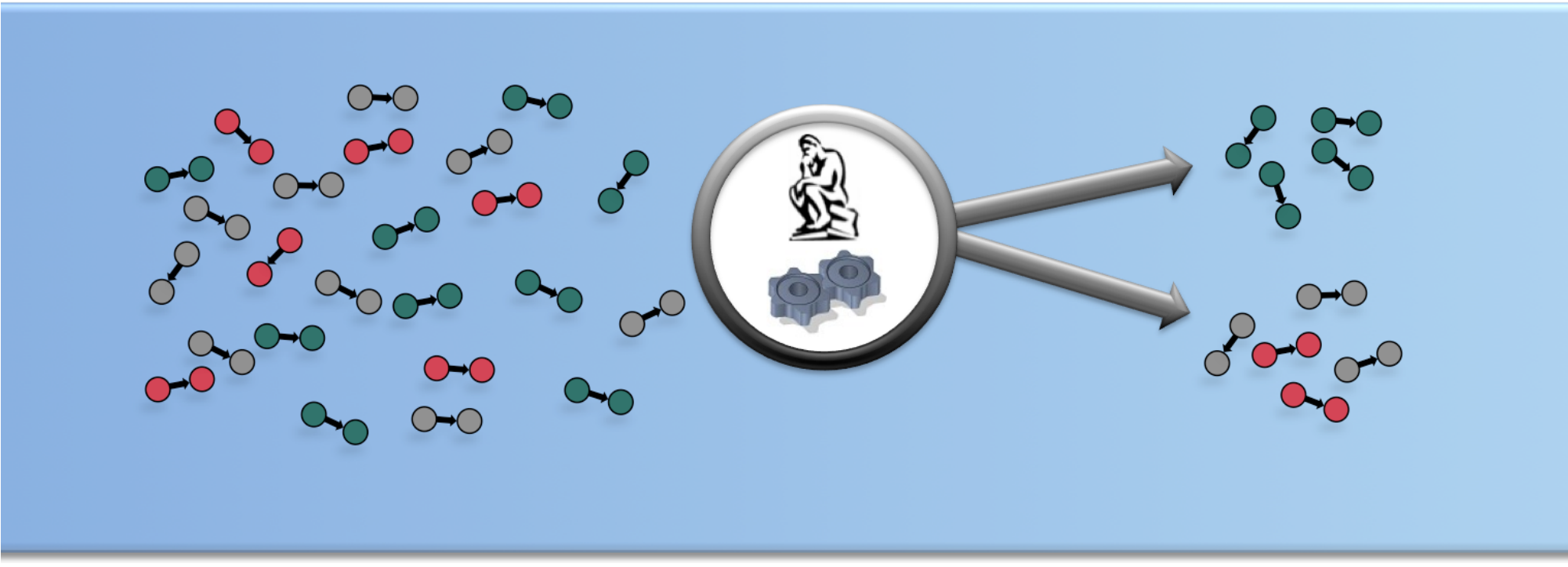
ulm university universität ulm

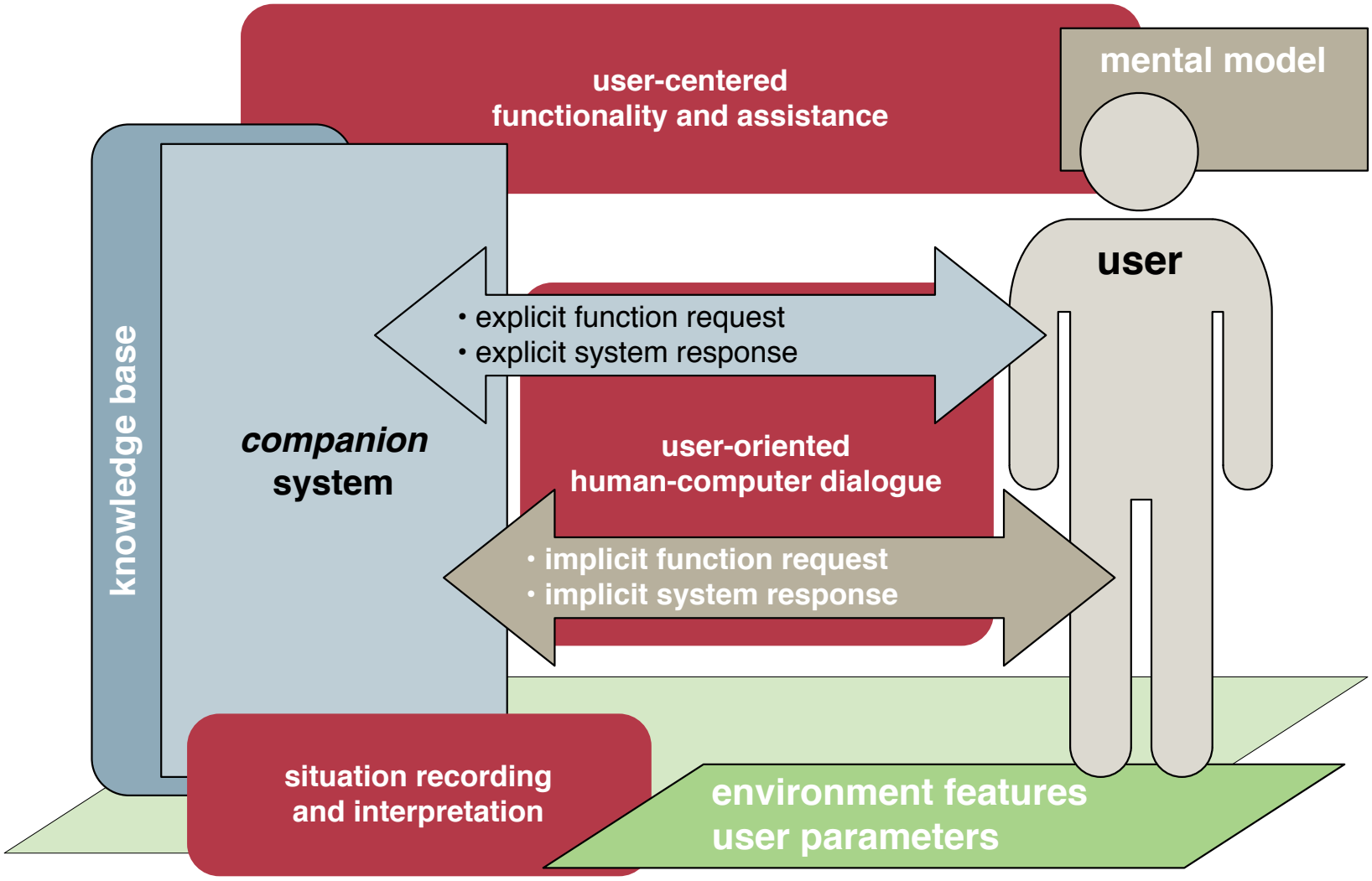


Interactive Reasoning-Based Ontology Revision

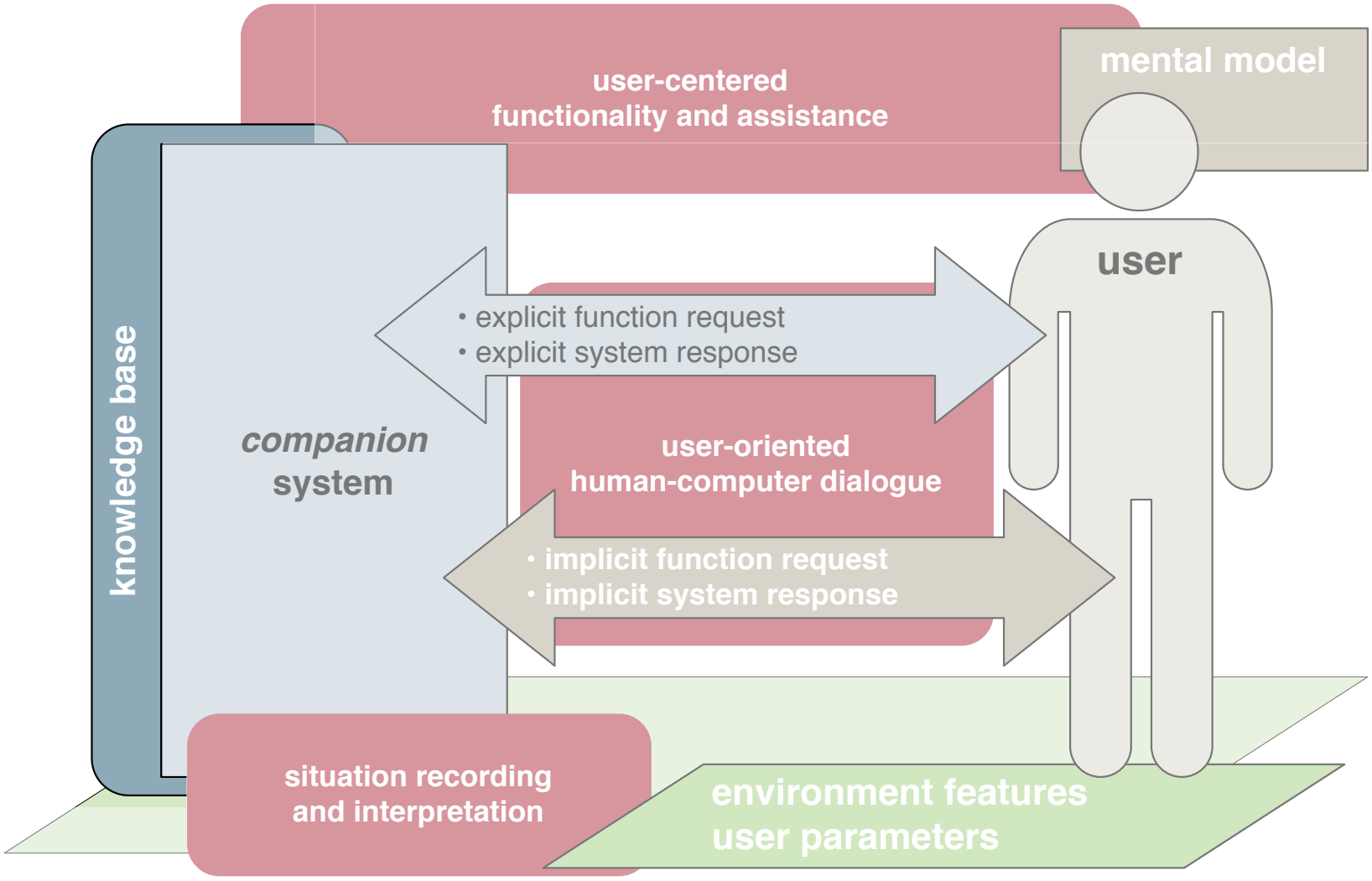
SFB/TRR 62 Kolloquium

B. Glimm, N. Nikitina, S. Rudolph
Institute of Artificial Intelligence | 14.11.2011

Ontology Revision in the Context of Companion Systems



Ontology Revision in the Context of Companion Systems



Ontology Revision in the Context of Companion Systems

Companion systems need some background knowledge

- ⇝ Ontologies
  - Ontology revision helps to ensure the quality of an ontology
  - Revision support for building up the knowledge base through reuse or automated learning methods

