title": "House of cards",
        "year": "2013",
        "character": [
                {
                  "name": "Francis",
                  "actor": "Kevin Spacey"
                },
                {
                  "name": "Claire",
                  "actor": "Robin Wright"
                }
        ]
}
```

JSON (Multi TB) success stories:

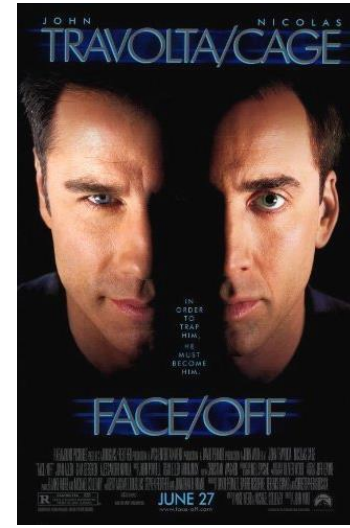
1. CouchDB
2. MongoDB
3. Jaql and Hive JSON SerDe for Hadoop

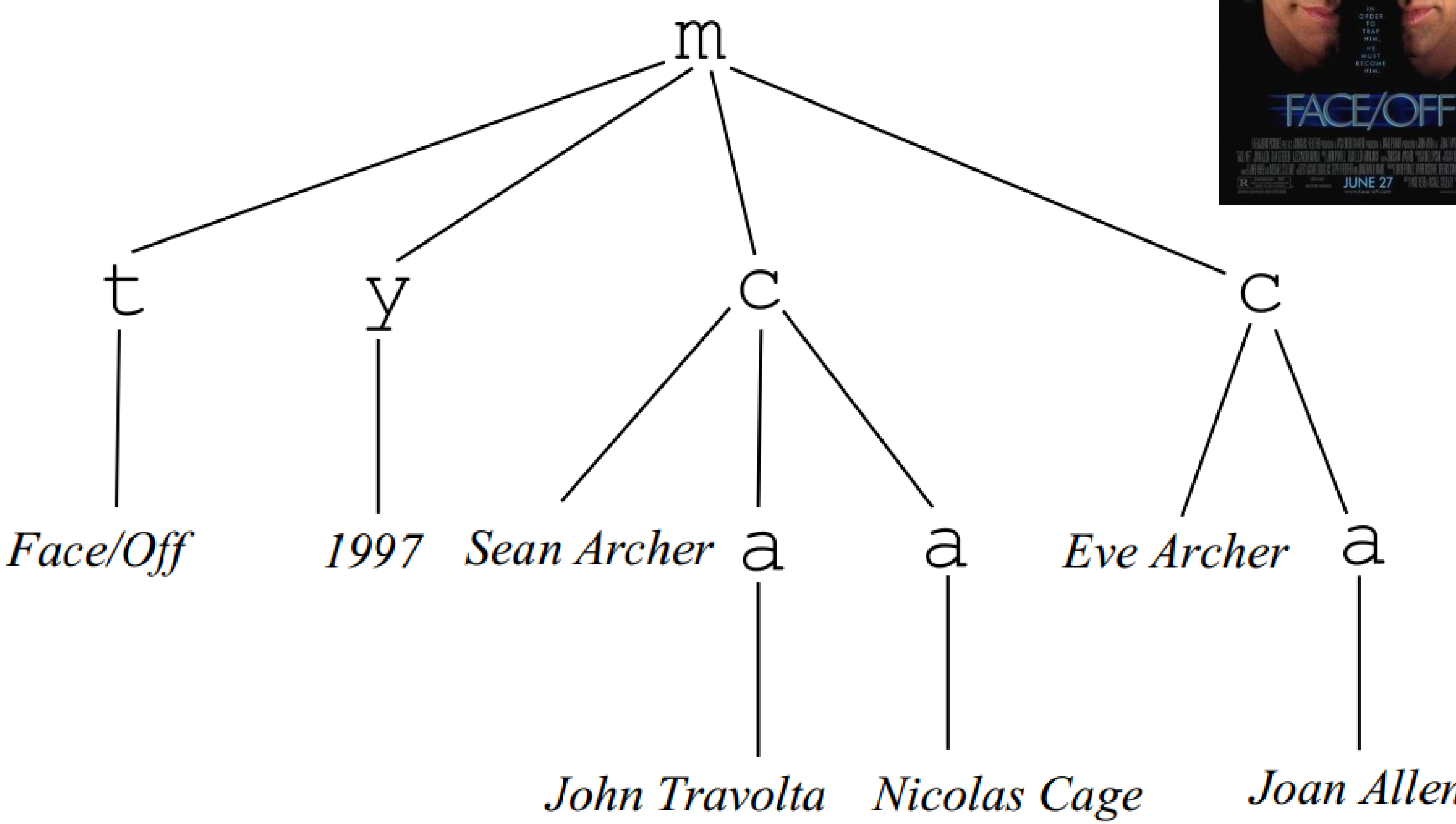
Mantra:
"Log first, ask questions later"

Trees visually



Another example





Importance of views (example)

- Big database of movies in a super-tree,
  - each movie being a sub-tree
- Query asks for all the movie sub-trees with a MAC.
  - small minority; number about 50.
  - Result materialized into a view.
- Tremendous help in answering new queries, e.g.
  - “find actors playing a MAC”.
  - Rewrite into: “find actors playing a MAC in a movie having a MAC”
  - answer it on the materialized view.

