Theorie des Algorithmischen Lernens
Sommersemester 2006

Teil 5: Informationsextraktion

Version 1.1

Gliederung der LV

Teil 1: Motivation

1. Was ist Lernen
2. Das Szenario der Induktiven Inf erenz
3. Natürlichkeitsanforderungen

Teil 2: Lernen formaler Sprachen

1. Grundlegende Begriffe und Erkennungstypen
2. Die Rolle des Hypothesenraums
3. Lernen von Patternsprachen
4. Inkrementelles Lernen

Teil 3: Lernen endlicher Automaten

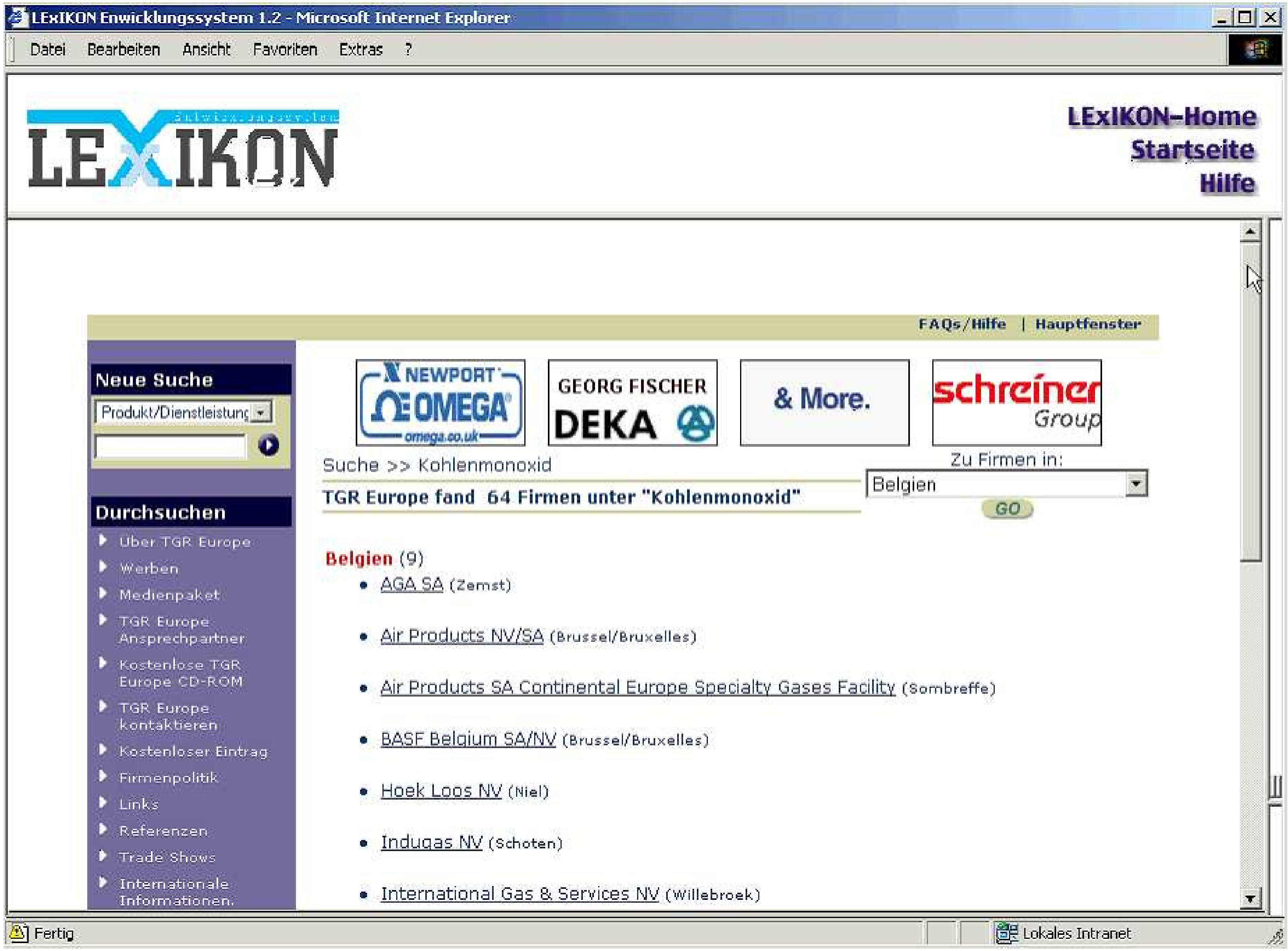
Teil 4: Lernen berechenbarer Funktionen

1. Grundlegende Begriffe und Erkennungstypen
2. Reflexion

Teil 5: Informationsextraktion

1. Island Wrappers
2. Query Scenarios

Information is embedded in structure



Gliederung der LV

Sometimes we can recognize the content 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
```

Sometimes we can recognize the content by its context

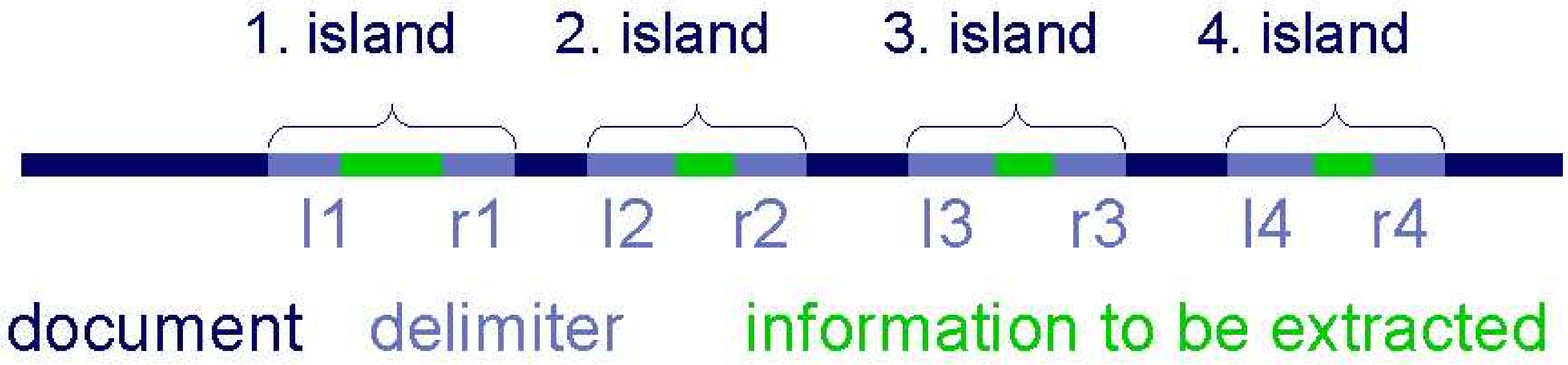
```
href="http://www.tgreurope.com/main/gotocompany/11307307302347372307350390"
                                   fontsize="+1">L' Air Liquide GmbH</A><FONT

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
  - the context

  as formal languages

Island Wrappers



- in general: delimiters not unique
  $\rightsquigarrow$ *Delimiter Languages*
- n: arity of the island wrapper
  $\rightsquigarrow$ $2n$ delimiter languages

Definition 5.1:
An ***island wrapper*** of arity n is a $2n$ tupel of formal languages $(L_1, R_1, \ldots, L_n, R_n)$.

Formal Model

Definition 5.2: $\Sigma_L^* = \Sigma^* \setminus (\Sigma^* \circ L \circ \Sigma^*)$ $\quad$ $\Sigma_L^+ = \Sigma_L^* \setminus \{\varepsilon\}$.

Definition 5.3:
Let $n \geq 1$, let $L_1, R_1, \ldots, L_n, R_n$ be delimiter languages, and let $W = (L_1, R_1, \ldots L_n, R_n)$ be the corresponding island wrapper.