Ontology Revision in the Context of Companion Systems

Companion systems need some background knowledge

- ⇝ Ontologies
  - Ontology revision helps to ensure the quality of an ontology
  - Revision support for building up the knowledge base through reuse or automated learning methods

Ontology Revision in the Context of Companion Systems

Companion systems need some background knowledge

- ⇝ Ontologies
  - Ontology revision helps to ensure the quality of an ontology
  - Revision support for building up the knowledge base through reuse or automated learning methods

Ontology might be incomplete

- ⇝ Ontology extension either at runtime or offline
  - Ontology extension at runtime needs fully automatic revision

Ontology Revision in the Context of Companion Systems

Companion systems need some background knowledge

- ⇝ Ontologies
  - Ontology revision helps to ensure the quality of an ontology
  - Revision support for building up the knowledge base through reuse or automated learning methods

Ontology might be incomplete

- ⇝ Ontology extension either at runtime or offline
  - Ontology extension at runtime needs fully automatic revision

Companion systems need some user specific knowledge

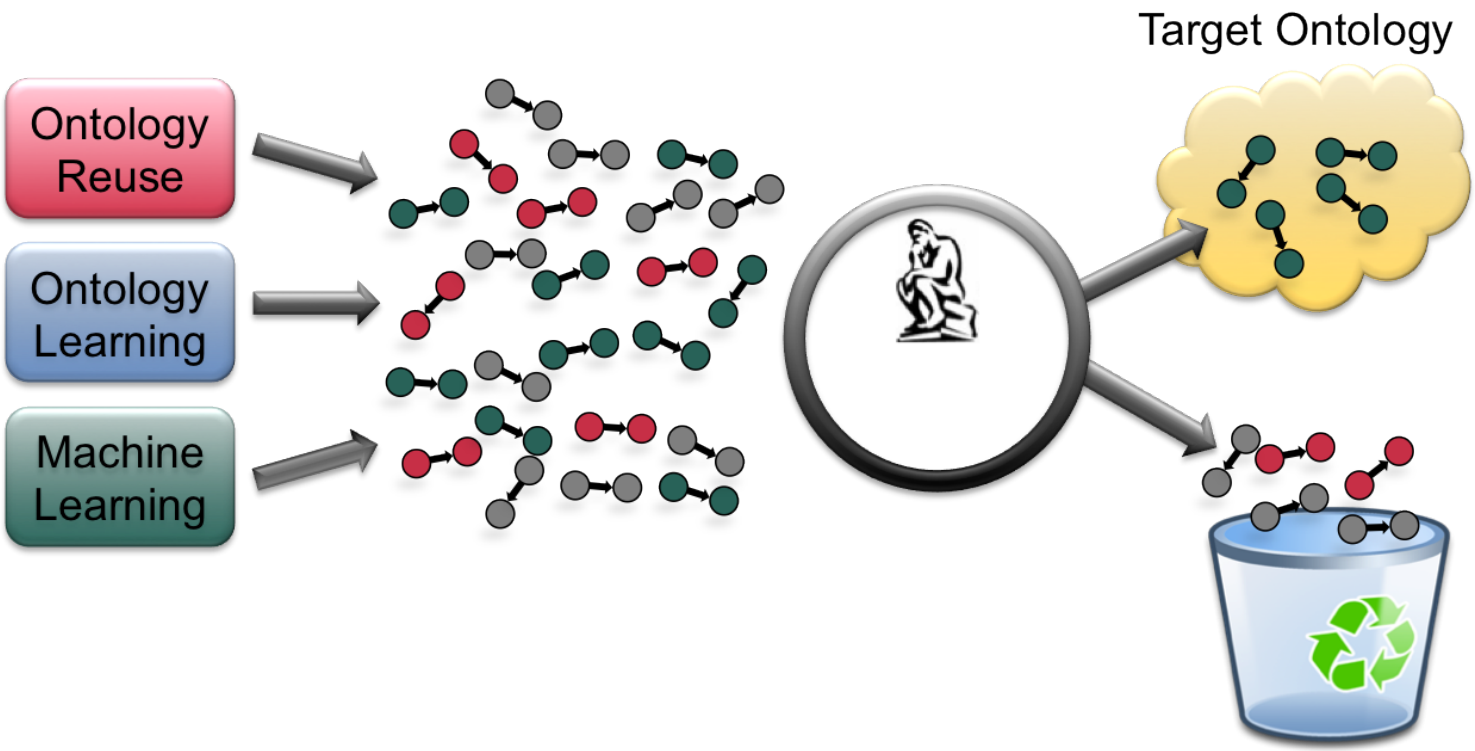
- More dynamic and uncertain knowledge
- ⇝ Reasoning over dynamically changing and uncertain knowledge

Outline

- Motivation
- Ontology and Reasoning Basics
- Semi-automatic Ontology Revision by Example
- Ontology Revision Terminology
- Axiom Ranking
- Decision Spaces
- Empirical Evaluation & Dynamic Ranking Strategies
- Summary & Outlook

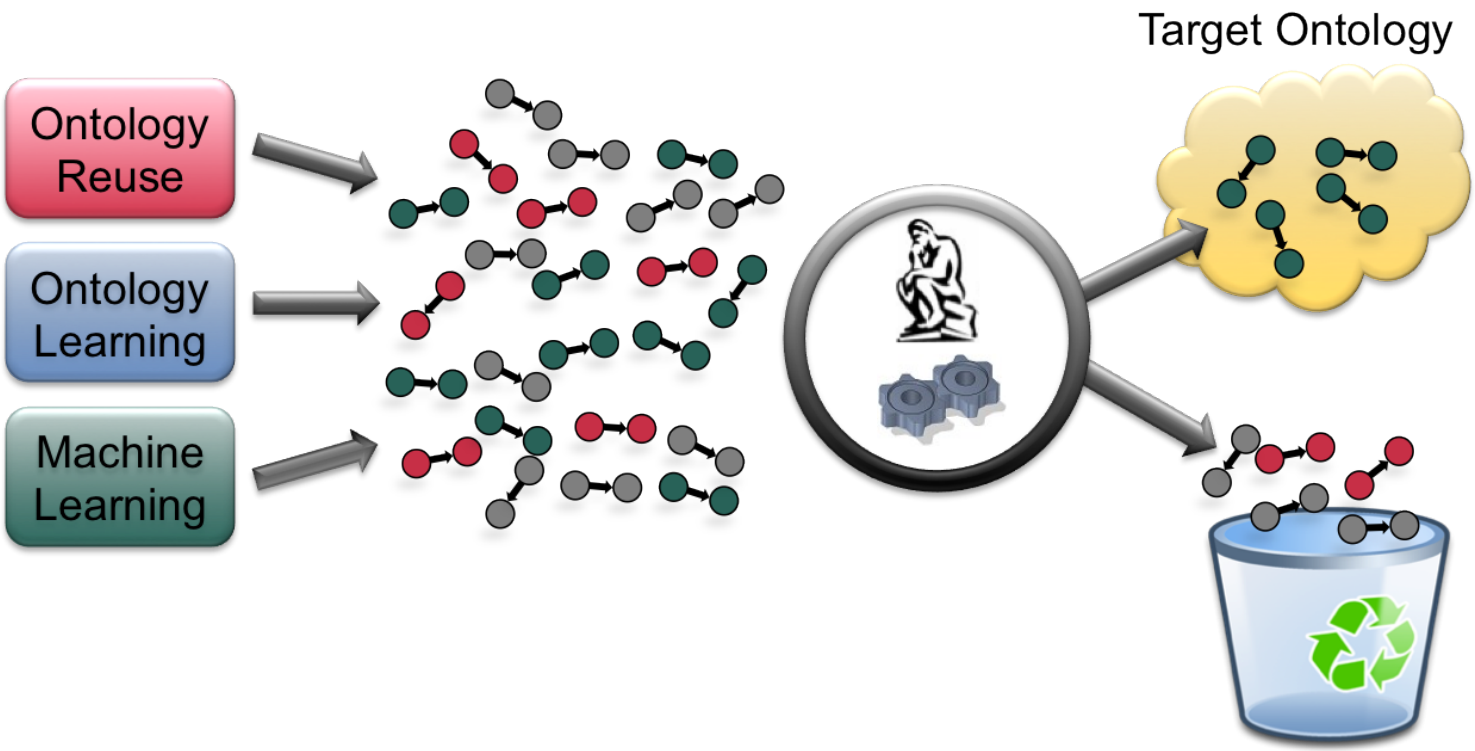
Motivation

- Noisy domain statements produced by knowledge acquisition and integration methods
- Manual revision of the acquired information is required to ensure high quality



Motivation

- Noisy domain statements produced by knowledge acquisition and integration methods
- Manual revision of the acquired information is required to ensure high quality



Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms

Example Ontology

Individual: Birte Type: Juniorprofessor

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms

Example Ontology

Individual: Birte Type: Juniorprofessor

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms
- Constructors can be used to build complex classes

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms
- Constructors can be used to build complex classes

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor

Ontology Basics

- The basic building blocks of an ontology are
  - individuals
  - classes (sets of individuals)
  - properties (relate pairs of individuals)
- An ontology can be understood as a set of facts and axioms
- Constructors can be used to build complex classes

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

- Individual: Birte Type: Professor

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

- Individual: Birte Type: Professor

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

- Individual: Birte Type: Professor
- Individual: CS6380.00 Type: Lecture

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

- Individual: Birte Type: Professor
- Individual: CS6380.00 Type: Lecture

Ontology Basics

- Ontologies are modelled in a formal language (e.g., OWL)
- The semantics is well-defined (based on First-Order Logic)

$\rightsquigarrow$ We can use the axioms to derive automatic consequences

Example Ontology

Individual:	Birte	Type:	Juniorprofessor
Individual:	Birte	Facts:	holds CS6380.00
Class:	Professor	SubClassOf:	holds some Lecture
Class:	JuniorProfessor	SubClassOf:	Professor
ObjectProperty:	holds	Range:	Lecture

The ontology entails, for example:

- Individual: Birte Type: Professor
- Individual: CS6380.00 Type: Lecture
- Class: JuniorProfessor SubClassOf: holds some Lecture

Reasoning Basics

- If an ontology $\mathcal{O}$ entails an axiom/fact α, we write $\mathcal{O} \models \alpha$