Regular Expressions and Automata

- Return all movie actors

$$\_{*}m\_{*}c\hat{a}$$

- Automaton

$$s_0 \xrightarrow{\Sigma} s_0$$

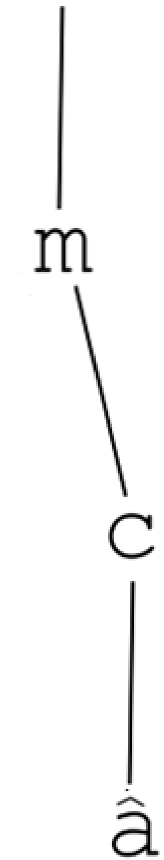
$$s_0 \xrightarrow{m} s_1$$

$$s_1 \xrightarrow{\Sigma} s_1$$

$$s_1 \xrightarrow{c} s_2$$

$$s_2 \xrightarrow{\hat{a}} s_3$$

A pattern

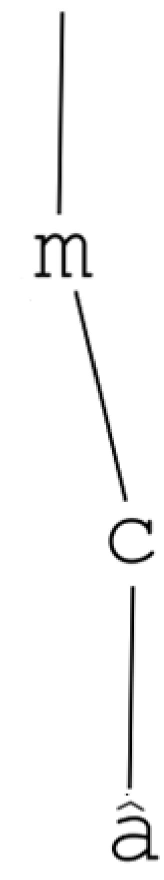


Reverse

- Return all movie actors

$$\_*m\_*c\hat{a}$$

A pattern



- Automaton

$$s \xrightarrow{\hat{a}} s_a$$

$$s_a \xrightarrow{c} s_c$$

$$s_c \xrightarrow{\Sigma} s_c$$

$$s_c \xrightarrow{m} s_m$$

$$s_m \xrightarrow{\Sigma} s_m$$

Bottom-up Tree Automata

- Return all movie actors

$$\_{*}m\_{*}c\hat{a}$$

- Automaton

$$s \xrightarrow{\hat{a}} s_a \qquad s \xrightarrow{\Sigma} s$$

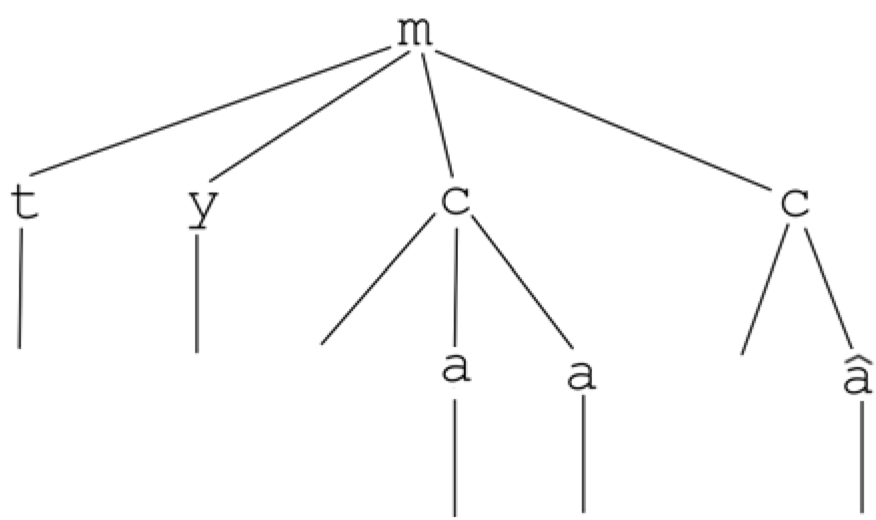
