Ontology Reasoning & Inconsistency

Übersicht

Was ist Inkonsistenz - Was ist das Problem?

Quellen für Inkonsistenzen (Ursachen)

Umgang mit Inkonsistenzen

- Schließen unter Inkonsistenz
- Inkonsistenz beheben

"Real-World"-Beispiele

Zuerst: Beschränkung auf OWL-DL

äquivalent zu *SHOIN* Beschreibungslogik

- Entscheidbar
- Fragment von *FOL*
- NExpTime aber reale Algorithmen zufriedenstellend

De facto Standard Ontologie-Sprache

- Vollständige Unterstützung durch Tools
- Stabiler Standard

Notation hier im Beschreibungslogik-Stil

Inkonsistente Ontologie

Semi-Formal: Eine Ontologie ist inkonsistent falls ein Konzept unerfüllbar ist d.h. die Interpretation immer leer ist.

Einfacher: Wenn Sie einen Widerspruch enthält

Fehlerhafte Terminologie (T-Box)

Fehlerhafte Fakten (A-Box)

Oft Beschränkung auf Fehlersuche in T-Box da Hinweis auf grundlegende Modellierungsfehler

Inkonsistente Ontologie - Beispiel

Inkonsistente T-Box:

$bird \sqsubseteq animal$ (All Birds are animals)

$bird \sqsubseteq fly$ (All birds can fly)

$penguin \sqsubseteq bird$

$penguin \sqsubseteq \neg fly$

Inkonsistente Ontologie - Beispiel

Inkonsistenz in der A-Box

$$tauchen(PeterPinguin)$$
$$vogel(PeterPinguin)$$
$$tauchen \sqsubseteq \neg\, vogel \text{ (disjunkte Konzepte)}$$

Inkonsistente Ontologie - Problem?

Kann PeterPinguin tauchen?

Antwort: Beides? Keines? Ja/Nein/Vielleicht?

Schlimmer noch: Standard Schließen ist Explosiv!

- Beliebiges lässt sich aus inkonsistenter Ontologie ableiten da
  $\Sigma \models \varphi \Leftrightarrow \Sigma \cup \{\neg\varphi\}$ unerfüllbar. Und
  Σ inkonsistent => Σ unerfüllbar => $\Sigma \cup \{\neg\varphi\}$ unerfüllbar.

Quellen von Inkonsistenzen

- Semantic WEB an sich, da
  - skalierbar
  - verteilt
  - viele Autoren
- Verschiedene Versionen einer Ontologie
  - Jede Version sei Konsistent
  - Relationen zwischen den Versionen müssen es nicht sein
- "Ungenaue" Modellierung (Es gibt Vögel die tauchen)
- Mehrdeutigkeit von Konzepten (Problem selbst für Experten)

Quellen von Inkonsistenzen

- Migration aus anderer Sprache
  - Weil Frame-Logic nicht mehr hip ist?
- Inkonsistenzen sprachlich unmöglich machen?
  - Kein Problem, nur dann verliert man unheimlich viel Ausdrucksstärke und ist nicht viel weiter als Google heute
  - z.B. Negation und Disjunktion sind essentiell für bedeutungsvolles Schließen aber Quelle vieler Inkonsistenzen

Umgang mit Inkonsistenzen

Schließen unter Inkonsistenzen

- Nicht-klassisches Schließen, 4-wertige Logik und sinnvolle Antworten

Inkonsistenz beheben

- Diagnosen und Pinpoint

Umgang - Inkonsistenz Lokalisieren

Lokalisieren mit üblichen Reasoner (z.B. Pellet)

$$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3 \qquad ax_2 : A_2 \sqsubseteq A \sqcap A_4$$
$$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5 \qquad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$$
$$ax_5 : A_5 \sqsubseteq \exists s.\neg B \qquad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$$
$$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$$

