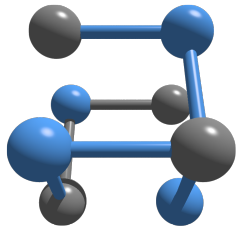
Learning OWL Class Expressions

betreut von Prof. Klaus-Peter Fähnrich

Dissertations-Verteidigungsvortrag

Dipl.-Inf. Jens Lehmann

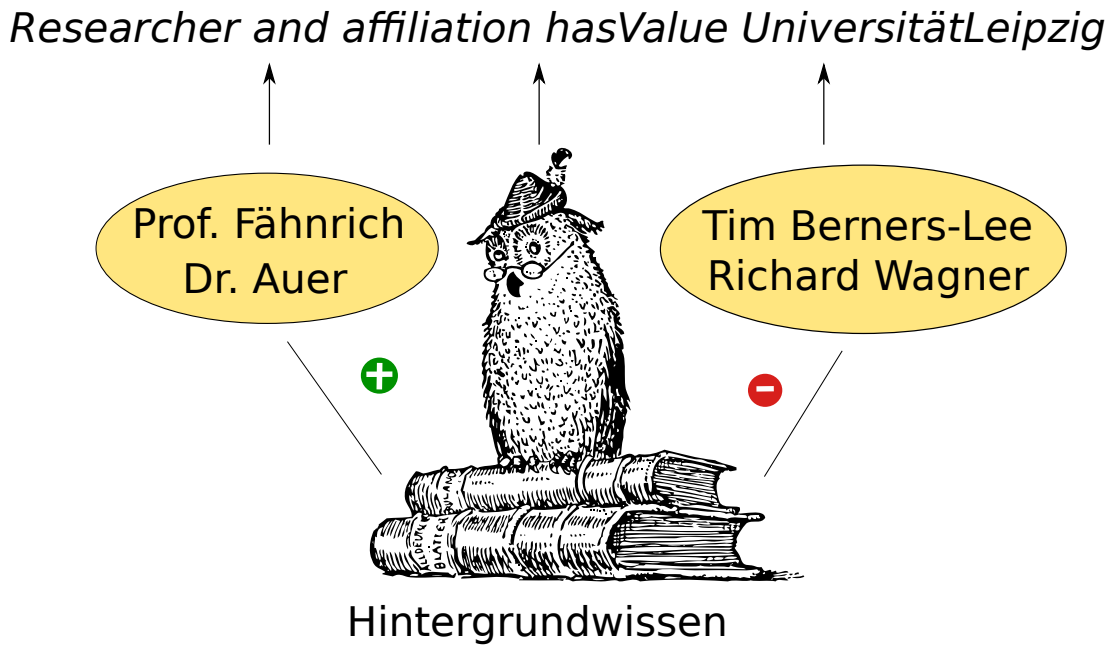


UNIVERSITÄT LEIPZIG

9. Juni 2010

Was bedeutet "Learning OWL Class Expressions"?

- gegeben:
  - Hintergrundwissen (speziell: OWL/DL Wissensbasis)
  - positive und negative Beispiele (speziell: Objekte in Wissensbasis)
- gesucht:
  - logische Formel (speziell: OWL Class Expression), die möglichst alle positiven und keines der negativen Beispiele abdeckt

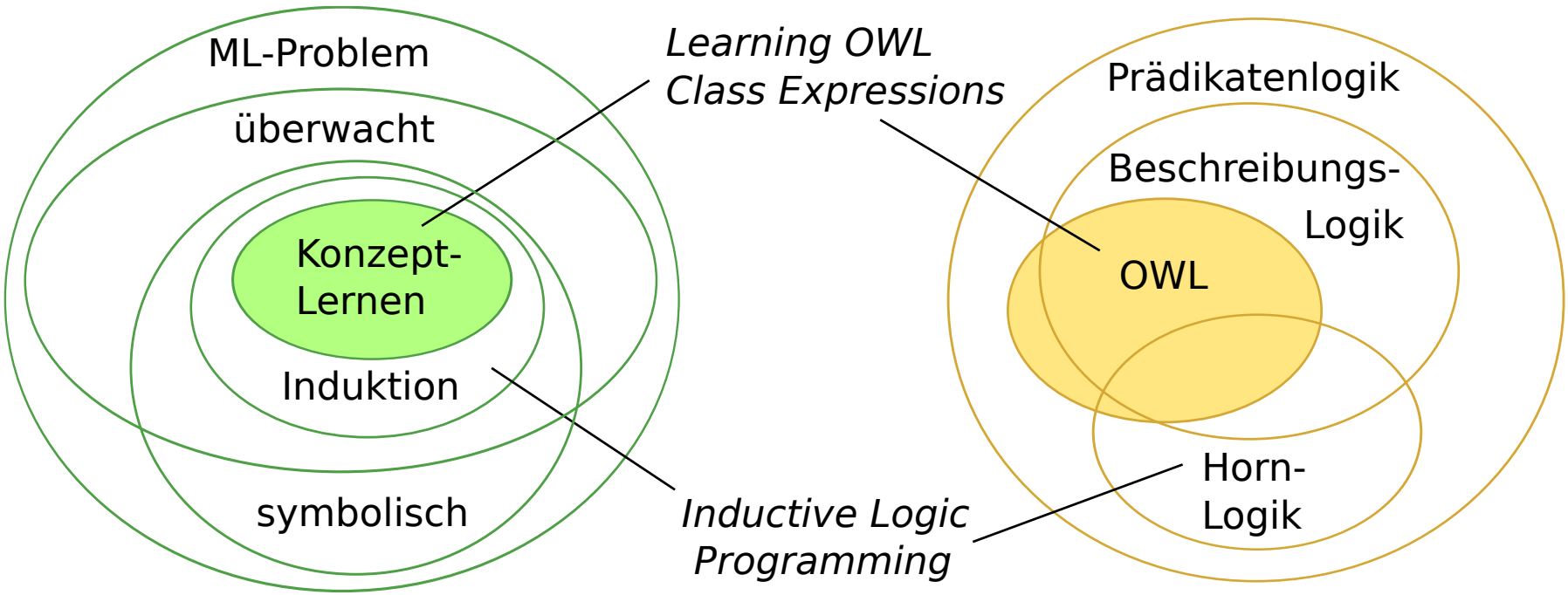


Gliederung

1. *Motivation*
2. *Überblick zu Beschreibungslogiken und OWL*
3. *Refinementoperatoren in OWL/DLs*
   - Theoretische Erkenntnisse
   - Entwicklung von Refinementoperatoren
4. *Lernalgorithmen auf Basis von Refinement-Operatoren*
   - OCEL
   - Skalierbarkeit
5. *Evaluation*
6. *Implementierung und Anwendung*
7. *Zusammenfassung und weitere Arbeit*

Gliederung

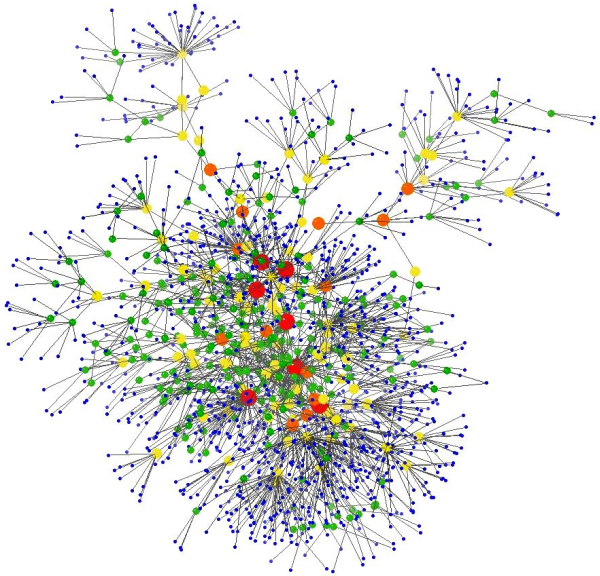
Motivation - Forschungsperspektive



- seit Anfang 90er Jahre Induktive Logikprogrammierung
- nur wenige Ansätze basieren auf Beschreibungslogiken
- Web Ontology Language (OWL) wird 2004 W3C-Standard
- steigende Anzahl von RDF/OWL Wissensbasen, aber ILP noch stark auf Logikprogrammierung fixiert $\rightsquigarrow$ Forschungslücke

Motivation - Anwendungsperspektive

- Ontology-Engineering schwierig und ein Hauptproblem im Semantic Web
  - Domänenexperten oft keine Ontologie-Experten
  - halbautomatische Hilfsmethoden zur Konstruktion ausdrucksstarker Ontologien
- Verbesserung von Lösungen existierender Lernprobleme
- direkte Nutzung des Wissens im Semantic Web statt Konvertierung um ML-Werkzeuge anwenden zu können



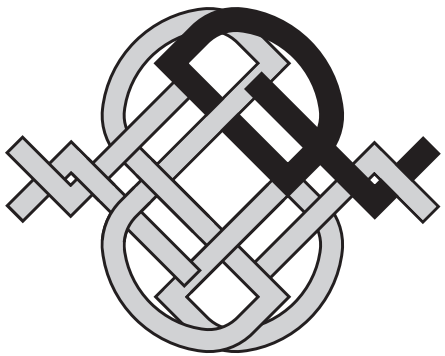
Ontologienetzwerk

Machine Learning Probleme

Gliederung

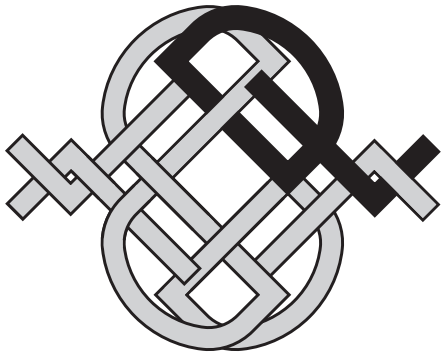
Kurzeinführung zu OWL/DLs

- Description Logics: Familie von Sprachen zur Wissensrepräsentation
- Fragmente der Prädikationlogik
- weniger ausdrucksstark als Prädikatenlogik, aber in der Regel entscheidbare Inferenzprobleme
- intuitive variablenfreie Syntax



Kurzeinführung zu OWL/DLs

- Description Logics: Familie von Sprachen zur Wissensrepräsentation
- Fragmente der Prädikationlogik
- weniger ausdrucksstark als Prädikatenlogik, aber in der Regel entscheidbare Inferenzprobleme
- intuitive variablenfreie Syntax
- Basis der Ontologiesprache OWL
- OWL steht für Web Ontology Language
- W3C recommendation seit 2004 (OWL 2 seit 2009)



Grundstrukturen in OWL/DLs

- Repräsentation von Wissen mit Rollen, Konzepten und Objekten

Grundstrukturen in OWL/DLs

- Repräsentation von Wissen mit Rollen, Konzepten und Objekten
- Objekte/Individuen
  - entsprechen Konstanten
  - Beispiele: `MARIA`, `LEIPZIG`

Grundstrukturen in OWL/DLs

- Repräsentation von Wissen mit Rollen, Konzepten und Objekten
- Objekte/Individuen
  - entsprechen Konstanten
  - Beispiele: `MARIA`, `LEIPZIG`
- Konzepte/Klassen
  - entsprechen einstelligen Prädikaten
  - Menge von Objekten
  - Beispiele: `Student`, `Auto`, `Land`

Grundstrukturen in OWL/DLs

- Repräsentation von Wissen mit Rollen, Konzepten und Objekten
- Objekte/Individuen
  - entsprechen Konstanten
  - Beispiele: `MARIA`, `LEIPZIG`
- Konzepte/Klassen
  - entsprechen einstelligen Prädikaten
  - Menge von Objekten
  - Beispiele: `Student`, `Auto`, `Land`
- Rollen/Properties
  - entsprechen zweistelligen Prädikaten
  - beschreiben Verbindung zwischen Objekten
  - Beispiele: `hatKind`, `istTeilVon`

Grundstrukturen in OWL/DLs

- Repräsentation von Wissen mit Rollen, Konzepten und Objekten
- Objekte/Individuen
  - entsprechen Konstanten
  - Beispiele: `MARIA`, `LEIPZIG`
- Konzepte/Klassen
  - entsprechen einstelligen Prädikaten
  - Menge von Objekten
  - Beispiele: `Student`, `Auto`, `Land`
- Rollen/Properties
  - entsprechen zweistelligen Prädikaten
  - beschreiben Verbindung zwischen Objekten
  - Beispiele: `hatKind`, `istTeilVon`
- können zu (komplexen) Konzepten kombiniert werden, z.B.: $\texttt{Kind} \sqcap \exists \texttt{hatElternteil.Professor}$
- komplexes Konzept $\widehat{=}$ OWL Class Expression

Wissensbasen

Eine DL-Wissensbasis hat folgende Struktur:

Wissensbasis

TBox $\mathcal{T}$ (Terminologie/Schema)

ABox $\mathcal{A}$ ("assertions"/Instanzdaten)

Wissensbasen

Eine DL-Wissensbasis hat folgende Struktur:

Wissensbasis

TBox $\mathcal{T}$ (Terminologie/Schema), z.B.

$\texttt{Student} \equiv \texttt{Person} \sqcap \exists \texttt{studiertAn.Universität}$

$\texttt{Doktorand} \sqsubseteq \exists \texttt{arbeitetAn.}(\texttt{Universität} \sqcup \texttt{Hochschule})$

ABox $\mathcal{A}$ ("assertions"/Instanzdaten)

Wissensbasen

Eine DL-Wissensbasis hat folgende Struktur:

Wissensbasis

TBox $\mathcal{T}$ (Terminologie/Schema), z.B.

$\texttt{Student} \equiv \texttt{Person} \sqcap \exists \texttt{studiertAn.Universität}$

$\texttt{Doktorand} \sqsubseteq \exists \texttt{arbeitetAn.}(\texttt{Universität} \sqcup \texttt{Hochschule})$

ABox $\mathcal{A}$ ("assertions"/Instanzdaten), z.B.