$$s{*}s_a s{*} \xrightarrow{c} s_c$$

$$s{*}s_c s{*} \xrightarrow{\Sigma} s_c$$

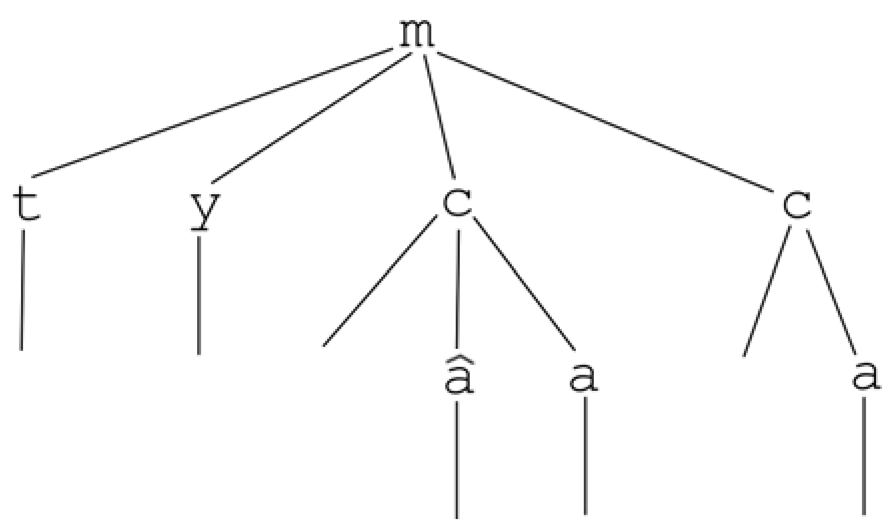
$$s{*}s_c s{*} \xrightarrow{m} s_m$$

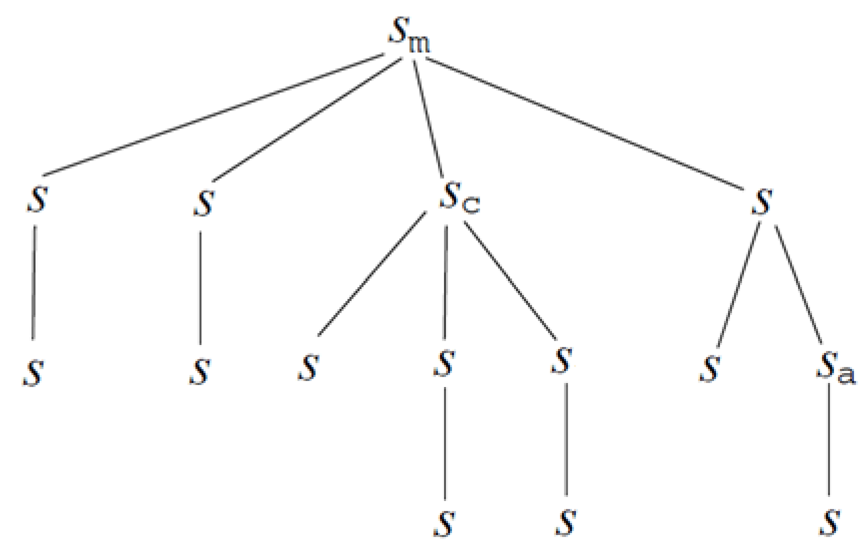
$$s{*}s_m s{*} \xrightarrow{\Sigma} s_m$$

A pattern



Run





Bottom-up Tree Automata (II)

- Return all movie actors of MACs

$$\_ * m \_ * c_{\mathrm{MAC}} \hat{a}$$

- Automaton

$$s \xrightarrow{\hat{a}} s_{\hat{a}} \qquad s \xrightarrow{a} s_a \qquad s \xrightarrow{\Sigma} s$$

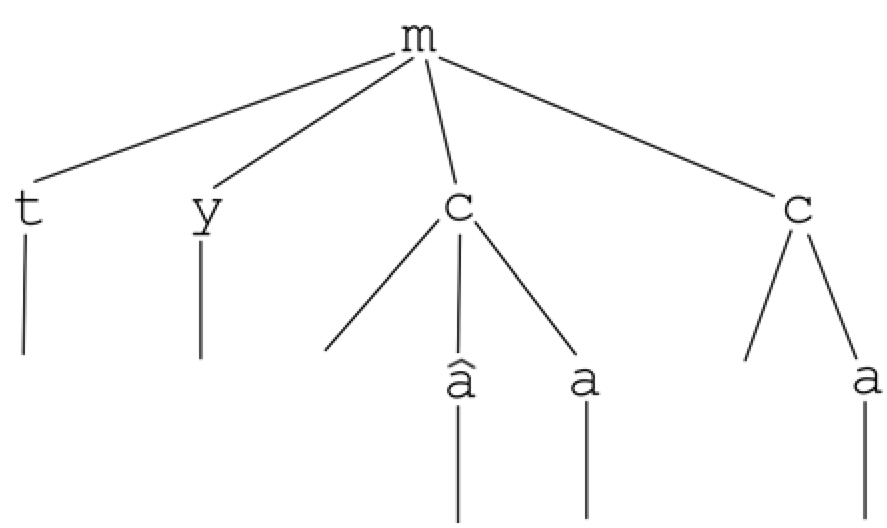
$$s*\left(s_{\hat{a}} s_a \mid s_a s_{\hat{a}}\right) s* \xrightarrow{c} s_c$$

$$s* s_c s* \xrightarrow{\Sigma} s_c$$

$$s* s_c s* \xrightarrow{m} s_m$$

$$s* s_m s* \xrightarrow{\Sigma} s_m$$

A pattern



Bottom-up Tree Automata (IV)

- Return all movies having some MACs