Reasoner gibt unerfüllbare Konzepte:

$$\{A_1, A_3, A_6, A_7\}$$

Aber es fehlen wichtige Informationen zu Abhängigkeiten der unerfüllbaren Konzepte - Des Ursprungs der Unerfüllbarkeit

Umgang - Schließen unter Inkonsistenz

„damit leben“

- u.U. einzige verbleibende Methode im Semantic Web
- importieren aus anderen Quellen
- (zu) große Ontologie
- fehlende Rechte

Antworten mit nicht-klassischem Reasoner

- Ziel: aussagekräftige Antwort trotz vorhandenen Inkonsistenzen

Schließen unter Inkonsistenz
4-wertige Logik & Nicht-klassischer Schluss

Klassische Logik $\Sigma \models \phi$

- Antwort: Aussage ist wahr oder falsch

Belnaps 4-wertige Logik

widersprüchlich: $\Sigma \mid\!\approx \phi \; and \; \Sigma \mid\!\approx \neg\phi$

wahr: $\Sigma \mid\!\approx \phi \; and \; \Sigma \not\mid\!\approx \neg\phi$

falsch: $\Sigma \not\mid\!\approx \phi \; and \; \Sigma \mid\!\approx \neg\phi$

unbekannt: $\Sigma \not\mid\!\approx \phi \; and \; \Sigma \not\mid\!\approx \neg\phi$