Then, the island wrapper W defines the following mapping $\boldsymbol{S_W}$ from documents to n-ary relations:
Given any document d, we let $S_W(d)$ be the set of all n-tuples $\langle v_1, \ldots, v_n \rangle \in (\Sigma^+)^n$ for which there are

- $x_0 \in \Sigma^*, \ldots, x_n \in \Sigma^*$,
- $l_1 \in L_1, \ldots, l_n \in L_n$ and $r_1 \in R_1, \ldots, r_n \in R_n$

such that:

1. $d = x_0 l_1 v_1 r_1 \ldots l_n v_n r_n x_n$.
2. for all $i \in \{1, \ldots, n\}$, v_i does not contain a substring belonging to R_i, i.e., $v_i \in \Sigma_{R_i}^+$.
3. for all $i \in \{1, \ldots, n-1\}$, x_i does not contain a substring belonging to L_{i+1}, i.e., $x_i \in \Sigma_{L_{i+1}}^*$.

conditions 2 and 3: ensure that that the extracted strings are as short as possible and that the distance between them is as small as possible.

without condition 2:

without condition 3:

Learning Scenario for Island Wrappers

remember:

available information / examples:

- user marks interesting n-tuple $\langle v_1, \ldots, v_n \rangle$ in a document d
  - marks the corresponding starting and end positions
- user samples the document into $2n+1$ consecutive text parts $u_0, v_1, u_1, \ldots, v_n, u_n$.
  - the string $u_0 v_1 u_1 \cdots v_n u_n$ equals d
- such a $2n+1$-tuple $\langle u_0, v_1, u_1, \ldots, v_n, u_n \rangle$ is said to be an ***n-marked document***

Definition 5.4:
Let $W = (L_1, R_1, \ldots, L_n, R_n)$ be an island wrapper and let $m = \langle u_0, v_1, u_1, \ldots, v_n, u_n \rangle$ be an n-marked document.
Then, m is said to be an ***example*** for W if

1. $u_0 \in \Sigma^* \circ L_1$ and $u_n \in R_n \circ \Sigma^*$.
2. for all $i \in \{1, \ldots, n\}$, $v_i \in \Sigma^+_{R_i}$.
3. for all $i \in \{1, \ldots, n-1\}$, $u_i \in R_i \circ \Sigma^*_{L_{i+1}} \circ L_{i+1}$.