$$\_ * \hat{m} \_ * c_{\text{MAC}} a$$

- Automaton

$$s \xrightarrow{a} s_a \qquad s \xrightarrow{\Sigma} s$$

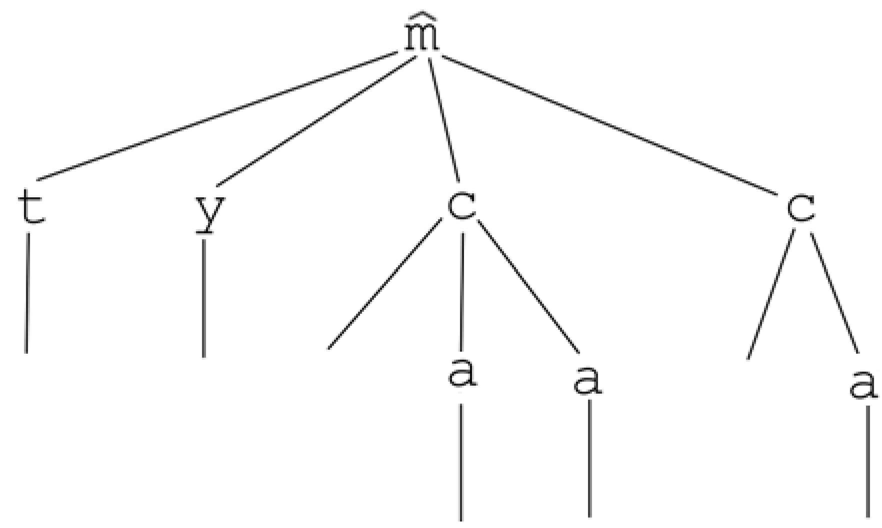
$$s^* s_a s_a s^* \xrightarrow{c} s_c$$

$$s^* s_c s^* \xrightarrow{\Sigma} s_c$$

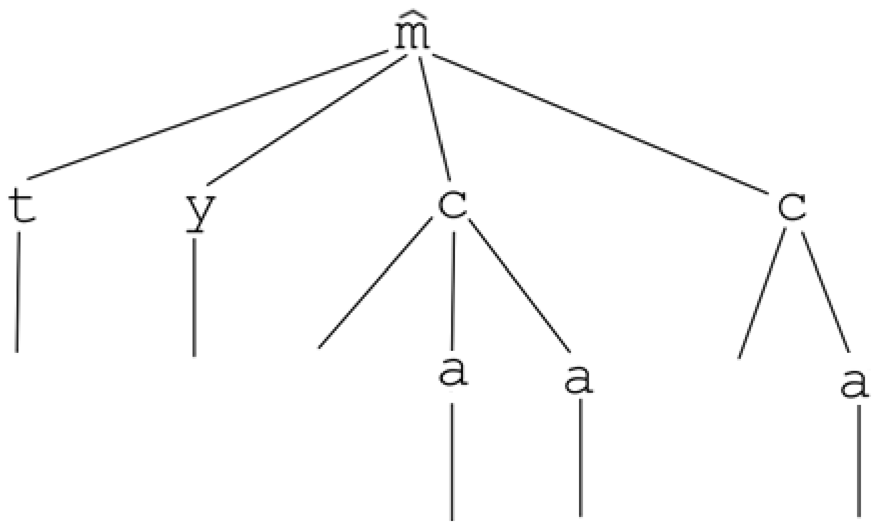
$$s^* s_c s^* \xrightarrow{\hat{m}} s_m$$

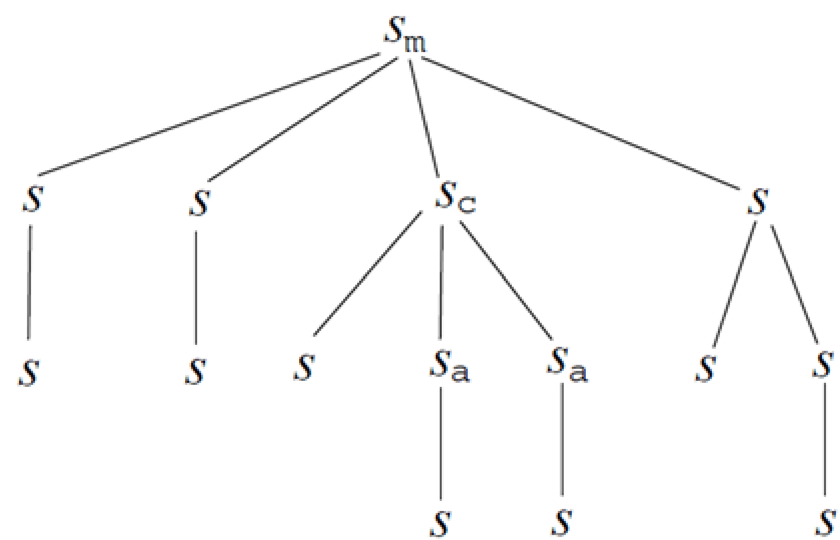
$$s^* s_m s^* \xrightarrow{\Sigma} s_m$$

A pattern



Run





Bottom-up Tree Automata (V)

- Regular tree languages (RTAs)
  - the sets of trees recognized by TAs.
  - closed under intersection and complement
- Deterministic TA
  - For any tree t, there can be at most one accepting run of A on t.
  - Power-wise, TA = DTA.
- Complement obtained from deterministic TA
- Intersection via a special construction preserves determinism.