Reasoning Basics

- If an ontology $\mathcal{O}$ entails an axiom/fact α, we write $\mathcal{O} \models \alpha$
- If $\mathcal{O}$ does not entail α, we write $\mathcal{O} \not\models \alpha$

Reasoning Basics

- If an ontology $\mathcal{O}$ entails an axiom/fact α, we write $\mathcal{O} \models \alpha$
- If $\mathcal{O}$ does not entail α, we write $\mathcal{O} \not\models \alpha$
- Reasoners implement algorithms to check entailment

Reasoning Basics

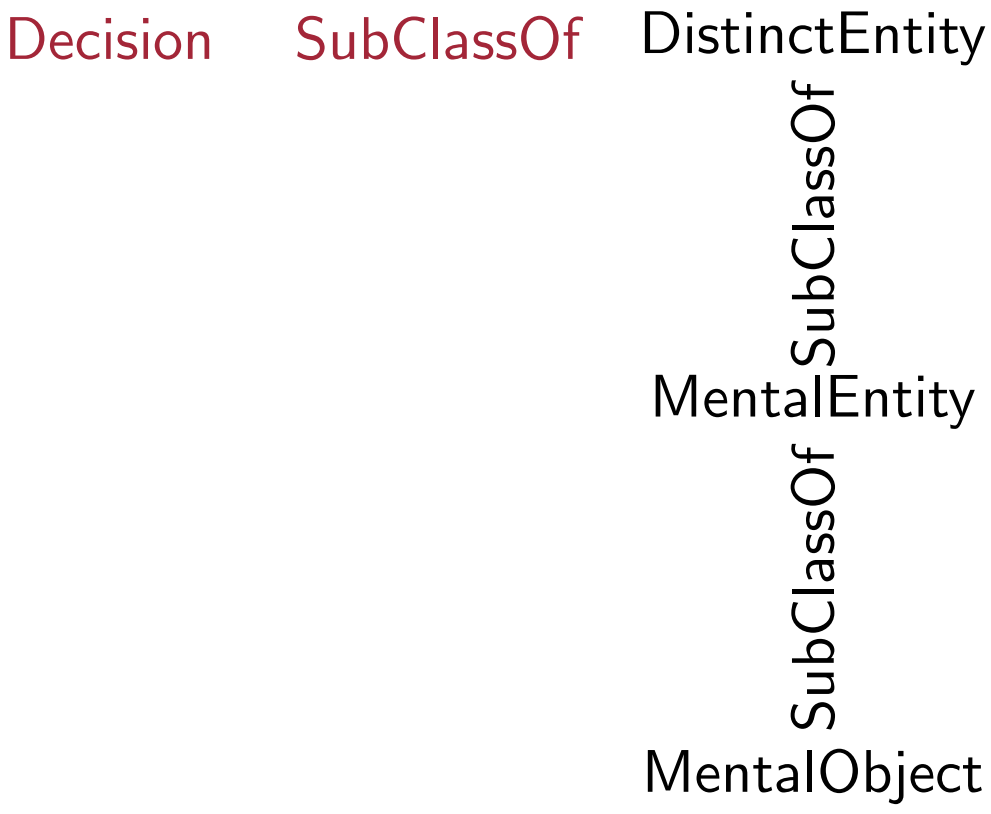
- If an ontology $\mathcal{O}$ entails an axiom/fact α, we write $\mathcal{O} \models \alpha$
- If $\mathcal{O}$ does not entail α, we write $\mathcal{O} \not\models \alpha$
- Reasoners implement algorithms to check entailment
- Depending on the expressivity of the ontology language such algorithms can be anything between tractable and super-exponential in worst-case complexity

Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms

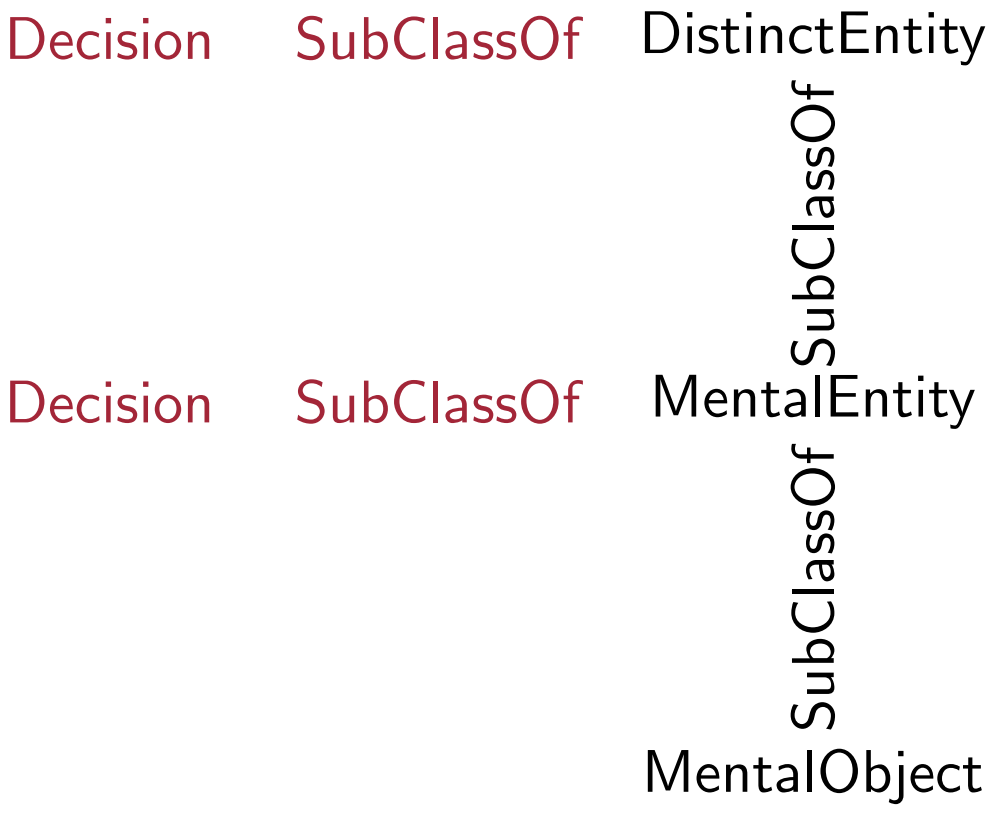
Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms



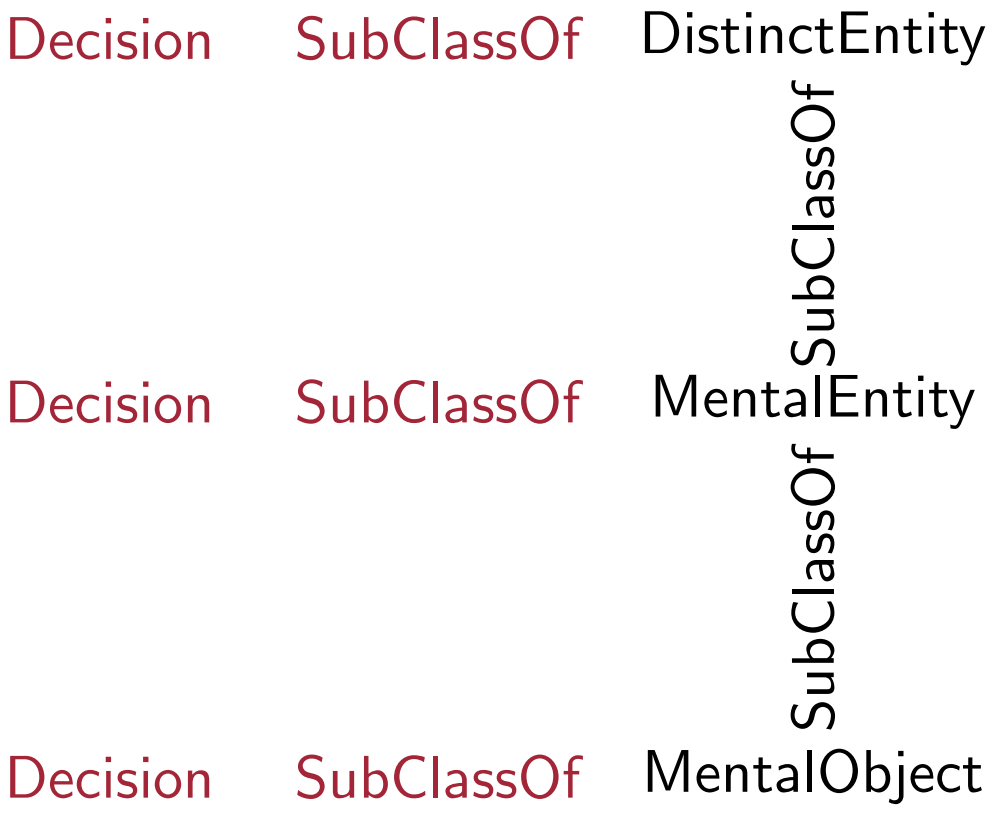
Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms



Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms



Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms

✓ Decision SubClassOf MentalObject

Decision SubClassOf MentalEntity

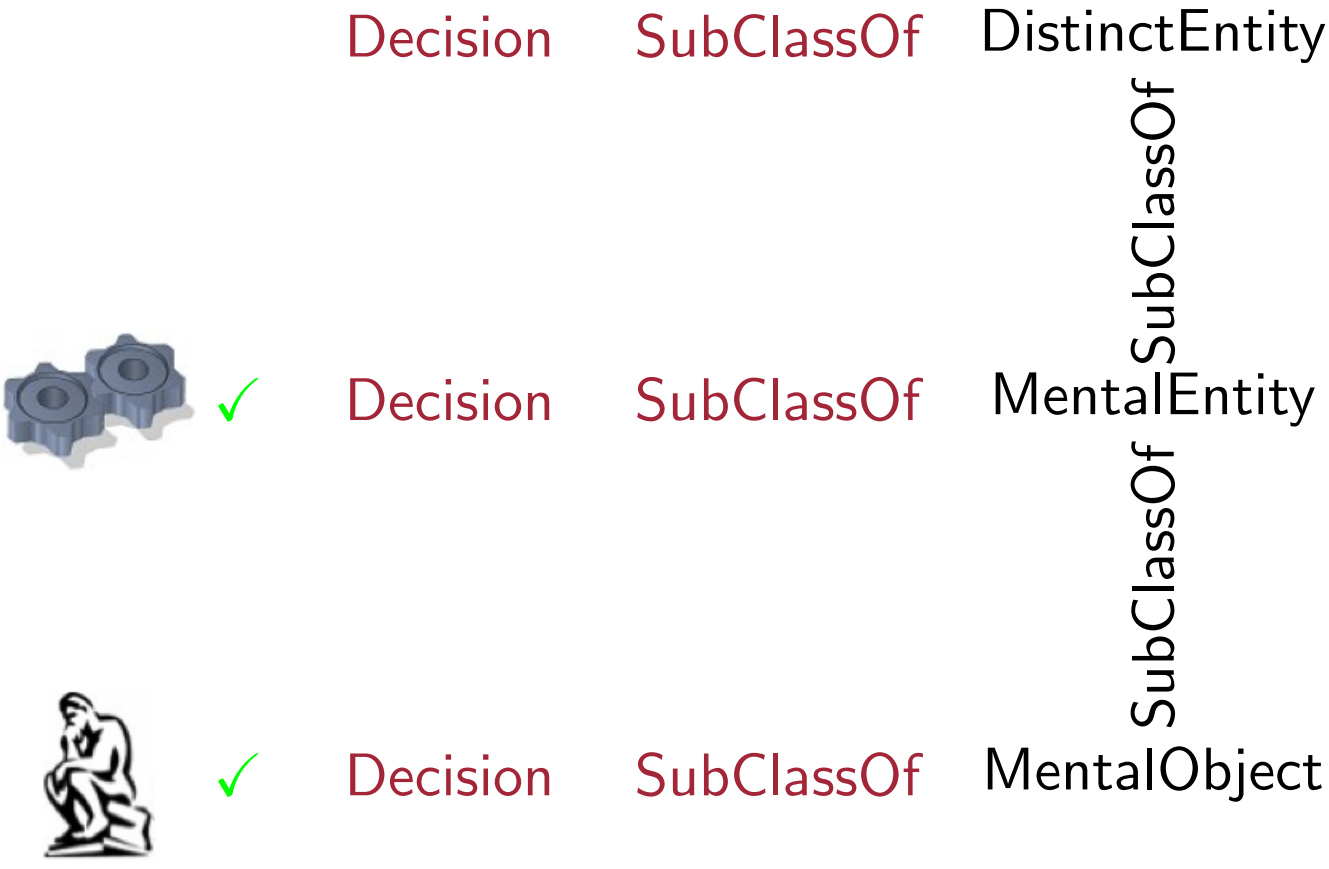
Decision SubClassOf DistinctEntity

SubClassOf

SubClassOf

Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms



Revision Example

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms


✓ Decision SubClassOf DistinctEntity

SubClassOf

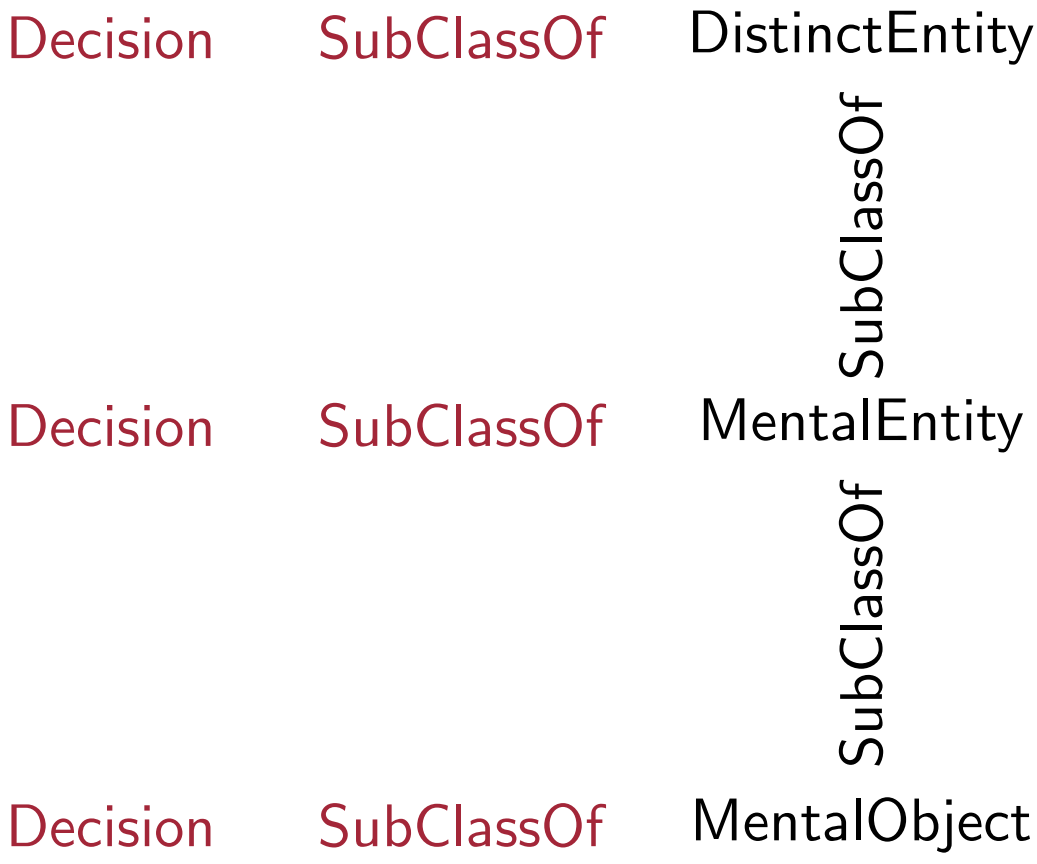

✓ Decision SubClassOf MentalEntity

SubClassOf


✓ Decision SubClassOf MentalObject

Revision Example

- Order is important to achieve a good grade of automation



Revision Example

- Order is important to achieve a good grade of automation

✓ Decision SubClassOf DistinctEntity

SubClassOf

Decision SubClassOf MentalEntity

SubClassOf

Decision SubClassOf MentalObject

Revision Example

- Order is important to achieve a good grade of automation

✓ Decision SubClassOf DistinctEntity

SubClassOf

✓ Decision SubClassOf MentalEntity

SubClassOf

Decision SubClassOf MentalObject

Revision Example

- Order is important to achieve a good grade of automation

✓ Decision SubClassOf DistinctEntity

SubClassOf

✓ Decision SubClassOf MentalEntity

SubClassOf

✓ Decision SubClassOf MentalObject

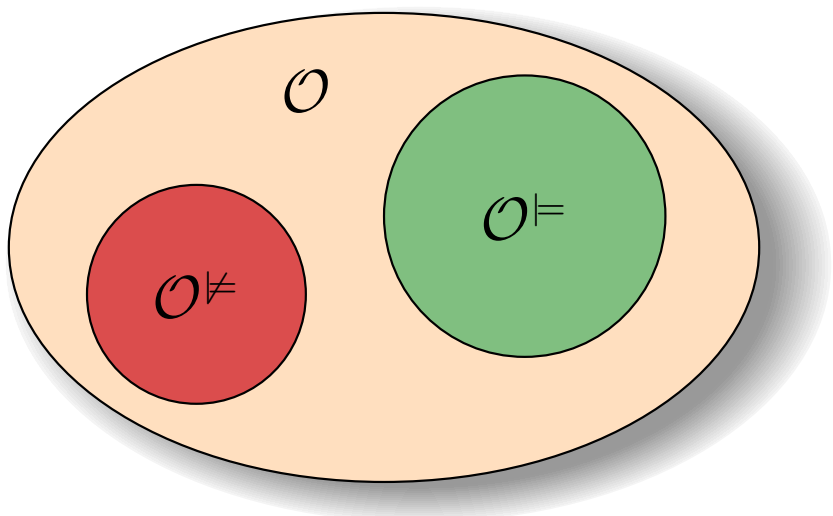
Ontology Revision

Assumption: Deductive closure of the intended consequences must not contain unintended consequences

- A single evaluation decision can predetermine the decision for several yet unevaluated axioms
- Order influences method effectiveness

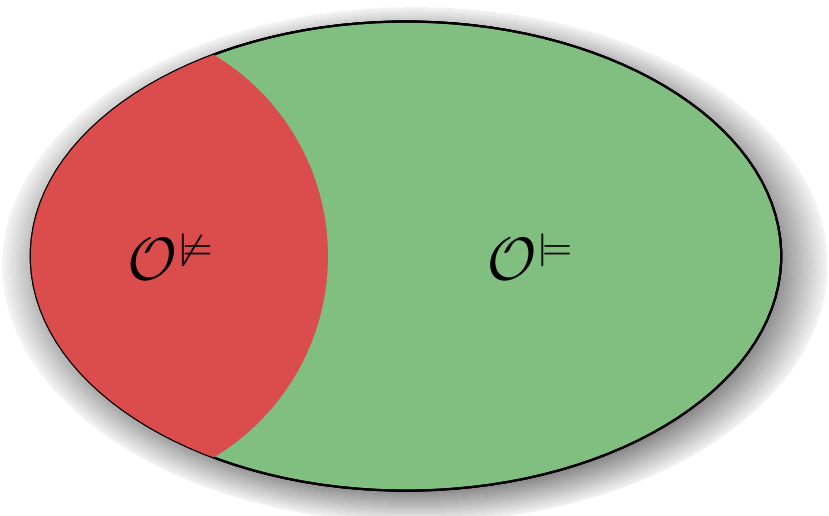
Revision States

- A revision state is defined as a tuple $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ of ontologies with $\mathcal{O}^{\models} \subseteq \mathcal{O}, \emptyset \neq \mathcal{O}^{\not\models} \subseteq \mathcal{O}$, and $\mathcal{O}^{\models} \cap \mathcal{O}^{\not\models} = \emptyset$.
  $\mathcal{O}^{\models}$: the set of desired consequences
  $\mathcal{O}^{\not\models}$: the set of undesired consequences



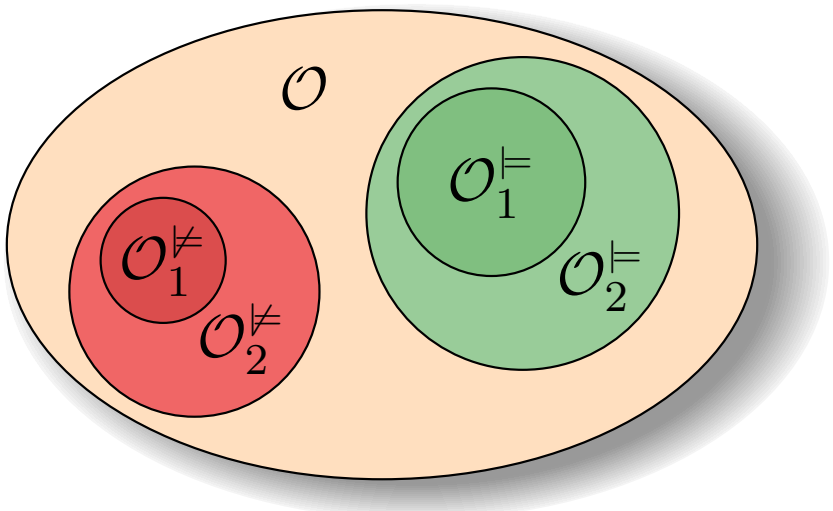
Revision State Completeness

- A revision state is complete, if $\mathcal{O} = \mathcal{O}^{\models} \cup \mathcal{O}^{\not\models}$, and incomplete otherwise.