$\texttt{Doktorand}(\texttt{SEBASTIAN})$

$\texttt{arbeitetAn}(\texttt{SEBASTIAN}, \texttt{LEIPZIG\_UNIVERSITÄT})$

Gliederung

Refinementoperator - Definition

- gegeben sei eine DL $\mathcal{L}$ und eine quasi-geordneten Raum $\langle \mathcal{C}(\mathcal{L}), \sqsubseteq_{\mathcal{T}} \rangle$ über Konzepten in $\mathcal{L}$
- $\rho : \mathcal{C}(\mathcal{L}) \rightarrow 2^{\mathcal{C}(\mathcal{L})}$ ist ein downward $\mathcal{L}$ refinement operator wenn für alle $C \in \mathcal{C}(\mathcal{L})$ gilt:

$$D \in \rho(C) \text{ impliziert } D \sqsubseteq_{\mathcal{T}} C$$

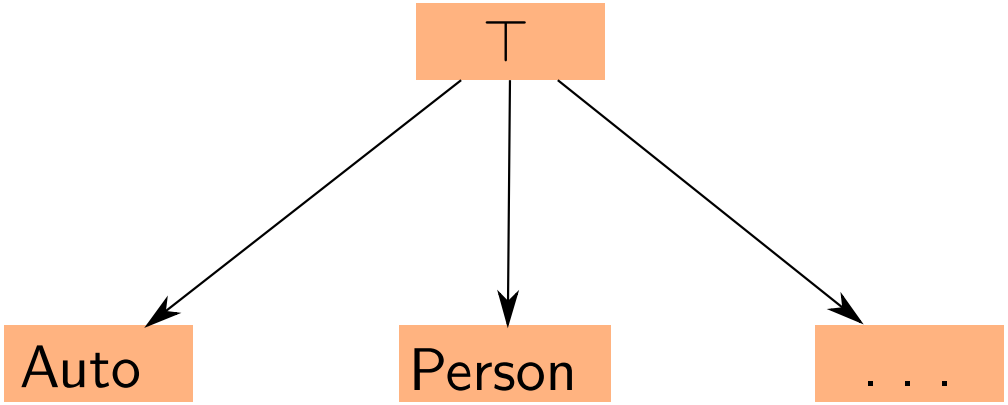
- Notation: oft $C \rightsquigarrow_{\rho} D$ statt $D \in \rho(C)$
- Refinementkette (refinement chain):

$$\top \rightsquigarrow_{\rho} \texttt{Person} \rightsquigarrow_{\rho} \texttt{Mann} \rightsquigarrow_{\rho} \texttt{Mann} \sqcap \exists \texttt{hatKind}.\top$$

Lernen mit Refinementoperatoren

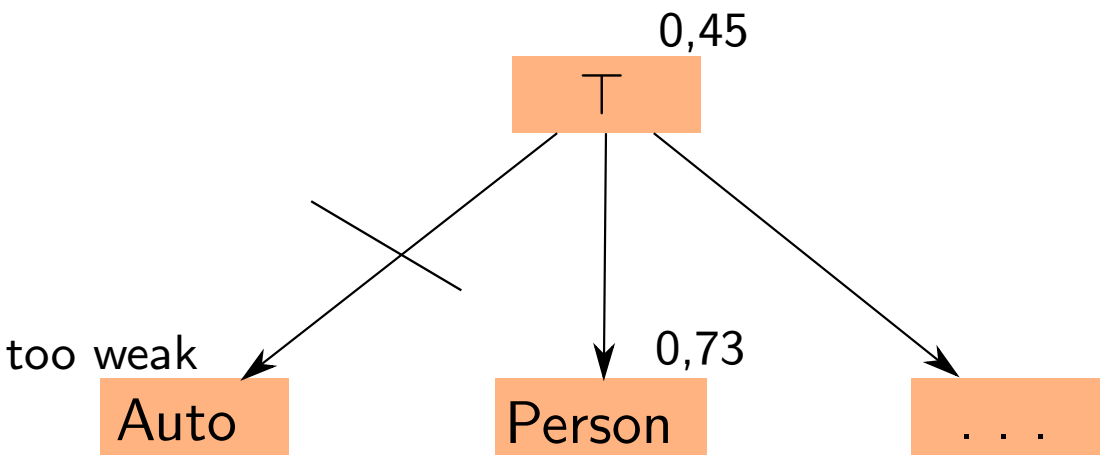
- Start mit allgemeinstem Konzept (top down)

Lernen mit Refinementoperatoren



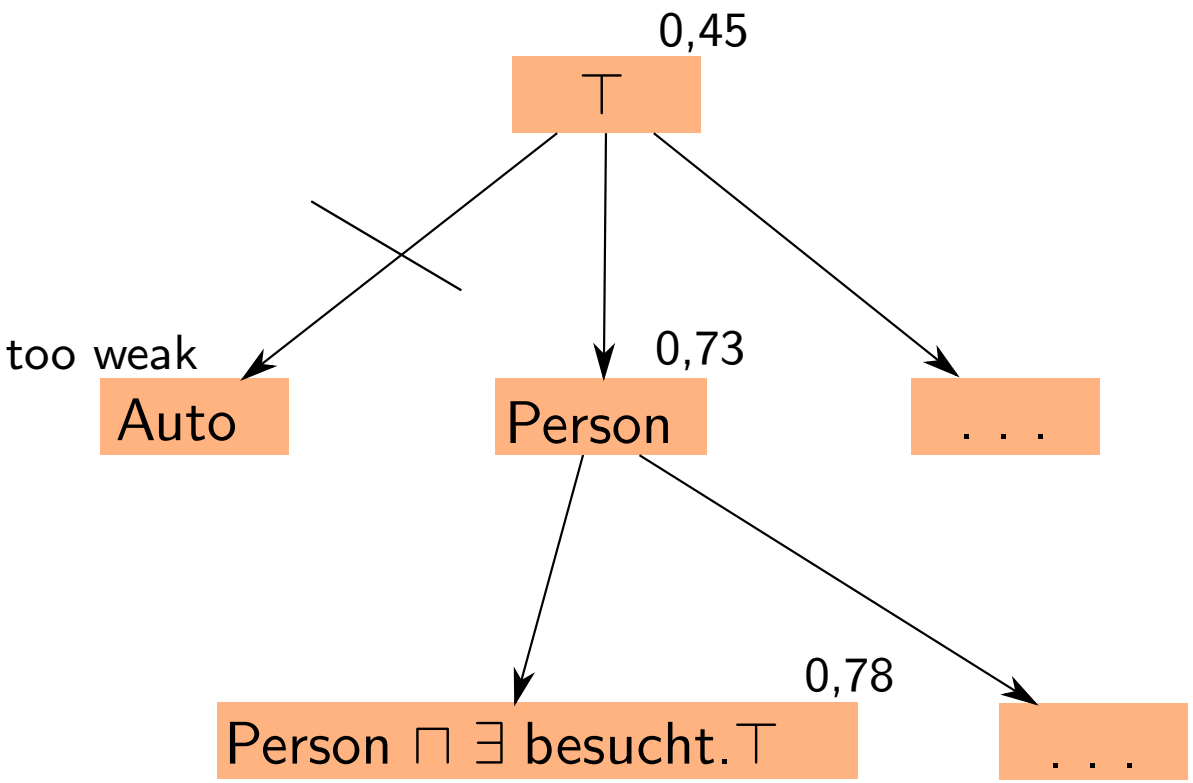
- Start mit allgemeinstem Konzept (top down)
- Operator spezialisiert

Lernen mit Refinementoperatoren



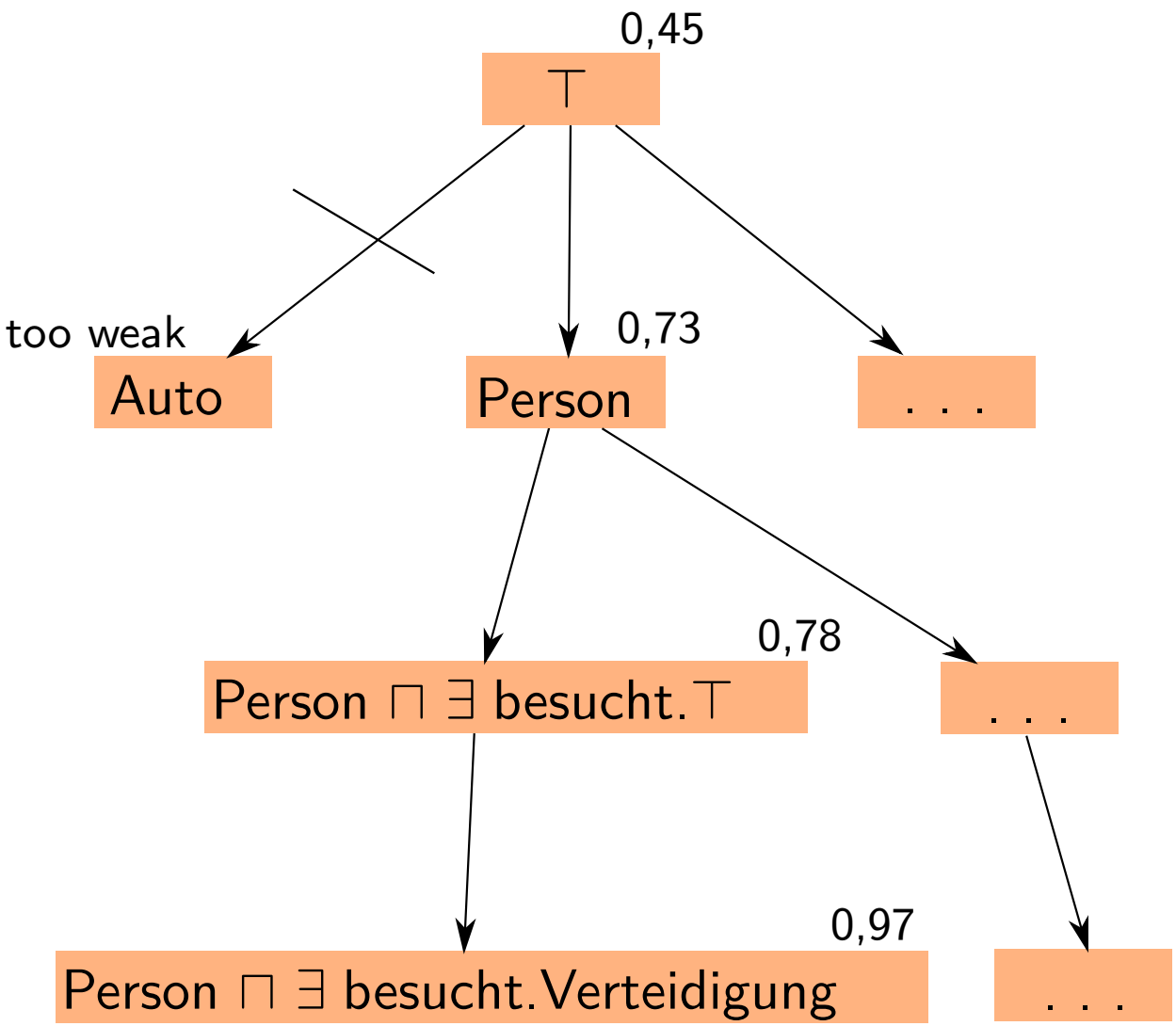
- Start mit allgemeinstem Konzept (top down)
- Operator spezialisiert
- Heuristik bewertet unter Nutzung von pos/neg Beispielen

Lernen mit Refinementoperatoren



- Start mit allgemeinstem Konzept (top down)
- Operator spezialisiert
- Heuristik bewertet unter Nutzung von pos/neg Beispielen
- Fortsetzung bis Abbruchkriterium

Lernen mit Refinementoperatoren



- Start mit allgemeinstem Konzept (top down)
- Operator spezialisiert
- Heuristik bewertet unter Nutzung von pos/neg Beispielen
- Fortsetzung bis Abbruchkriterium

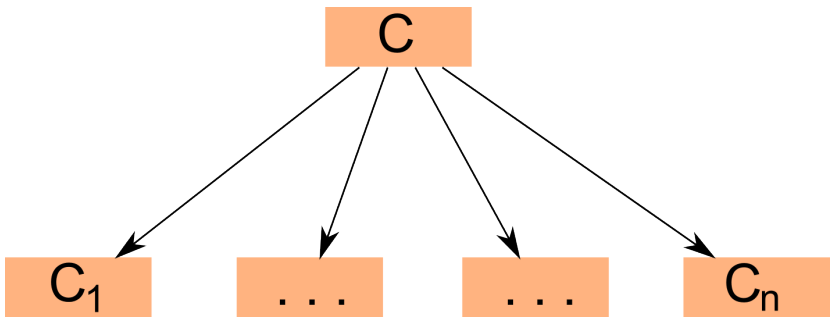
=

Lernalgorithmus

Eigenschaften von Refinementoperatoren

Ein $\mathcal{L}$ downward refinement operator ρ ist

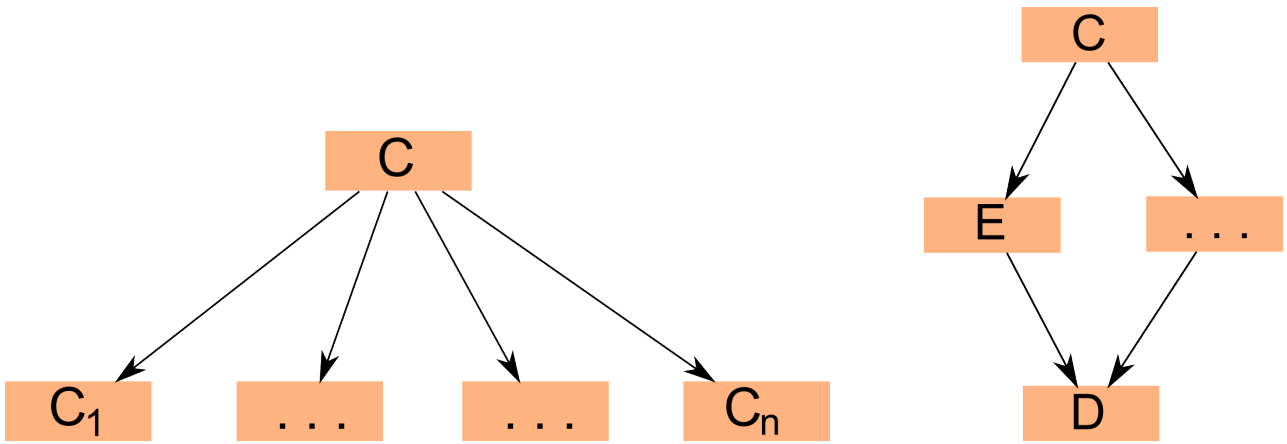
- finite (endlich) falls $\rho(C)$ für jedes $C \in \mathcal{C}(\mathcal{L})$ endlich ist



Eigenschaften von Refinementoperatoren

Ein $\mathcal{L}$ downward refinement operator ρ ist

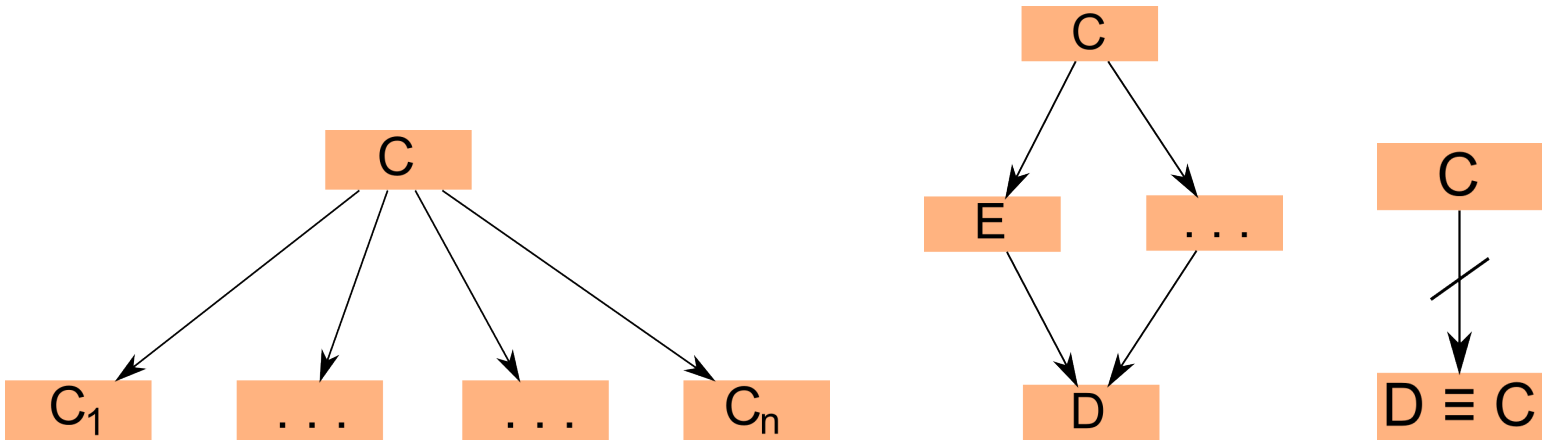
- finite (endlich) falls $\rho(C)$ für jedes $C \in \mathcal{C}(\mathcal{L})$ endlich ist
- redundant falls mehrere ρ-Refinementketten von einem Konzept C zu einem Konzept D existieren