Encoding: Represent $\langle u_0, v_1, u_1, \ldots, v_n, u_n \rangle$ as $u_0 \# v_1 \# u_1 \# \ldots \# v_n \# u_n$ with $\# \notin \Sigma$

Learning of complete wrappers

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

Theorem 5.1:
IW$(\mathcal{IC})$ $\in$ *LimInf*

Idea: identifiction by enumeration

Theorem 5.2:
IW$(\mathcal{IC})$ $\notin$ *LimTxt*

$L_1 = \{a\}, L_2 = \{a\}, R_2 = \{a\}$

$R_1 = \{a^n \mid n > 0\}$ or $\{a\}$ or $\{a, a^2\}$ or $\{a, a^2, a^3\} \ldots$

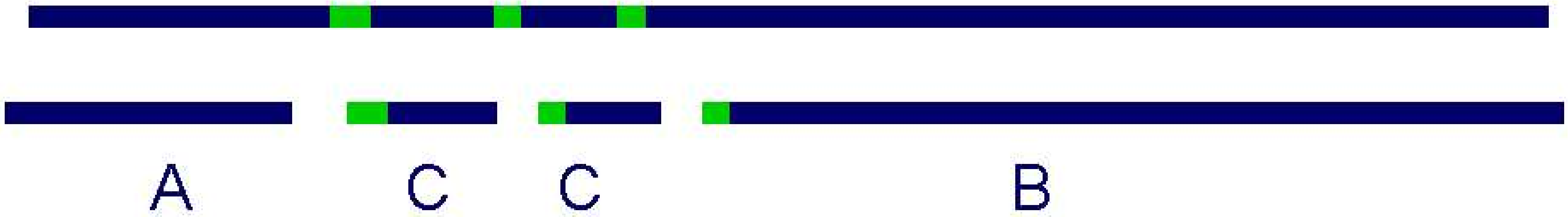
Learning of complete wrappers

$\boldsymbol{\Sigma^{\leq k}}$: set of all words over Σ of length $\leq k$

Theorem 5.3:
IW$(\wp(\Sigma^{\leq k})) \in$ *LIMTxt* for all $k \in \mathbb{N}$

Proof.

Observation: Learning an Island Wrapper from text can be decomposed!



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 $\Sigma^+_{R_m}\{\#\}R_m\Sigma^+_{L_{m+1}}L_{m+1}$

Learning of complete wrappers

The IIM M_A for learning problems of type A:

IIM M_A: On input $S = u_0, \ldots, u_m$ do the following:

Set $h = \emptyset$. Determine the set E of all non-empty suffixes of strings in S. For all strings $e \in E$ check whether or not, for all $a \in \Sigma$, $u = a \circ e$ for some $u \in S$. Let T be the set of all strings e passing this test.

While $T \neq \emptyset$ do:

Determine a shortest string e in T. Set $h = h \cup \{e\}$ and $T = T \setminus T_e$, where T_e contains all strings in T with the suffix e.

Output h.

IIM M_B can be obtained from M_A by replacing everywhere the term suffix by prefix and ignoring the part before the $\#$ in the examples

---

IIM M_C: On input $S = u_0\#w_0, \ldots, u_m\#w_m$ do the following:

Let B and E be the set of all non-empty prefixes and suffixes of the strings $w_0, \ldots, w_m$.

Let H be the collection of all sets $h \subseteq (B, E)$ such that no string in h is longer than k .

Search for an $h \in H$ such that, for every $u \in S$ it holds $u \in \Sigma_B^+\{\#\}B\Sigma_E^*E$. If such an h is found, let h' be the lexicographically first of them. Otherwise, set $h' = (\emptyset, \emptyset)$.

Output h'.

Learning an Island Wrapper from text can be decomposed

Question: What is the relation between the learning tasks?

Definition 5.5:

- $T_1(L) = \Sigma^* \circ L$.
- $T_2(L) = \Sigma_L^+ \circ \{\#\} \circ L \circ \Sigma^*$.
- $T_3(L, L') = \Sigma_L^+ \circ \{\#\} \circ L \circ \Sigma_{L'}^* \circ L'$.

  Let $\mathcal{L}$ be an indexable language class. For all $i \in \{0, \ldots, 3\}$, the learning problem ***LP****$_i(\mathcal{L})$* can be solved iff $T_i(\mathcal{L}) \in$ *LimTxt*, where
  $T_0(\mathcal{L}) = \mathcal{L}$ (reference problem),
  $T_1(\mathcal{L}) = \{T_1(L) \mid L \in \mathcal{L}\}$,
  $T_2(\mathcal{L}) = \{T_2(L) \mid L \in \mathcal{L}\}$, and
  $T_3(\mathcal{L}) = \{T_3(L, L') \mid L, L' \in \mathcal{L}\}$.

Learning an Island Wrapper from text can be decomposed

Theorem 5.4:
Let $i, j \in \{0, \ldots, 3\}$ with $i \neq j$. Then, there is an indexable class $\mathcal{L}$ such that assertions

1. it is possible to solve problem $LP_i(\mathcal{L})$.
2. it is impossible to solve problem $LP_j(\mathcal{L})$.

Consequently, there are indexable classes $\mathcal{L}$ such that

1. knowing that there is a solution for one of the learning problems does not help to solve the other ones and, vice versa,
2. knowing that some learning problem cannot be solved does not mean that one cannot solve the other ones.

Learning an Island Wrapper from text can be decomposed

Proof.

We only discuss some cases.

