Information Extraction by Grammatical Inference

Gunter Grieser
FG Intellektik, FB Informatik
TU Darmstadt

Overview

- Information extraction
- wrappers
    - island wrappers
- representation language
    - EFS, AEFS
    - representability
- learning
    - learning models LIM and PAC
    - learning of AEFS, of island wrappers, and of the subtasks

Computers: from toolboxes to assistents

computer as tool

does what I say

- artificial communication
- machine logic
- no world knowledge, no context

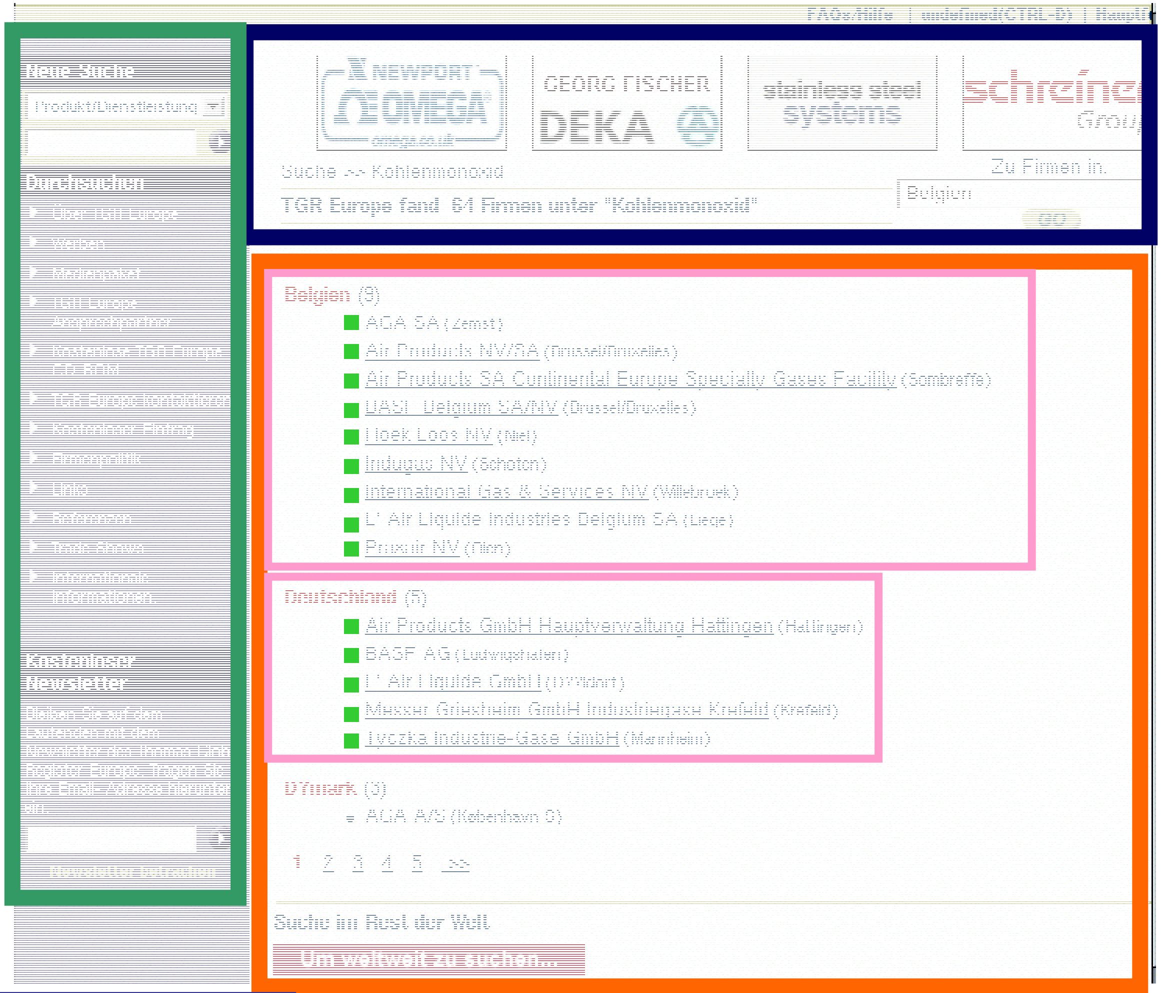
computer as assistent

does what I mean

- my communication style
- thinking amplifier
- context, world knowledge

Information Extraction by Grammatical Inference
G. Grieser

Consider information in a web page



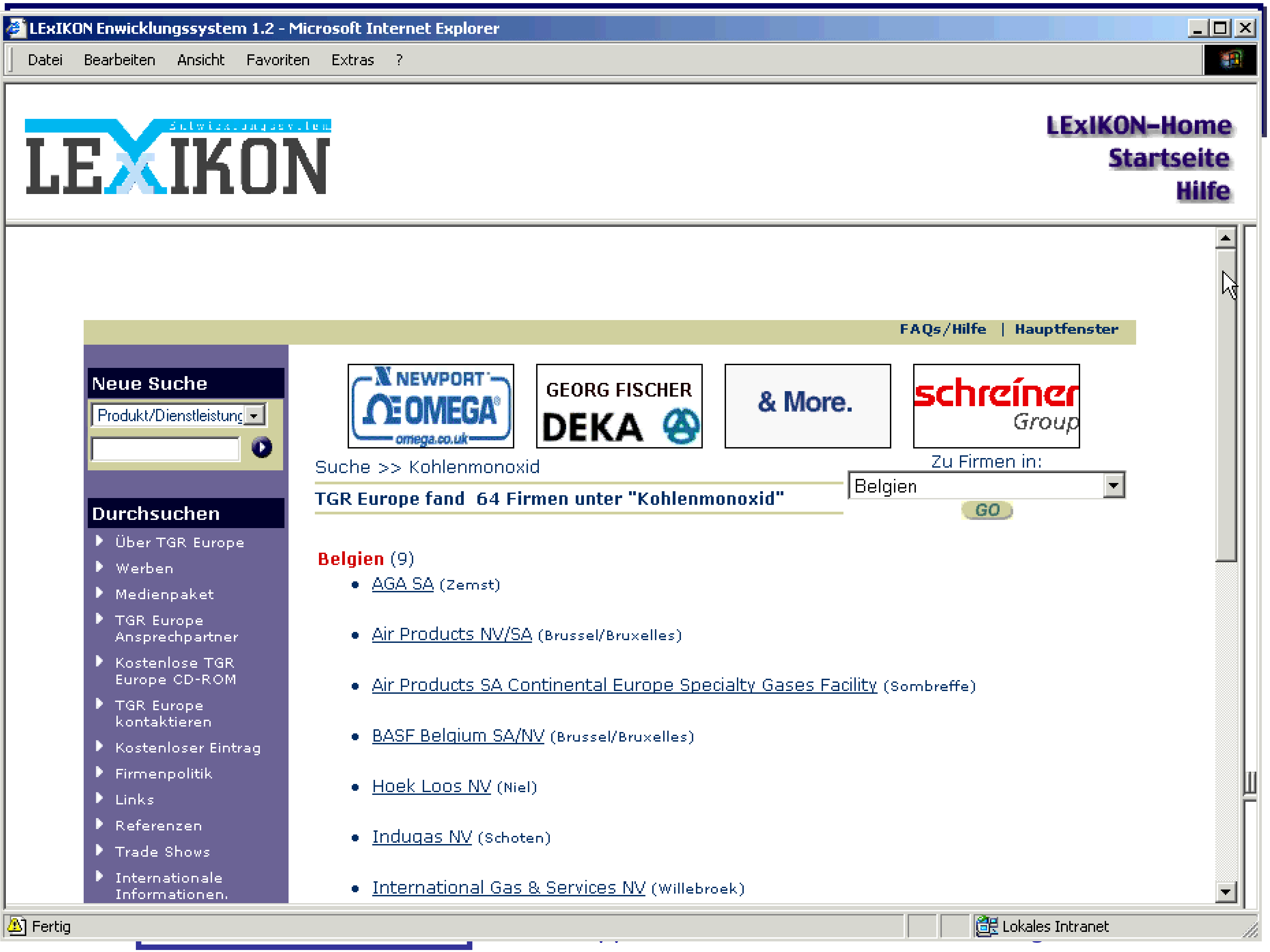
Motivation for IE

How to extract information from such documents?

there is some growing interest in powerful information extraction procedures, e.g.

