

Interactive Learning of Node Selection Queries in Web Documents

J. Carme, R. Gilleron, **A. Lemay**, J. Niehren

INRIA FUTURS, University of Lille 3

Web Information Extraction

Search Results for: *bruss*

Found 36 total matches in 1.32067 seconds.

Score: 1

Name: [Ingo Bruss](#)

First Entered: 03/06/96

Email: bruss@wbkst15.mach.uni-karlsruhe.de

Score: 1

Name: [Trevor Bruss](#)

First Entered: 03/06/96

Email: bruss@pa621a.inland.com

Score: 1

Name: [Trevor Bruss](#)

First Entered: 03/06/96

Email: bruss@virgo.cpe.valpo.edu

Data organisation is :

- adapted to interface with human
- unadapted to automatic extraction
- specific to each web site

Example of task :

extract e-mails from a web page

⇒ Extraction tasks are uneasy

Queries for web documents

A query :

- **Input** : HTML or XML
- **Output** : Parts of document

Several possible representations :

- **programs** (perl, ...)
- monadic queries by **path expressions** : XPath (W3C), modal logic PDL [Marx, PODS 2004]
- monadic queries in **monadic Datalog** [Gottlob & Koch PODS, LICS 2002]
- n-ary queries in **MSO** over trees [Thatcher & Wright 68]
- **tree automata** [Neven & Van de Bussche JACM 02, Neven & Schwentick TCS 02, ...]

Conception of Queries

Conception :

- **"Hand-made"** (Hard, Error-prone, need for expert)
- **With visual tools** like Lixto (Easier but still not easy) [Gottlob, Koch 2000; ...]
- **Automatically induced** (Limited expressiveness and need a lot of examples) [Freitag, Kushmerick 2000; Kosala et al 2003; Muslea et al 2001; ...]

Solution ? automatically induced using visual tools

Web documents as trees

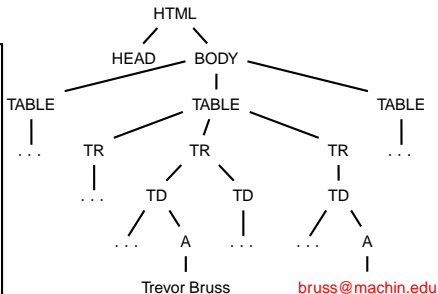
Search Results for: bruss

Found 36 total matches in 1.32067 seconds.

Score: 1
 Name: [Ingo Bruss](#) First Entered: 03/06/96
 Email: bruss@wbkst15.mach.uni-karlsruhe.de

Score: 1
 Name: [Trevor Bruss](#) First Entered: 03/06/96
 Email: bruss@pa621a.inland.com

Score: 1
 Name: [Trevor Bruss](#) First Entered: 03/06/96
 Email: bruss@virgo.cpe.valpo.edu



- Documents = Trees (no text and no attributes)
- Elements = Nodes
- Query a web page = select nodes on a tree
- Query = node selector

- 1 Demo
- 2 Learning from completely annotated documents
- 3 Learning from partially annotated examples
- 4 Experiments
- 5 Conclusion

Plan

- 1 Demo
- 2 Learning from completely annotated documents
- 3 Learning from partially annotated examples
- 4 Experiments
- 5 Conclusion

Demonstration of SQUIRREL

Demonstration

The SQUIRREL Algorithm integrated in Mozilla Firefox

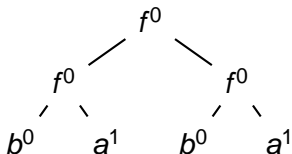
(available at www.grappa.univ-lille3.fr/~carme/squirrel)

Plan

- 1 Demo
- 2 Learning from completely annotated documents**
- 3 Learning from partially annotated examples
- 4 Experiments
- 5 Conclusion

Learning from completely annotated documents

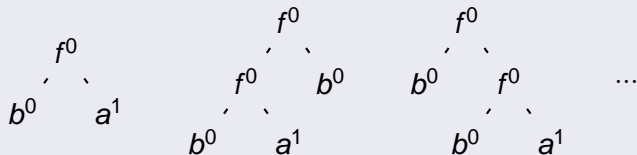
Input : Completely annotated documents (trees on $\Sigma \times \text{Bool}$)



Output : A Node Selection Query q "consistent" with the input

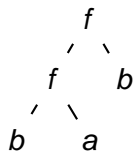
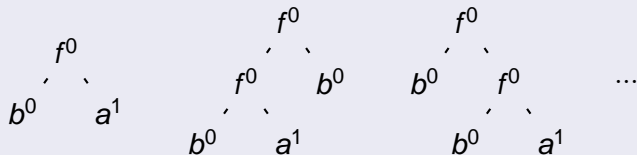
Queries as tree languages