$\mathcal{L}_A$: collection of the following languages over $\Sigma = \{a, b, c\}$: For all $n \in \mathbb{N}$, let $L_0 = \{a^m b \mid m \geq 1\} \cup \{c\}$ and $L_{n+1} = \{a^m b \mid 1 \leq m \leq n+1\} \cup \{c, ca\}$.

$T_0(\mathcal{L}_A) \in$ *LimTxt*: trivial

$T_1(\mathcal{L}_A) \in$ *LimTxt*:

IIM M: On input $w_0, \ldots, w_m$, check whether some of the strings $w_0, \ldots, w_m$ ends with a. If no such string occurs, output a description for $\Sigma^* \circ L_0$. Otherwise, return a description for $\Sigma^* \circ L_1$.

Reason: $\Sigma^* \circ L_1 = \Sigma^* \circ L_2 = \Sigma^* \circ L_3 = \ldots$

Learning an Island Wrapper from text can be decomposed

$T_2(\mathcal{L}_A) \notin$ *LimTxt*:

assume the contrary, i.e., let M be an IIM that learns $T_2(\mathcal{L}_A)$ in the limit from text.

- since $\Sigma^+_{L_i} = \Sigma^+_{L_j}$ for any $i, j \in \mathbb{N}$, one can easily transform M into an IIM M' that *LimTxt*–identifies the indexable class $\{L \circ \Sigma^* \mid L \in \mathcal{L}_A\}$.
- hence, there is a finite telltale set $S_0 \subseteq L_0 \circ \Sigma^*$ such that $S_0 \subseteq L \circ \Sigma^*$ implies $L \circ \Sigma^* \not\subset L_0 \circ \Sigma^*$, for any $L \in \mathcal{L}_A$.
- for the ease of argumentation assume that some string in S_0 has a prefix of form $a^{n'}b$
- let n be the maximal index n'
- clearly, $L_n \circ \Sigma^* \subset L_0 \circ \Sigma^*$
- on the other hand, $S_0 \subseteq L_n \circ \Sigma^*$.
- this contradicts our assumptions that S_0 serves as a finite tell-tale set for L_0

$T_3(\mathcal{L}_A) \in$ *LimTxt*: Exercise

Learning an Island Wrapper from text can be decomposed

$\mathcal{L}_B$: collection of the following languages L_n over $\Sigma = \{a, b\}$, where, for all $n \in \mathbb{N}$, $L_0 = \{ab^m a \mid m \geq 1\}$ and $L_{n+1} = L_0 \setminus \{ab^{n+1}a\}$.

$T_0(\mathcal{L}_B) \notin$ *LimTxt*: trivial

$T_1(\mathcal{L}_B) \notin$ *LimTxt*: Exercise

Observation:

- for all $n \in \mathbb{N}$, $\Sigma^+_{L_{n+1}}$ contains exactly one string that belongs to L_0, namely the string $ab^{n+1}a$.
- this allows one to distinguish the languages $T_2(L_0)$ and $T_2(L_{n+1})$ as well as $T_3(L_0)$ and $T_3(L_{n+1})$

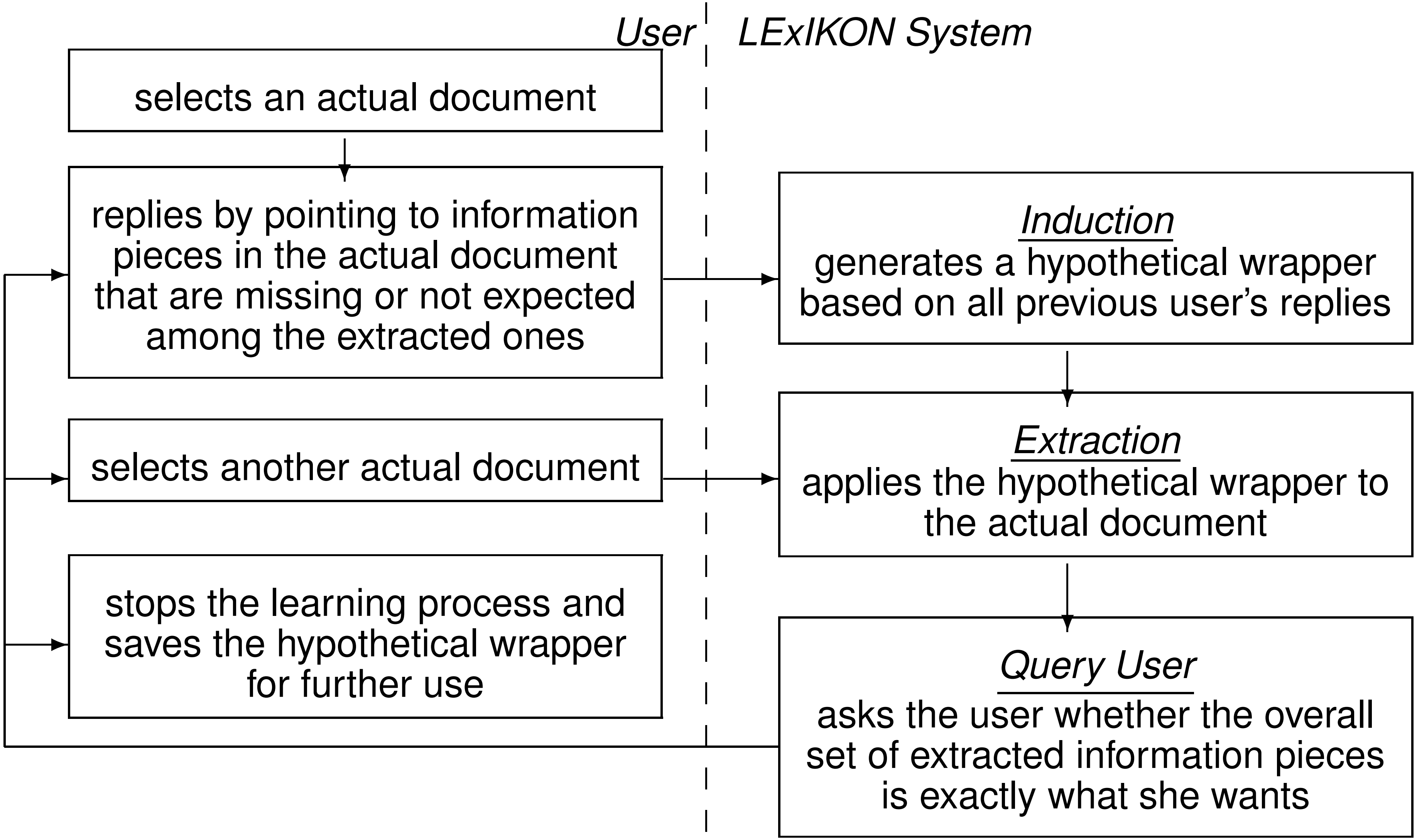
$T_2(\mathcal{L}_B) \in$ *LimTxt*.