Eigenschaften von Refinementoperatoren

Ein $\mathcal{L}$ downward refinement operator ρ ist

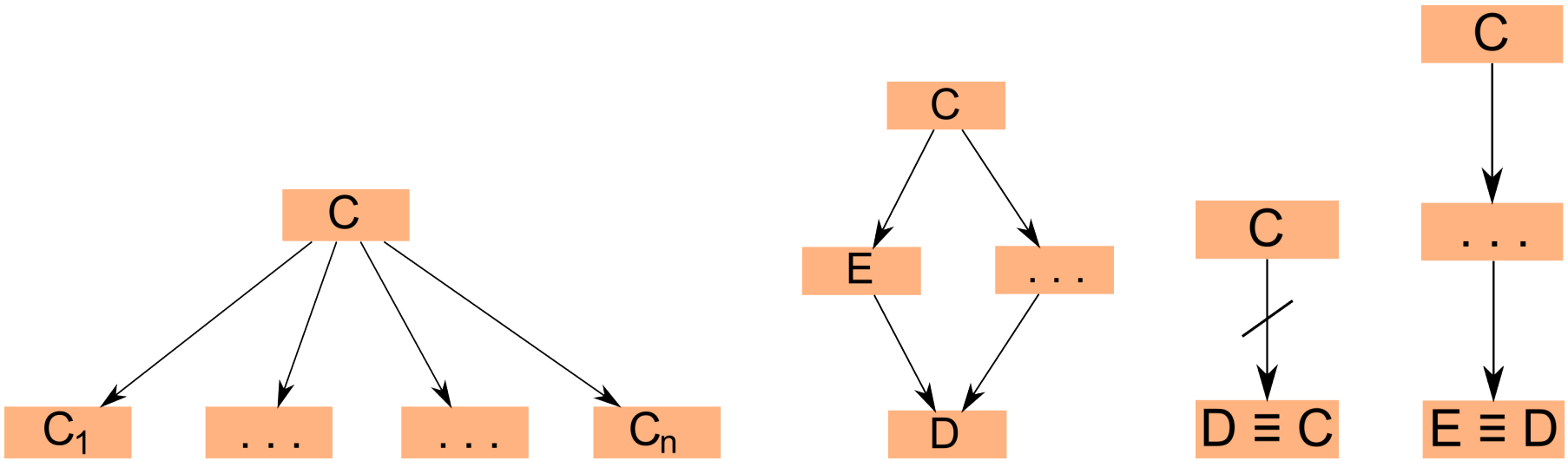
- finite (endlich) falls $\rho(C)$ für jedes $C \in \mathcal{C}(\mathcal{L})$ endlich ist
- redundant falls mehrere ρ-Refinementketten von einem Konzept C zu einem Konzept D existieren
- proper (echt) falls $C \rightsquigarrow_\rho D$ impliziert $C \not\equiv_\mathcal{T} D$



Eigenschaften von Refinementoperatoren

Ein $\mathcal{L}$ downward refinement operator ρ ist

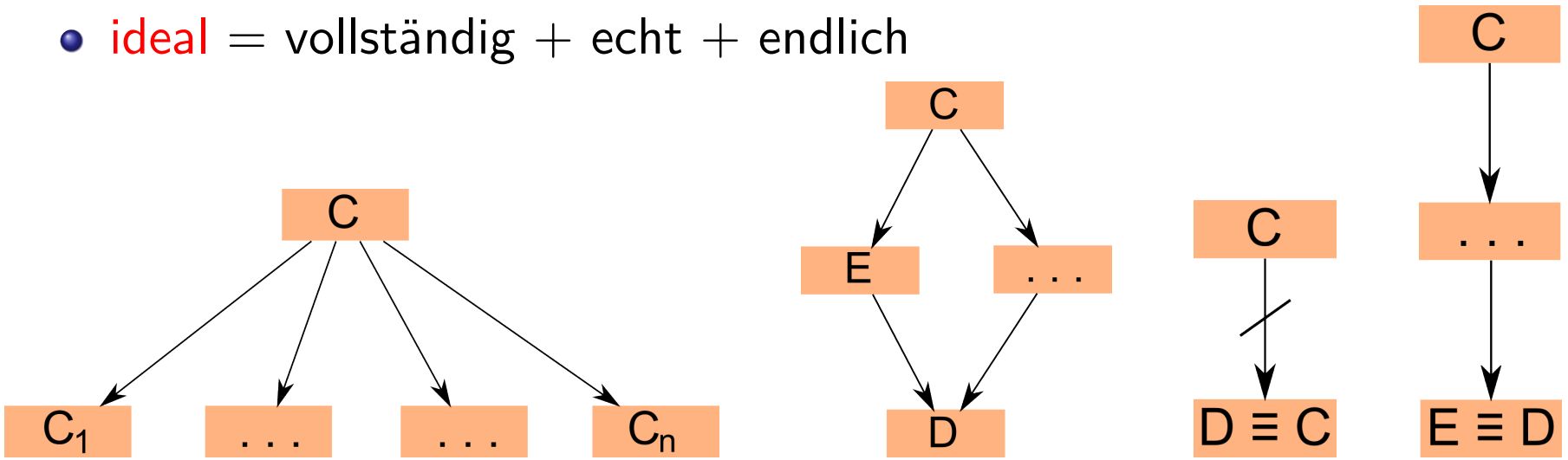
- finite (endlich) falls $\rho(C)$ für jedes $C \in \mathcal{C}(\mathcal{L})$ endlich ist
- redundant falls mehrere ρ-Refinementketten von einem Konzept C zu einem Konzept D existieren
- proper (echt) falls $C \rightsquigarrow_\rho D$ impliziert $C \not\equiv_\mathcal{T} D$
- complete (vollständig) falls es für alle $C, D \in \mathcal{C}(\mathcal{L})$ mit $D \sqsubset_\mathcal{T} C$ ein Konzept $E \equiv_\mathcal{T} D$ mit $C \rightsquigarrow_\rho \cdots \rightsquigarrow_\rho E$ gibt
- weakly complete (schwach vollständig) falls für alle $C \in \mathcal{C}(\mathcal{L})$ mit $C \sqsubset_\mathcal{T} \top$ ein Konzept $D \equiv C$ mit $\top \rightsquigarrow_\rho \cdots \rightsquigarrow_\rho D$ gibt



Eigenschaften von Refinementoperatoren

Ein $\mathcal{L}$ downward refinement operator ρ ist

- finite (endlich) falls $\rho(C)$ für jedes $C \in \mathcal{C}(\mathcal{L})$ endlich ist
- redundant falls mehrere ρ-Refinementketten von einem Konzept C zu einem Konzept D existieren
- proper (echt) falls $C \rightsquigarrow_\rho D$ impliziert $C \not\equiv_\mathcal{T} D$
- complete (vollständig) falls es für alle $C, D \in \mathcal{C}(\mathcal{L})$ mit $D \sqsubset_\mathcal{T} C$ ein Konzept $E \equiv_\mathcal{T} D$ mit $C \rightsquigarrow_\rho \cdots \rightsquigarrow_\rho E$ gibt
- weakly complete (schwach vollständig) falls für alle $C \in \mathcal{C}(\mathcal{L})$ mit $C \sqsubset_\mathcal{T} \top$ ein Konzept $D \equiv C$ mit $\top \rightsquigarrow_\rho \cdots \rightsquigarrow_\rho D$ gibt
- ideal = vollständig + echt + endlich



Eigenschaften von Refinementoperatoren

Eigenschaft	**Effekt**
unvollständig	Verpassen von Lösungen
redundant	doppelte Teilbäume, schlechtere Performance
unecht	schlechtere Performance
unendlich	Refinementberechnung terminiert nicht immer

Schlüsselfrage: Welche Eigenschaften können kombiniert werden?

Theorem zu Eigenschaften von $\mathcal{L}$ Refinementoperatoren

Theorem

Maximale kombinierbare Mengen von Eigenschaften von $\mathcal{L}$ Refinementoperatoren für $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCN}, \mathcal{SHOIN}, \mathcal{SROIQ}\}$ sind:

1. *{schwach vollständig, vollständig, endlich}*
2. *{schwach vollständig, vollständig, echt}*
3. *{schwach vollständig, nicht-redundant, endlich}*
4. *{schwach vollständig, nicht-redundant, echt}*
5. *{nicht-redundant, endlich, echt}*

"Foundations of Refinement Operators for Description Logics", J. Lehmann, P. Hitzler, ILP conference, 2008

"Concept Learning in Description Logics Using Refinement Operators", J. Lehmann, P. Hitzler, Machine Learning journal, 2010

Theorem zu Eigenschaften von $\mathcal{L}$ Refinementoperatoren

- keine idealen Operatoren in OWL und den meisten Beschreibungslogiken außer $\mathcal{EL}$-Familie
- Theorem widerlegt teilweise vorher publizierte Resultate
- Indikator dafür dass Lernen in DLs schwierig ist
- Algorithmen sollten Maßnahmen ergreifen um Nachteile der Operatoren auszugleichen
- Ziel 1: Entwicklung eines OWL Refinementoperators nahe an theoretischen Beschränkungen
- Ziel 2: effizienter idealer Refinementoperator für leichtgewichtige $\mathcal{EL}$ Beschreibungslogik

Definition von ρ

$$\rho(C) = \begin{cases} \{\bot\} \cup \rho_\top(C) & \text{if } C = \top \\ \rho_\top(C) & \text{otherwise} \end{cases}$$

$$\rho_B(C) = \begin{cases}
\emptyset & \text{if } C = \bot \\
\{C_1 \sqcup \cdots \sqcup C_n \mid C_i \in M_B\ (1 \leq i \leq n)\} & \text{if } C = \top \\
\{A' \mid A' \in sh_\downarrow(A)\} & \text{if } C = A\ (A \in N_C) \\
\quad \cup \{A \sqcap D \mid D \in \rho_B(\top)\} & \\
\{\neg A' \mid A' \in sh_\uparrow(A)\} & \text{if } C = \neg A\ (A \in N_C) \\
\quad \cup \{\neg A \sqcap D \mid D \in \rho_B(\top)\} & \\
\{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \exists r.D \\
\quad \cup \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\} & \\
\quad \cup \{\exists s.D \mid s \in sh_\downarrow(r)\} & \\
\{\forall r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \forall r.D \\
\quad \cup \{\forall r.D \sqcap E \mid E \in \rho_B(\top)\} & \\
\quad \cup \{\forall r.\bot \mid & \\
\qquad D = A \in N_C \text{ and } sh_\downarrow(A) = \emptyset\} & \\
\quad \cup \{\forall s.D \mid s \in sh_\downarrow(r)\} & \\
\{C_1 \sqcap \cdots \sqcap C_{i-1} \sqcap D \sqcap C_{i+1} \sqcap \cdots \sqcap C_n \mid & \text{if } C = C_1 \sqcap \cdots \sqcap C_n \\
\qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\
\{C_1 \sqcup \cdots \sqcup C_{i-1} \sqcup D \sqcup C_{i+1} \sqcup \cdots \sqcup C_n \mid & \text{if } C = C_1 \sqcup \cdots \sqcup C_n \\
\qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\
\quad \cup \{(C_1 \sqcup \cdots \sqcup C_n) \sqcap D \mid & \\
\qquad D \in \rho_B(\top)\} &
\end{cases}$$

Basisoperator (Auszug)

Definition von ρ

$$\rho(C) = \begin{cases} \{\bot\} \cup \rho_\top(C) & \text{if } C = \top \\ \rho_\top(C) & \text{otherwise} \end{cases}$$

$$\rho_B(C) = \begin{cases}
\emptyset & \text{if } C = \bot \\
\{C_1 \sqcup \cdots \sqcup C_n \mid C_i \in M_B\ (1 \leq i \leq n)\} & \text{if } C = \top \\
\{A' \mid A' \in sh_\downarrow(A)\} & \text{if } C = A\ (A \in N_C) \\
\quad \cup \{A \sqcap D \mid D \in \rho_B(\top)\} & \\
\{\neg A' \mid A' \in sh_\uparrow(A)\} & \text{if } C = \neg A\ (A \in N_C) \\
\quad \cup \{\neg A \sqcap D \mid D \in \rho_B(\top)\} & \\
\{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \exists r.D \\
\quad \cup \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\} & \\
\quad \cup \{\exists s.D \mid s \in sh_\downarrow(r)\} & \\
\{\forall r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \forall r.D \\
\quad \cup \{\forall r.D \sqcap E \mid E \in \rho_B(\top)\} & \\
\quad \cup \{\forall r.\bot \mid & \\
\qquad D = A \in N_C \text{ and } sh_\downarrow(A) = \emptyset\} & \\
\quad \cup \{\forall s.D \mid s \in sh_\downarrow(r)\} & \\
\{C_1 \sqcap \cdots \sqcap C_{i-1} \sqcap D \sqcap C_{i+1} \sqcap \cdots \sqcap C_n \mid & \text{if } C = C_1 \sqcap \cdots \sqcap C_n \\
\qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\
\{C_1 \sqcup \cdots \sqcup C_{i-1} \sqcup D \sqcup C_{i+1} \sqcup \cdots \sqcup C_n \mid & \text{if } C = C_1 \sqcup \cdots \sqcup C_n \\
\qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\
\quad \cup \{(C_1 \sqcup \cdots \sqcup C_n) \sqcap D \mid & \\
\qquad D \in \rho_B(\top)\} &
\end{cases}$$

Basisoperator (Auszug)

Definition von ρ

$$\{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} \quad \text{if } C = \exists r.D$$
$$\cup\ \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\}$$
$$\cup\ \{\exists s.D \mid s \in sh_{\downarrow}(r)\}$$

Beispiele:

$$\exists \texttt{nimmtTeilAn.SozialeZusammenkunft} \rightsquigarrow \exists \texttt{nimmtTeilAn.Meeting}$$

Definition von ρ

$$\{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} \qquad \text{if } C = \exists r.D$$
$$\cup\ \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\}$$
$$\cup\ \{\exists s.D \mid s \in sh_{\downarrow}(r)\}$$