> to allow for an explicit access to information that is hidden in various documents (knowledge mangement)

as a result thereof, there is some growing need for techniques that allow for an ‚interactive' creation of powerful information extraction procedures


LExIKON Enwicklungssystem 1.2 - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
LEXIKON
LExIKON-Home
Startseite
Hilfe
FAQs/Hilfe | Hauptfenster
Neue Suche
Produkt/Dienstleistung
Durchsuchen
Über TGR Europe
Werben
Medienpaket
TGR Europe Ansprechpartner
Kostenlose TGR Europe CD-ROM
TGR Europe kontaktieren
Kostenloser Eintrag
Firmenpolitik
Links
Referenzen
Trade Shows
Internationale Informationen.
NEWPORT OMEGA omega.co.uk
GEORG FISCHER DEKA
& More.
schreiner Group
Suche >> Kohlenmonoxid
TGR Europe fand 64 Firmen unter "Kohlenmonoxid"
Zu Firmen in:
Belgien
GO
Belgien (9)
AGA SA (Zemst)
Air Products NV/SA (Brussel/Bruxelles)
Air Products SA Continental Europe Specialty Gases Facility (Sombreffe)
BASF Belgium SA/NV (Brussel/Bruxelles)
Hoek Loos NV (Niel)
Indugas NV (Schoten)
International Gas & Services NV (Willebroek)
Fertig
Lokales Intranet

Information Extraction by Grammatical Inference
G. Grieser

Documents are available as source code only!

```
href="http://www.tgreurope.com/main/gotocompany/11307307302347372307350390"
                                fontsize="+1">L' Air Liquide GmbH</A><FONT

size=1> (D&uuml;sseldorf)  </FONT><BR></LI></BLOCKQUOTE></TD></
TR>
                                <TR>
                                <TD align=left>
                                <BLOCKQUOTE>
                                <LI><A

href="http://www.tgreurope.com/main/gotocompany/12309309317335386346340304"
                                fontsize="+1">Messer Griesheim GmbH
 Industriegase Krefeld</A><FONT

size=1> (Krefeld)  </FONT><BR></LI></BLOCKQUOTE></TD></TR>
                                <TR>
                                <TD align=left>
                                <BLOCKQUOTE>
                                <LI><A

href="http://www.tgreurope.com/main/gotocompany/13307300307355305339354390"
                                fontsize="+1">Tyczka Industrie-Gase
                                GmbH</A><FONT
```

IE and formal languages

- documents are strings over a certain alphabet
- information is contained in the documents

- can view documents as well as contained information as formal languages

Often, information can be identified by its context

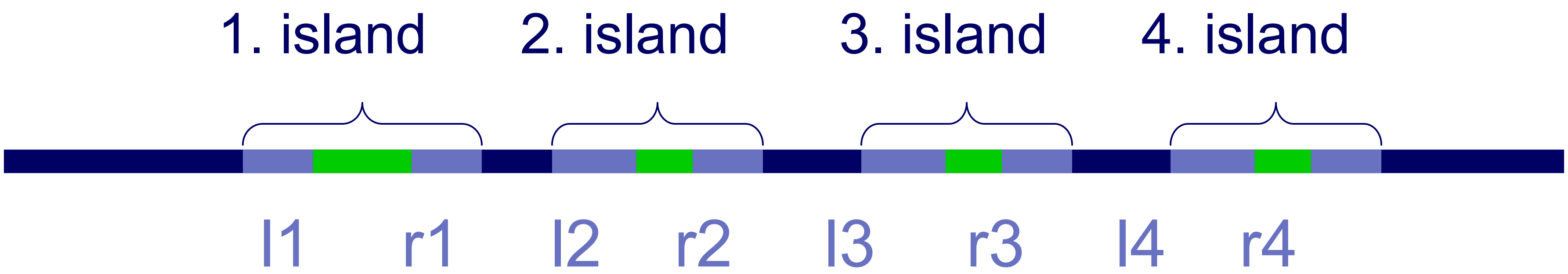
```
href="http://www.tgreurope.com/main/gotocompany/11307307302347372307350390"
                                fontsize="+1">L' Air Liquide GmbH</A><FONT

size=1> (D&uuml;sseldorf)  </FONT><BR></LI></BLOCKQUOTE></TD></
TR>
                                <TR>
                                <TD align=left>
                                <BLOCKQUOTE>
                                <LI><A

href="http://www.tgreurope.com/main/gotocompany/12309309317335386346340304"
                                fontsize="+1">Messer Griesheim GmbH
Industriegase Krefeld</A><FONT

size=1> (Krefeld)  </FONT><BR></LI></BLOCKQUOTE></TD></TR>
                                <TR>
                                <TD align=left>
                                <BLOCKQUOTE>
                                <LI><A

href="http://www.tgreurope.com/main/gotocompany/13307300307355305339354390"
                                fontsize="+1">Tyczka Industrie-Gase
                                GmbH</A><FONT
```