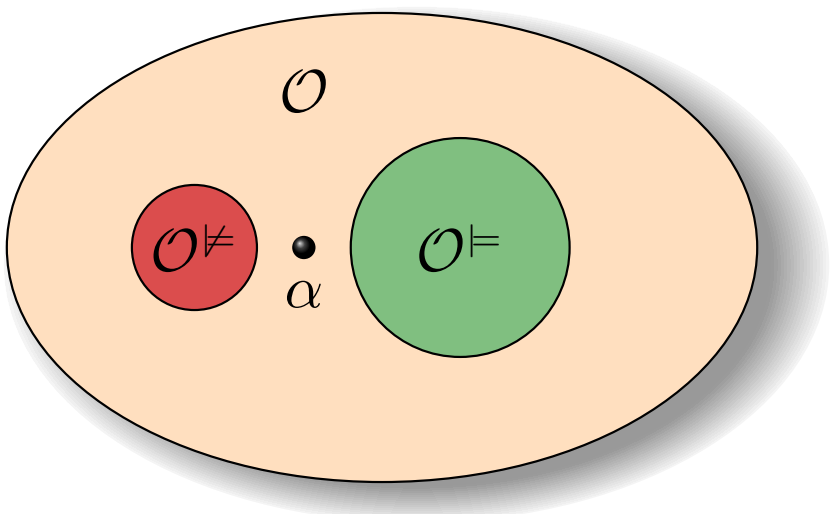
Revision State Refinements

- Given two revision states $(\mathcal{O}, \mathcal{O}_1^{\models}, \mathcal{O}_1^{\not\models})$ and $(\mathcal{O}, \mathcal{O}_2^{\models}, \mathcal{O}_2^{\not\models})$, we call $(\mathcal{O}, \mathcal{O}_2^{\models}, \mathcal{O}_2^{\not\models})$ a refinement of $(\mathcal{O}, \mathcal{O}_1^{\models}, \mathcal{O}_1^{\not\models})$, if $\mathcal{O}_1^{\models} \subseteq \mathcal{O}_2^{\models}$ and $\mathcal{O}_1^{\not\models} \subseteq \mathcal{O}_2^{\not\models}$.



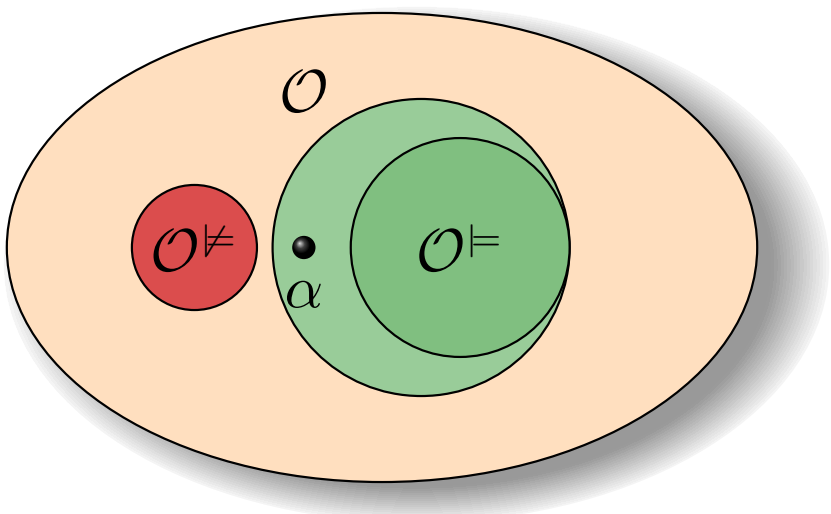
Elementary Refinements of Revision States

- An incomplete revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ can be refined by evaluating a further axiom $\alpha \in \mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})$, obtaining $(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models})$ or $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\})$.



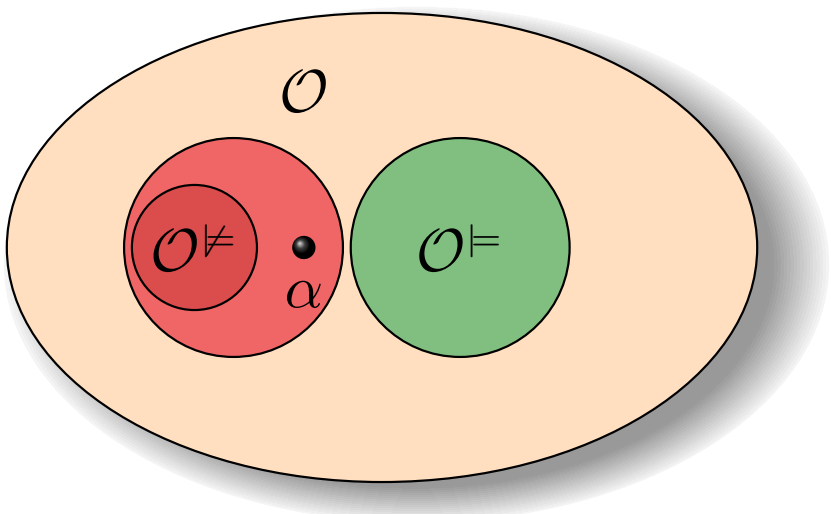
Elementary Refinements of Revision States

- An incomplete revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ can be refined by evaluating a further axiom $\alpha \in \mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})$, obtaining $(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models})$ or $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\})$.



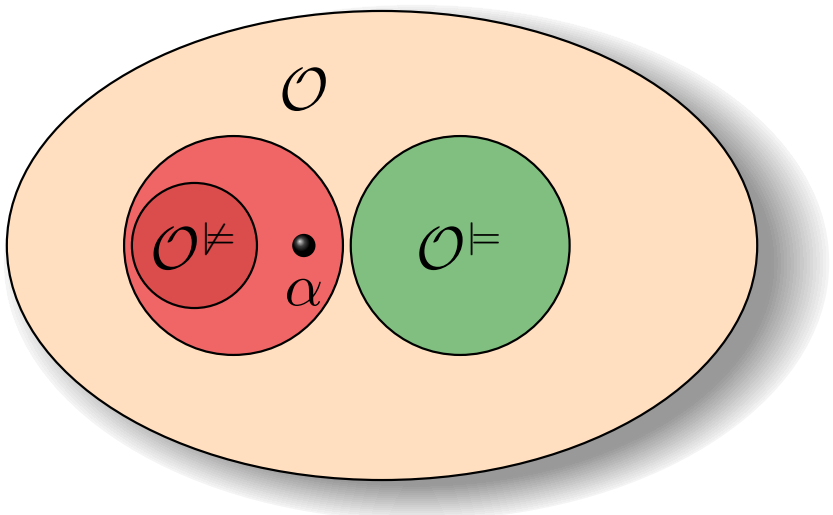
Elementary Refinements of Revision States

- An incomplete revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ can be refined by evaluating a further axiom $\alpha \in \mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})$, obtaining $(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models})$ or $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\})$.



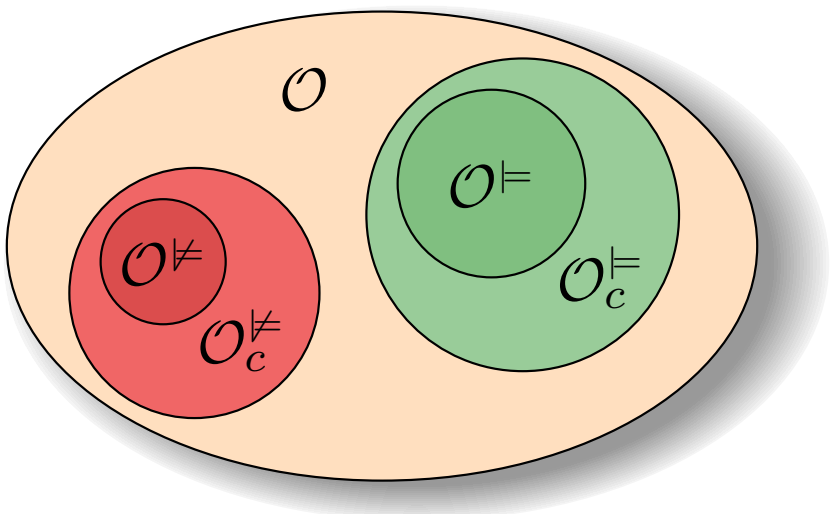
Elementary Refinements of Revision States

- An incomplete revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ can be refined by evaluating a further axiom $\alpha \in \mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})$, obtaining $(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models})$ or $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\})$.
- We call the resulting revision state an elementary refinement of $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$.



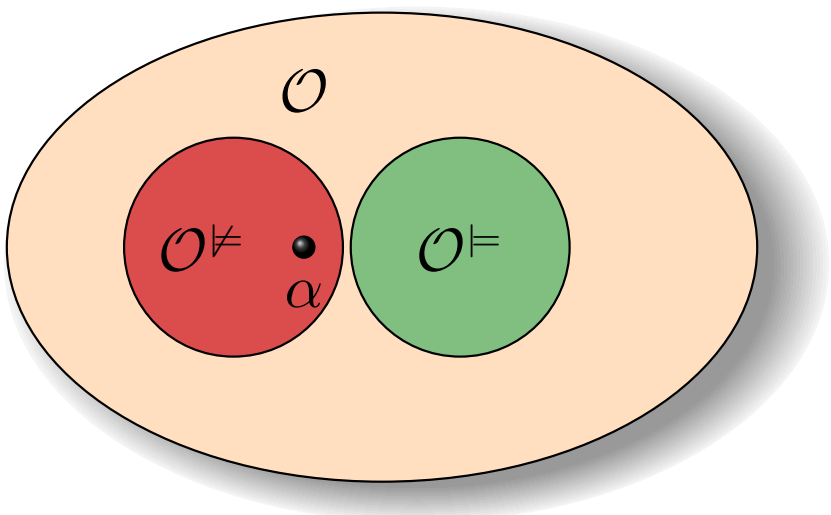
Revision Closure

- The revision closure $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ of $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is $(\mathcal{O}, \mathcal{O}_c^{\models}, \mathcal{O}_c^{\not\models})$ with
  - $\mathcal{O}_c^{\models} := \{\alpha \in \mathcal{O} \mid \mathcal{O}^{\models} \models \alpha\}$ and
  - $\mathcal{O}_c^{\not\models} := \{\alpha \in \mathcal{O} \mid \mathcal{O}^{\models} \cup \{\alpha\} \models \beta \text{ for some } \beta \in \mathcal{O}^{\not\models}\}$.