Queries

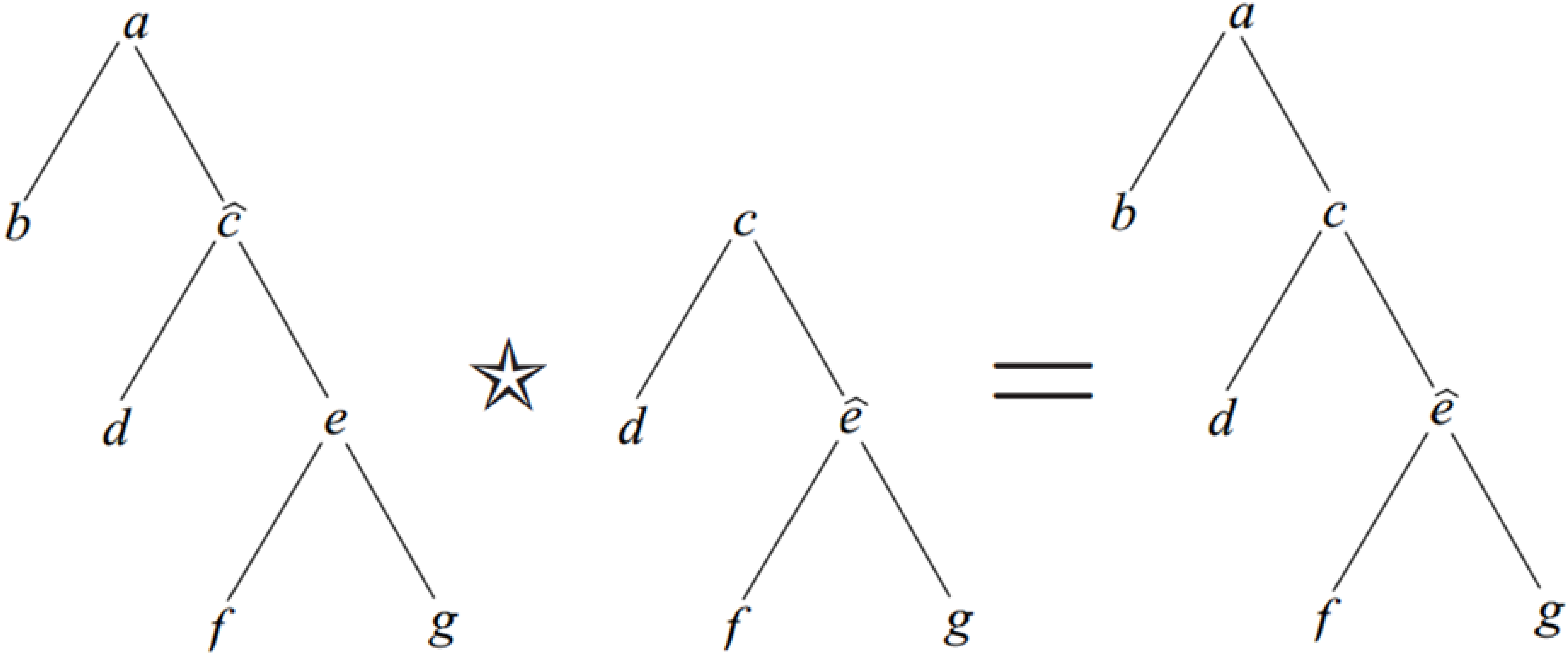
- Queries are regular sets of trees over

$$\Sigma \cup \hat{\Sigma}$$

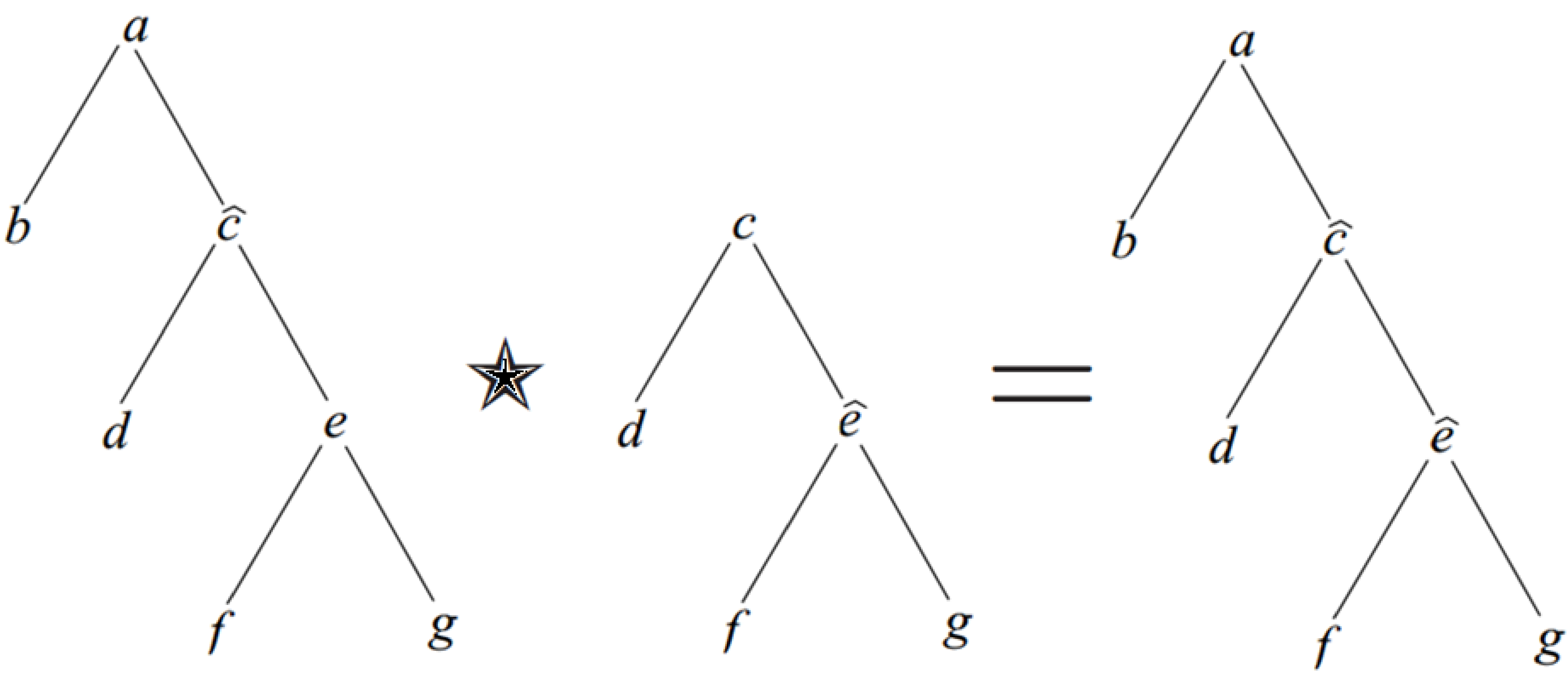
- Containment Lemma

$$Q_1 \subseteq Q_2 \quad \text{implies} \quad \text{ans}(Q_1) \subseteq \text{ans}(Q_2)$$

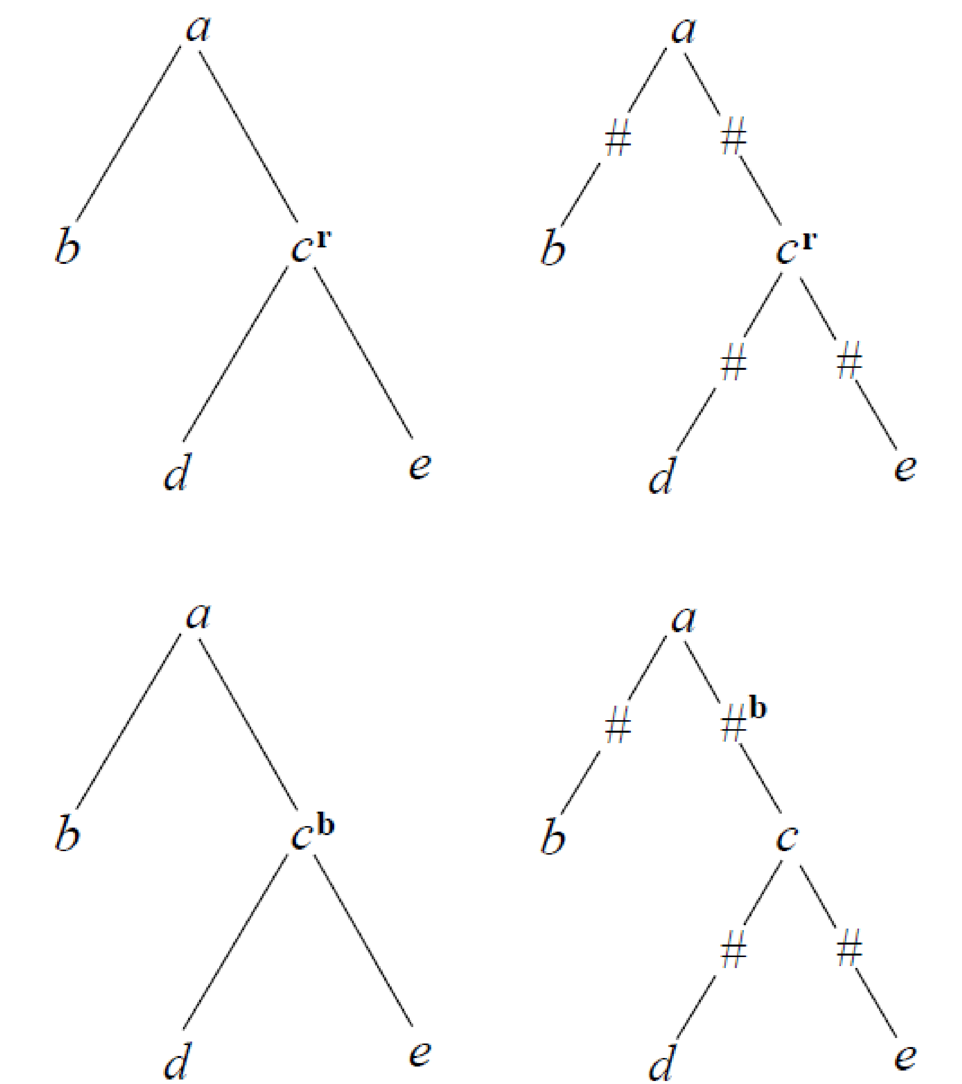
Star Operation



Filled Star Operation



Transformation for avoiding marker overlap



Rewriting, and two sets

Maximally contained rewriting:

$$R = \{\xi \in \Upsilon^{\mathbf{x}} : V \star \xi \subseteq Q\}$$

The bad set:

$$X = \{\xi \in \Upsilon^{\mathbf{x}} : \text{ there exists } \upsilon \in V \text{ such that } \upsilon \star \xi \in Q^c\}$$

The promising set:

$$Y = \{\xi \in \Upsilon^{\mathbf{x}} : \text{ there exists } \upsilon \in V \text{ such that } \upsilon \star \xi \in Q\}$$

Proposition. $R = Y \setminus X$

Example with chains

$$Q = \{aac\hat{d}e,\ bbc\hat{d}e,\ ccc\hat{d}e\}$$

$$V = \{aa\hat{c}de,\ bb\hat{c}de\}$$

$$c\hat{d}e \in R$$

$$V \star c\hat{d}e = \{aac\hat{d}e,\ bbc\hat{d}e\} \subseteq Q$$

Example with chains (II)

$$Q = \{aac\hat{d}e,\ ccc\hat{d}e\}$$

$$V = \{aa\hat{c}de,\ bb\hat{c}de\}$$

$$c\hat{d}e \notin R$$

$$bbc\hat{d}e \in V \star c\hat{d}e \qquad bbc\hat{d}e \notin Q$$

$$c\hat{d}e \in X \qquad c\hat{d}e \in Y$$

Inverse of the star operation

$$v \circledast v' = \begin{cases} \xi & \text{if } v' = v \star \xi \\ \text{undefined} & \text{otherwise} \end{cases}$$

Proposition.

