

# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

# Aufgabe 1



- Gegeben ist folgende Ontologie:

```
:Tier owl:disjointWith :Mensch .  
:hatHaustier rdfs:domain :Mensch .  
:hatHaustier rdfs:range :Tier .  
:Tom :hatHaustier :Sternchen .
```

- Wie schließt ein Tableau-Reasoner daraus, dass Sternchen kein Mensch ist?

# Aufgabe 1



- Erster Schritt: DL-Negations-Normalform
  - $\neg \text{Tier} \sqcup \neg \text{Mensch}$
  - $\neg \exists \text{hatHaustier.T} \sqcup \text{Mensch} \quad (\leftarrow \exists \text{hatHaustier.T} \sqsubseteq \text{Mensch})$
  - $\forall \text{hatHaustier.Tier} \quad (\leftarrow \text{T} \sqsubseteq \forall \text{hatHaustier.Tier})$
  - $\text{hatHaustier}(\text{Tom}, \text{Sternchen})$
- Wir wollen zeigen, dass  $\text{Mensch}(\text{Sternchen})$  nicht gilt
- Strategie: Fakt in Wissensbasis aufnehmen und nach Widerspruch fahnden

# Aufgabe 1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Mensch(Sternchen)

Nr	Aussage	Aktion
1	C(a)	Füge C(a) hinzu

# Aufgabe 1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Mensch(Sternchen), hatHaustier(Tom,Sternchen)

Nr	Aussage	Aktion
2	$R(a,b)$	Füge $R(a,b)$ hinzu

# Aufgabe 1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Mensch(Sternchen), hatHaustier(Tom,Sternchen),  
 $(\neg \text{Tier} \sqcup \neg \text{Mensch})(\text{Sternchen})$

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

# Aufgabe 1



Mensch(Sternchen), hatHaustier(Tom,Sternchen),  
( $\neg$ Tier  $\sqcup$   $\neg$ Mensch)(Sternchen), $\neg$ Tier(Sternchen)

Mensch(Sternchen), hatHaustier(Tom,Sternchen),  
( $\neg$ Tier  $\sqcup$   $\neg$ Mensch)(Sternchen), $\neg$ Mensch(Sternchen)

Nr	Aussage	Aktion
5	$(C \sqcup D)(a)$	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

# Aufgabe 1



Mensch(Sternchen), hatHaustier(Tom,Sternchen),  
 $(\neg\text{Tier} \sqcup \neg\text{Mensch})(\text{Sternchen}), \neg\text{Tier}(\text{Sternchen}),$   
 $\forall\text{hatHaustier.Tier}(\text{Tom})$

Mensch(Sternchen), hatHaustier(Tom,Sternchen),  
 $(\neg\text{Tier} \sqcup \neg\text{Mensch})(\text{Sternchen}), \neg\text{Mensch}(\text{Sternchen})$

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu



# Aufgabe 1



Mensch(Sternchen), hatHaustier(Tom, Sternchen),  
( $\neg$ Tier  $\sqcup$   $\neg$ Mensch)(Sternchen),  $\neg$ Tier(Sternchen),  
 $\forall$ hatHaustier.Tier(Tom), Tier(Sternchen)

Mensch(Sternchen), hatHaustier(Tom, Sternchen),  
( $\neg$ Tier  $\sqcup$   $\neg$ Mensch)(Sternchen),  $\neg$ Mensch(Sternchen)

Nr	Aussage	Aktion
7	$(\forall R.C)(a)$	Für alle b mit $R(a,b) \in T$ : füge $C(b)$ hinzu

# Aufgabe 1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Damit sind alle Teiltableaus abgeschlossen
  - d.h., es ergibt sich in jedem Fall ein Widerspruch
- Die Aussage Mensch(Sternchen) kann also nicht stimmen
  - ausgehend davon, dass unsere restliche Wissensbasis wahr ist

# Aufgabe 2

- Ein Schachverein möchte seine Daten mit einem ontologiebasierten System verwalten. Folgende Spezifikation ist gegeben:

*Der Verein hat Mitglieder. Jedes Jahr veranstaltet der Verein ein Turnier, das aus mehreren Runden besteht. In jeder Runde werden parallel mehrere Partien gespielt, bei denen jeweils zwei Mitglieder gegeneinander spielen.*

Finden Sie zunächst die Basisklassen und -relationen. Bauen Sie dann zwei Ontologien. Nutzen Sie dazu einmal die Top-Kategorien der Ontologie DOLCE, einmal die Top-Kategorien der Ontologie SUMO.