Beispiele:

$$\texttt{∃nimmtTeilAn.SozialeZusammenkunft} \rightsquigarrow \texttt{∃nimmtTeilAn.Meeting}$$

$$\texttt{Student} \sqcap \texttt{∃nimmtTeilAn.SozialeZusammenkunft}$$

Definition von ρ

$$\{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} \quad \text{if } C = \exists r.D$$
$$\cup\ \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\}$$
$$\cup\ \{\exists s.D \mid s \in sh_{\downarrow}(r)\}$$

Beispiele:

$$\texttt{∃nimmtTeilAn.SozialeZusammenkunft} \rightsquigarrow$$
$$\texttt{∃nimmtTeilAn.Meeting}$$
$$\texttt{Student} \sqcap \texttt{∃nimmtTeilAn.SozialeZusammenkunft}$$
$$\texttt{∃leitet.SozialeZusammenkunft}$$

ρ Eigenschaften

- $\rho_\downarrow$ ist vollständig
- $\rho_\downarrow$ ist unendlich, z.B. gibt es unendlich viele Refinementschritte der Form:

$$\top \rightsquigarrow_{\rho_\downarrow} C_1 \sqcup C_2 \sqcup C_3 \sqcup \ldots$$

- $\rho_\downarrow$ ist unecht, aber kann erweitert werden zu einem echten Operator $\rho_\downarrow^{cl}$ (Refinements zeitaufwändiger zu berechnen)
- $\rho_\downarrow$ ist redundant:

$$\begin{array}{ccc} \forall r_1.A_1 \sqcup \forall r_2.A_1 & \rightsquigarrow_{\rho_\downarrow} & \forall r_1.(A_1 \sqcap A_2) \sqcup \forall r_2.A_1 \\ \uparrow_D \downarrow & & \uparrow_D \downarrow \\ \forall r_1.A_1 \sqcup \forall r_2.(A_1 \sqcap A_2) & \rightsquigarrow_{\rho_\downarrow} & \forall r_1.(A_1 \sqcap A_2) \sqcup \forall r_2.(A_1 \sqcap A_2) \end{array}$$


"A Refinement Operator Based Learning Algorithm for the $\mathcal{ALC}$ Description Logic", J. Lehmann, P. Hitzler, ILP conference, 2008


"Concept Learning in Description Logics Using Refinement Operators", J. Lehmann, P. Hitzler, Machine Learning journal, 2010

$\mathcal{EL}$-Operator

Konstruktor	Syntax
top	$\top$
Konjunktion	$C \sqcap D$
existenzielle Restriktion	$\exists r.C$

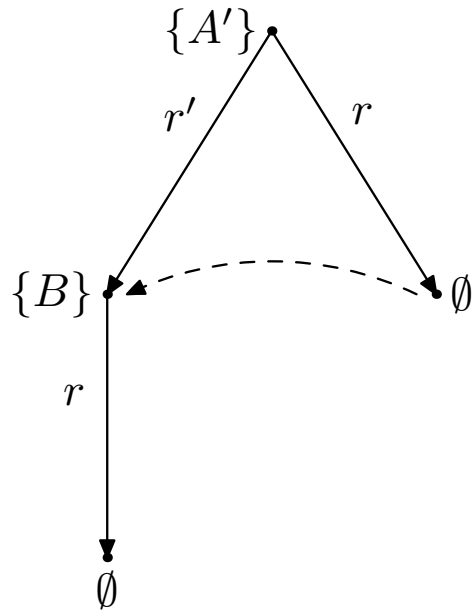
Tabelle: $\mathcal{EL}$ Konzeptkonstruktoren

- motiviert durch niedrige Reasoning-Komplexität von $\mathcal{EL}$ und EL Profil in OWL 2
- $\mathcal{EL}$ weit verbreitet z.B. SNOMED-CT, GEN-ONTOLOGIEN

$\mathcal{EL}$-Operator

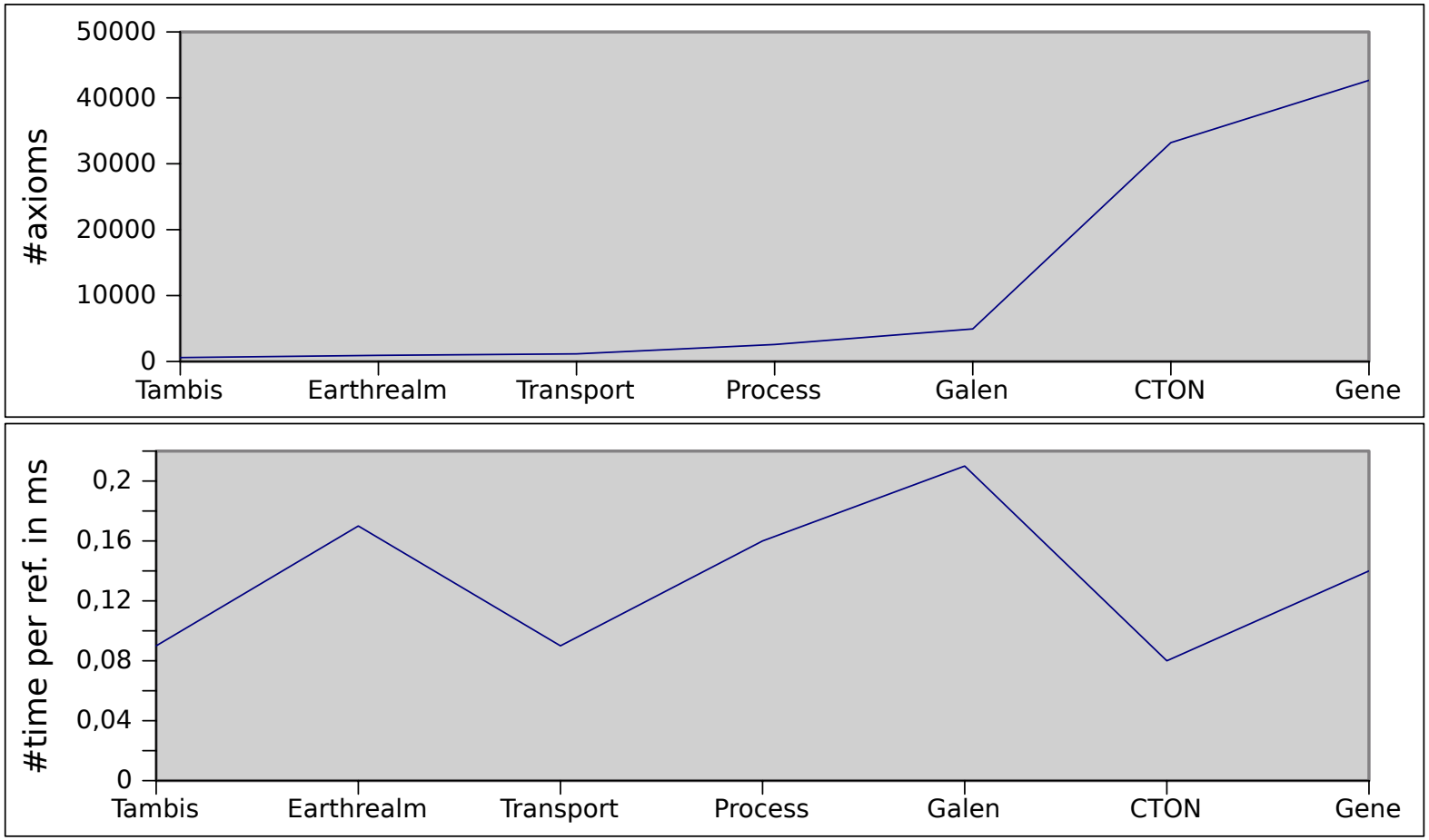
Konstruktor	Syntax
top	$\top$
Konjunktion	$C \sqcap D$
existenzielle Restriktion	$\exists r.C$

Tabelle: $\mathcal{EL}$ Konzeptkonstruktoren



- motiviert durch niedrige Reasoning-Komplexität von $\mathcal{EL}$ und EL Profil in OWL 2
- $\mathcal{EL}$ weit verbreitet z.B. SNOMED-CT, GEN-ONTOLOGIEN
- $\mathcal{EL}$-Konzepte (class expressions) als Bäume repräsentieren
- Reasoning durch sogenannte "Simulationen" auf Bäumen
- Refinement und Reasoning sind stark miteinander verwoben

Operatoreffizienz auf verschiedenen Ontologien



$<$ 1 ms pro Refinement (gemessen über "random refinement chains")

"Ideal Downward Refinement in the EL Description Logic",
J. Lehmann, C. Haase, Int. Conf. on Inductive Logic Programming, 2009

Gliederung

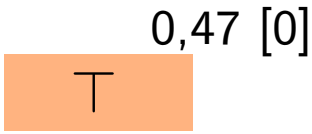
Algorithmen - Übersicht

1. OCEL - OWL Class Expression Learner
2. ELTL - $\mathcal{EL}$ Tree Learner
3. CELOE - Class Expression Learning for Ontology Engineering

OCEL

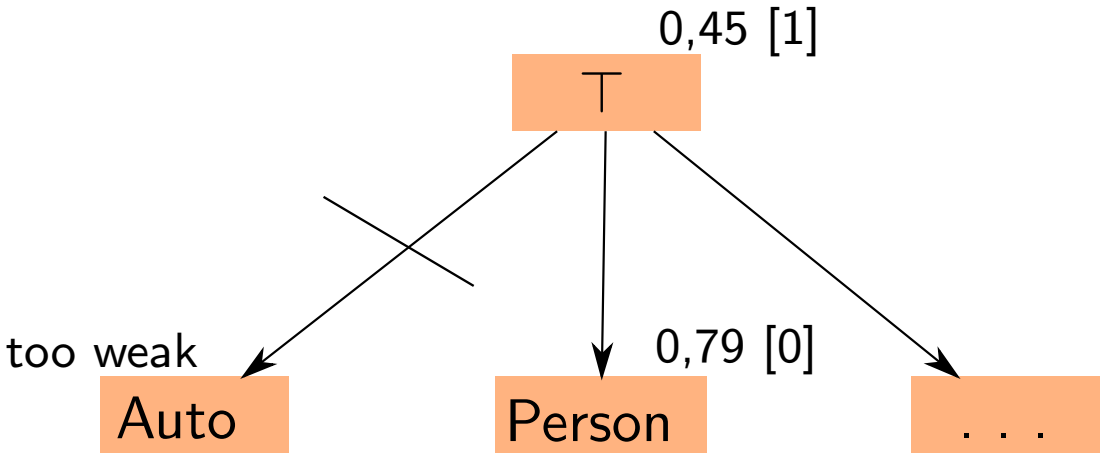
- verwendet ρ für top-down Suche
- OCEL ist vollständig - falls Lösung existiert, kann sie gefunden werden
- stark konfigurierbar, z.B. flexible Wahl von Ziel-/Hypothesensprache, Terminierungskriterium und Heuristik
- implementiert Redundanzeliminierung mit niedriger Komplexität bzgl. Größe des Suchbaums
- kann unendliche Operatoren verwenden durch schrittweise längenbeschränkte Knotenexpansion

Schrittweise Längenbeschränkte Knotenexpansion



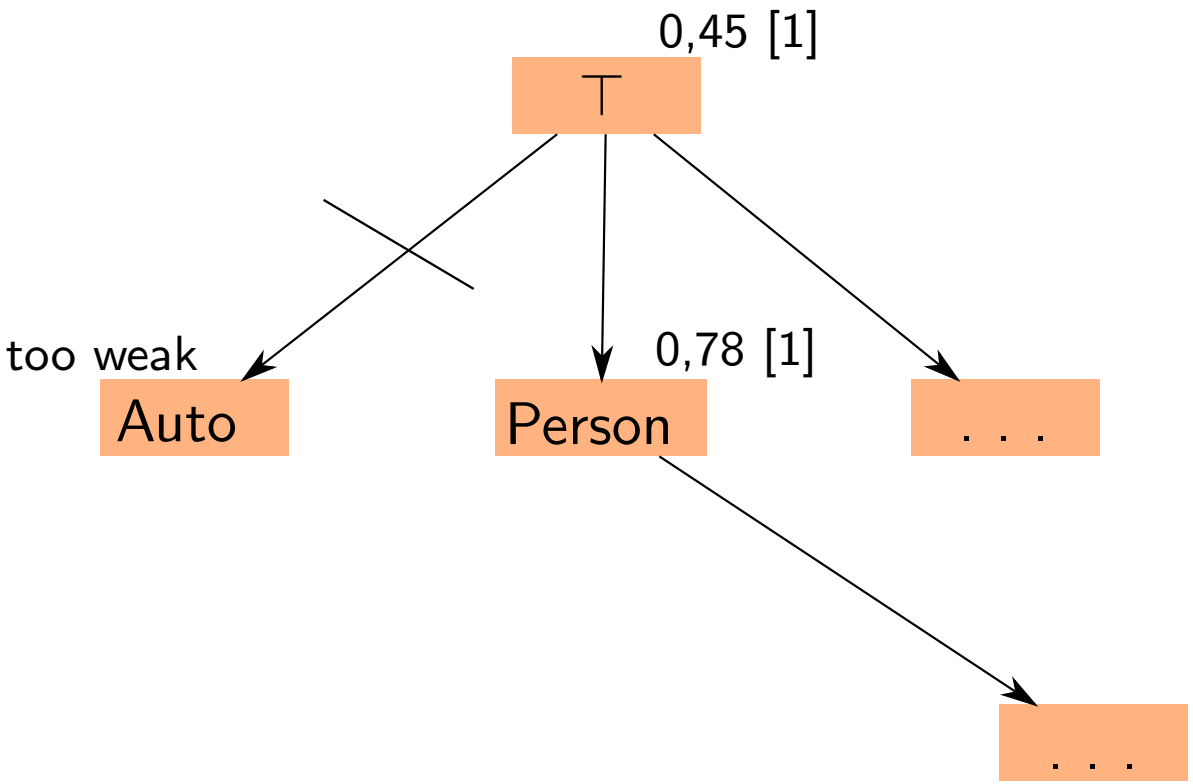
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)