$T_3(\mathcal{L}_B) \in$ *LimTxt*.

qed

Gliederung der LV

The LExIKON Interaction Scenario



Prototypical Questions

one may/may not expect that most powerful learning algorithms have one of the following features ...

- all wrappers constructed in the learning phase are consistent with the information they are built upon
- all wrappers constructed in the learning phase are applicable to all possible documents
- one can see whether or not the wrapper most recently constructed is a correct one, i.e. that the learning phase is already finished
- explicit acces to the information provided in the previous steps of the learning phase is not needed

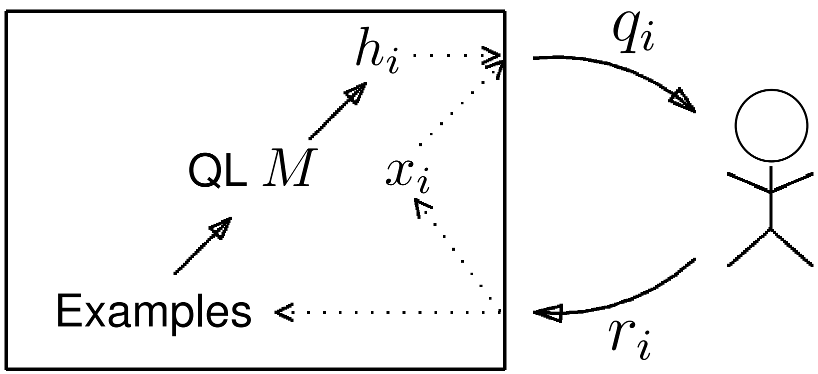
5.1 Our Model: IE by CQ

two players query learner M and user U

purpose U wants to exploit the capabilities of M in order to create a particular wrapper

internal actions of the learner M synthesizes a wrapper based on all information seen so far

internal actions of the user U checks whether or not the synthesized wrapper behaves on the recent document as expected



Technicalities

Notions and Notations

- By convention, $\varphi_0(x) = 0$ for all $x \in \mathbb{N}$.
- $(F_i)_{i\in\mathbb{N}}$ is the canonical enumeration of all finite subsets of $\mathbb{N}$, *where* $F_0 = \emptyset$.

Wrappers

- *wrapper*: function that, given a document, returns a finite set of information pieces contained in the document.
- formal: use natural numbers to describe both documents as well as the information pieces extracted.
- a wrapper can be seen as a computable mapping from $\mathbb{N}$ to $\wp(\mathbb{N})$

More formally:

- wrapper: computable mapping f from $\mathbb{N}$ to $\mathbb{N}$, where
  - for all $x \in \mathbb{N}$ with $f(x)\downarrow$, $f(x)$ is interpreted to denote the finite set $F_{f(x)}$.
  - If $f(x) = 0$, the wrapper f fails to extract anything of interest from the document x

(since $F_0 = \emptyset$).

Interaction Sequence

– system starts with a default wrapper $h_0 = 0$
($\varphi_0(x) = 0$ for all x and $F_0 = \emptyset \rightarrow h_0$ does not extract any data from any document)
– the user selects an initial document d and presents d to the system.