# Aufgabe 2



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

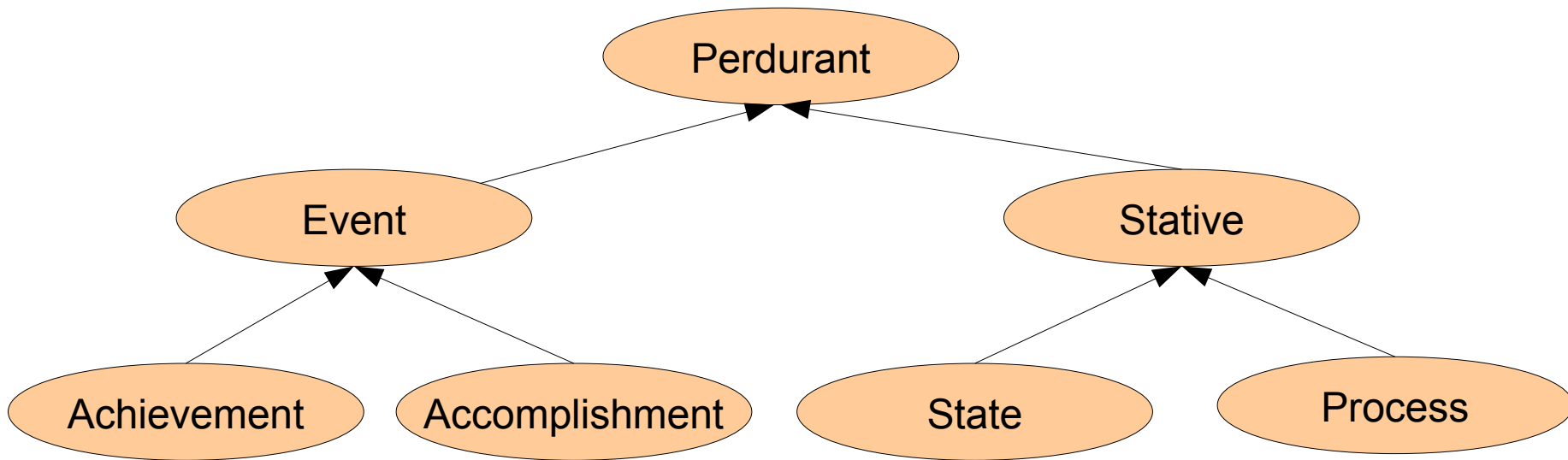
- Basisklassen:
  - Verein
  - Mitglied
  - Turnier
  - Runde
  - Partie

# Aufgabe 2



- Passende Superklassen in DOLCE
  - Verein < AgentiveSocialObject
    - Vereine können auch handeln (z.B. Mitglieder aufnehmen)
  - Mitglied < AgentivePhysicalObject
    - Faustregel: das passt meist auf Menschen
  
- Was machen wir mit Turnier, Runde und Partie?
  - auf jeden Fall sind es Perdurants

# Aufgabe 2



Masolo et al. (2003): *Ontology Library (final)*. WonderWeb Deliverable D18.