Schrittweise Längenbeschränkte Knotenexpansion



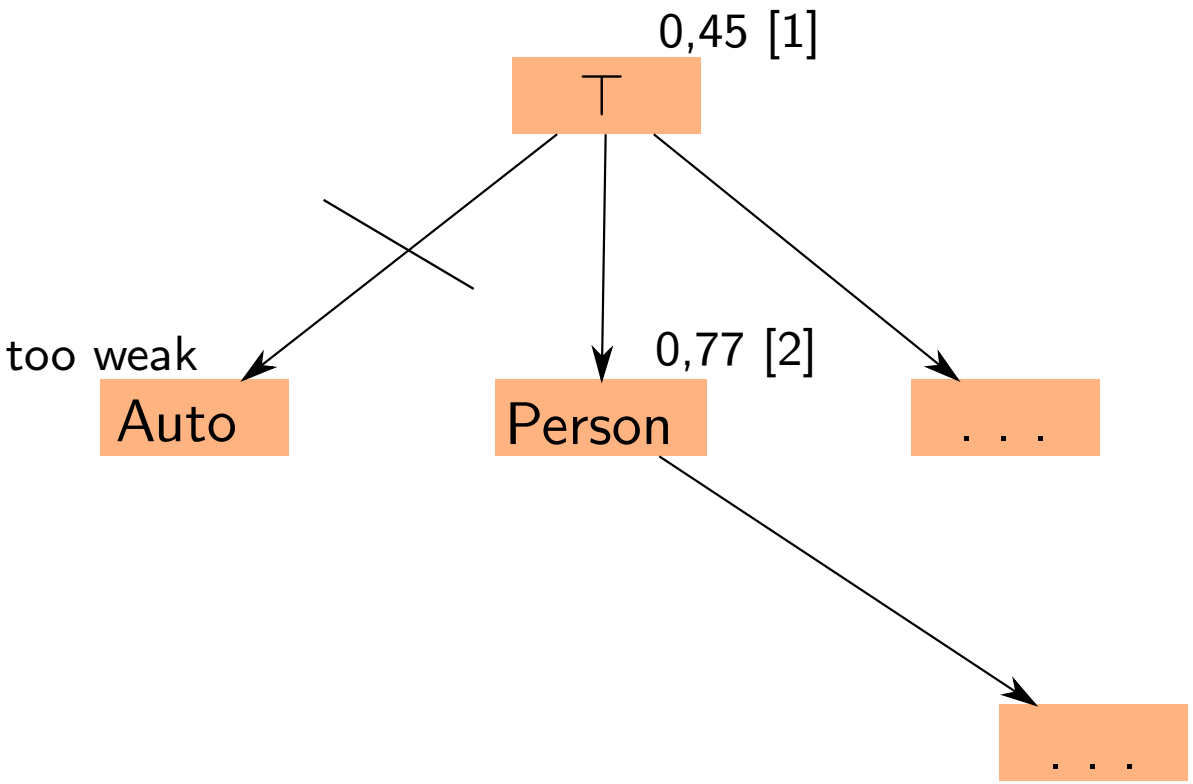
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar

Schrittweise Längenbeschränkte Knotenexpansion



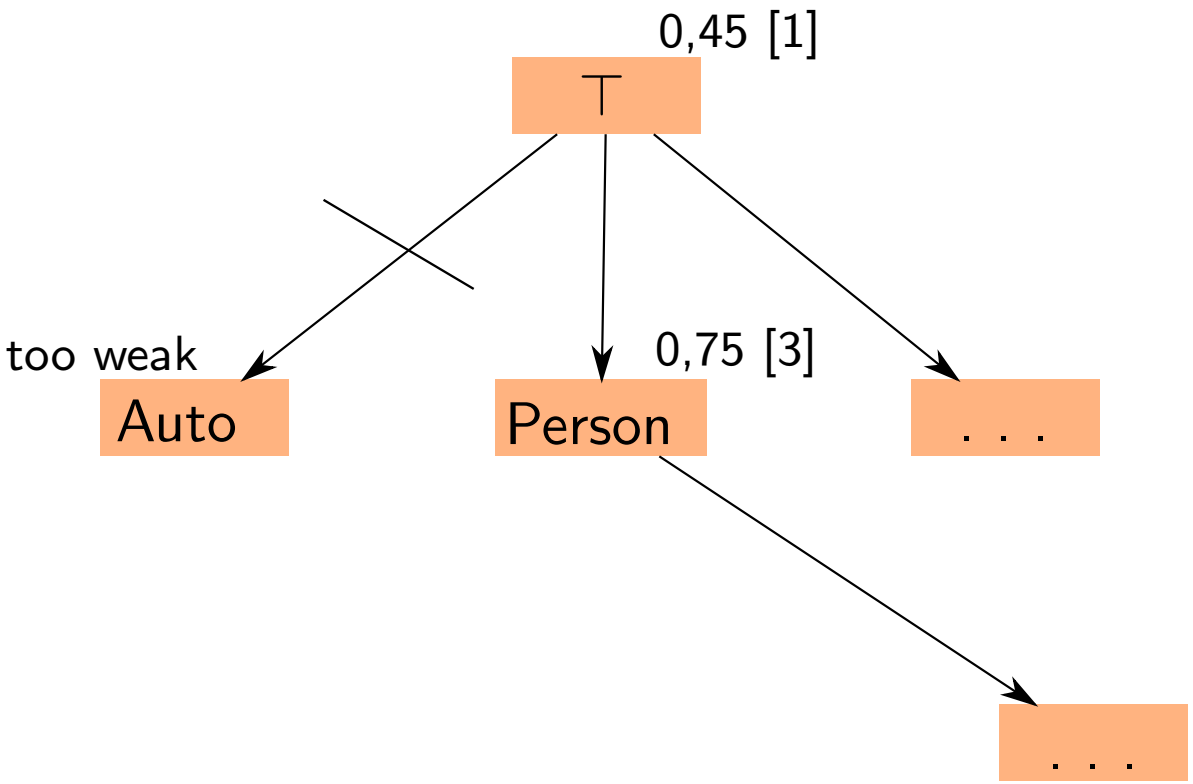
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Schrittweise Längenbeschränkte Knotenexpansion



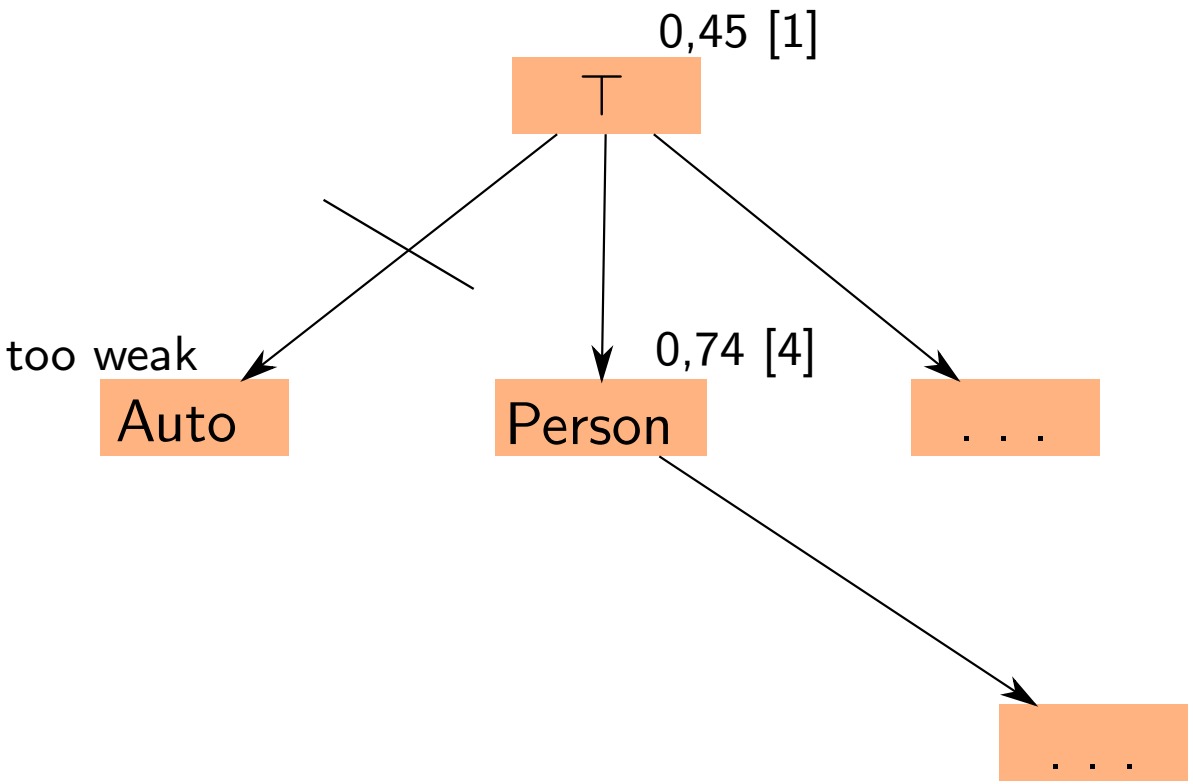
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Schrittweise Längenbeschränkte Knotenexpansion



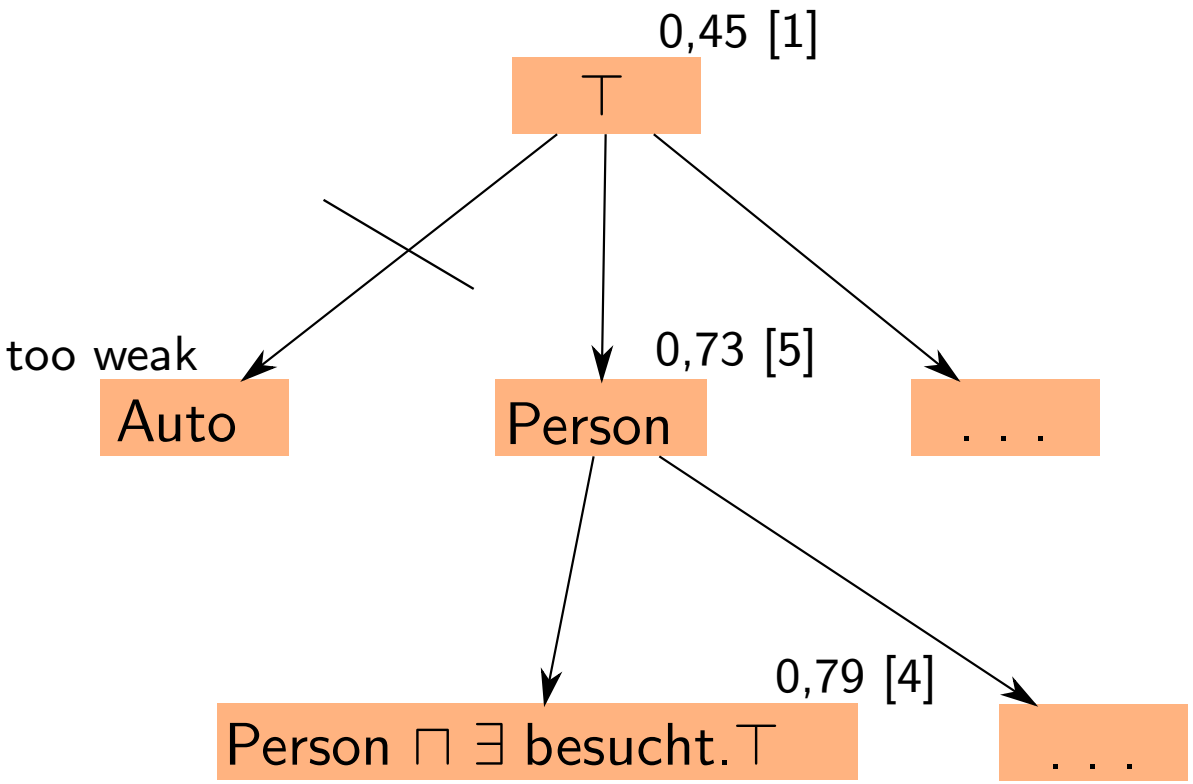
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Schrittweise Längenbeschränkte Knotenexpansion



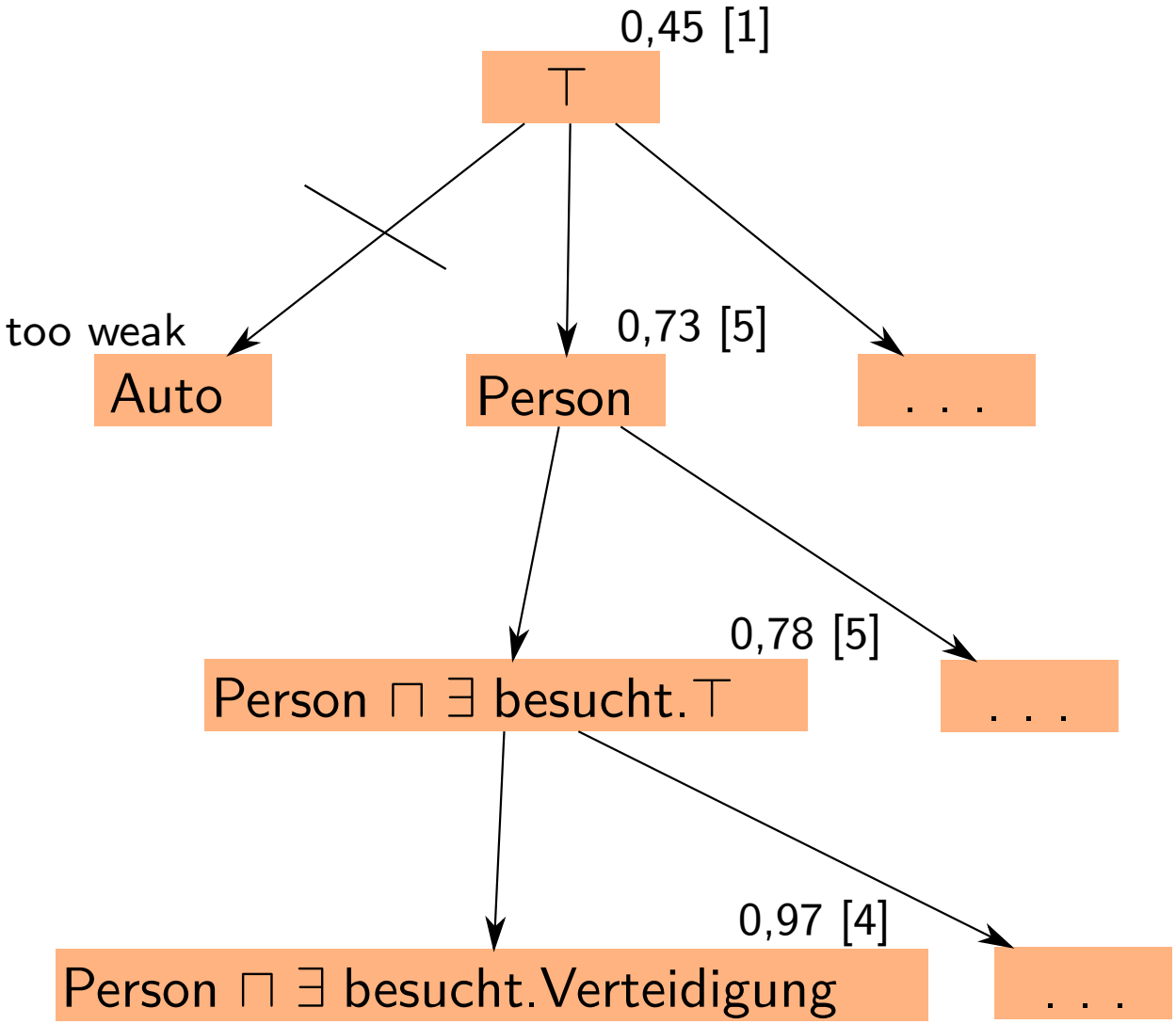
- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Schrittweise Längenbeschränkte Knotenexpansion



- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Schrittweise Längenbeschränkte Knotenexpansion



- Länge der Kinderkonzepte begrenzt durch horizontal expansion (he)
- dadurch ρ (infinite) anwendbar
- he geht in Heuristik ein (Bias zu kurzen Konzepten - Occam's Razor, größere Streuung)