1. system applies the most recently stored wrapper h_i to the current document x_i ($x_0 = d$)
2. let $F_{h_i(x_i)}$ be the set of data that has have been extracted from x_i by the wrapper h_i.
   - we demand that the wrapper h_i is defined on input x_i. Otherwise, the interaction between the system and the user will definitely crash.
3. the ***consistency query*** $q_i = (x_i, F_{h_i(x_i)})$ is presented to the user for evaluation.
4. is $F_{h_i(x_i)}$ correct (i.e., $F_{h_i(x_i)}$ contains only interesting data) and complete (i.e., $F_{h_i(x_i)}$ contains all interesting data)?
   - if yes: signal that wrapper h_i is accepted for the current document x_i:
     – select another document d' subject to further interrogation
     – return the *accepting reply* $r_i = d'$
   - otherwise: select either
     – a data item n_i which was erroneously extracted from x_i (i.e., a negative example) or
     – a data item p_i which is of interest in x_i and which was not extracted (i.e., a positive example).

     i.e. return the *rejecting reply* $r_i = (n_i, -)$ or $r_i = (p_i, +)$.
5. system: generates wrapper h_{i+1} (new hypothesis) based on all previous interactions, the last consistency query q_i, and the corresponding reply r_i.
6. Goto 1.

Interaction Sequence

Definition 5.6:
Let $d \in \mathbb{N}$ and $I = ((q_i, r_i))_{i \in \mathbb{N}}$ be an infinite sequence.
I is said to be an ***interaction sequence*** between a query learner M and a user U with respect to a target wrapper f iff for every $i \in \mathbb{N}$ the following conditions hold:

1. $q_i = (x_i, E_i)$, where
   - $x_0 = d$ and $E_0 = \emptyset$.
   - $x_{i+1} = r_i$, if r_i is an accepting reply.
   - $x_{i+1} = x_i$, if r_i is a rejecting reply.
   - $E_{i+1} = F_{\varphi_{M(I_i)}(x_{i+1})}$.*
2. If $F_{f(x_i)} = E_i$, then r_i is an accepting reply, i.e., $r_i \in \mathbb{N}$.
3. If $F_{f(x_i)} \neq E_i$, then r_i is a rejecting reply, i.e., it holds either $r_i = (n_i, -)$ with $n_i \in E_i \setminus F_{f(x_i)}$ or $r_i = (p_i, +)$ with $p_i \in F_{f(x_i)} \setminus E_i$.

* It is assumed that $\varphi_{M(I_i)}(x_{i+1}) \downarrow$, i.e. M's most recent hypothesis, i.e. the wrapper $w = \varphi_{M(I_i)}$ has to be applicable to the most recent document x_{i+1}.

Interaction Sequence

- interaction sequence: pairs of queries and responses
  $(q_0, r_0), (q_1, r_1), (q_2, r_2), \ldots$
- (hidden) sequence of hypotheses
  $h_0 = M((q_0, r_0)), h_1 = M((q_0, r_0), (q_1, r_1)), h_2 = M((q_0, r_0), (q_1, r_1), (q_2, r_2)), \ldots$

Fairness Requirements

- ensure that the learner does not get stuck in a single document

Definition 5.7:
A query learner M is said to be *open-minded* with respect to $\mathcal{L}$ iff

- for all users U, all wrappers $f \in \mathcal{L}$, and all interaction sequences $I = ((q_i, r_i))_{i \in \mathbb{N}}$ between M and U with respect to f
- there are infinitely many $i \in \mathbb{N}$ such that r_i is an accepting reply.

- if M is not open-minded, the user might not get the opportunity to inform the system adequately about her expectations

Fairness Requirements

- a query learner can only be successful in case when the user illustrates her intentions on various different documents

Definition 5.8:
A user U is said to be *co-operative* with respect to $\mathcal{L}$ iff

- for all open-minded query learners M, for all wrappers $f \in \mathcal{L}$, all interaction sequences $I = ((q_i, r_i))_{i \in \mathbb{N}}$ between M and U with respect to f, and all $x \in \mathbb{N}$
- there is an accepting reply r_i with $r_i = x$.

LIMCQ

Definition 5.9:
Let $\mathcal{L} \subseteq \mathcal{R}$ and let M be an *open-minded* query learner.
$\mathcal{L} \subseteq$ *LIMCQ*(M) iff

- for all *co-operative* users U, all wrappers $f \in \mathcal{L}$, and all interaction sequences I between M and U with respect to f
- there is a $j \in \mathbb{N}$ with $\varphi_j = f$ such that, for almost all $n \in \mathbb{N}$, $j = h_{n+1} = M(I_n)$.

By *LIMCQ* we denote the collection of all $\mathcal{L}' \subseteq \mathcal{R}$ for which there is an open-minded query learner M' such that $\mathcal{L}' \subseteq$ *LIMCQ*(M').