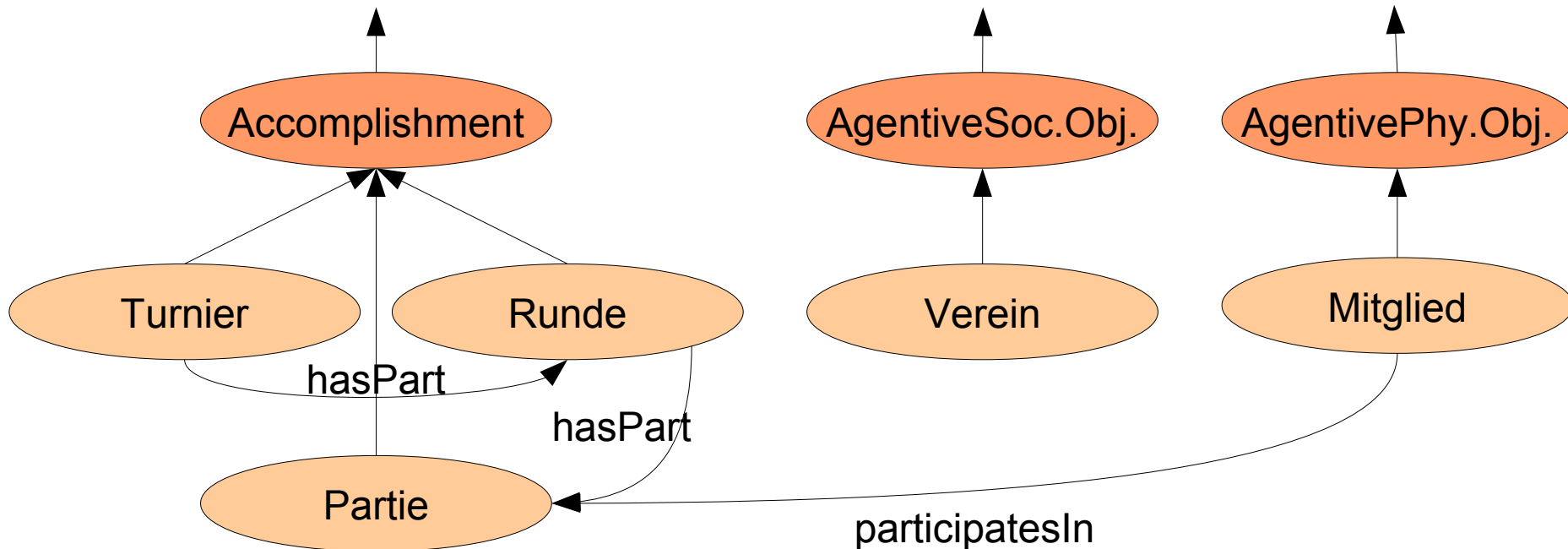
# Aufgabe 2



- Event vs. Stative
  - Summe von zwei Statives ist ein Stative vom gleichen Typ
  - zwei Partien/Runden/Turniere hintereinander gespielt sind aber nicht einfach ein(e) längere(s) Partie/Runde/Turnier
  - also ein Event
  
- Achievement vs. Accomplishment
  - Achievements sind unteilbar
  - Aber: Turniere bestehen aus Runden, Runden bestehen aus Partien, Partien bestehen aus Zügen
  - also: Accomplishment

# Aufgabe 2

- Lösung mit DOLCE:





# Aufgabe 2

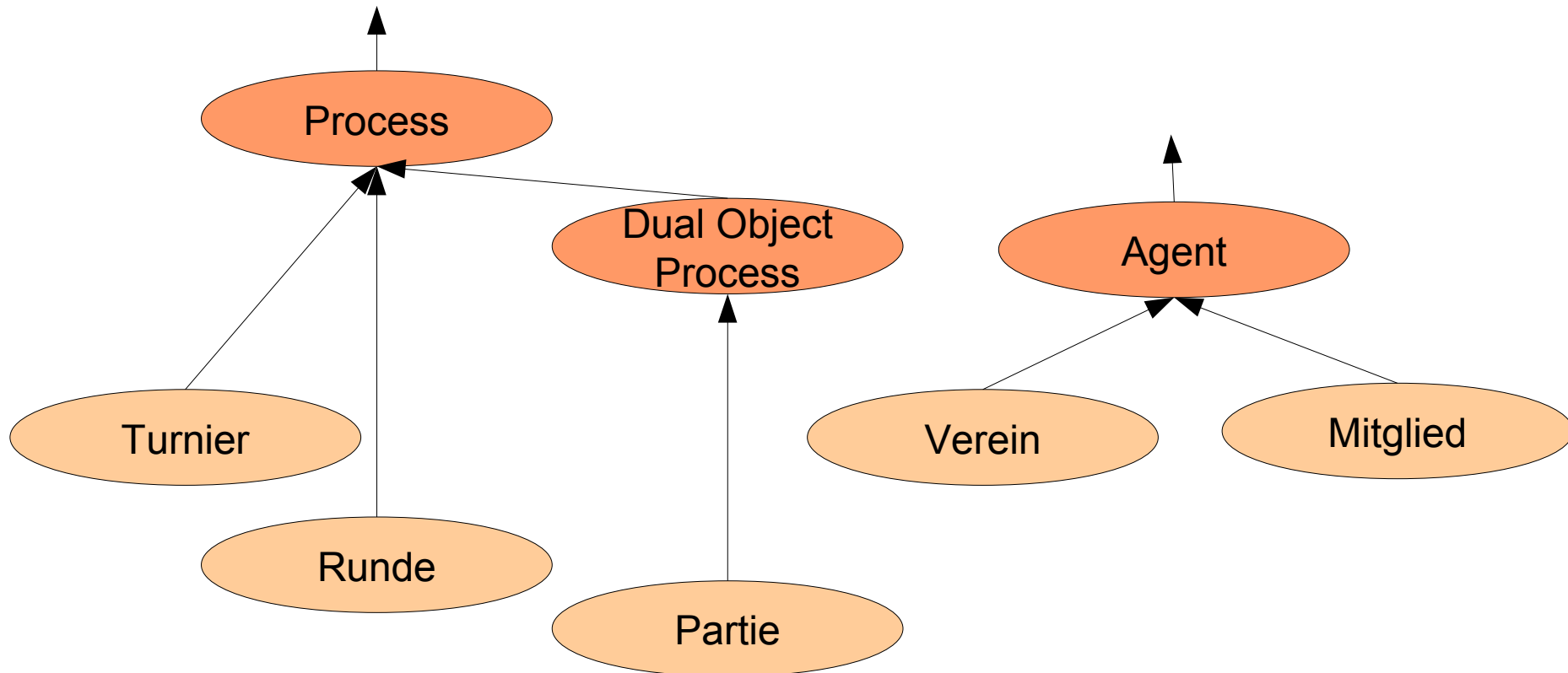


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Passende Superklassen in SUMO:
  - Verein < Agent
  - Mitglied < Agent
  - Turnier < Process
  - Runde < Process
  - Partie < Process (evtl. Dual Object Process)