A query : a tree language on $(\Sigma \times \text{Bool})$



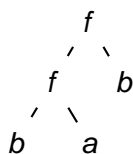
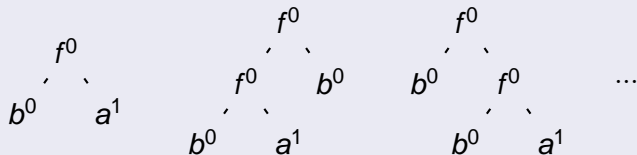
Queries as tree languages

A query : a tree language on $(\Sigma \times \text{Bool})$

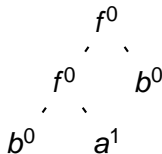


Queries as tree languages

A query : a tree language on $(\Sigma \times \text{Bool})$

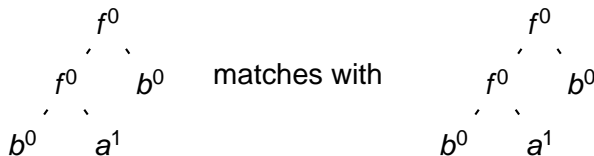
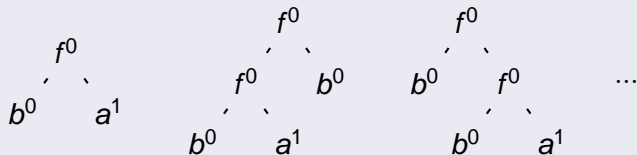


matches with



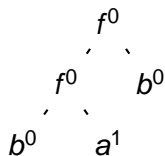
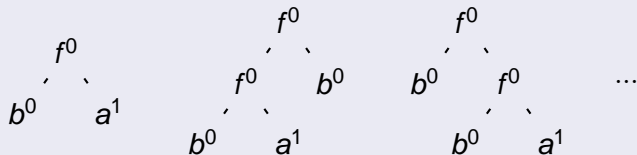
Queries as tree languages

A query : a tree language on $(\Sigma \times \text{Bool})$



Queries as tree languages

A query : a tree language on $(\Sigma \times \text{Bool})$



Tree languages representing queries are functional :
for each tree on T_Σ , only one tree on $T_{\Sigma \times \text{Bool}}$ is in the language

Node Selecting Tree Transducers

Definition

NSTT = functional tree automata on $\Sigma \times \text{Bool}$

Node Selecting Tree Transducers

Definition

NSTT = functional tree automata on $\Sigma \times \text{Bool}$

Properties

- **Efficient** : queries can be answered in polynomial time
- **Expressive** : as expressive as MSO queries
- **Adapted to learning** : Tree Automata can be learned using RPNI [Oncina, Garcia 93]

Inference Algorithm for completely annotated examples

RPNI [Oncina, Garcia 92, 93]

- **Input** : positive and negative (tree) examples
- Based on state merging methods
- Uses **consistency** checks
- **Output** : Tree automata

Inference Algorithm for completely annotated examples

RPNI [Oncina, Garcia 92, 93]

- **Input** : positive and negative (tree) examples
- Based on state merging methods
- Uses **consistency** checks
- **Output** : Tree automata

RPNI_{NSTT} [Carme, Gilleron, Lemay, Niehren 04]

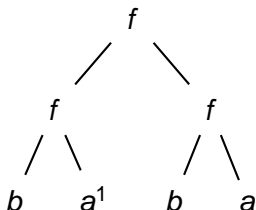
- **Input** : (positive) completely annotated examples
- Based on state merging methods
- Uses **functionality** tests
- **Output** : NSTT

Plan

- 1 Demo
- 2 Learning from completely annotated documents
- 3 Learning from partially annotated examples**
- 4 Experiments
- 5 Conclusion