Nicht-klassischer Schluss $\mid\!\approx$

Schließen unter Inkonsistenz

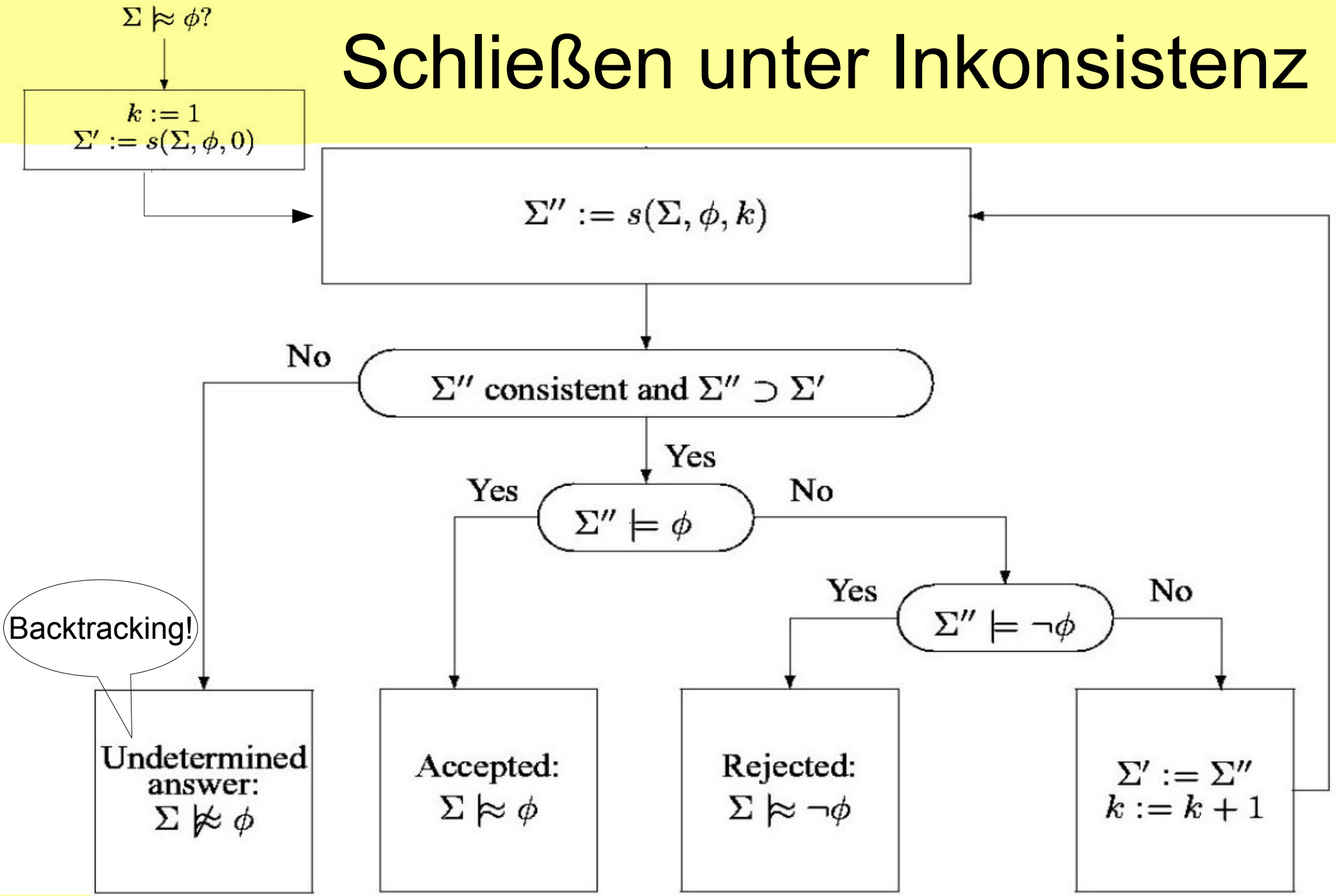
Von der Anfrage ausgehend

(1) Konsistente Sub-Ontologie mittels einer Selektions-Funktion s auswählen

(2) Mittels Standard-Schließen versuchen eine sinnvolle Antwort zu bekommen

(3) Falls es keine zufriedenstellende Antwort gibt die Sub-Ontologie erweitern und goto:(2)

Schließen unter Inkonsistenz



Schließen unter Inkonsistenz - Sinnvolle Antworten

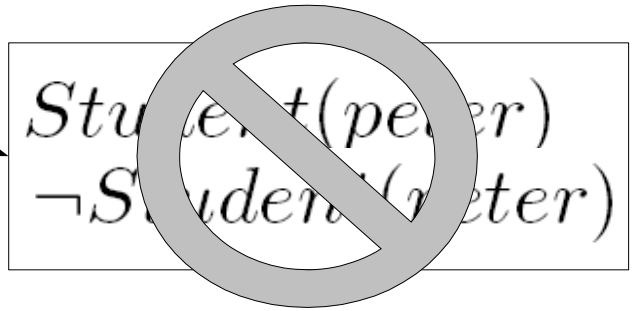
Antwort ist sinnvoll (meaningfull) wenn

- Sie mittels klassischem Schluss aus konsistenter Subontologie folgt
- Sie eine konsistente Antwort mittels nicht-klassischem Schluss ist - d.h.

$$T \mid\!\approx \phi \Rightarrow T \not\mid\!\approx \neg\phi$$

Schließen unter Inkonsistenz - Beispiel

$\models Student(peter)?$



$\{Employee \sqsubseteq Person,$
$Student \sqsubseteq Person,$
$PhDStudent \sqsubseteq Student,$
$PhDStudent \sqsubseteq Employee,$
$Employee \sqsubseteq \neg Student,$
$PhDStudent(peter)\}.$

$\not\approx Student(peter)?$

$\{PhDStudent(peter),$
$PhDStudent \sqsubseteq Employee,$
$PhDStudent \sqsubseteq Student\}.$

$\not\approx Student(peter)$

Umgang - Inkonsistenzen beheben

Diagnose

- Die Wurzel der Inkonsistenz finden

Debugging

- effiziente "Diagnosen"

MUPS → MIPS → Pinpoints

Reperatur

Umgang - Inkonsistenzen beheben

Inkonsistente T-Box:

$$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3 \qquad ax_2 : A_2 \sqsubseteq A \sqcap A_4$$
$$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5 \qquad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$$
$$ax_5 : A_5 \sqsubseteq \exists s.\neg B \qquad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$$
$$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$$

Reasoner gibt unerfüllbare Konzepte:

$$\{A_1, A_3, A_6, A_7\}$$

Aber es fehlen wichtige Informationen zu Abhängigkeiten der Konzepte.