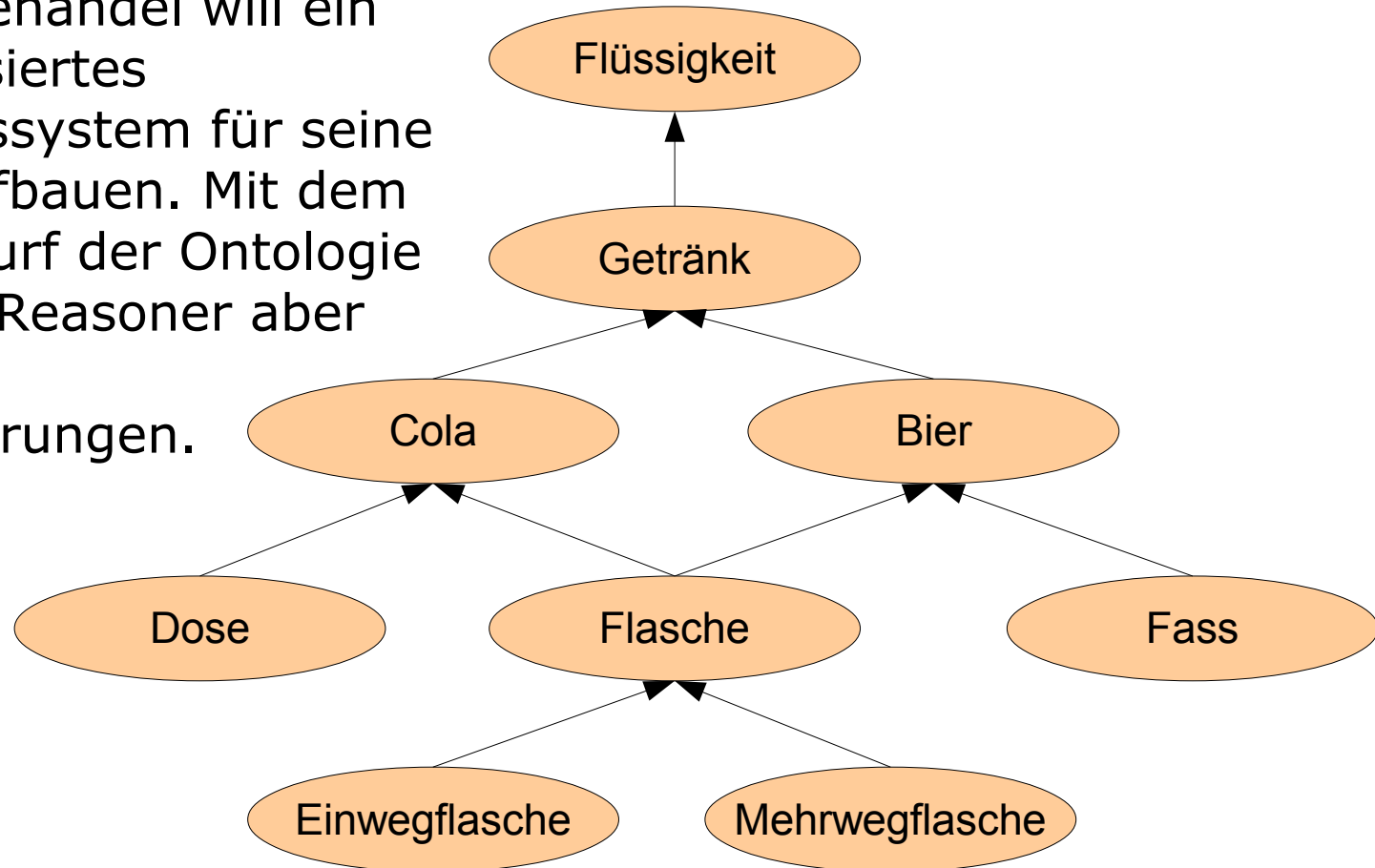
# Aufgabe 2

- Lösung mit SUMO:



# Aufgabe 3

- Ein Getränkehandel will ein ontologiebasiertes Informationssystem für seine Produkte aufbauen. Mit dem ersten Entwurf der Ontologie erzeugt der Reasoner aber unsinnige Schlussfolgerungen.



# Aufgabe 3



- Geben Sie einige Beispiele für unsinnige Schlussfolgerungen mit dieser Ontologie an.
- Gegeben:  
:0.5lFlascheSchädelbräu a :Einwegflasche .
- Dann folgt daraus (unter anderem):  
:0.5lFlascheSchädelbräu a :Bier .  
:0.5lFlascheSchädelbräu a :Cola .
- Das ist irgendwie merkwürdig.