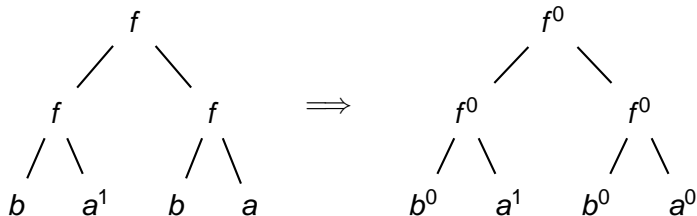
Partial annotations and pruning

Target query : Extract a -leaves after a b -leaf



Partial annotations and pruning

Target query : Extract *a*-leaves after a *b*-leaf

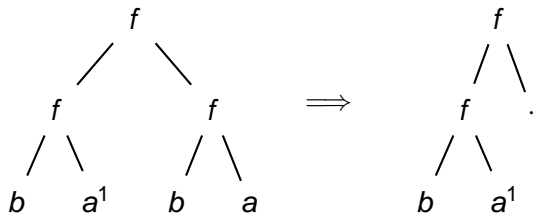


The user provides *partially* annotated examples

- Complete annotation ? : **Errors!**

Partial annotations and pruning

Target query : Extract *a*-leaves after a *b*-leaf

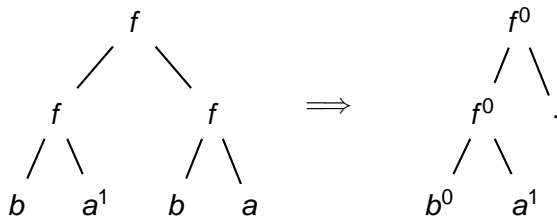


The user provides *partially* annotated examples

- Prune

Partial annotations and pruning

Target query : Extract a -leaves after a b -leaf



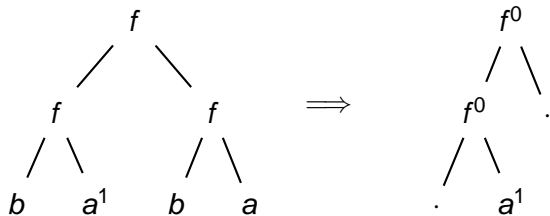
The user provides *partially* annotated examples

- Prune and complete : Ok!

We obtain a completely annotated **pruned** tree

Partial annotations and pruning

Target query : Extract a -leaves after a b -leaf

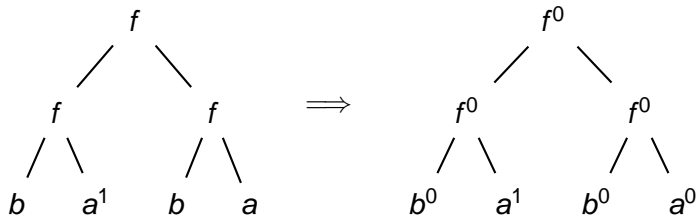


How to prune ?

- Too Strong : loss of expressibility
- Too Weak : need more annotations

Partial annotations and pruning

Target query : Extract a -leaves after a b -leaf

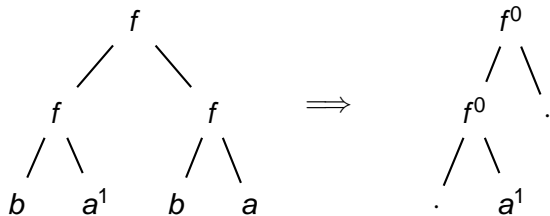


How to prune ?

- Too Strong : loss of expressibility
- Too Weak : need more annotations

Partial annotations and pruning

Target query : Extract a -leaves after a b -leaf



How to prune ?

- Too Strong : loss of expressibility
- Too Weak : need more annotations

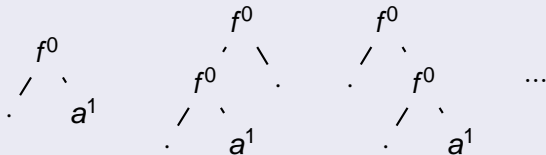
Our choice : keep the path and the position

pruning NSTT

NSTT that prunes

NSTT on $(\Sigma \times \text{Bool}) \cup \{\cdot\}$

language :

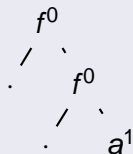
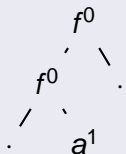


pruning NSTT

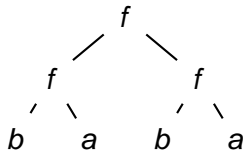
NSTT that prunes

NSTT on $(\Sigma \times \text{Bool}) \cup \{\cdot\}$

language :



...