Inkonsistenzen beheben - Diagnose

Gesucht ist eine minimale Teilmenge (Diagnose) einer T-Box die die Inkonsistenz der T-Box verantwortet.

Also die Menge, die weggelassen, eine konsistente T-Box hinterlässt.

Problem: Erstellen der Diagnosen ist mit extrem hohen Kosten verbunden (naiv 2^n Teilmengen zu testen) deshalb heuristische Algorithmen

→ Pinpoints

Inkonsistenzen beheben - Diagnose

$$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3 \qquad ax_2 : A_2 \sqsubseteq A \sqcap A_4$$
$$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5 \qquad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$$
$$ax_5 : A_5 \sqsubseteq \exists s.\neg B \qquad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$$
$$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$$

Unerfüllbare Konzepte:

$$\{A_1, A_3, A_6, A_7\}$$

Menge der Diagnosen der Beispiel T-Box:

$$\Delta_{\mathcal{T}1} = \{\{ax_1, ax_4\}, \{ax_2, ax_4\}, \{ax_1, ax_3, ax_7\}, \{ax_2, ax_3, ax_7\}, \{ax_1, ax_5, ax_7\}, \{ax_2, ax_5, ax_7\}\}$$

Inkonsistenzen beheben - Debugging

MUPS (Minimal Unsatisfiability-Preserving Sub-T-Boxes)

- Was ist verantwortlich für Unerfüllbarkeit eines Konzeptes

MIPS (Minimal Incoherence-Preserving Sub-T-Boxes)

- Was ist verantwortlich für Inkonsistenz einer T-Box

Pinpoint

- Wo muss man bestmöglichst ansetzen um die Konsistenz der T-Box wieder herzustellen

Inkonsistenzen beheben - MUPS

$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3$
$ax_2 : A_2 \sqsubseteq A \sqcap A_4$
$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5$
$ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$
$ax_5 : A_5 \sqsubseteq \exists s.\neg B$
$ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$
$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$

$mups\ (\mathcal{T}_1, A_1) : \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$

$mups\ (\mathcal{T}_1, A_3) : \{\{ax_3, ax_4, ax_5\}\}$

$mups\ (\mathcal{T}_1, A_6) : \{\{ax_1, ax_2, ax_4, ax_6\}, \{ax_1, ax_3, ax_4, ax_5, ax_6\}\}$

$mups\ (\mathcal{T}_1, A_7) : \{\{ax_4, ax_7\}\}$

Inkonsistenzen beheben - MIPS

$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3$ $\quad ax_2 : A_2 \sqsubseteq A \sqcap A_4$

$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5$ $\quad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$

$ax_5 : A_5 \sqsubseteq \exists s.\neg B$ $\quad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$

$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$

$mups\ (\mathcal{T}_1, A_1) : \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$

$mups\ (\mathcal{T}_1, A_3) : \{\{ax_3, ax_4, ax_5\}\}$

$mups\ (\mathcal{T}_1, A_6) : \{\{ax_1, ax_2, ax_4, ax_6\}, \{ax_1, ax_3, ax_4, ax_5, ax_6\}\}$

$mups\ (\mathcal{T}_1, A_7) : \{\{ax_4, ax_7\}\}$

$\bigcup$

$$mips(\mathcal{T}_1) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$$

MIPS ist Vereinigung aller MUPS mit anschließender subset-reduction

Inkonsistenzen beheben - Pinpoints

Intuitiv: Je öfter ein Axiom in MIPS vorkommt desto 'verantwortlicher' ist es für die Inkohärenz