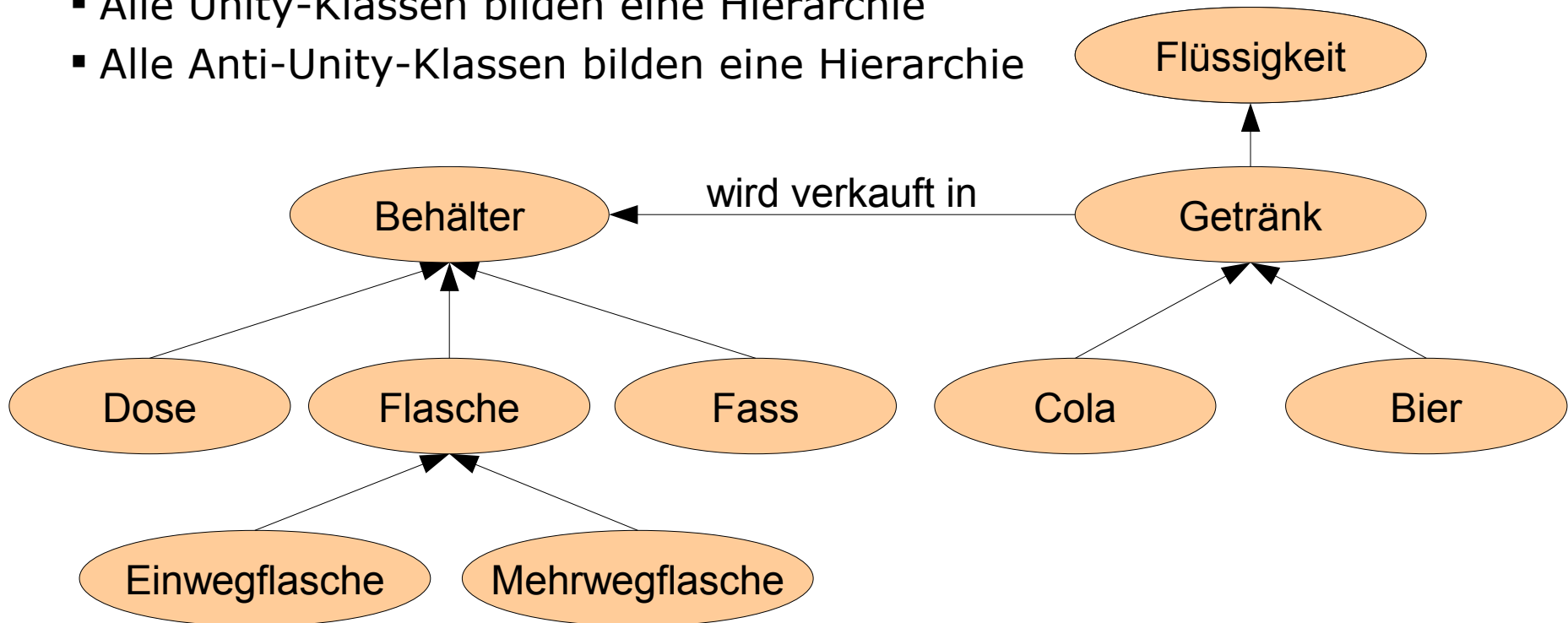
# Aufgabe 3



- Gegen welche Regel in OntoClean wurde hier verstoßen?
- *Flüssigkeit* ist eine Anti-Unity-Klasse
  - Eine Menge Flüssigkeit zerfällt in zwei Teilmengen Flüssigkeit
- *Flasche* ist eine Unity-Klasse
  - Flaschen sind ganze, zählbare Objekte
- OntoClean:
  - Unity-Klassen dürfen nicht von Anti-Unity-Klassen erben (und umgekehrt)

# Aufgabe 3

- Neu sortierte Ontologie:
  - Alle Unity-Klassen bilden eine Hierarchie
  - Alle Anti-Unity-Klassen bilden eine Hierarchie