IE and formal languages

- documents are strings over a certain alphabet
- information is contained in the documents

- can view documents as well as contained information as well as context as formal languages

Island Wrappers



in general: delimiters not unique

$\Rightarrow$ delimiter languages

n: arity of the island wrapper

$\Rightarrow$ 2n delimiter languages: $L_1, R_1, \ldots, L_n, R_n$

island wrapper: 2n-tuple of formal languages

$(L_1, R_1, \ldots, L_n, R_n)$

Island Wrapper: definition

an island wrapper $(L_1,R_1, \ldots,L_n,R_n)$ extracts a tuple $(v_1,v_2,\ldots,v_n)$ from document d iff:

- $d = x_1 l_1 v_1 r_1 x_2 l_2 v_2 r_2 x_3 \ldots x_n l_n v_n r_n x_{n+1}$
- $x_1 \in \Sigma^* \quad x_{n+1} \in \Sigma^*$
- $l_1 \in L_1 \quad r_1 \in R_1 \quad l_2 \in L_2 \quad r_2 \in R_2 \quad \ldots \quad l_n \in L_n \quad r_n \in R_n$
- $v_1 \in \Sigma^+ \backslash (\Sigma^* R_1 \Sigma^*) \ \ldots \ v_n \in \Sigma^+ / (\Sigma^* R_n \Sigma^*)$

Island wrapper: definition

an island wrapper $(L_1,R_1, \ldots,L_n,R_n)$ extracts a tuple $(v_1,v_2,\ldots,v_n)$ from document d iff:

- $d = x_1 l_1 v_1 r_1 x_2 l_2 v_2 r_2 x_3 \ldots x_n l_n v_n r_n x_{n+1}$
- $x_1 \in \Sigma^*$ $x_{n+1} \in \Sigma^*$
- $l_1 \in L_1$ $r_1 \in R_1$ $l_2 \in L_2$ $r_2 \in R_2$ ... $l_n \in L_n$ $r_n \in R_n$
- $v_1 \in \Sigma^+ \backslash (\Sigma^* R_1 \Sigma^*)$ ... $v_n \in \Sigma^+ / (\Sigma^* R_n \Sigma^*)$
- $x_2 \in \Sigma^* \backslash (\Sigma^* L_2 \Sigma^*)$ ... $x_n \in \Sigma^* / (\Sigma^* L_n \Sigma^*)$

How to represent such wrappers?

Elementary formal systems (EFS)

```
p(baX):- p(aX).
p(bbX):- p(bX).
p(abX):- p(bX).
p(a).
p(b).
p(ab).
p(ba).
p(bb).
```

- Σ = {a,b} ... characters
- Π = {p} ... predicate symbols
- X = {X} ... variables
- patterns like baX, aX, a
- atoms like p(baX), p(aX), p(a)
- rules like p(baX) :- p(aX)., p(a).

- EFS S = (Σ,Π,Γ), where Γ is a set of rules

EFS Semantics

- relies on a well-known idea from logic programming; i.e., we focus our attention on ground atoms (g.a.)
  - for an EFS S = (Σ,Π,Γ), we let

    Sem(S) = { g.a. | g.a. holds in all Herbrand models for S }

- characterizations of Sem(S)
  - Sem(S) = { g.a. | g.a. holds in the least Herbrand model for S }
  - thus, it suffices to enumerate the g.a. that hold in a distinguished model (using a simple operator, starting with the empty set)

Advanced elementary formal systems (AEFS)

```
q(X)  :- not p(X).
p(XY):- p(X).
p(YX):- p(X).
p(aa).
```

- characters, variables, patterns, atoms ... as for EFS
- rules as for EFS and, additionally, rules like q(X) :- not p(X).
- AEFS S = (Σ,Π,Γ), where Γ is a set of rules that meet particular syntactical constraints

Why syntactical constraints at all?

if negation is allowed for, there is generally no least Herbrand model and, thus, the idea to enumerate the ground facts that hold in a distinguished model doesn‘t work