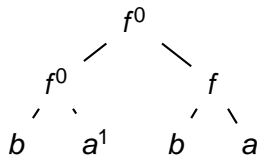
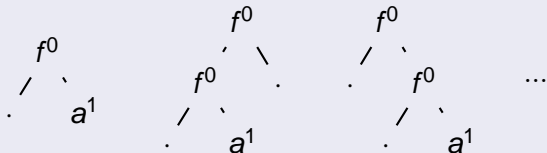


pruning NSTT

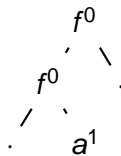
NSTT that prunes

NSTT on $(\Sigma \times \text{Bool}) \cup \{\cdot\}$

language :



matches with

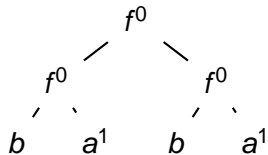
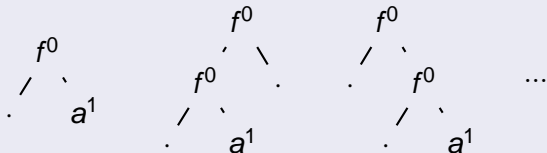


pruning NSTT

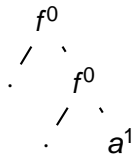
NSTT that prunes

NSTT on $(\Sigma \times \text{Bool}) \cup \{\cdot\}$

language :



matches with

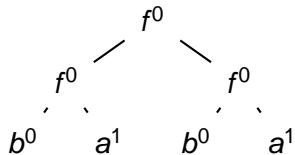
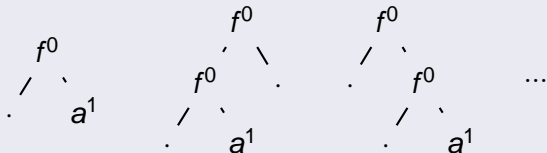


pruning NSTT

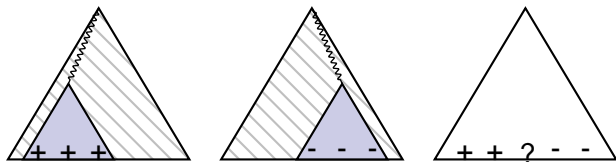
NSTT that prunes

NSTT on $(\Sigma \times \text{Bool}) \cup \{.\}$

language :

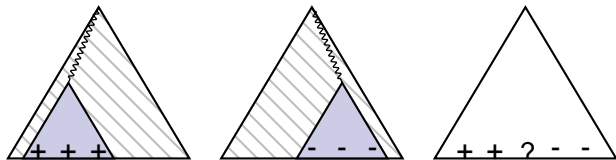


Cut-functionality



Problem : several runs on different pruning may give different results on their common parts

Cut-functionality



Problem : several runs on different pruning may give different results on their common parts

Cut-functionality

There is no inconsistency on annotation from different pruning of the same tree.

Learning pruning NSTT

pruning NSTT

pruning NSTT = cut-functional NSTT on pruned trees

$RPNI_{pNSTT}$

- **Input** : **partially** annotated examples
- Based on state merging methods
- Uses **cut-functionality** tests
- **Output** : pruning NSTT

Cut-functionality test can be done in polynomial time

Querying with pruning NSTT

Querying with a pruning NSTT

Can be done in polynomial time (equivalent to a run with a non deterministic automata)

Plan

- 1 Demo
- 2 Learning from completely annotated documents
- 3 Learning from partially annotated examples
- 4 Experiments**
- 5 Conclusion

Off-line Experiments

Tests on 26 benchmarks (adapted from Kushmerick and Muslea)

	success	partial success	failure
Squirrel	22	-	4
WIEN (Kushmerick)	16	-	10
Stalker (Muslea, Minton, Knoblock)	17	7	2

Failure when the textual information is necessary or when pruning is too strong.

Interactive Learning

	# pages	# annotations
Okra-names	1.6	3.48
Bigbook-addresses	1	3.02
Yahoo	6.18	11.36
E-bay	1.06	2.62
NYTimes	1.44	1.44
Google	1.86	4.78

Active Learning

	1	2	5	10
	random			
Yahoo	71.8	80.9	82.1	93.7
NYTimes	91.2	92.4	95.3	100
Google	98.7	99.1	99.5	99.8
	active			
Yahoo	71.8	86.1	98.4	99.4
NYTimes	91.2	100	100	100
Google	98.7	100	100	100

F-score with respect to the number of *visited* documents

Plan

- 1 Demo
- 2 Learning from completely annotated documents
- 3 Learning from partially annotated examples
- 4 Experiments
- 5 Conclusion**

Conclusions

Squirrel :

- outputs tree wrappers
- is based on grammatical inference techniques (RPNI)
- is integrated in a visual interactive environment
- builds the wrapper through simple interaction
- performances are good :
 - good expressivity
 - limited amount of interactions

Perspectives

Ways of improvement

- Better pruning techniques
- Use of texts and attributes
- N-ary queries (multi-slot)