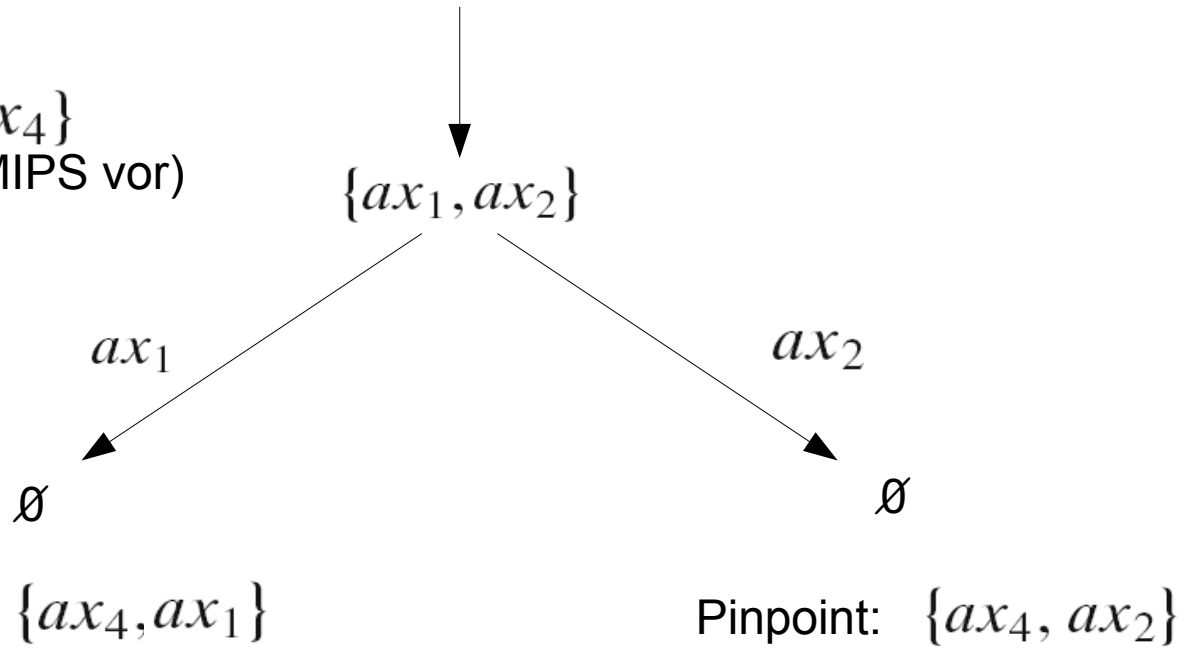
- Das am häufigsten vorkommende Axiom wird in die Pinpoint Menge aufgenommen und MIPSe die dieses Axiom enthalten gestrichen. Wiederholen bis keine MIPSe mehr übrig sind
- Nichtdeterministisch weil mehrere Axiome mit gleicher Häufigkeit auftreten können -> zufällige Auswahl

Nicht jeder Pinpoint ist eine Diagnose aber die meisten (keine Garantie der Minimalität).

Inkonsistenzen beheben - Pinpoint

$$mips(\mathcal{T}_1) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$$

2-wertiger Kern $\{ax_4\}$
(kommt zweimal im MIPS vor)



Es existieren zwei Kerne mit Wertigkeit eins. Es gibt also zwei mögliche Pinpoints.

Inkonsistenzen beheben - Reparatur der Terminologie

MUPS → MIPS → Pinpoints

Man erhält eine kleinstmögliche Menge an Axiomen, die die Inkonsistenz verursachen

Auf Grundlage der Pinpoints kann man nun diejenigen 'verantwortlichen' Axiome auswählen die sich gut korrigieren (löschen) lassen

"Real-World"-Beispiele

DICE - Diagnoses for Intensive Care Evaluation (Inkonsistenz behebung)

BT Digital Library (Machine Learning + Ontology Reasoning)

PION (Schließen unter Inkonsistenz)

DICE

- medizinische Ontologie - wird zur Aufnahme in die Intensivstation eingesetzt
- ursprünglich Frame-basiert wurde nach ACL-DL transformiert
- Enthält etwa 2500 Konzepte
- 65 der Konzepte waren nach der Transformation unerfüllbar