AEFS Semantics

- similarly as before, for an AEFS S = (Σ,Π,Γ), we let

  Sem(S) = { g.a. | g.a. holds in all Herbrand models for S }

- the introduced syntactical constraints on the rules in Γ guarantee that we obtain the same characterizations of Sem(S), i.e.,

  Sem(S) = { g.a. | g.a. holds in the least Herbrand model }

EFS/AEFS definable languages

```
q(X)  :- not p(X).
p(XY):- p(X).
p(YX):- p(X).
p(aa).
```

- let an AEFS S = (Σ,Π,Γ) and some distinguished predicate symbol p from Π be fixed, then

  $L(S,p) = \{ w \in \Sigma^+ \mid (w) \in Sem(S) \}$

Variable-bounded EFS/AEFS

examples:

```
q(X)  :- not p(X).
p(XY):- p(X).
p(aa).
```

- every variable in the body of a rule has to appear in the head, as well

counterexamples:

```
p(XY):- p(X), q(Y,Z).
```

Theorem:
L $\in$ L(vb-EFS) iff* L is a r.e. language.

Theorem:
There are L $\in$ L(vb-AEFS) that are not r.e.

Length-bounded EFS/AES

examples:

```
q(X)  :- not p(X).
p(XY):- p(X).
p(aa).
```

counterexamples:

```
p(XY):- p(X), q(Y,Y).
```

- variable-bounded
- if some X appears k times in the body of a rule, it must occur at least k times in its head

Theorem:
L $\in$ L(lb-EFS) iff* L is context-sensitive.

Theorem:
L $\in$ L(lb-AEFS) iff L is context-sensitive.

Regular EFS/AEFS

examples:

```
q(X)  :- not p(X).
p(XY):- p(X).
p(aa).
```

counterexamples:

```
p(XYX):- p(X).
p(XY):- q(X,Y).
```

- length-bounded
- only unary predicate symbols
- only regular patterns in the head of a rule

Theorem:
L $\in$ L(reg-EFS) iff L is context-free.

Theorem:
There are L $\in$ L(reg-AEFS) that are not context-free.

Closedness properties

Theorem:

The AEFS definable language classes L(reg-AEFS), L(lb-AEFS), and L(vb-AEFS) are closed under the operation union, intersection, and complement.