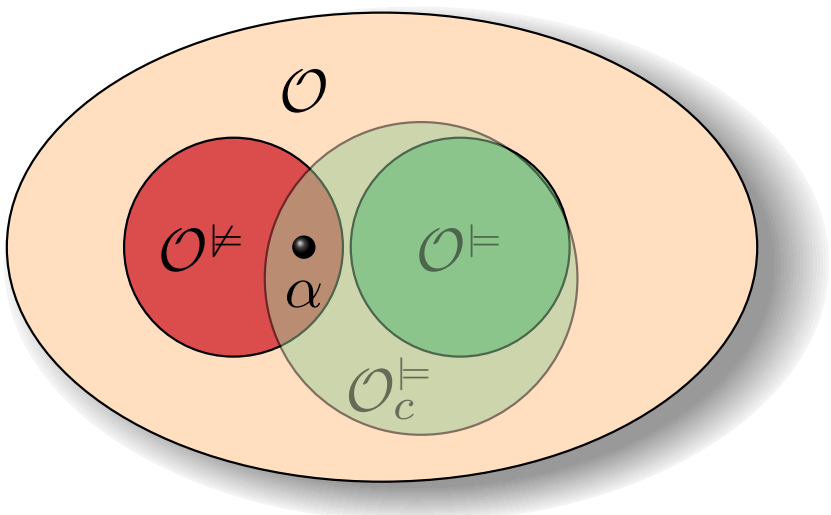
Revision State Consistency

- A revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent if there is no $\alpha \in \mathcal{O}^{\not\models}$ such that $\mathcal{O}^{\models} \models \alpha$.



Revision State Consistency

- A revision state $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent if there is no $\alpha \in \mathcal{O}^{\not\models}$ such that $\mathcal{O}^{\models} \models \alpha$.



Revision States

- New evaluation decisions are reflected in elementary refinements
- The revision closure reflects the automatic decisions

Revision States

- New evaluation decisions are reflected in elementary refinements
- The revision closure reflects the automatic decisions

Goals:

- Obtain complete and consistent revision state
- Reduce number of manual decisions
- Reduce the time for automatic decisions

Properties of the Revision Closure

For $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ a consistent revision state:

1. $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent

Properties of the Revision Closure

For $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ a consistent revision state:

1. $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent
2. every elementary refinement of $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent

Properties of the Revision Closure

For $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ a consistent revision state:

1. $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent
2. every elementary refinement of $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent
3. every consistent complete refinement of $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is a refinement of $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$

Properties of the Revision Closure

For $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ a consistent revision state:

1. $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent
2. every elementary refinement of $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is consistent
3. every consistent complete refinement of $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ is a refinement of $\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$

$\rightsquigarrow$ Revision closures reduce the manual effort of revision and ensure the consistency of revision states

Axiom Impact

- The evaluation order has an impact on the degree of atomatization

$(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: a consistent revision state with $\alpha \in \mathcal{O}$

$?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: $|\mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})|$

- Approval impact: Number of automatically evaluated axioms in case α is approved:
  $\text{impact}^{+}(\alpha) = ?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models}) - ?(\text{clos}(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models}))$,

Axiom Impact

- The evaluation order has an impact on the degree of atomatization

$(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: a consistent revision state with $\alpha \in \mathcal{O}$

$?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: $|\mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})|$

- Approval impact: Number of automatically evaluated axioms in case α is approved:
  $\text{impact}^{+}(\alpha) = ?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models}) - ?(\text{clos}(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models}))$,
- Decline impact: number of automatically evaluated axioms in case α is declined:
  $\text{impact}^{-}(\alpha) = ?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models}) - ?(\text{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\}))$,

Axiom Impact

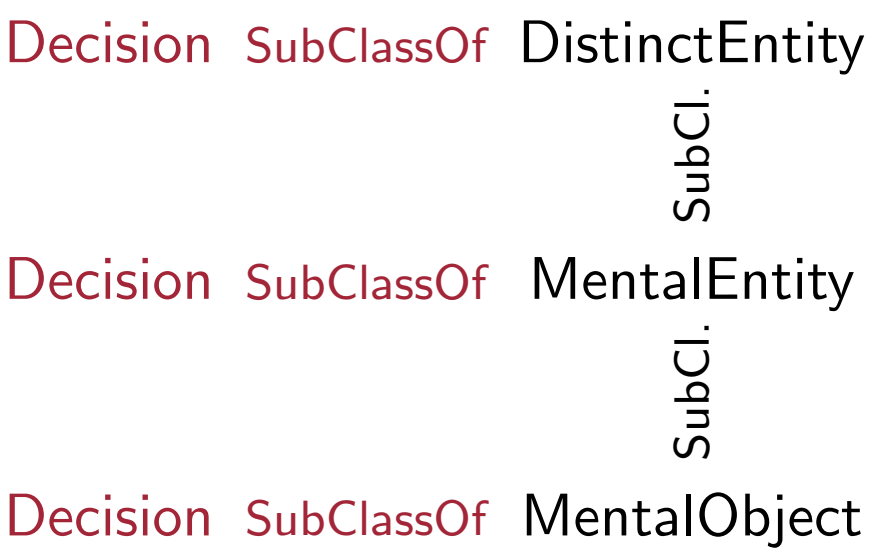
- The evaluation order has an impact on the degree of atomatization

$(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: a consistent revision state with $\alpha \in \mathcal{O}$

$?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$: $|\mathcal{O} \setminus (\mathcal{O}^{\models} \cup \mathcal{O}^{\not\models})|$

- Approval impact: Number of automatically evaluated axioms in case α is approved:
  $\mathsf{impact}^{+}(\alpha) = ?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models}) - ?(\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models} \cup \{\alpha\}, \mathcal{O}^{\not\models}))$,
- Decline impact: number of automatically evaluated axioms in case α is declined:
  $\mathsf{impact}^{-}(\alpha) = ?(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models}) - ?(\mathrm{clos}(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models} \cup \{\alpha\}))$,
- Guaranteed impact:
  $\mathsf{guaranteed}(\alpha) = \min(\mathsf{impact}^{+}(\alpha), \mathsf{impact}^{-}(\alpha))$.

Impact Computation



impact^+	impact^-	guaranteed
0	2	0
1	1	1
2	0	0

Decision Spaces: Saving Computational Effort

- Computing the closure and impact measures requires entailment checking:
  - $\alpha E \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha\} \models \beta$
  - $\alpha C \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha, \beta\} \models \gamma$ for some $\gamma \in \mathcal{O}^{\not\models}$

Decision Spaces: Saving Computational Effort

- Computing the closure and impact measures requires entailment checking:
  - $\alpha E \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha\} \models \beta$
  - $\alpha C \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha, \beta\} \models \gamma$ for some $\gamma \in \mathcal{O}^{\not\models}$
- Decision Space for a particular revision state:
  - Graph with nodes $\mathcal{O}^{?}$ – the unevaluated axioms after closure
  - Relations E and C induce edges

Decision Spaces: Saving Computational Effort

- Computing the closure and impact measures requires entailment checking:
  - $\alpha E \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha\} \models \beta$
  - $\alpha C \beta$ iff $\mathcal{O}^{\models} \cup \{\alpha, \beta\} \models \gamma$ for some $\gamma \in \mathcal{O}^{\not\models}$
- Decision Space for a particular revision state:
  - Graph with nodes $\mathcal{O}^?$ – the unevaluated axioms after closure
  - Relations E and C induce edges

Decision spaces exploit the following properties of E and C:

P1 $(\mathcal{O}^?, E)$ is a quasi-order (i.e., reflexive and transitive),

P2 C is symmetric,

P3 $\alpha E \beta$ and $\beta C \gamma$ imply $\alpha C \gamma$ for all $\alpha, \beta, \gamma \in \mathcal{O}^?$, and

P4 if $\alpha E \beta$ then $\alpha C \beta$ does not hold.

Pruning the Space of Unknown Elements of E and C

R0		$\rightarrow E(x,x)$	reflexivity of E
R1	$E(x,y) \wedge E(y,z)$	$\rightarrow E(x,z)$	transitivity of E
R2	$E(x,y) \wedge C(y,z)$	$\rightarrow C(x,z)$	(P3)
R3	$C(x,y)$	$\rightarrow C(y,x)$	symmetry of C
R4	$E(x,y)$	$\rightarrow \overline{C}(x,y)$	disjointness of E and C
R5	$\overline{C}(x,y)$	$\rightarrow \overline{C}(y,x)$	symmetry of C
R6	$E(x,y) \wedge \overline{C}(x,z)$	$\rightarrow \overline{C}(y,z)$	(P3)
R7	$C(x,y)$	$\rightarrow \overline{E}(x,y)$	disjointness of E and C
R8	$\overline{C}(x,y) \wedge C(y,z)$	$\rightarrow \overline{E}(x,z)$	(P3)
R9	$E(x,y) \wedge \overline{E}(x,z)$	$\rightarrow \overline{E}(y,z)$	transitivity of E

$\overline{E}$: complement of E

$\overline{C}$: complement of C

Computing E and C

- Rules have an acyclic structure

$\rightsquigarrow$ E, C, $\overline{C}$, and $\overline{E}$ can be saturated one after another

Computing E and C

- Rules have an acyclic structure

$\rightsquigarrow$ E, C, $\overline{C}$, and $\overline{E}$ can be saturated one after another

Condensed rule set:

$$
\begin{array}{lcl}
E & \leftarrow & E^* \\
C & \leftarrow & E \circ (C \cup C^-) \circ E^- \\
\overline{C} & \leftarrow & E^- \circ (\overline{C} \cup Id \cup \overline{C}^-) \circ E \\
\overline{E} & \leftarrow & E^- \circ (\overline{C} \circ C \cup \overline{E}) \circ E^-
\end{array}
$$

Computing E and C

- Rules have an acyclic structure

$\rightsquigarrow$ E, C, $\overline{C}$, and $\overline{E}$ can be saturated one after another

Condensed rule set:

$$
\begin{array}{lcl}
E & \leftarrow & E^* \\
C & \leftarrow & E \circ (C \cup C^-) \circ E^- \\
\overline{C} & \leftarrow & E^- \circ (\overline{C} \cup Id \cup \overline{C}^-) \circ E \\
\overline{E} & \leftarrow & E^- \circ (\overline{C} \circ C \cup \overline{E}) \circ E^-
\end{array}
$$

- Algorithm now initializes the relations and applies the rules
- Executes an entailment check to clarify a missing case
- Closes the relations under the rule set

Complexity of Computing E and C

Assuming that entailment checking is a constant time operation:

Lemma

Let $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ with $n = |\mathcal{O}|$ be a revision state. Computing E and C can be done in time $O(n^5)$ and space $O(n^2)$.