Skalierbarkeit: Reasoning

$$\mathcal{K} = \{\texttt{Männlich} \sqsubseteq \texttt{Person},$$
$$\texttt{Männlich}(\texttt{MICHAEL}), \texttt{Männlich}(\texttt{JONAS}), \texttt{Männlich}(\texttt{PAUL}),$$
$$\texttt{hatKind}(\texttt{MICHAEL}, \texttt{JONAS}), \texttt{hatKind}(\texttt{MICHAEL}, \texttt{PAUL})\}$$

- positives Beispiel: `MICHAEL`
- $C = \texttt{Person} \sqcap \forall \texttt{hatKind}.\texttt{Männlich}$ scheinbar gute Lösung, aber Objekt "`MICHAEL`" keine Instanz von C unter OWA
- Idee: materialisieren von $\mathcal{K}$ mit Standard (OWA) DL-Reasoner, aber Instance Checks unter CWA ausführen
- näher an ML-Intuition und führt zu starker Performanceverbesserung

Heuristiken benötigen oft viele Instance Checks oder Retrieval, z.B.:

$$\frac{1}{2} \cdot \left(\frac{|R(A) \cap R(C)|}{|R(A)|} + \sqrt{\frac{|R(A) \cap R(C)|}{|R(C)|}} \right)$$

Skalierbarkeit: Stochastic Coverage Computation

Heuristiken benötigen oft viele Instance Checks oder Retrieval, z.B.:

$$\frac{1}{2} \cdot \left(\frac{a}{|R(A)|} + \sqrt{\frac{a}{b}} \right)$$

- $|R(A) \cap R(C)|$ und $|R(C)|$ durch abzuschätzende Variablen a und b ersetzen
- Wald-Methode zur Berechnung des 95% Konfidenz-Intervalls
- erst a abschätzen, danach gesamten Ausdruck (beweisbar pessimistische Abschätzung)
- Verfahren anwendbar auf verschiedene Heuristiken

Skalierbarkeit: Stochastic Coverage Computation

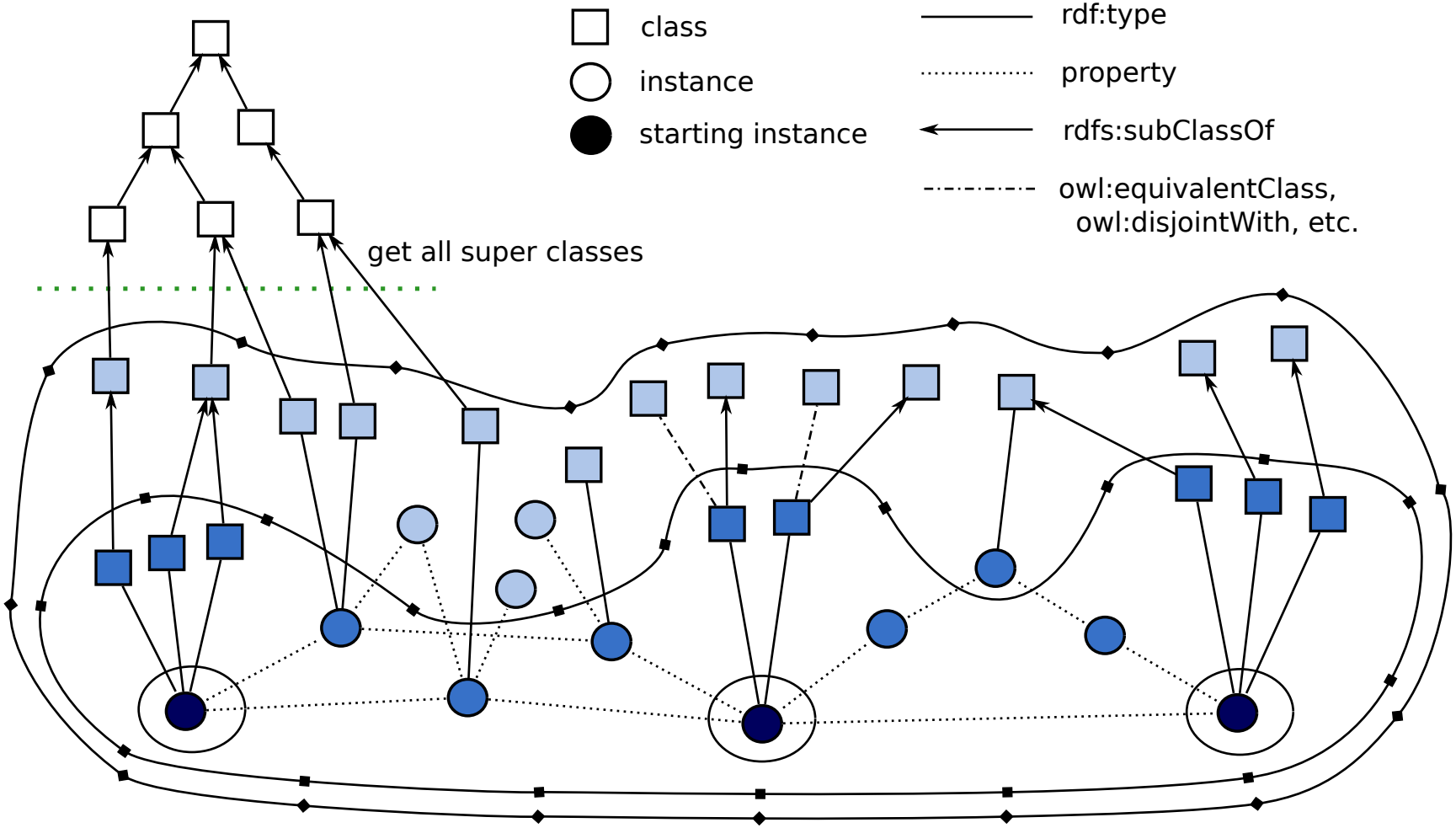
Heuristiken benötigen oft viele Instance Checks oder Retrieval, z.B.:

$$\frac{1}{2} \cdot \left(\frac{a}{|R(A)|} + \sqrt{\frac{a}{b}} \right)$$

- $|R(A) \cap R(C)|$ und $|R(C)|$ durch abzuschätzende Variablen a und b ersetzen
- Wald-Methode zur Berechnung des 95% Konfidenz-Intervalls
- erst a abschätzen, danach gesamten Ausdruck (beweisbar pessimistische Abschätzung)
- Verfahren anwendbar auf verschiedene Heuristiken
- in Tests auf realen Ontologien bis zu 99% weniger Instance Checks und dadurch Algorithmus bis zu 30mal schneller
- geringer Einfluss auf Lernergebnisse empirisch in 380 Lernproblemen auf 7 realen Ontologien nachgewiesen (Abweichung ca. $0,2\% \pm 0,4\%$)

Skalierbarkeit: Fragmentextraktion

Extraktion von Fragmenten aus SPARQL Endpunkten / Linked Data:



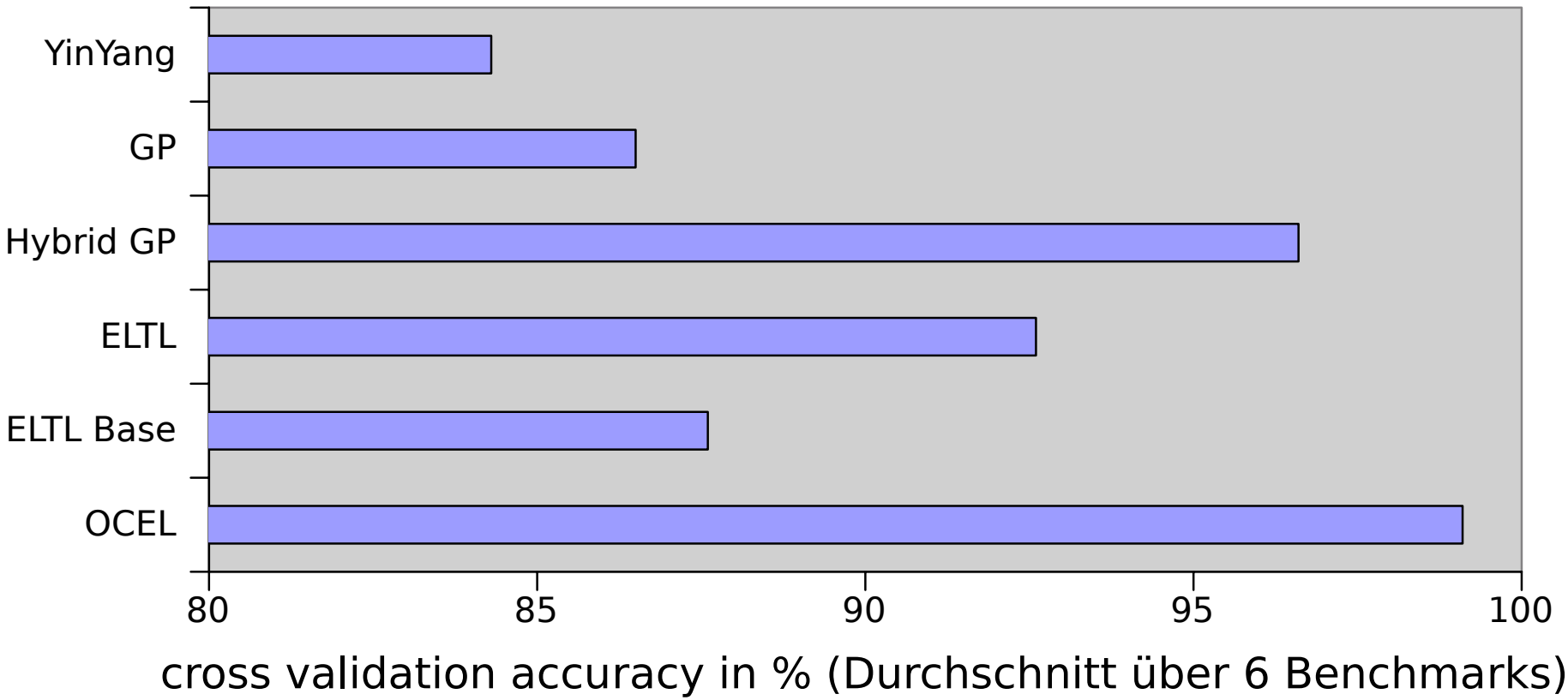
"Learning of OWL Class Descriptions on Very Large Knowledge Bases", Hellmann, Lehmann, Auer, Int. Journal Semantic Web Inf. Syst, 2009

Gliederung

Evaluation

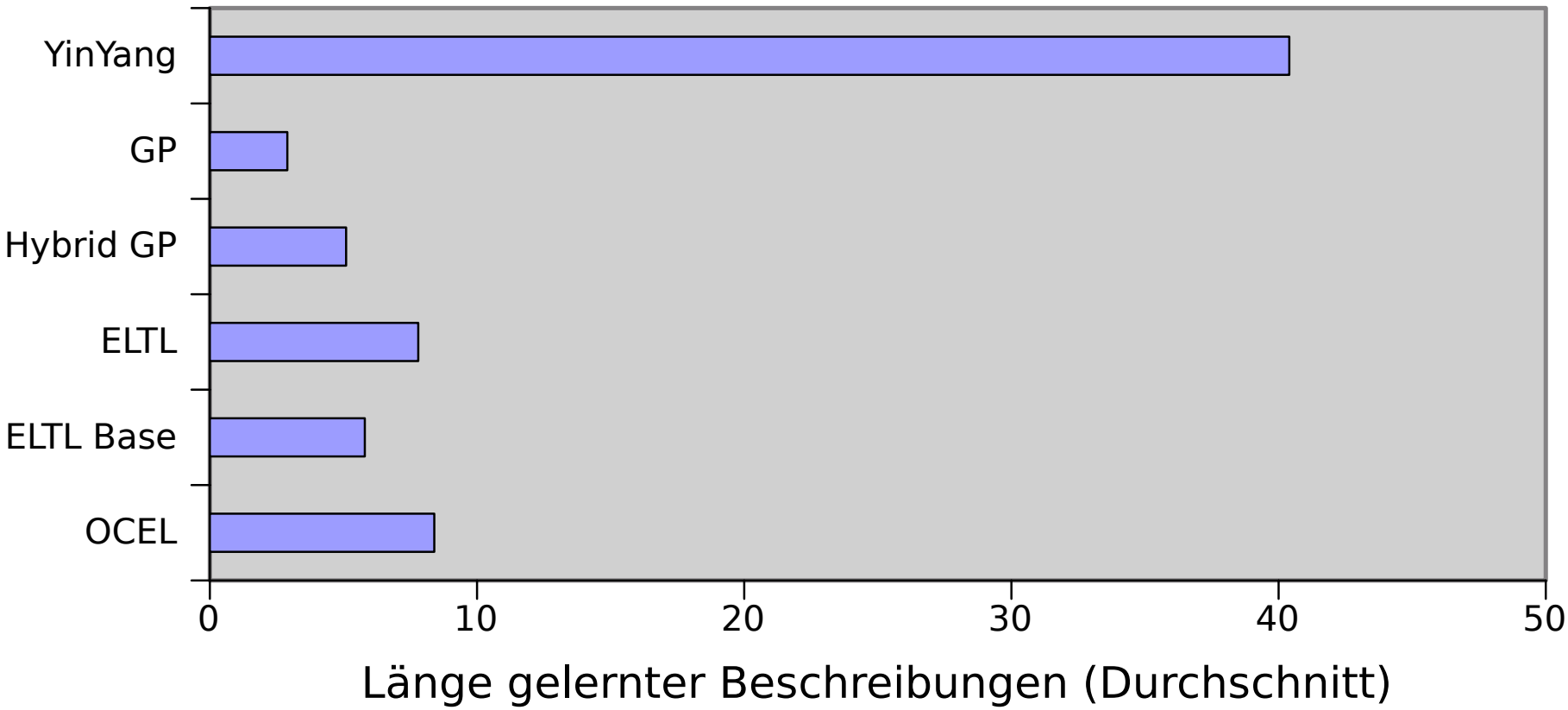
- Mangel an Evaluationsstandards für Lernen in OWL/DL
- Vorgehen: Konvertieren von existierenden Benchmarks nach OWL (zeitaufwändig, erfordert Domänenwissen)
- Messen von predictive accuracy mit ten fold cross validation
- Teil 1: Evaluation gegenüber OWL/DL Lernsystemen
- Teil 2: Evaluation gegenüber anderen ML-Systemen (Krebserkennungsproblem)
- Teil 3: Evaluation im Bereich Ontology Engineering