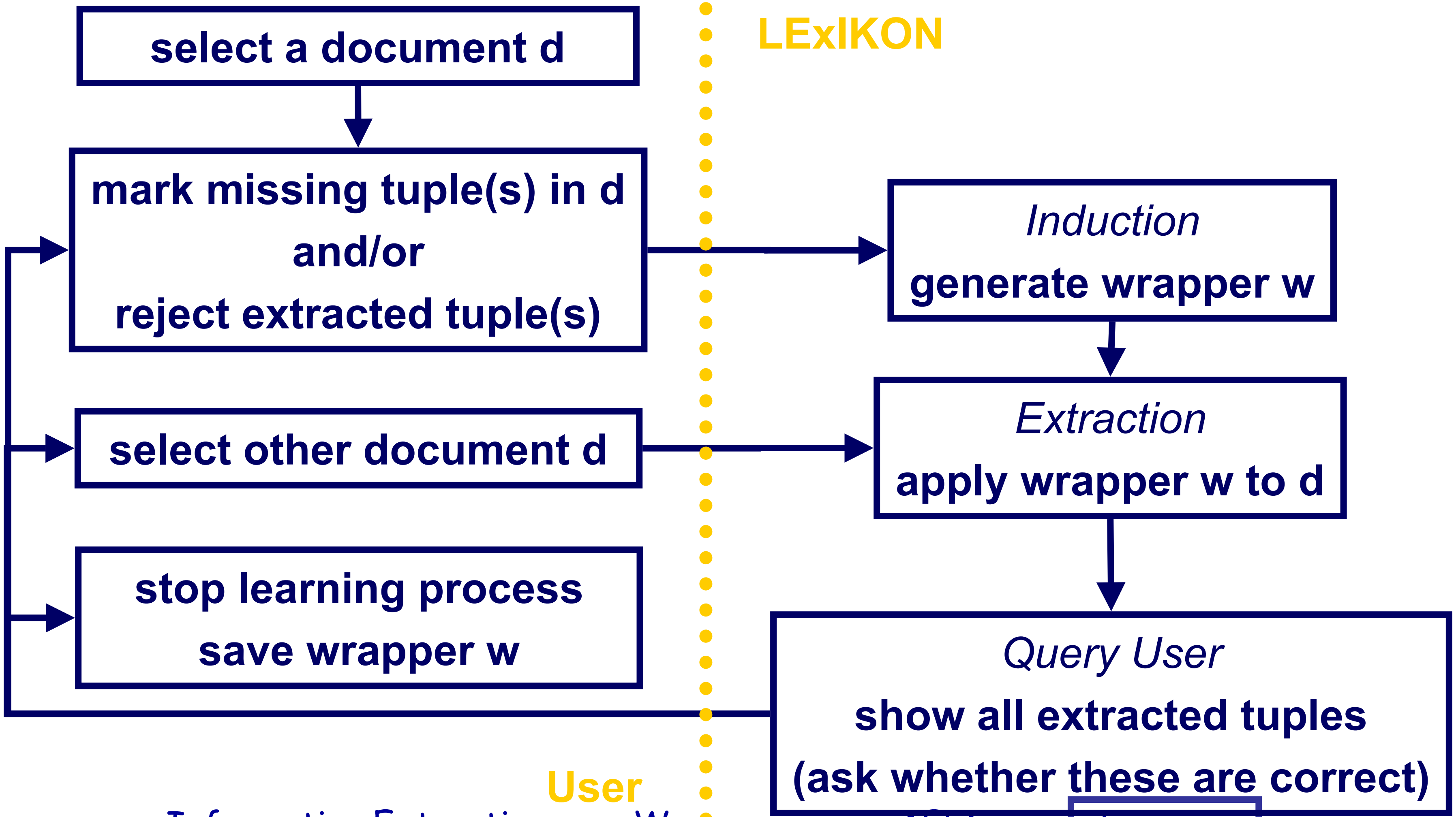
Representing island wrappers as AEFS

```
extract(V1, V2, X1L1V1R1X2L2V2R2X3):-
      l1(L1), r1(R1), l2(L2), r2(R2),
      nc-r1(V1), nc-r2(V2), nc-l2(X2).
nc-r1(X) :- not c-r1(X).
c-r1(X) :- r1(X).
c-r1(XY) :- c-r1(X).
c-r1(XY) :- c-r1(Y).
nc-r2(X) :- analogously
nc-l2(X) :- analogously
```

$l_1(X), r_1(X), l_2(X), r_2(X)$ *freely definable*

Learning

Interaction in LExIKON



Learning

- When is this interaction cycle successful?

$\rightarrow$ Learning

- 2 different models
  - learning in the limit
  - PAC learning

- learnability results for
  - representation language (AEFS)
  - island wrappers
  - composite learning tasks

Learning in the limit

- learning goal
  - a finite description of a target language L
- information available about a target language L
  - learning from positive data (text)

    sequence of words exhausting L
  - learning from positive and negative data (informant)

    sequence of labelled words that exhausts Σ^+; the words are labelled by \`+´ and \`-´ according to their membership in L
- IIM
  - receives as input finite segments of a text (an informant) and outputs a hypothesis about the target language
  - learns L in the limit iff, on every text/informant, the sequence of hypotheses stabilizes on a correct description of the target language L

Results

LimInf/LimTxt: set of all languages learnable from Informant/Text

Theorem:
L(lb-EFS) $\in$ LimInf

Theorem:
L(lb-AEFS) $\in$ LimInf

Theorem:
(i) L(lb-EFS) $\notin$ LimTxt
(ii) L(lb-EFS(k)) $\in$ LimTxt for k $\in$ N

Theorem:
(i) L(lb-AEFS) $\notin$ LimTxt
(ii) L(lb-AEFS(1)) $\in$ LimTxt
(iii) L(lb-AEFS(k)) $\notin$ LimTxt for all k > 1

Learning island wrappers

- remember:

- available information / examples:

- task: learn delimiter languages $L_1, R_1, \ldots, L_n, R_n$ from examples of form

$$\Sigma^* L_1\{\#\}\Sigma_{R_1}\{\#\}R_1\Sigma_{L_2}L_2 \ldots L_n\{\#\}\Sigma_{R_n}\{\#\}R_n\Sigma^*$$

where $\Sigma_L=\Sigma^*\backslash(\Sigma^* L\Sigma^*)$

Results

IW(L): set of all island wrappers with delimiter languages from L

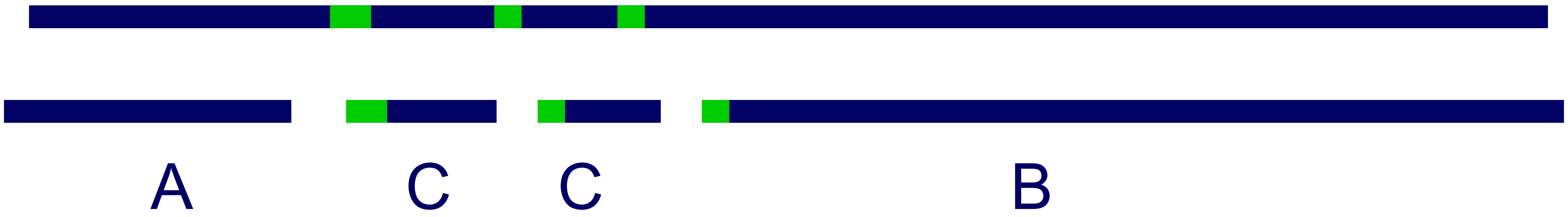
Theorem:
$IW(\wp(\Sigma^*)) \in LimInf$

Theorem:
$IW(\wp(\Sigma^*)) \notin LimTxt$

Theorem:
$IW(\wp(\Sigma^k)) \in LimTxt$ for $k \in N$

Subtasks when learning island wrappers

- problem A: learn L_1 from $\Sigma^* L_1$
- problem B: learn R_n from $\Sigma_{R_n}\{\#\}R_n\Sigma^*$
- problem C: learn R_m and L_{m+1} from $\Sigma_{R_m}\{\#\}R_m\Sigma_{L_{m+1}}L_{m+1}$



- problem D: learn delimiter languages from standard information (reference problem)

Results

Theorem:

The learning problems A, B, C, and D are incomparable.

example

- $\Sigma=\{a,b,c\}$
- $L_0=\{a^m b \mid m \geq 1\} \cup \{c\}$
- $L_{n+1}=\{a^m b \mid 1 \leq m \leq n+1\} \cup \{c, ca\}$

problem A (learn L from $\Sigma^* L$) solvable

M: on input $w_0,\ldots,w_m$ check whether some string ends with a. If no such string occurs, output a description for $\Sigma^* L_0$, otherwise for $\Sigma^* L_1$

problem B (learn R from $\Sigma_R \{\#\} R \Sigma^*$) not solvable

PAC learning

- learning goal
  - finite description that approximates L sufficiently well
- learning algorithm
  - receives a finite set of positive and negative examples and computes a hypothesis about the target language L
- C is polynomial-time PAC-learnable iff

  there exists a learning algorithm A such that given ε, $\delta \in [0,1]$,

  $n \in N$, and any probability distribution Pr over Σ^n

  - A takes $q(1/\varepsilon, 1/\delta, n, s)$ examples randomly generated with respect to Pr and outputs, in polynomial time, a hypothesis h such that, with probability $1 - \delta$, $Pr(w \in L \Delta h) < \varepsilon$

  here, s denotes the size of the smallest description of L

Hereditary EFS/AEFS

examples:

```
q(X)  :- not p(X).
p(abXaY):- p(bX), q(Y).
```

- every pattern in the body of a rule is a subword of a pattern in its head

counterexamples:

```
p(aXbY):- p(aaX).
```

- h-(A)EFS(m,k,t,r) - set of all hereditary (A)EFS with
  - at most m rules
  - at most k variables occurences in head of every rule
  - at most t atoms in the body of every rule
  - arity of each predicate symbol at most r

Results

Theorem:
For all m,k,t,r $\in$ N, L(h-EFS(m,k,t,r)) is polynomial time PAC learnable.

Theorem:
For all m,k,t,r $\in$ N, L(h-AEFS(m,k,t,r)) is polynomial time PAC learnable.

Note, that already L(h-AEFS(2,1,1,1)) \ L(h-EFS) $\neq \emptyset$.

Corollary:
If L is polynomial time PAC learnable then also IW(L) is polynomial time PAC learnable.

Overview

- Information extraction
- wrappers
  - island wrappers
- representation language
  - EFS, AEFS
  - representability
- learning
  - learning models LIM and PAC
  - learning of AEFS, of island wrappers, and of the subtasks