Complexity of Computing E and C

Assuming that entailment checking is a constant time operation:

Lemma

Let $(\mathcal{O}, \mathcal{O}^{\models}, \mathcal{O}^{\not\models})$ with $n = |\mathcal{O}|$ be a revision state. Computing E and C can be done in time $O(n^5)$ and space $O(n^2)$.

- Entailment checking usually outweighs the other operations
- For the Web Ontology Language OWL entailment checking is N2ExpTime-complete
- Lower complexities for fragments of OWL

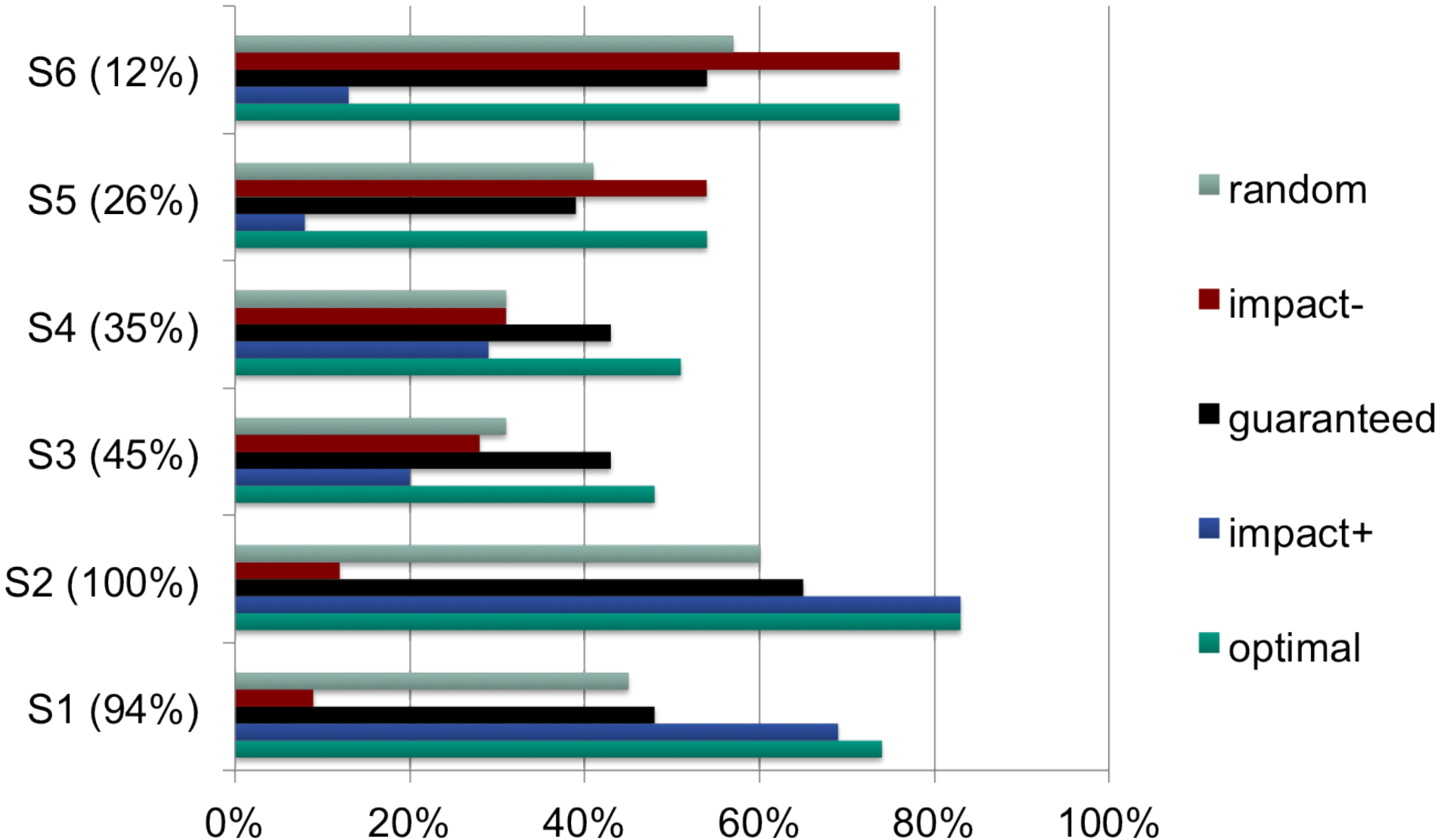
Advantages of Decision Spaces

- Revision closures can be read off the E and C relations
- Axiom impact can be read off the E and C relations
- Updating a decision space after an axiom approval or decline can be done incrementally

Ontology Revision Use Case

- Empirical evaluation based on data from the NanOn project
- NanOn goals: semi-automatic ontology generation for nano technology
- Ontology captures substances, structures, and procedures used in the domain of nano technology
- Scientific documents are automatically analyzed for the occurrence of NanOn classes and properties by the means of lexical patterns
- Documents are annotated with these terms to facilitate topic-specific information retrieval
- Revision is needed to ensure the quality

Evaluation Results: Reduction of Manual Effort



Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

$\rightsquigarrow$ Develop a ranking function parametrized by the validity ratio

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

$\rightsquigarrow$ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

$\rightsquigarrow$ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

$\rightsquigarrow$ Estimate and lear the validity ratio for the parametrized ranking function

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

$\rightsquigarrow$ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

$\rightsquigarrow$ Estimate and lear the validity ratio for the parametrized ranking function

- Achieves a near maximum automation

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

⇝ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

⇝ Estimate and lear the validity ratio for the parametrized ranking function

- Achieves a near maximum automation
- Gain is particularly important for datasets with a validity ratio close to 50%

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

⇝ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

⇝ Estimate and lear the validity ratio for the parametrized ranking function

- Achieves a near maximum automation
- Gain is particularly important for datasets with a validity ratio close to 50%
- Even for small datasets (50–100 axioms) validity ratio can be learned effectively

Parametrizing the Impact Function

- Axiom ranking functions are tailored towards validity ratios of 100% and 0%

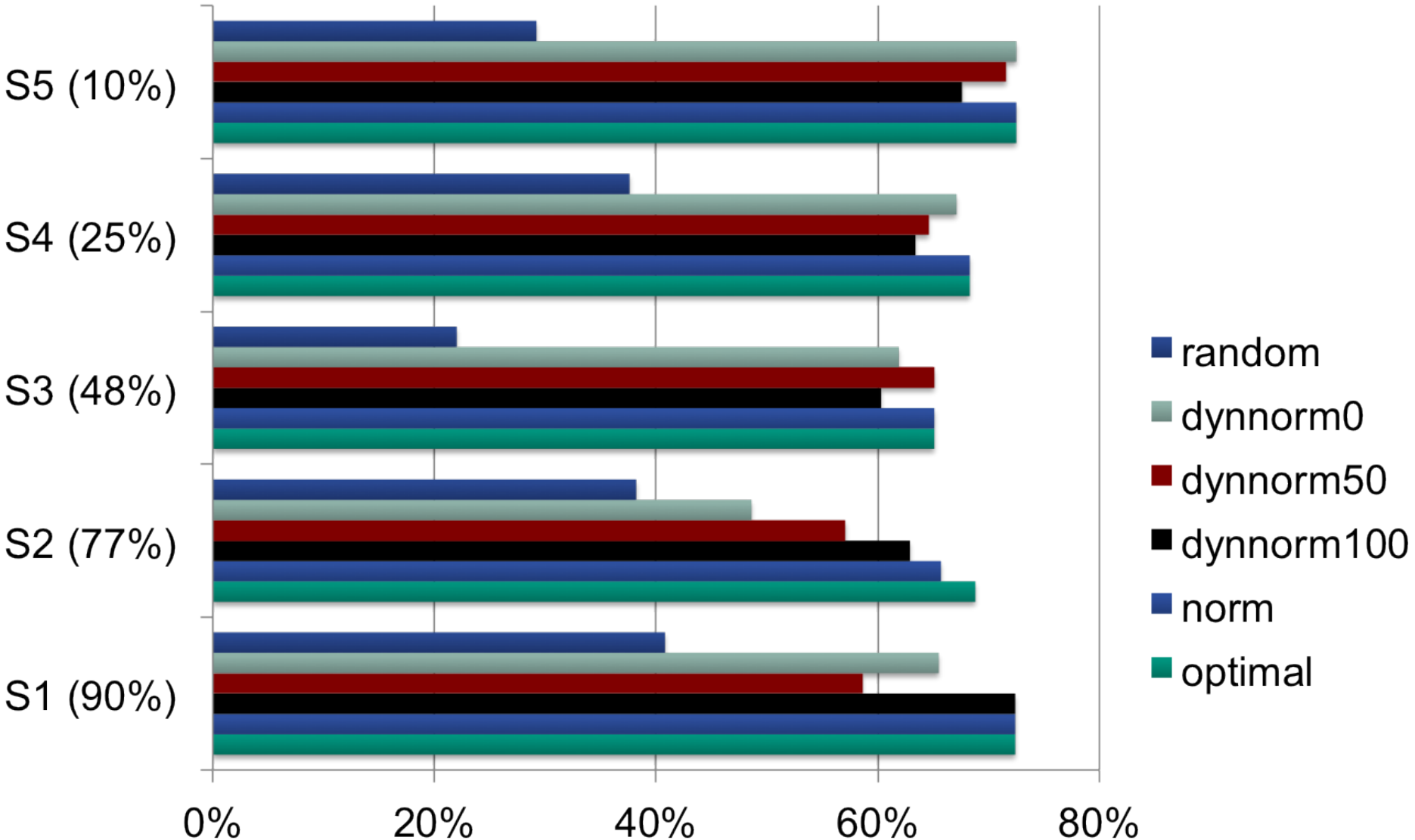
⇝ Develop a ranking function parametrized by the validity ratio

- The validity ratio is rarely known in advance

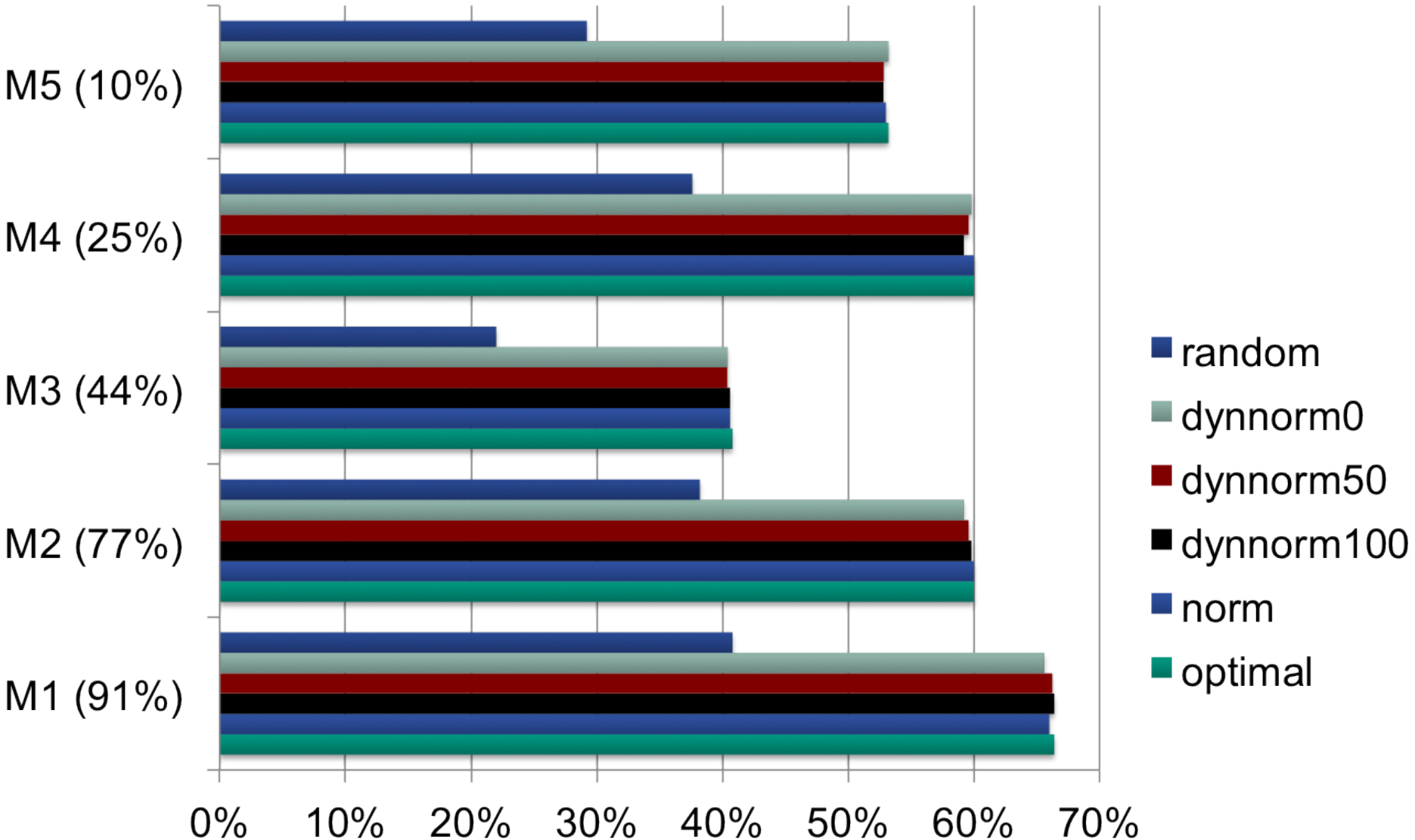
⇝ Estimate and lear the validity ratio for the parametrized ranking function

- Achieves a near maximum automation
- Gain is particularly important for datasets with a validity ratio close to 50%
- Even for small datasets (50–100 axioms) validity ratio can be learned effectively
- For larger datasets (e.g., 5,000 axioms and more) the learned validity ratio deviates only 0.3% from the known one

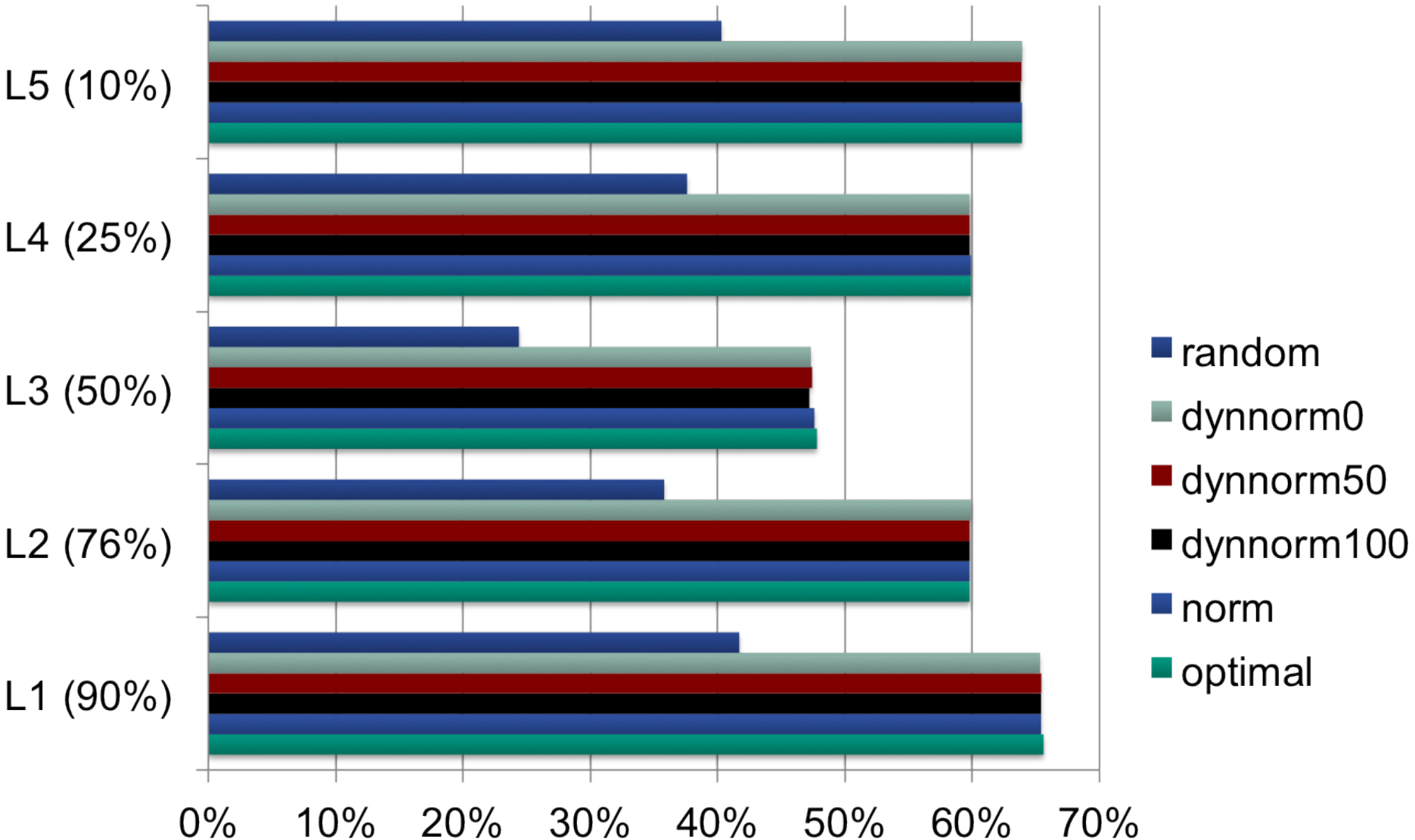
Evaluation Results with Learned Parametrized Ranking



Evaluation Results with Learned Parametrized Ranking



Evaluation Results with Learned Parametrized Ranking



Summary

- Revision closure partially automatizes ontology revision
  - Significantly reduces manual revision effort
  - Guarantees the consistency of approved axioms

Summary

- Revision closure partially automatizes ontology revision
  - Significantly reduces manual revision effort
  - Guarantees the consistency of approved axioms
- Choosing an appropriate order usually yields a higher effort reduction

Summary

- Revision closure partially automatizes ontology revision
  - Significantly reduces manual revision effort
  - Guarantees the consistency of approved axioms
- Choosing an appropriate order usually yields a higher effort reduction
- Impact function ($\rightsquigarrow$ determines oder) can be parametrized with the validity ratio

Summary

- Revision closure partially automatizes ontology revision
  - Significantly reduces manual revision effort
  - Guarantees the consistency of approved axioms
- Choosing an appropriate order usually yields a higher effort reduction
- Impact function ($\rightsquigarrow$ determines oder) can be parametrized with the validity ratio
- Validity ratio can effectively be learned over the course of the revision

Summary

- Revision closure partially automatizes ontology revision
  - Significantly reduces manual revision effort
  - Guarantees the consistency of approved axioms
- Choosing an appropriate order usually yields a higher effort reduction
- Impact function ($\rightsquigarrow$ determines oder) can be parametrized with the validity ratio
- Validity ratio can effectively be learned over the course of the revision
- Decision spaces are efficient for determining the revision closure and axiom impact and saved 75% of reasoning calls in our experiments

Outlook – Ontology Techniques for Companion-Systems

Ontologies as background knowledge for companion systems

- Revision support for building up the ontology (reuse)

Outlook – Ontology Techniques for Companion-Systems

Ontologies as background knowledge for companion systems

- Revision support for building up the ontology (reuse)
- Ontology might be incomplete
  - $\rightsquigarrow$ Ontology extension either at runtime or offline

Outlook – Ontology Techniques for Companion-Systems

Ontologies as background knowledge for companion systems

- Revision support for building up the ontology (reuse)
- Ontology might be incomplete
  - $\rightsquigarrow$ Ontology extension either at runtime or offline
- Ontology extension at runtime needs fully automatic revision

Outlook – Ontology Techniques for Companion-Systems

Ontologies as background knowledge for companion systems

- Revision support for building up the ontology (reuse)
- Ontology might be incomplete
  - $\rightsquigarrow$ Ontology extension either at runtime or offline
- Ontology extension at runtime needs fully automatic revision
- Can be combined with modularization techniques to extract relevant knowledge

Outlook – Ontology Techniques for Companion-Systems

Companion systems also need user specific knowledge

- Longer-term than current sensor data
- More dynamic knowledge
- For example DS-3:
  - We know that a user is unfamiliar with the exercises
  - Detection that user looks puzzled $\rightsquigarrow$ explain exercises
  - We know that a user is familiar with the exercises
  - Detection that user looks puzzled $\rightsquigarrow$ clarification why puzzled

Outlook – Ontology Techniques for Companion-Systems

Companion systems also need user specific knowledge

- Longer-term than current sensor data
- More dynamic knowledge
- For example DS-3:
  - We know that a user is unfamiliar with the exercises
  - Detection that user looks puzzled $\rightsquigarrow$ explain exercises
  - We know that a user is familiar with the exercises
  - Detection that user looks puzzled $\rightsquigarrow$ clarification why puzzled
- Also includes forgetting of knowledge

$\rightsquigarrow$ Stream reasoning: reasoning with continuously changing knowledge

- Uncertainty has to be taken into account