FINCQ, CONSCQ and the like

Definition 5.10: $\mathcal{L} \subseteq$ *ET*(M)(*ET* $\in$ {***FINCQ, TOTALCQ, CONSCQ, ITCQ***}) iff there is an open-minded query learner M with $\mathcal{L} \subseteq$ *LIMCQ*(M) such that

- for all co-operative users U, U', for all $f, f' \in \mathcal{L}$, all interaction sequences I and I' between M and U with respect to f resp. between M and U' with respect to f', and all $n, m \in \mathbb{N}$:

FINCQ	$M(I_n) = M(I_{n+1})$ implies $\varphi_{M(I_n)} = f$.
TOTALCQ	$\varphi_{M(I_n)} \in \mathcal{R}$.
CONSCQ	For all $(x, y) \in I_n^+$ and all $(x, y') \in I_n^-$, it holds $y \in F_{\varphi_{M(I_n)}(x)}$ and $y' \notin F_{\varphi_{M(I_n)}(x)}$.
ITCQ	$M(I_n) = M(I'_m)$ and $I(n+1) = I'(m+1)$ imply $M(I_{n+1}) = M(I'_{m+1})$.

where, for any prefix σ of an interaction sequence

- σ^+: set of all pairs (x, y) such that there is a consistency query (x, E) in σ that
  - receives the rejecting reply $(y, +)$ or – receives an accepting reply and $y \in E$
- σ^-: set of all pairs (x, y') such that there is a consistency query (x, E) in σ that
  - receives the rejecting reply $(y', -)$ or – receives an accepting reply and $y' \notin E$

Results

$$
\begin{tikzcd}
 & \mathit{LIM} = \mathit{LIM}^{arb} & \\
 & \mathit{IT} \arrow[u] & \\
 \mathit{CONS} \arrow[ur] & & \\
 \mathit{CONS}^{arb} \arrow[u] & & \mathit{IT}^{arb} \arrow[uul] \\
 \mathit{TOTALCQ} = \mathit{CONSCQ} = \mathit{LIMCQ} = \mathit{TOTAL} = \mathit{TOTAL}^{arb} \arrow[u] & & \\
 & \mathit{INCCQ} \arrow[ul] \arrow[uur] & \\
 & \mathit{FIN} = \mathit{FIN}^{arb} = \mathit{FINCQ} \arrow[u] &
\end{tikzcd}
$$

Results

- *LIMCQ* far below in large hierarchy of identification types
  - IE is quite ambitious and doomed to fail in situations where other more theoretical learning approaches still work
- coincides with well-known identification type *TOTAL*
  - power of IE is well-understood
- IE can always be *consistent* and can return *fully defined* wrappers that work on every document
- IE can not always work incrementally by taking wrappers developed before and just presenting new samples
- query learner can not always decide when the work is done

Results

Theorem 5.5:

For all *ET* $\in \{$*FIN*, *TOTAL*, *CONS*, *LIM*$\}$: *ETCQ* $\subseteq$ *ET*^{arb}.

Proof.

let M be a query learner, let *ET* $\in \{$*FIN*, *TOTAL*, *CONS*, *LIM*$\}$, let $f \in$ *ETCQ*(M), and let $((x_j, f(x_j)))_{j \in \mathbb{N}}$ be any representation of f

define IIM M' such that *ETCQ*$(M) \subseteq$ *ET*$^{arb}(M')$:

- *main idea: M' uses the information which it receives about the graph of f in order to interact with M on behalf of a user. Then, in case where M's actual consistency query will receive an accepting reply, M' takes over the actual hypothesis generated by M.*

Results

- Initially, for the input $(x_0, f(x_0))$, M' presents x_0 as initial document to M, and the first round of the interaction between M' and M starts.
- In general, the $(i+1)$-st round of the interaction between M' and M can be described as follows.
  - Let (x_i, E_i) be the actual consistency query posed by M. (Initially: $(x_0, \emptyset)$)
  - M' checks whether or not E_i equals $F_{f(x_i)}$.
    - If not: M' selects the least element z from the symmetrical difference of E_i and $F_{f(x_i)}$ and returns the counterexample $(z, b(z))$.
      ($b(z) = +$, if $z \in F_{f(x_i)} \setminus E_i$ and $b(z) = -$, if $z \in E_i \setminus F_{f(x_i)}$.)
      In addition, M' and M continue the $(i+1)$-st round of their interaction.
    - Otherwise, the actual round is finished and M' takes over M's actual hypothesis. Moreover, M' answers M's last consistency query with the accepting reply x_{i+1} and the next round of the interaction between M' and M starts.

Results

Theorem 5.6:
FIN^{arb} $\subseteq$ *FINCQ*.

Proof.

- If a consistency query (x_i, E_i) receives an accepting response, one knows for sure that $f(x_i)$ equals y_i, where y_i is the unique index with $F_{y_i} = E_i$.
  - notation: *content*(τ) is the set of all pairs $(x, f(x))$ from the graph of f that can be determined according to the accepting responses in the interaction sequence τ

Results

Let an IIM M be given and let $f \in \mathit{FIN}^{arb}(M)$. The query learner M' works as follows:

- Let τ be the most recent initial segment of the resulting interaction sequence between M and U with respect to f. (* Initially, τ is empty. *) M' arranges all elements in *content*(τ) in lexicographical order, let σ be the resulting sequence. Then, M' simulates M when fed σ.
- If M outputs a final hypothesis, say j, M' generates the hypothesis j. Past that point, M' will never change its mind and will formulate all consistency queries with respect to φ_j.
- If M does not output a final hypothesis, M' starts a new interaction cycle with U. Let x_i be either the document that was initially presented or the document that M' received as its last accepting response. Informally speaking, in order to find $f(x_i)$, M' subsequently asks the consistency queries $(x_i, F_0), (x_i, F_1), \ldots$ until it receives an accepting reply. Obviously, this happens, if M' queries $(x_i, F_{f(x_i)})$. At this point, the actual interaction cycle between M' and U is finished and M' continues as described above, i.e., M' determines σ based on the longer initial segment of the interaction sequence.