DICE

Zur Inkonsistenz Behebung wurde mittels zweier verschiedener Algorithmen Pinpoints ermittelt

175 MUPS

142 MIPS

Von den gefundenen Pinpoints enthielt einer NUR 5 Axiome

Basierend auf diesen Ergebnissen konnte DICE relativ einfach repariert werden

DICE - Benchmark

Table 3 Comparing top-down and bottom-up methods

	Top-down: MUPSTER time for all MIPS	Bottom-up: DION time for all MIPS	RacerPro time for coherence
DICE-A	12 s	timed out	4.3 s
DICE	54 s	32 s	16.62 s

Trotz hoher Komplexität ist diese reale Ontologie in annehmbarer Zeit zu reparieren.

Allerdings gibt es weitaus komplexere Ontologien (z.B. FMA mit 69.000 Konzepten) ob dann Ergebnisse in realistischer Zeit gewonnen werden können ist fraglich.

BT Digital Library - Ontology Learning

Aus Dokumenten zu Knowledge Management automatisiert Ontologie erstellen (mittels machine learning + Sprachverarbeitung)

Erstellen eines Learned Ontology Models (LOM)

- angelehnt an OWL-DL
- zusätzlich Speicherung von Wahrscheinlichkeiten zu den Aussagen

Transformation von LOM nach OWL-DL

- nutzen der zusätzlichen Informationen zur Auswahl der "wahrscheinlichsten" konsistenten Ontologie

Ontology Learning - Beispiel

Der Hase, Igel und Maulwurf trafen sich im Walde.

- gelernt werden die Disjunkten Konzepten Hase, Igel und Maulwurf ($Hase \sqsubseteq \neg Igel$, ...)
- $Hase \sqsubseteq \neg Igel$ bekommt Vertrauenswert abhängig von weiterem Auftreten der Konzepte

Ontology Learning - Inkonsistenzen

Axiom t	Confidence
$Data \sqsubseteq Information$	1.0
$Data \sqsubseteq Knowledge$	1.0
$Information \sqsubseteq \neg Knowledge$	0.7

T-Box

Axiom t	Confidence
$Application(kavido)$	0.46
$Tool(kavido)$	0.46
$Tool \sqsubseteq \neg Application$	0.3

A-Box

Unterschied zu gewöhnlichem Inkosistency Repair ist zusätzliche Information durch Confidenz-Werte

PION - Nicht-klassischer Reasoner

Test mit 4 kleinen, inkonsistenten Ontologien

- (Bird, Brain, Married-Woman, Mad Cow)

Antwort von PION wird mit Antwort verglichen, die ein Mensch erwartet

- Erwartete Antwort
- gegenteilige Antwort
- vorsichtige Antwort: PION antwortet „unbekannt“, obwohl „wahr“ oder „falsch“
- leichtfertige Antwort: PION antwortet „wahr“ oder „falsch“, obwohl „unbekannt“

Evaluation PION

396 Fragen an PION gestellt

- 20 vorsichtige, 4 leichtfertige, 2 gegenteilige Antworten
- 85,7% erwartete Antworten

Trotz einfacher rein syntaktischer Selektions-Funktion sehr gute Ergebnisse.

ENDE

Quellen

Haase, P. u. a., 2005. A Framework for Handling Inconsistency in Changing Ontologies. LECTURE NOTES IN COMPUTER SCIENCE, 3729, 353.

Haase, P. & Volker, J., Ontology Learning and Reasoning—Dealing with Uncertainty and Inconsistency. W3: Uncertainty Reasoning for the Semantic Web.

Hitzler, P., Krötzsch, M. & Rudolph, S., 2008. Semantic web : Grundlagen Erste Auflage., Berlin: Springer.

Schlobach, S. u. a., 2007. Debugging Incoherent Terminologies. Journal of Automated Reasoning, 39(3), 317-349.