# Aufgabe 3

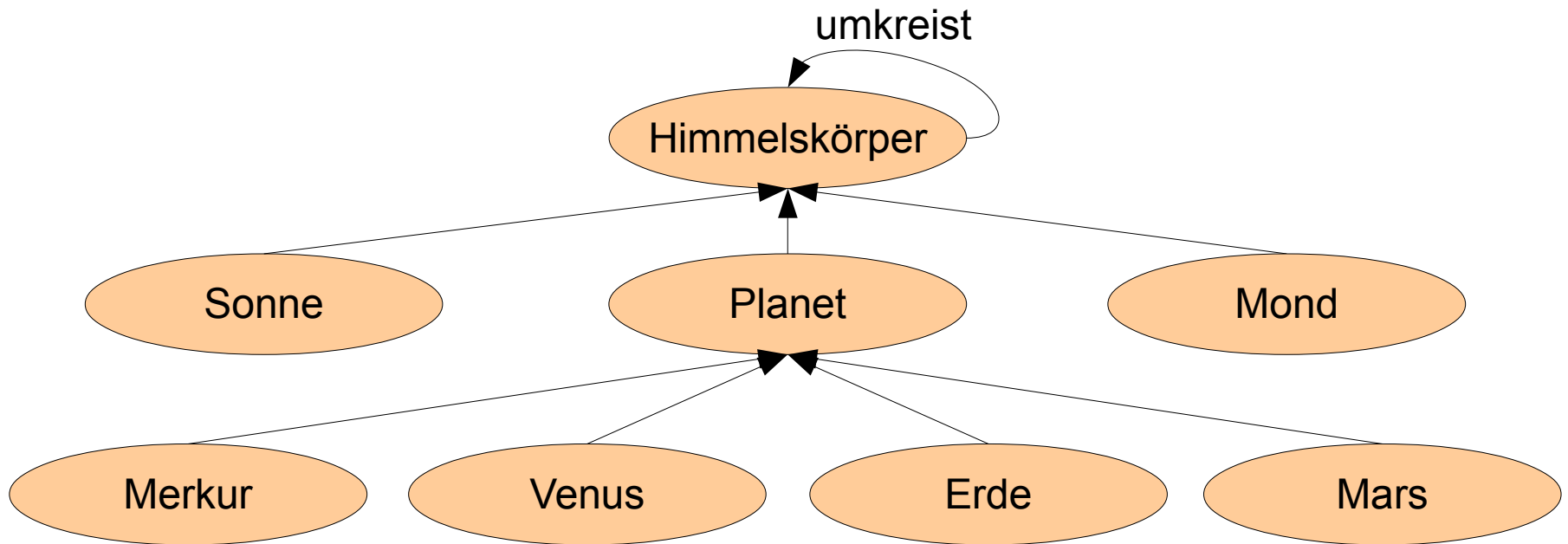


- Die ursprüngliche Ontologie sollte wahrscheinlich noch andere Aussagen enthalten
  - z.B.: Cola wird in Dosen und Flaschen verkauft
- Das können wir jetzt ordentlich modellieren:

```
:Cola rdfs:subClassof [  
  a owl:Restriction ;  
  owl:onProperty wirdVerkauftIn ;  
  owl:allValuesFrom [  
    owl:unionOf (Dose Flasche) ] ] .
```

# Aufgabe 4

- Die Astronomie-AG hat folgende Ontologie gebaut:





# Aufgabe 4



- Folgende Axiome sind gegeben:

```
:umkreist a owl:ObjectProperty .  
:umkreist rdfs:subPropertyOf :wirdAngezogenVon .  
:wirdAngezogenVon a owl:SymmetricProperty .  
:Mond :umkreist :Erde .  
:Erde :umkreist :Sonne .
```

- Warum kann man damit nicht z.B. folgendes schließen:

```
:Mond :wirdAngezogenVon :Erde .
```

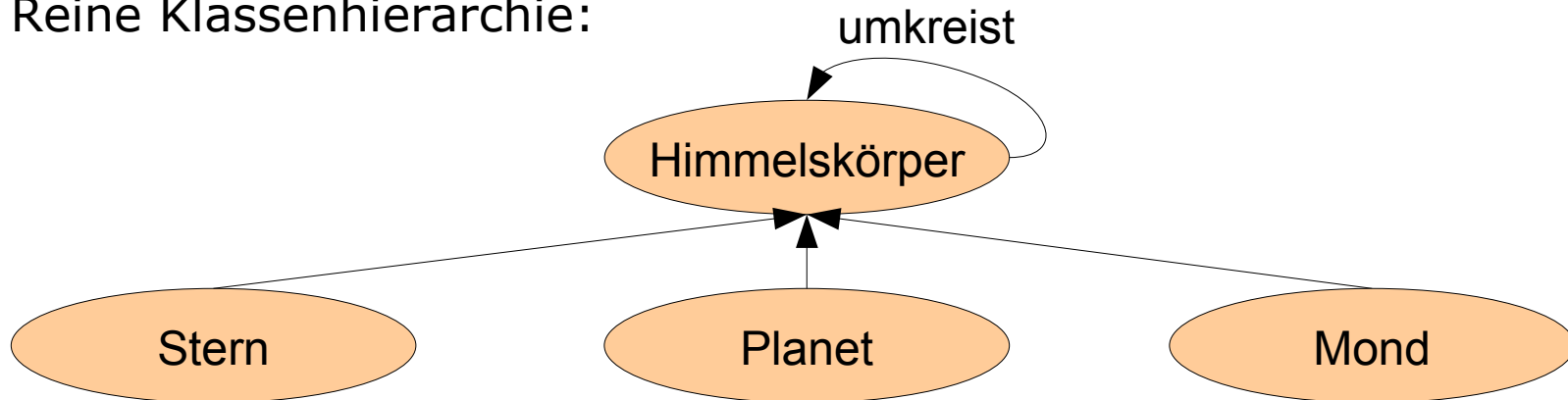
# Aufgabe 4

- In OWL DL müssen wir Klassen und Instanzen strikt trennen
- Das haben wir nicht gemacht...
  - Mond als Klasse:  
`:Mond rdfs:subClassOf :Himmelskörper .`
  - Mond als Instanz:  
`:Mond :umkreist :Erde .`
- Das ist kein gültiges OWL DL!
  - ...und damit für einen Reasoner nicht zu gebrauchen