Results

Theorem 5.7:
TOTAL^{arb} $\subseteq$ *TOTALCQ*.

Proof.

analogously to last proof

Results

Theorem 5.8:
LIMCQ $\subseteq$ *TOTALCQ*.

Proof.

Let M be an open-minded query learner and let τ be an initial segment of any interaction sequence.

Notations:

- τ^l is the last element of τ and τ^{-1} is the initial segment of τ without the last element τ^l.
- we fix some effective enumeration $(\rho_i)_{i\in\mathbb{N}}$ of all non-empty finite initial segments of all possible interaction sequences which end with a query q that received an accepting reply $r \in \mathbb{N}$.
- $\#\tau$: least index of τ in this enumeration.
- Let $i \in \mathbb{N}$. We call ρ_i a *candidate stabilizing segment* for τ iff
  1. *content*$(\rho_i) \subseteq$ *content*(τ),
  2. $M(\rho_i^{-1}) = M(\rho_i)$, and
  3. $M(\rho_j) = M(\rho_i^{-1})$ for all ρ_j with $j \leq \#\tau$ that meet *content*$(\rho_j) \subseteq$ *content*(τ) and that have the prefix ρ_i^{-1}.

Results

Let τ be the most recent initial segment of the interaction sequence between M' and user U and x be the most recent document.

M' searches for the least index $i \leq \#\tau$ such that ρ_i is a candidate stabilizing segment for τ.

Case A. No such index is found.

Now, M' simply generates an index j as auxiliary hypothesis such that φ_j is a total function that meets $\varphi_j(x) = \varphi_{M(\tau)}(x)$. *($\varphi_{M(\tau)}(x)$ has to be defined.)*

Case B. Otherwise.

M determines an index of a total function as follows. Let $\rho_i^l = (q, r)$.

($\varphi_{M(\rho_i^{-1} \diamond (q,x))}(x)$ and $\varphi_{M(\tau)}(x)$ have to be defined.)

Subcase B1. $\varphi_{M(\rho_i^{-1} \diamond (q,x))}(x) = \varphi_{M(\tau)}(x)$.

M determines an index k of a function meeting $\varphi_k(z) = \varphi_{M(\rho_i^{-1} \diamond (q,z))}(z)$ for all $z \in \mathbb{N}$.

($M(\rho_i^{-1} \diamond (q, z))$ is defined for all $z \in \mathbb{N}$, since ρ_i ends with an accepting reply.)

Subcase B2. $\varphi_{M(\rho_i^{-1} \diamond (q,x))}(x) \neq \varphi_{M(\tau)}(x)$.

M generates an index j of a total function as in Case A.

Results

Verification:

Let $f \in$ *LIMCQ*(M), let I be the resulting interaction sequence between M and U w.r.t. f.

Have to show that M' is an open-minded query learner with $f \in$ *TOTALCQ*(M'):

1. M' obviously outputs exclusively indices of total functions
2. M' is an open-minded query learner:
   Let x be the most recent document. By definition, it is guaranteed that the most recent hypotheses of M and M''s yield the same output on document x.

   $\leadsto$ interaction sequence I equals the corresponding interaction sequence between M and U (although M' may generate hypotheses that are different from that ones produced by M).

   M is an open-minded learner $\leadsto$ M' is open-minded, too

Results

3. M' learns as required:

   $f \in \mathit{LIMCQ}(M) \rightsquigarrow$ there is a *locking interaction sequence* σ of M for f

   - i.e., $\varphi_{M(\sigma^{-1})} = f$ and for all interaction sequences I' of M and U with respect to f and all $n \in \mathbb{N}$, we have that $M(I'_n) = M(\sigma)$ provided that σ is an initial segment of I'_n.

   Let ρ_i be the least (w.r.t. $(\rho_i)_{i \in \mathbb{N}}$) locking interaction sequence of M for f that ends with an accepting reply.

   - I equals an interaction sequence between M and U w.r.t. $f \rightsquigarrow M$ has to stabilize on I.
   - M' is open-minded $\rightsquigarrow$ there is an n such that *content*$(\rho_i) \subseteq$ *content*(I_n) and M outputs its final hypothesis when fed I_n.
   - $\rightsquigarrow$ past this point M' always forms its actual hypothesis according to Subcase B1
   - M stabilizes on I to $M(I_n)$, $\varphi_{M(I_n)} = f$, and $\varphi_{M(\rho_i^{-1})} = f \rightsquigarrow$ $\varphi_{M'(I_n)} = f$.

Literatur:

[6] G. Grieser & S. Lange: Learning Approaches to Wrapper Induction. *In: Russell & Kolen (eds.), Proc. 14th International FLAIRS Conference*, pp. 249–253, AAAI-Press 2001.

[7] G. Grieser & K.P. Jantke & S. Lange: Consistency Queries in Information Extraction. *In: Cesa-Bianchi & Numao & Reischuk (eds.), Proc. 13th International Conference on Algorithmic Learning Theory*, pp. 173–187, Springer-Verlag 2002.

Changelog

- V1.1:
  - Folie 15: last line changed
  - Folie 27: request $\longrightarrow$ reply