Evaluation: Genauigkeit/Accuracy



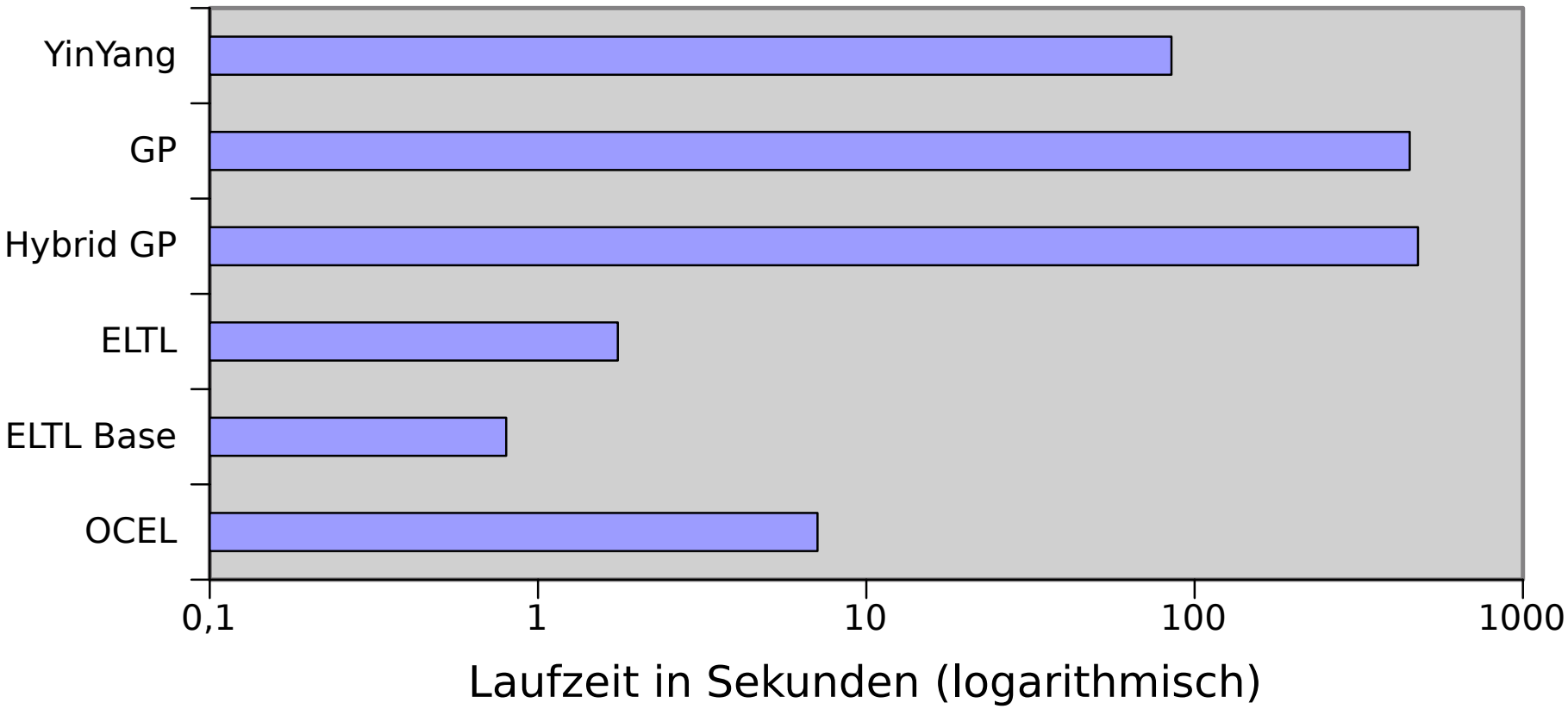
- Zusammenfassung von 6 Benchmarks
- OCEL häufig statistisch signifikant besser als andere Algorithmen für die meisten Benchmarks
- ELTL beschränkt durch Ausdrucksstärke bei einigen Benchmarks

Evaluation: Länge/Lesbarkeit



- YinYang erzeugt statistisch signifikant längere Lösungen

Evaluation: Laufzeit

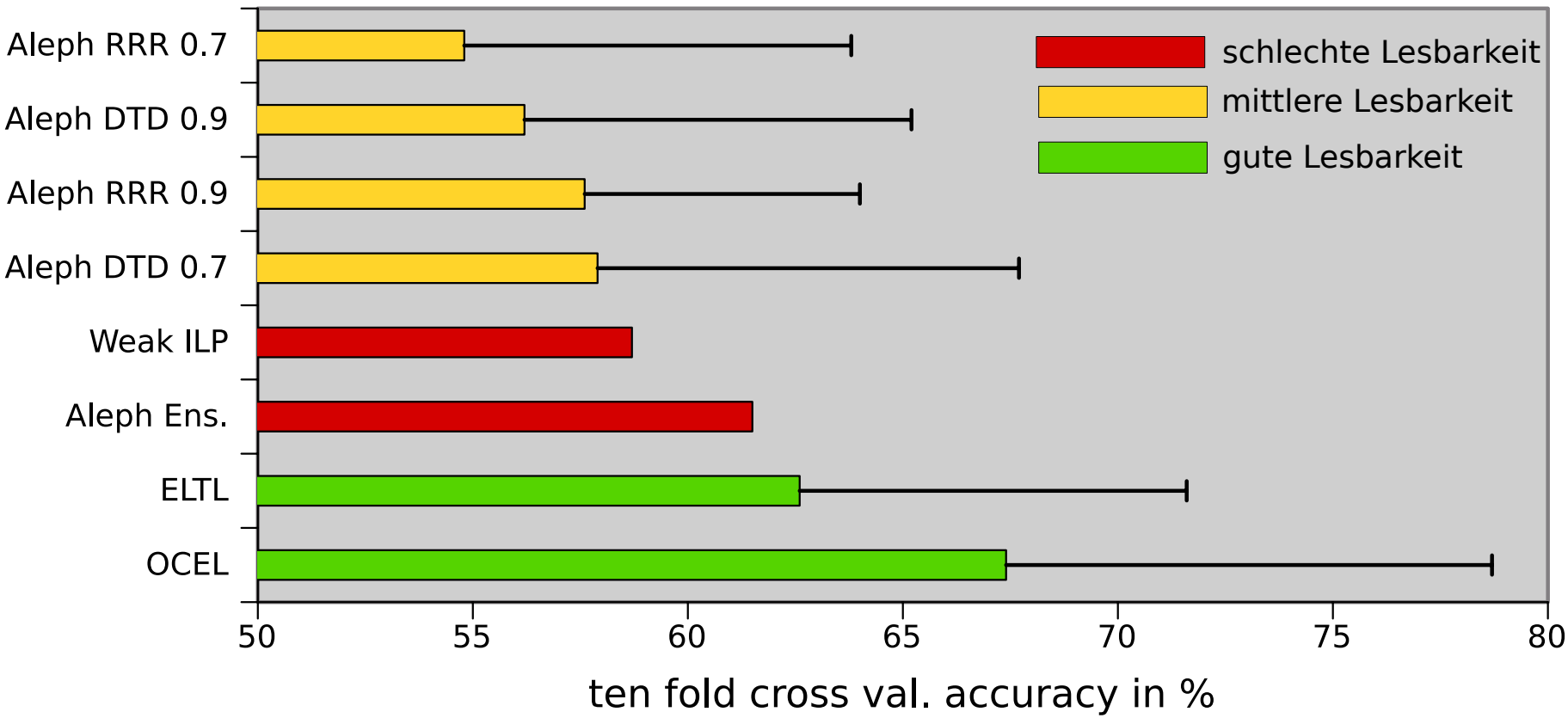


Carcinogenesis

- Ziel: vorhersagen ob Substanzen krebserregend sind
- Begründung:
  - jedes Jahr 1000 neue Substanzen
  - Substanzen können nur teilweise durch langwierige und kostspielige Experimente mit Ratten/Mäusen auf Krebsrisiko untersucht werden
- Hintergrundwissen:
  - Datenbank des US National Toxicology Program (NTP)
  - Konvertiert von Prolog zu OWL

"Obtaining accurate structural alerts for the causes of chemical cancers is a problem of great scientific and humanitarian value." (A. Srinivasan, R.D. King, S.H. Muggleton, M.J.E. Sternberg 1997)

Carcinogenesis



- sehr schwieriges Problem: niedrige Genauigkeit, hohe Standardabweichung
- OCEL stat. signifikant besser als die meisten anderen Ansätze

Ontology Engineering Evaluation

- initialer Test:
  - zwei Masterstudenten mit Einarbeitungszeit in verschiedene Domänen
  - Test auf 7 realen Ontologien
  - 383 Lernprobleme
  - für 60% der Probleme konnte CELOE akzeptierte Vorschläge zur Erweiterung der Ontologie machen
  - 4197 neue Instanzzuordnungen
  - 2 "versteckte Inkonsistenzen" aufgedeckt
- momentan: umfangreicherer Test mit Protégé Evaluationsplugin

Gliederung

DL-Learner Projekt

- DL-Learner Open-Source-Projekt: `http://dl-learner.org`, `http://sf.net/projects/dl-learner`
- leicht erweiterbare Plattform für verschiedene Lernprobleme und Lernansätze (4 Komponententypen, 21 Komponenten)
- mehrere Interfaces: Kommandozeile, GUI, Web-Service
- unterstützt gängige OWL-Formate und SPARQL Endpunkte
- erlaubt verschiedene Reasoner-Anbindungen (OWL API, DIG)
- sourceforge.net (Open Source Portal): 3500 Downloads
- mloss.org (ML & Open Source Software): 1400 Downloads

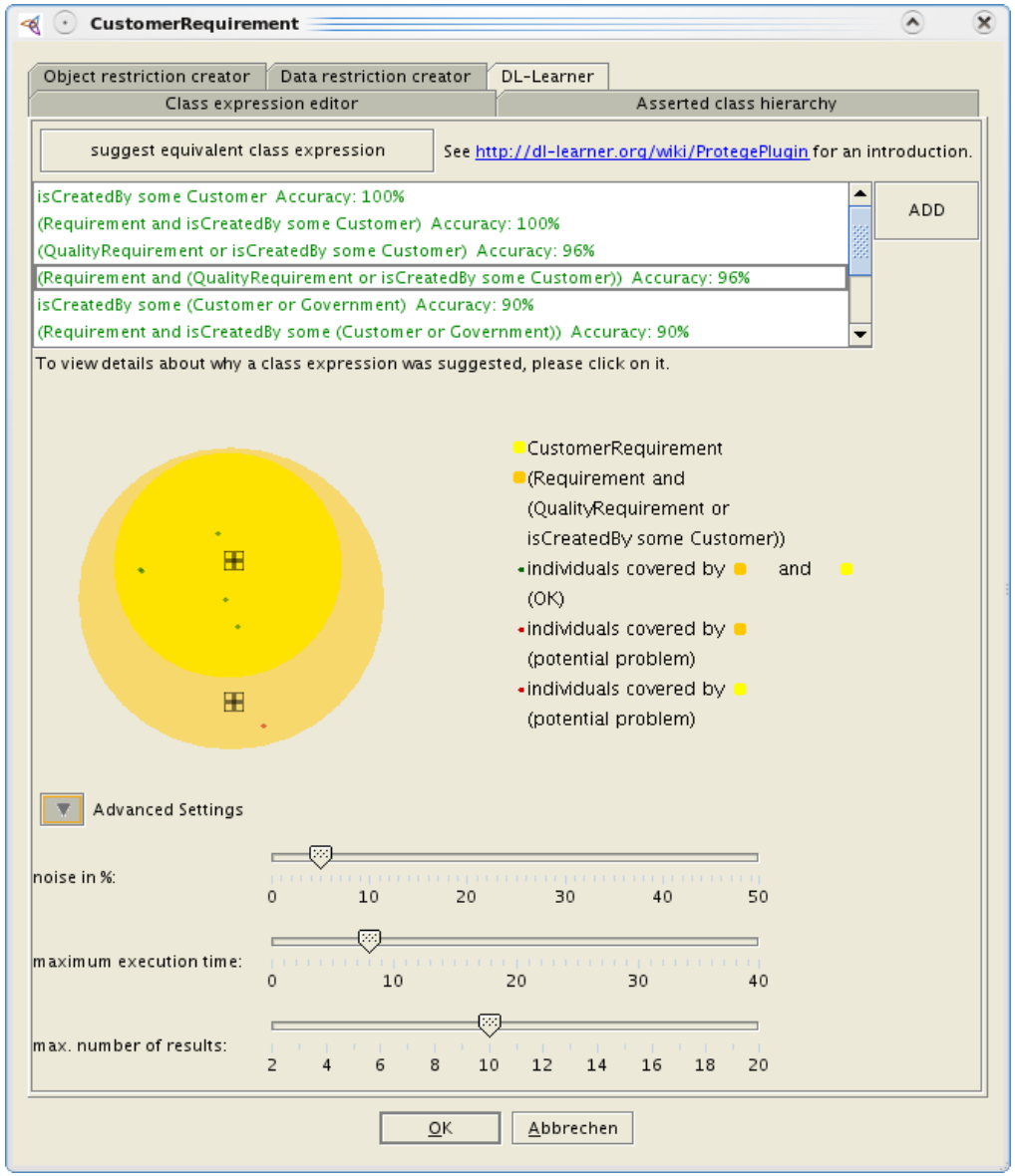
"DL-Learner: Learning Concepts in Description Logics",
Jens Lehmann, Journal of Machine Learning Research (JMLR), 2009

Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)

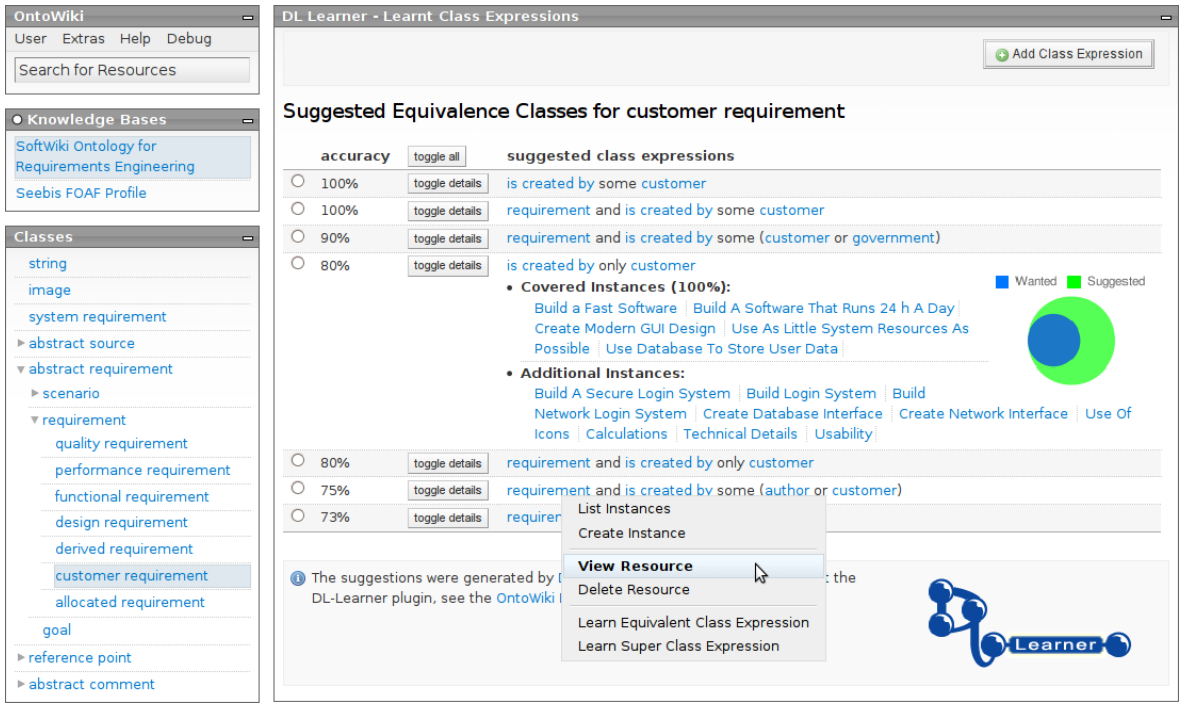
Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin



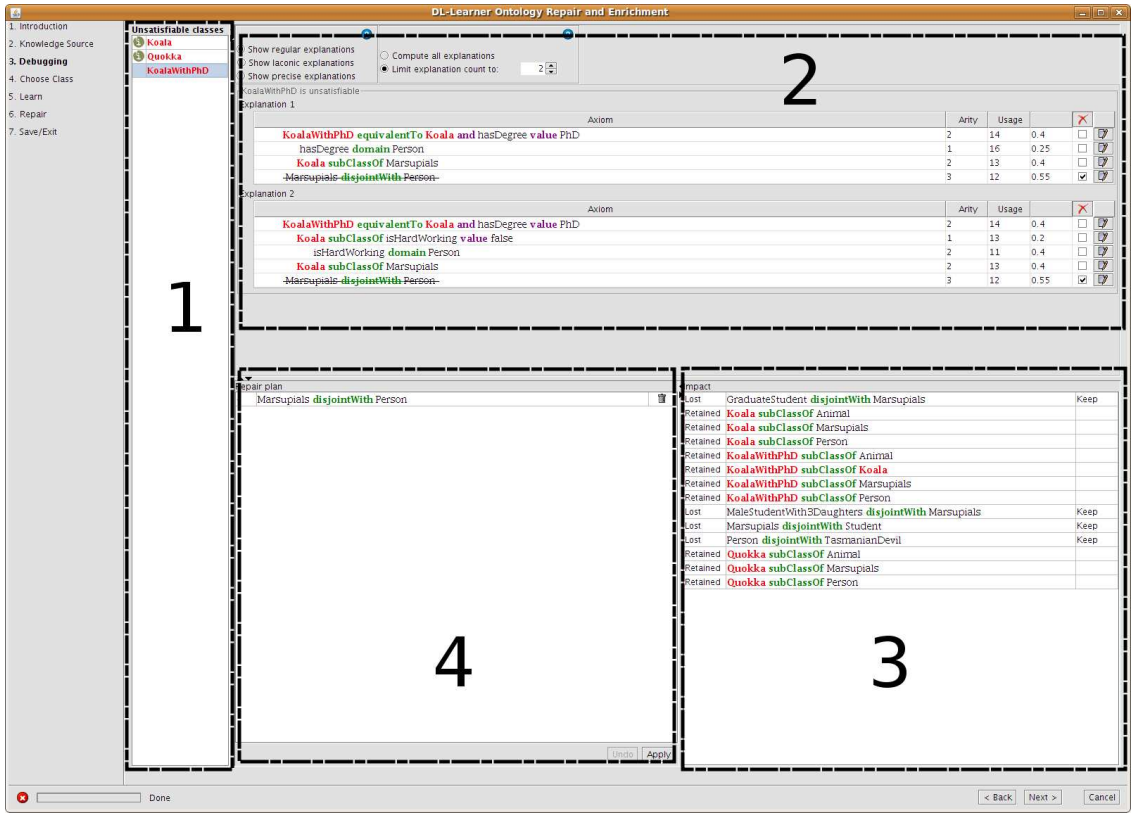
Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin
  - OntoWiki Plugin



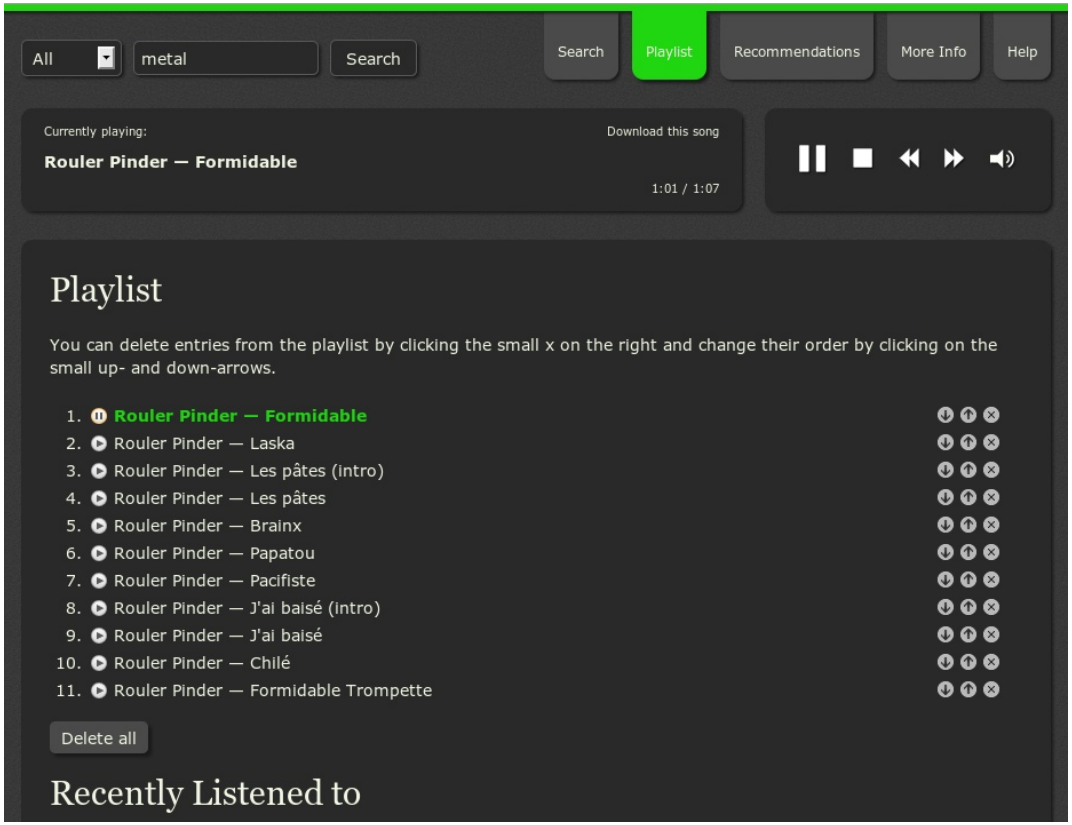
Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin
  - OntoWiki Plugin
  - ORE



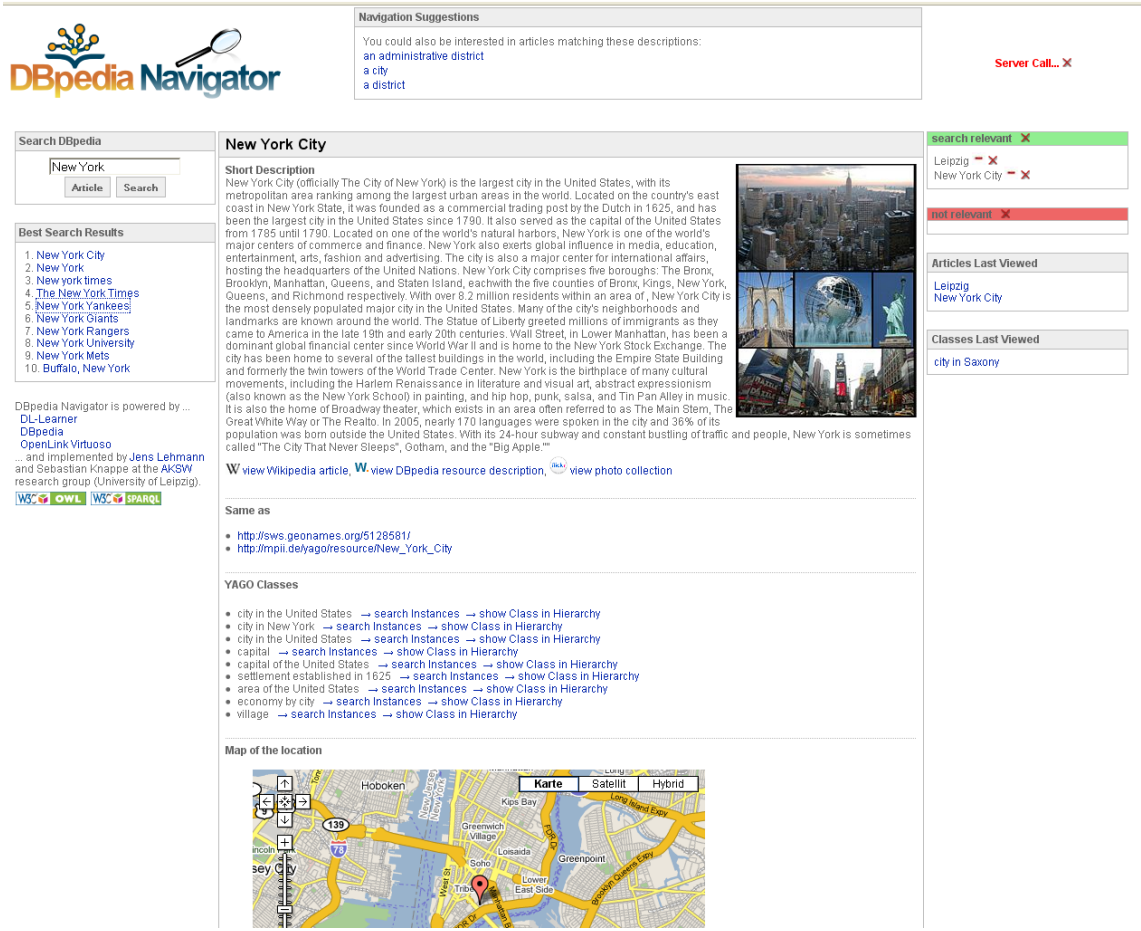
Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin
  - OntoWiki Plugin
  - ORE
- Empfehlungen/Navigation
  - moosique.net



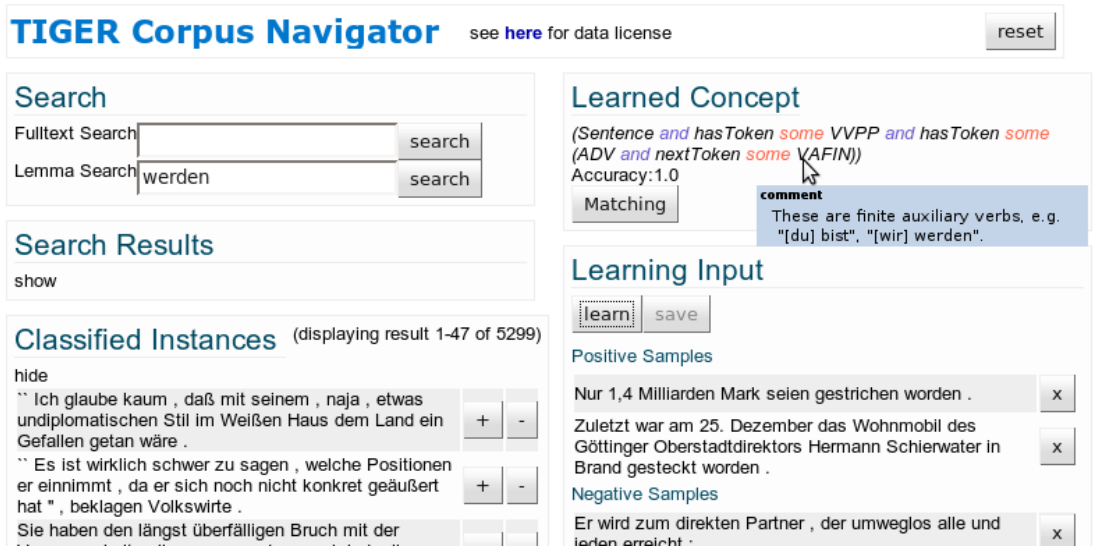
Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin
  - OntoWiki Plugin
  - ORE
- Empfehlungen/Navigation
  - moosique.net
  - DBpedia Navigator



Anwendungen

- „klassische“ ML-Probleme
  - krebserregende Substanzen
  - Erbkrankheiten (Mutation)
- Ontology Engineering
  - Protégé Plugin
  - OntoWiki Plugin
  - ORE
- Empfehlungen/Navigation
  - moosique.net
  - DBpedia Navigator
- nicht betreut:
  - ISS (Gerken et al.)
  - Learning in Probabilistic DLs (Ochoa Luna et al.)
  - TIGER Corpus Navigator (Hellmann et al.)



Gliederung

Zusammenfassung

- theoretische Analyse der Eigenschaften von DL-Refinementoperatoren
- Entwicklung von zwei Operatoren und Beweis ihrer Eigenschaften
- drei Algorithmen mit Erweiterungen gegenüber dem State of the Art
- signifikante Skalierbarkeitsverbesserungen
- Evaluation mit mehreren Benchmarks, die in OWL konvertiert worden
- Implementation im Open Source Projekt DL-Learner
- Entwicklung von mehreren Anwendungen/Plugins

Weitere Arbeit

- Lösen weiterer schwieriger ML-Probleme mit DL-Learner Framework
- Untersuchen von bottom-up Ansätzen
- Nutzung der Techniken für Matching von Instanzen in Ontologien
- Integration von NLP Techniken
- breitere Evaluation des Ontology Engineering Use Cases
- Buch "Perspectives of Ontology Learning" (mit Johanna Völker)
- allgemein: weiterhin Anwendungsbereich von ML/ILP-Techniken auf das Semantic Web erweitern

Danke für Ihre Aufmerksamkeit!

ρ *Eigenschaften*

- verwendet Konzept- und Rollenhierarchie im Gegensatz zu anderen Operatoren
- verwendet Domain und Range von Rollen (ähnlich zu "mode declarations" in ILP-Programmen) $\rightarrow$ reduziert Suchraum
- endliche Anzahl von Reasoneranfragen = durch einen Cache wird der Reasoner nur in der Aufwärmphase vom Operator verwendet
- unterstützte Konstrukte: $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$, $\top$, $\bot$
- existierende Erweiterungen: Zahlenrestriktionen (min/max), hasValue-Konstruktur, Datentypen: boolean, double, string

ELTL

- verwendet idealen Refinementoperator ψ
- (optionale) Verwendung eines “covering” Ansatzes d.h. schrittweise Erstellung einer Disjunktion
- vereinfachen von Lösungen (über OWL-Reasoner) um Lesbarkeit weiter zu erhöhen
- geeignet für einfache Lernaufgaben und Ontologien im EL Profil von OWL 2

CELOE

- Adaptierung von OCEL für Ontology Engineering
- verwendet existierende Klassen als Eingabe und nutzt deren Instanzen für überwachtes Lernen
- kann existierende Definitionen wiederverwenden (durch initiale upward refinement Phase)
- Heuristik optimiert für breite Suche (verpasst häufiger komplexe Lösungen, aber findet fast immer einfache Lösungen)

Carcinogenesis