# Aufgabe 4



- Beobachtet: Das Anti-Pattern "Klassenwildwuchs"
  - viele der Klassen sind eigentlich Instanzen!
  - Reine Klassenhierarchie:



- Instanzen:
  - Stern: Sonne
  - Planet: Merkur, Venus, Erde, Mars ...
  - Mond: Erdmond

# Aufgabe 5



- Gegeben ist folgende Instanzmenge:
  - :Darmstadt a :Stadt, :GeographischesObjekt .
  - :Frankfurt a :Stadt, :GeographischesObjekt .
  - :München a :Stadt, :GeographischesObjekt .
  - :Hamburg a :Stadt , :Bundesland .
  - :Berlin a :Stadt, :Bundesland .
  - :Bremen a :Stadt, :Bundesland, GeographischesObjekt .
  - :Hessen a :Bundesland .
  - :Bayern a :Bundesland, :GeographischesObjekt .
  - :Sachsen a :Bundesland, :GeographischesObjekt .
  - :Deutschland a :Staat, :GeographischesObjekt .
- Lernen Sie mit Hilfe des Apriori-Algorithmus eine Klassenhierarchie aus diesen Fakten. Wählen Sie dabei zunächst  $\text{minSupport}=0.25$  und  $\text{minConfidence}=0.5$ .

# Aufgabe 5



- Erster Schritt: Frequent Itemsets finden
- Starte mit 1-elementigen, bestimme Support
  - {GeographischesObjekt} (0.7)
  - {Stadt} (0.6)
  - {Bundesland} (0.6)
- fahre fort: Bilde 2-elementige Itemsets
  - {GeographischesObjekt, Stadt} (0.4)
  - {GeographischesObjekt, Bundesland} (0.3)
  - {Stadt, Bundesland} (0.3)

# Aufgabe 5

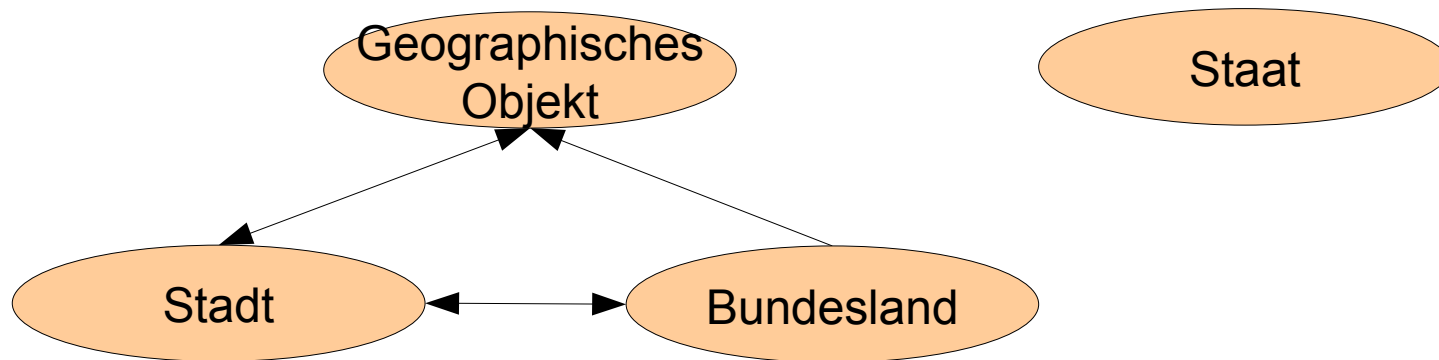


- Zweiter Schritt: Itemsets zu Regeln, Confidence bestimmen
  - Stadt → GeographischesObjekt (0.66)
  - GeographischesObjekt → Stadt (0.57)
  - Bundesland → GeographischesObjekt (0.5)
  - GeographischesObjekt → Bundesland (0.43)
  - Stadt → Bundesland (0.5)
  - Bundesland → Stadt (0.5)

# Aufgabe 5



- Gelernte Klassenhierarchie:



- Defekte:
  - Zu viele Vererbungshierarchien
  - Staat ist nicht enthalten
- Lösung: Parameter verändern