$$X = V \circledast Q^c \qquad\qquad Y = V \circledast Q$$

Compute $\quad K = J \circledast J' \quad$ where *J* and *J’* are RTQ

Colored Alphabets

- Markers will be colors
  - Blue for *J*
  - Red for *J'*

$$\Sigma^{\mathbf{r}} = \{a^{\mathbf{r}} : a \in \Sigma\}$$

$$\Sigma^{\mathbf{b}} = \{a^{\mathbf{b}} : a \in \Sigma\}$$

Colored Languages

$\Upsilon^{\mathbf{b}}$ set of all trees having one node blue

$\Upsilon^{\mathbf{r}}$ set of all trees having one node red

$\Upsilon^{\mathbf{b},\mathbf{r}}$ set of all trees having one node blue and another red as descendant of the blue node

$\Phi^{\mathbf{r}}$ set of all trees having all nodes black, except root which is red

$\Phi^{\mathbf{b},\mathbf{r}}$ set of all trees having all nodes black, except for the root which is blue and another node which is red.

Colored Languages (II)

$J \subseteq \Upsilon^{\mathbf{b}}$

$J' \subseteq \Upsilon^{\mathbf{r}}$

$K \subseteq \Upsilon^{\mathbf{r}}$ $\quad K \subseteq \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$

$J \star K \subseteq \Upsilon^{\mathbf{r}}$ $\quad J \bigstar K \subseteq \Upsilon^{\mathbf{b},\mathbf{r}}$

Colored Languages (III)

p over $\Sigma \cup \Sigma^{\mathbf{r}} \cup \Sigma^{\mathbf{b}}$

$\mathsf{p}^{\neg\mathbf{b}}$ same as p, but with blue nodes turned black

$\mathsf{p}^{\neg\mathbf{r}}$ same as p, but with red nodes turned black

L over $\Sigma \cup \Sigma^{\mathbf{r}} \cup \Sigma^{\mathbf{b}}$

$$L^{\neg\mathbf{b}} = \{\mathsf{p}^{\neg\mathbf{b}} : \mathsf{p} \in L\}$$

$$L^{\neg\mathbf{r}} = \{\mathsf{p}^{\neg\mathbf{r}} : \mathsf{p} \in L\}$$

Colored Languages (IV)

$$B_L = \{\mathsf{p} \in \Upsilon^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{b}} \in L\} \text{ for } L \subseteq \Upsilon^{\mathbf{r}}$$

$\mathcal{A} = (S, \Sigma \cup \Sigma^{\mathbf{r}}, F, \Delta)$ automaton for L

$$\mathcal{B} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F, \Delta_{\mathcal{B}})$$

$$\Delta_{\mathcal{B}} = \Delta \cup \{H \xrightarrow{a^{\mathbf{b}}} s : H \xrightarrow{a} s \text{ in } \Delta \text{ and } a \in \Sigma\}$$

$$L(\mathcal{B}) \cap \Upsilon^{\mathbf{b},\mathbf{r}} = B_L$$

Similarly:

$$B'_L = \{\mathsf{p} \in \Upsilon^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{r}} \in L\} \text{ for } L \subseteq \Upsilon^{\mathbf{b}}$$

$$C_L = \{\mathsf{p} \in \Phi^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{b}} \in L\} \text{ for } L \subseteq \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$$

Rewriting Algorithm $K = J \varostar J'$

Compute: B_J $B'_{J'}$

$B_J \cap B'_{J'}$

$\mathcal{D} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F, \Delta)$ $L(\mathcal{D}) \subseteq \Upsilon^{\mathbf{b},\mathbf{r}}$

$\mathcal{E} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F_{\mathcal{E}}, \Delta)$ $L(\mathcal{E}) \subseteq \Phi^{\mathbf{b},\mathbf{r}}$

$$F_{\mathcal{E}} = \{s \in S : \text{ there exists } H \xrightarrow{a^{\mathbf{b}}} s \text{ in } \Delta\}$$

Theorem. $L(\mathcal{E}) = C_K$ $C_K^{\neg\mathbf{b}} = K$

Rewriting Algorithm (II)

$X = V \circledast Q^c$ $\qquad$ $Y = V \circledast Q$

$R = Y \setminus X$

Complexity

- **Proposition**. $K = J \circledast J'$ can be computed in polynomial time.
- **Theorem**. The MCR of *Q* using *V* can be computed in exponential time.
- **Theorem**. Computing the MCR of *Q* using *V* is EXPTIME-hard.

Final Notes

- Query automata formalism used is equivalent in power to MSO (golden standard)
  - For specifying node-selecting queries.
  - Colors correspond to Boolean markings
    - J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. DBPL, 2005

- XPath rewriting is NP-hard.
- XPath is a subclass of our formalism.
  - Our automata-based algorithm can be used as well for rewriting XPath queries.

K-ary queries

- **Example**: Find the 2-forests of actor tree pairs for actors who have played the same character together in some movie.

- Automaton

$$s \xrightarrow{\hat{a}} s_{\hat{a}} \qquad s \xrightarrow{\Sigma} s$$

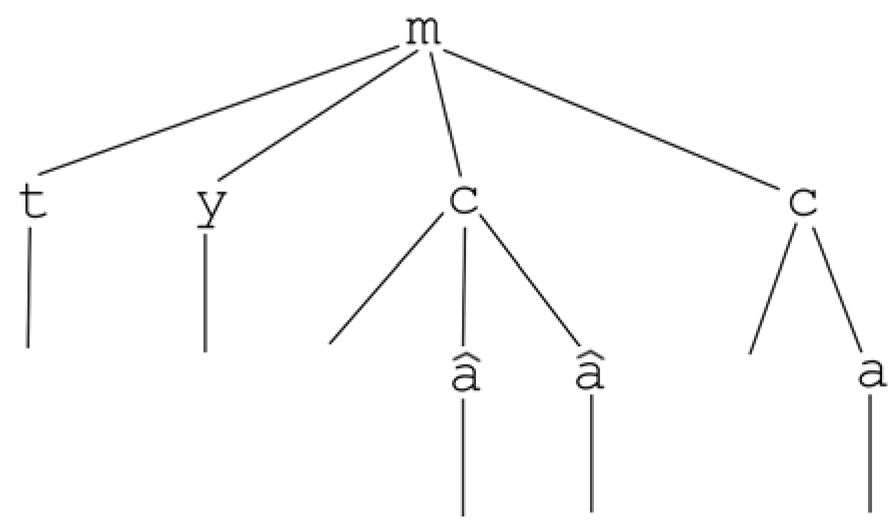
$$s^* s_{\hat{a}} s_{\hat{a}} s^* \xrightarrow{c} s_c$$

$$s^* s_c s^* \xrightarrow{\Sigma} s_c$$

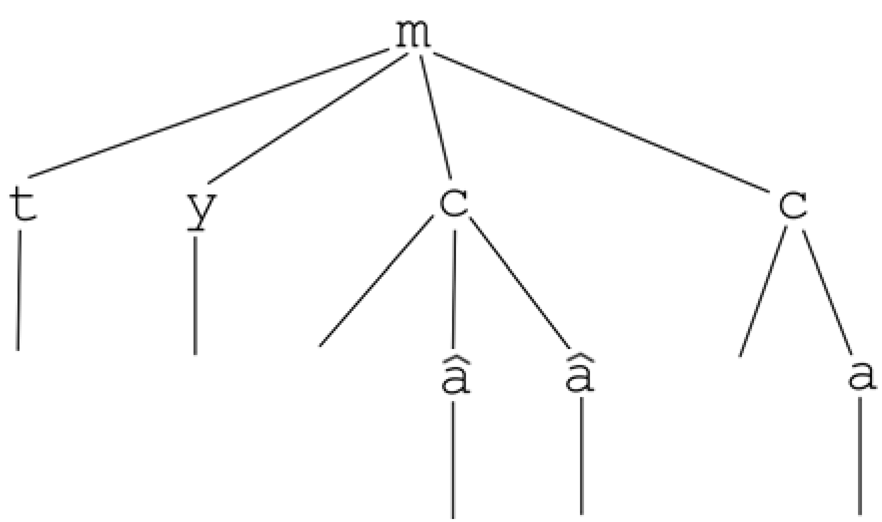
$$s^* s_c s^* \xrightarrow{m} s_m$$

$$s^* s_m s^* \xrightarrow{\Sigma} s_m$$

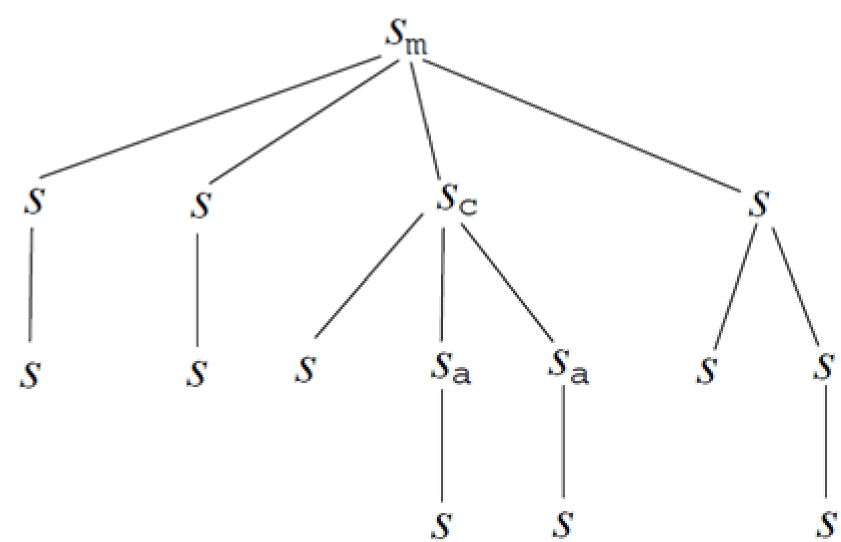
A pattern



Run



m
t
y
c
c
â
â
a



$s_{\mathtt{m}}$
s
s
$s_{\mathtt{c}}$
s
s
s
s
$s_{\mathtt{a}}$
$s_{\mathtt{a}}$
s
s
s
s
s

Why is rewriting K-ary queries challenging

- It has been shown that k-ary queries can be encoded by unary queries
  - T. Schwentick. On diving in trees. In MFCS, 2000.
  - Done by going through MSO formulas.
  - Going from a k-ary query to an MSO encoding and then back to automata incurs non elementary complexity.

- Therefore we need a another algorithm for rewriting k-ary queries
  - that doen't go via MSO formulas

Conclusions

- Characterized view-based rewriting as solving a lang. equation
  - Defined appropriate tree operators
- Defined colored languages
  - Gave automata constructions
- Computed rewriting as a series of operations on automata
- Characterized the complexity of computing rewriting
  - Tight lower bound provided
- Extended the results to k-ary queries
  - Common in XQuery

Thank You