# Aufgabe 5



- Alle möglichen Regeln mit Support und Confidence:
  - **Stadt** → **GeographischesObjekt** (0.4, 0.66)
  - GeographischesObjekt → Stadt (0.4, 0.57)
  - **Bundesland** → **GeographischesObjekt** (0.3, 0.5)
  - GeographischesObjekt → Bundesland (0.3, 0.43)
  - Bundesland → Stadt (0.3, 0.5)
  - Stadt → Bundesland (0.3, 0.5)
  - **Staat** → **GeographischesObjekt** (0.1, 1.0)
  - GeographischesObjekt → Staat (0.1, 0.14)



# Aufgabe 5



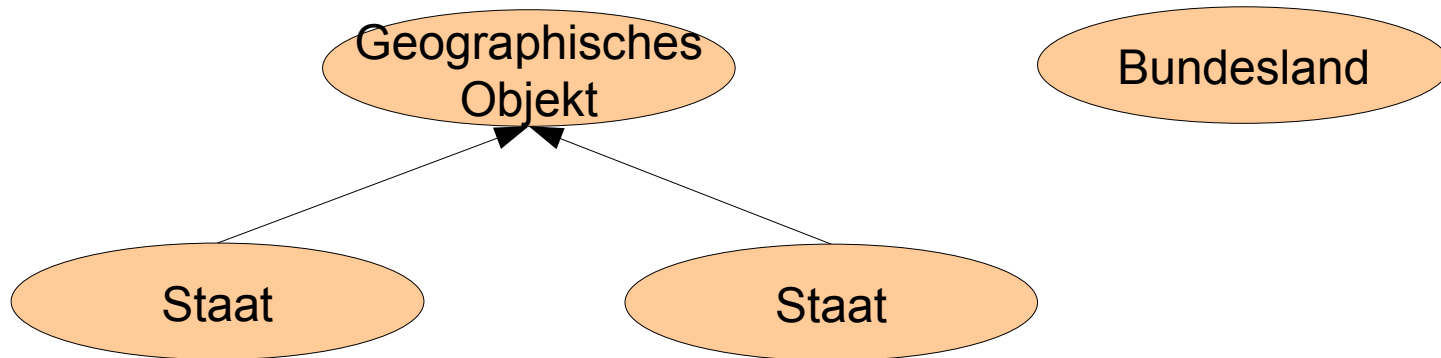
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Beobachtungen:
  - Um Staat < GeographischesObjekt zu finden, brauchen wir  $\text{minSupport} = 0.1$
  - Alle sinnvollen Regeln haben mindestens Confidence 0.5.
  - ab  $\text{minConfidence} = 0.6$  findent man **nur** sinnvolle Regeln
  - Bestimmte Fehler lassen sich nicht kostenlos ausschließen!

# Aufgabe 5



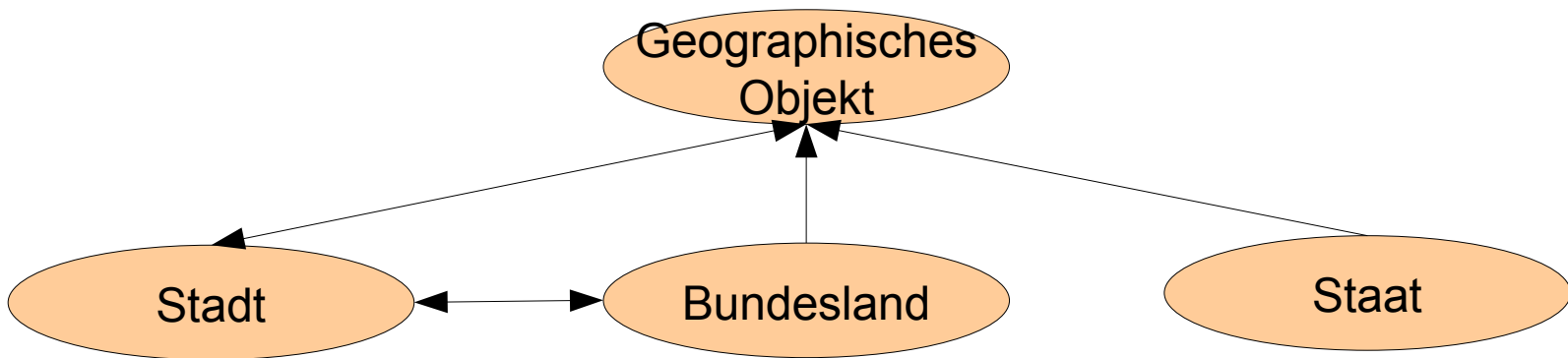
- Pessimistische Lösung (Precision-optimiert):
  - $\text{minSupport} = 0.1$ ,  $\text{minConfidence} = 0.6$



# Aufgabe 5



- Optimistische Lösung (Recall-optimiert):
  - $\text{minSupport} = 0.1$ ,  $\text{minConfidence} = 0.5$



# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering