Projekt SmartWeb - Webdienste und Integration von semantischen Webressourcen

Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Knowledge Engineering

Diplomarbeit

Grigori Babitski

Projekt: SmartWeb

Firma: **E**uropean **M**edia **L**aboratory

Projekt: SmartWeb

1. WWW zugänglicher durch Semantic Web
2. Zugang an Semantic Web von mobilen Geräten über multimodale Schnittstelle

Teilprojekt: Webdienste und Integration von semantischen Webressourcen

Modul: SitCom

Semantische Repräsentation mit der Kontextinformation bereichern

„Ich möchte zum Schloss"

↓ **Kontextmodell**

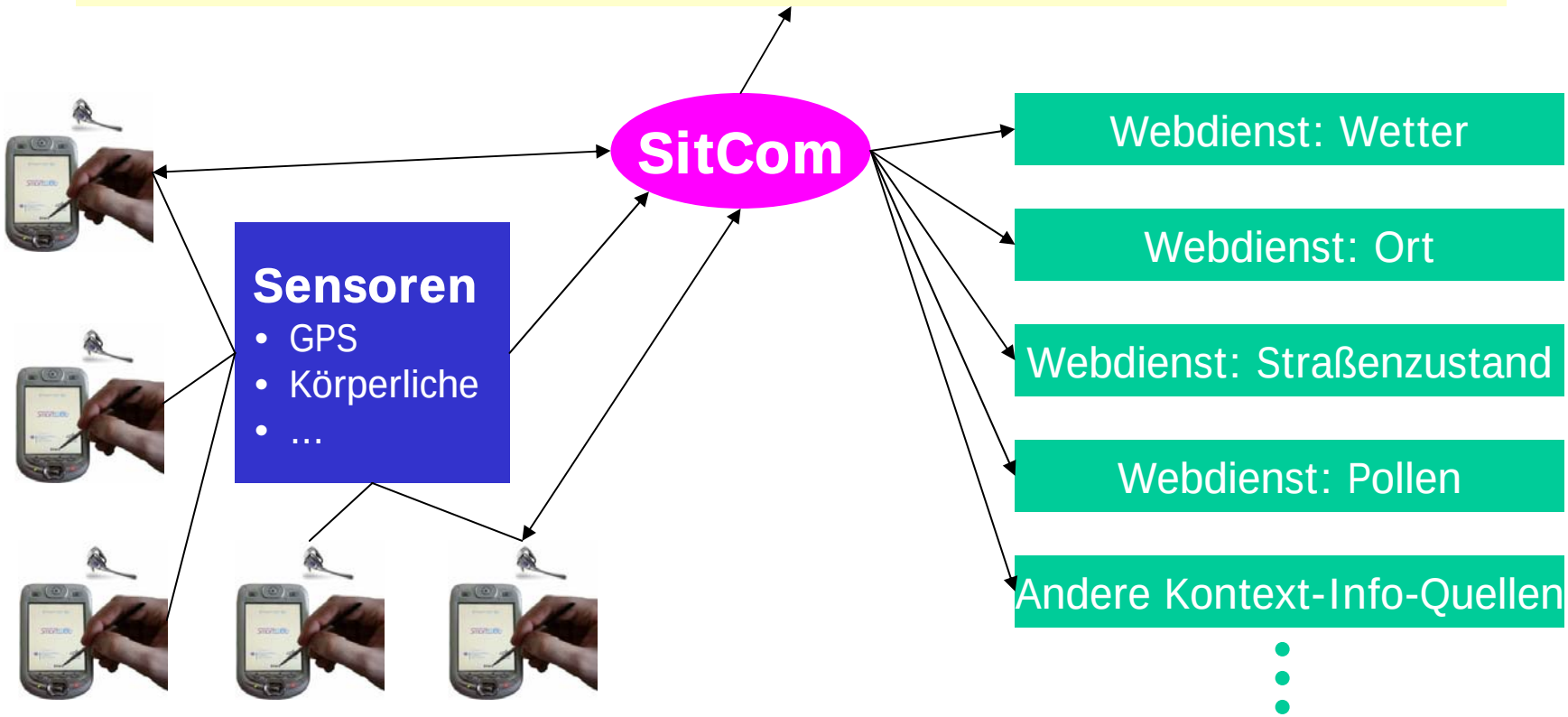
„Ich möchte zu Fuß zum Schloss (Adresse…)"

SitCom und Cache

User X	Zeit t0	Ort 1	Wetter 1
User X	Zeit t1		Wetter 2
User Y	Zeit t0	Ort 1	

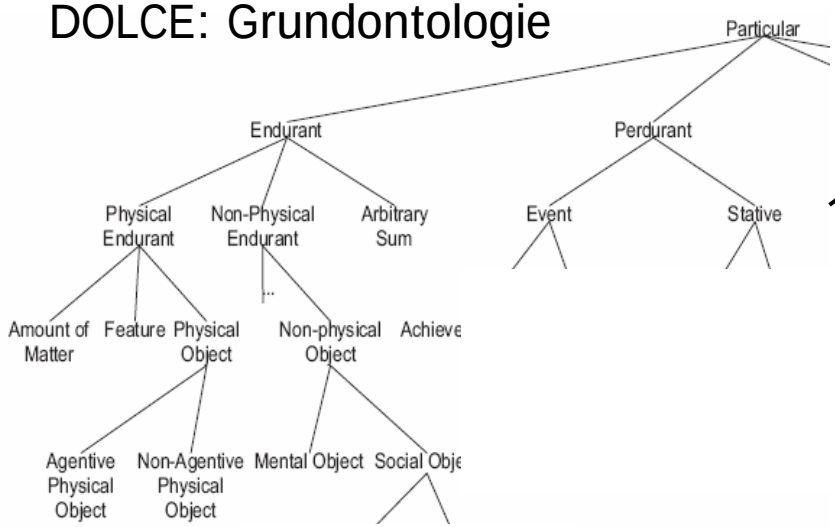
Cache:

flexibel, schnell



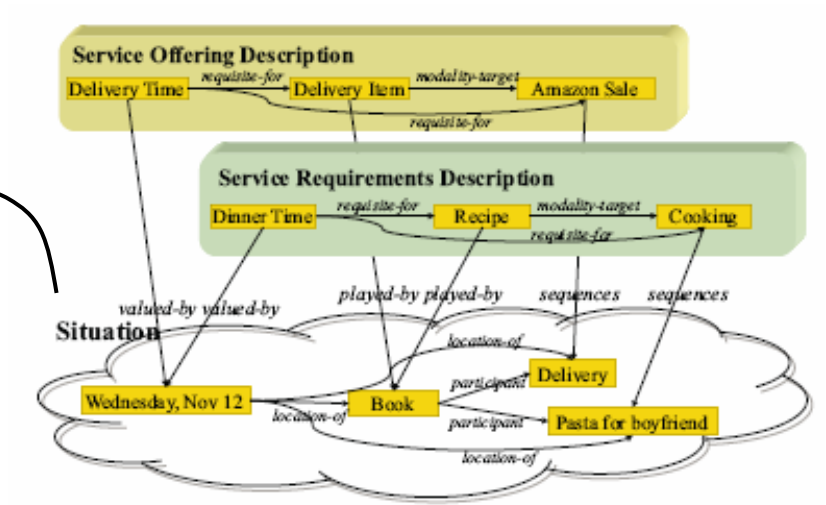
Ontologien

DOLCE: Grundontologie



„Ich möchte zum Schloss“

„Ich möchte zu Fuß zum Schloss



Descriptions & Situations Ontologie

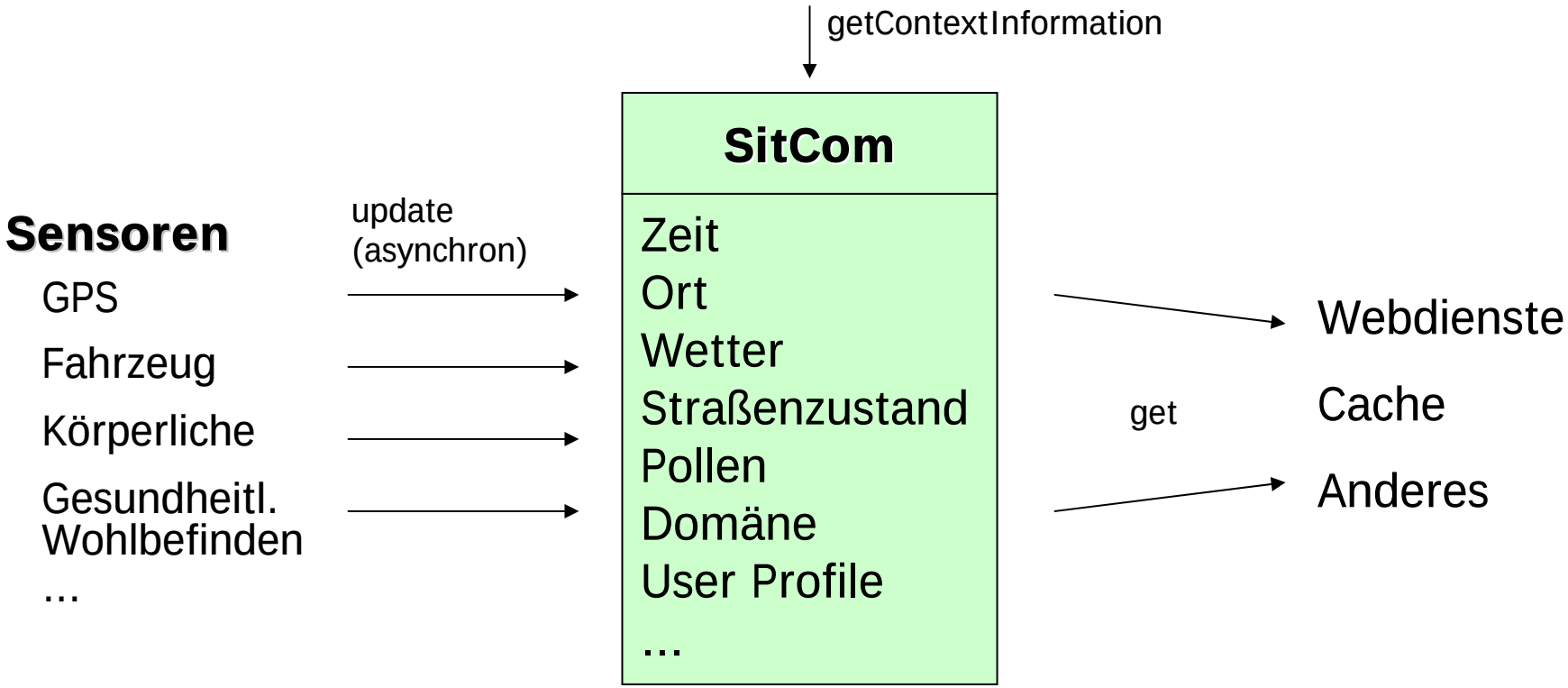
Kontext

- Hintergrundinformationen (Ort, Zeit, Wetter…)
- Gemeintes, aber nicht Gesagtes

  „ich möchte ***zu Fuß*** zum Schloss“
- Kontext einer Aussage – ihr Gegenstand / Thema

  Intention der Fortbewegung zum Ziel (Locomotion)

Kontextmodell



- Erweiterbarkeit
- Multithreading

Cache

Type:	UserID	Zeit	Ort	Wetter
Eintrag 1:	User X	Zeit t0	Ort 1	Wetter 1
Eintrag 2:	User X	Zeit t1		Wetter 2
Eintrag 3:	User Y	Zeit t0	Ort 1	

Value

Anfragen können time- und value thresholds spezifizieren

time threshold: Zeitpunkt, zu dem etwas im Cache gespeichert wurde

value threshold: Abweichung eines value im Cache von der value in der Anfrage

„gib alle User zurück, über die Cache Eintrag aus Zeit $t \pm \Delta$ gibt, der beinhaltet, dass diese User am Ort $x \pm \Delta$ sich aufhalten"

Cache: Methoden

store ($type_1$,..., $type_n$,
$value_1$,...,$value_n$)

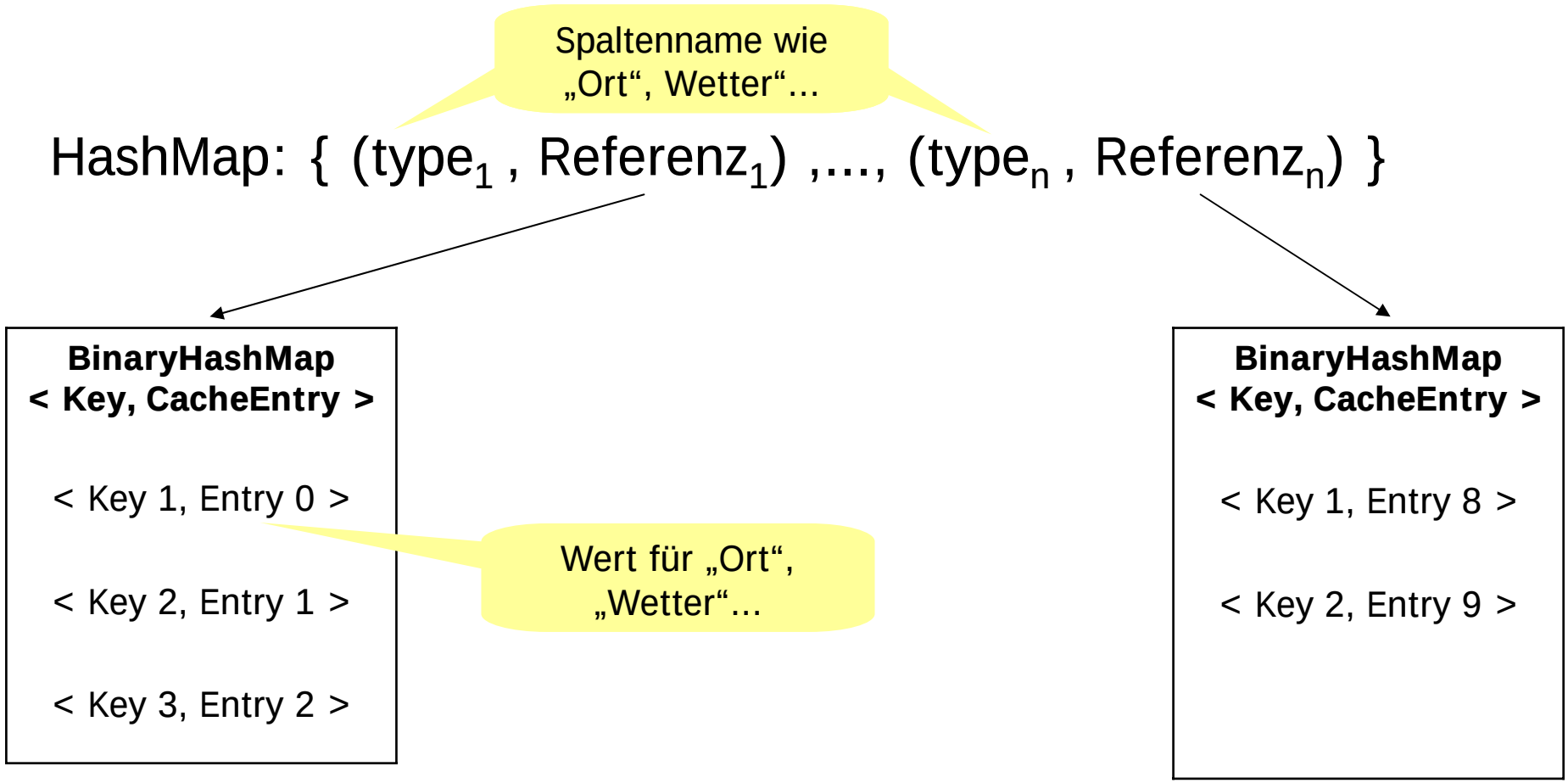
get ($type_1$,..., $type_n$,
$value_1$,..., $value_n$,
$value\_threshold_1$,..., $value\_threshold_n$,
$time\_threshold_1$,..., $time\_threshold_n$,
$obligatory\_request\_type_1$,..., $obligatory\_request\_type_m$,
$optional\_request\_type_1$,..., $optional\_request\_type_k$)

„aus jedem Eintrag in Cache, der alle vorgegebene type-value-Paare hat (mit Berücksichtigung von thresholds) und zu allen obligatorisch angefragten types irgendwelche values hat, gib diese values, sowie falls vorhanden, values optional angefragter types zurück"

removeRecords: „entferne jeden Eintrag, der alle vorgegebene type-value-Paare hat (mit Berücksichtigung von thresholds)"

removeValues: „aus einem Eintrag in Cache, der alle vorgegebene type-value-Paare hat (mit Berücksichtigung von thresholds) entverne alle values von vorgegebenen types"

Cache: Aufbau

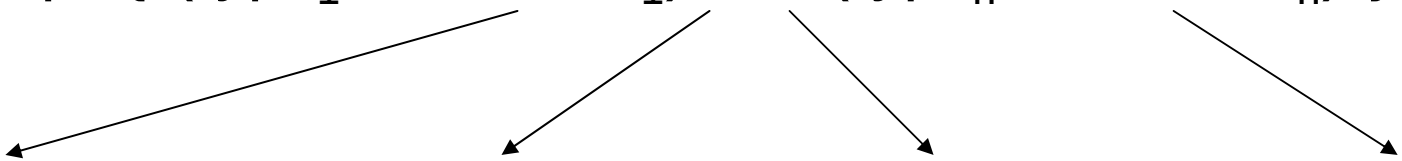


BinaryHashMap – HashMap, in dem key über value (mit gleicher Komplexität) abrufbar

Cache: Aufbau

Spaltenname wie „Ort“, Wetter“…

HashMap: { (type$_1$, Referenz$_1$) ,…, (type$_n$, Referenz$_n$) }



BinaryHashMap < Key, CacheEntry >	**BinaryHashMap < Key, CacheEntry >**	**BinaryHashMap < Key, CacheEntry >**	**BinaryHashMap < Key, CacheEntry >**
< Key 1, Entry 0 >	< Key 1, Entry 3 >	< Key 1, Entry 6 >	< Key 1, Entry 8 >
< Key 2, Entry 1 >	< Key 2, Entry 4 >		< Key 2, Entry 9 >
< Key 3, Entry 2 >	< Key 3, Entry 5 >	< Key 3, Entry 7 >	

BinaryHashMap – HashMap, in dem key über value (mit gleicher Komplexität) abrufbar

Cache: Aufbau

Spaltenname wie „Location“,Weather“…

HashMap: { ($type_1$, $Referenz_1$) ,…, ($type_n$, $Referenz_n$) }

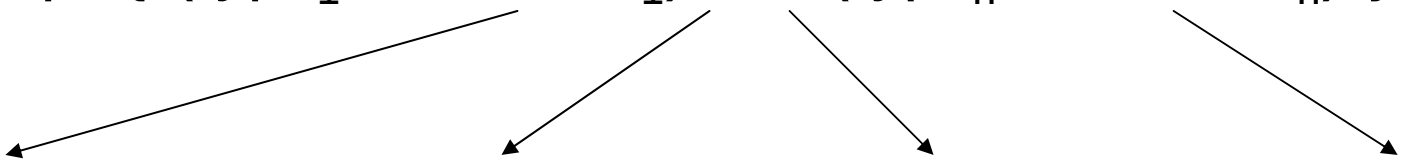
BinaryHashMap < Key, CacheEntry > UserID	**BinaryHashMap < Key, CacheEntry > Zeit**	**BinaryHashMap < Key, CacheEntry > Ort**	**BinaryHashMap < Key, CacheEntry > Wetter**
< Key 1, Entry 0 > User X	< Key 1, Entry 3 > Zeit t0	< Key 1, Entry 6 > Ort 1	< Key 1, Entry 8 > Wetter 1
< Key 2, Entry 1 > UserX	< Key 2, Entry 4 > Zeit t1		< Key 2, Entry 9 > Wetter 2
< Key 3, Entry 2 > User Y	< Key 3, Entry 5 > Zeit t0	< Key 3, Entry 7 > Ort 1	

BinaryHashMap – HashMap, in dem key über value (mit gleicher Komplexität) abrufbar

Cache: Aufbau

Spaltenname wie „Location“,Weather“…

HashMap: { (type$_1$, Referenz$_1$) ,…, (type$_n$, Referenz$_n$) }



UserID	Zeit	Ort	Wetter
User X	Zeit t0	Ort 1	Wetter 1
UserX	Zeit t1		Wetter 2
User Y	Zeit t0	Ort 1	

BinaryHashMap – HashMap, in dem key über value (mit gleicher Komplexität) abrufbar

Cache

Anfrage liefert i.A. Menge von Ergebnissen. Ist value threshold gesetzt, so liegen die Ergebnisse ungleich „nah“ zu der Anfrage

Wetter 2 Ort 8 User ? ⟶

1) Wetter 1 Ort 9 User 0

2) Wetter 4 Ort 6 User 1

Ob 1) oder 2) naher ist hängt vom Wertebereich von Wetter und Ort ab

Es wird bestimmt, welches Ergebnis das beste ist (ohne Verlust an Komplexität)

Ontologien

Ontologie – formal definiertes System von Begriffen und Relationen

i.d.R. mittels Beschreibungslogik

Begrenzt als Graph darstellbar

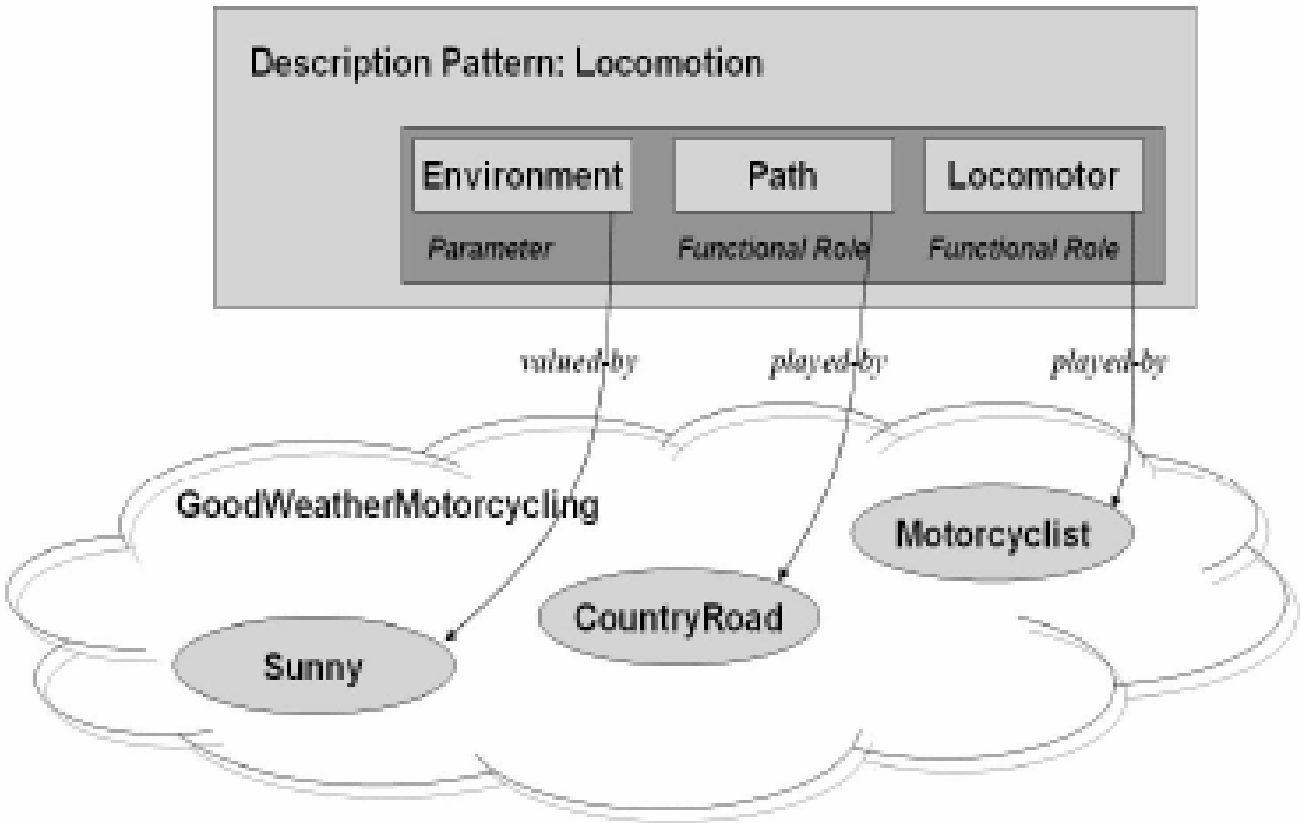
1. Grundontologie (z.B. DOLCE) – für Oberbegriffe; rel. Abstract
2. Domänenontologie – repräsentiert Wissen über eine bestimmte Domäne (z.B. Sportereignisse)
3. Descritpitions & Situations (DnS) – plug-in für DOLCE

   Ontologie, mit deren Hilfe z.B. Kontexte (von Aussagen) modellierbar sind

   Eine Aussage ist in 1 und 2 repräsentiert

„Ich möchte zum Schloss" → Kontext: Fortbewegung

DnS

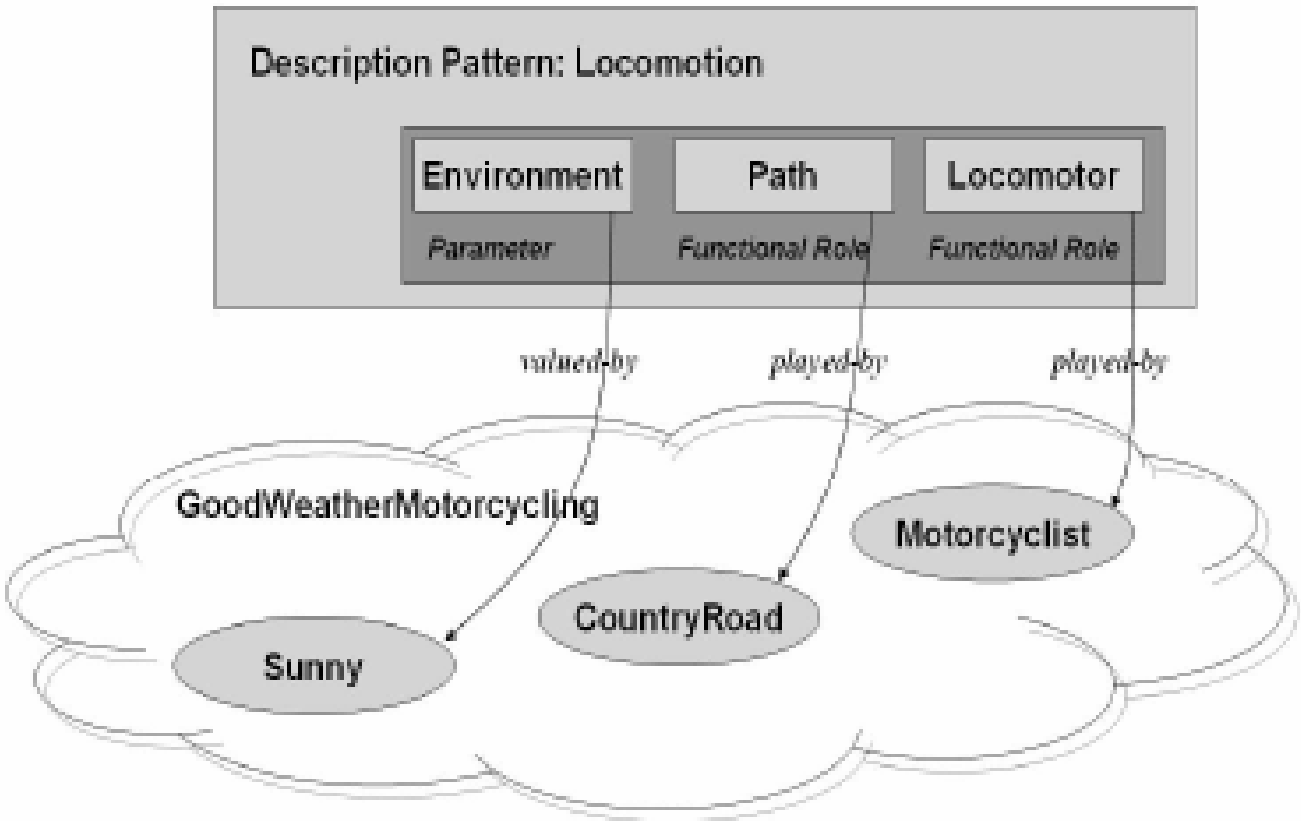


← Deskription

← Situation

DnS legt fest wie man nicht physikalische Begriffe repräsentiert und stellt dafür ontologische Mittel zur Verfügung

DnS



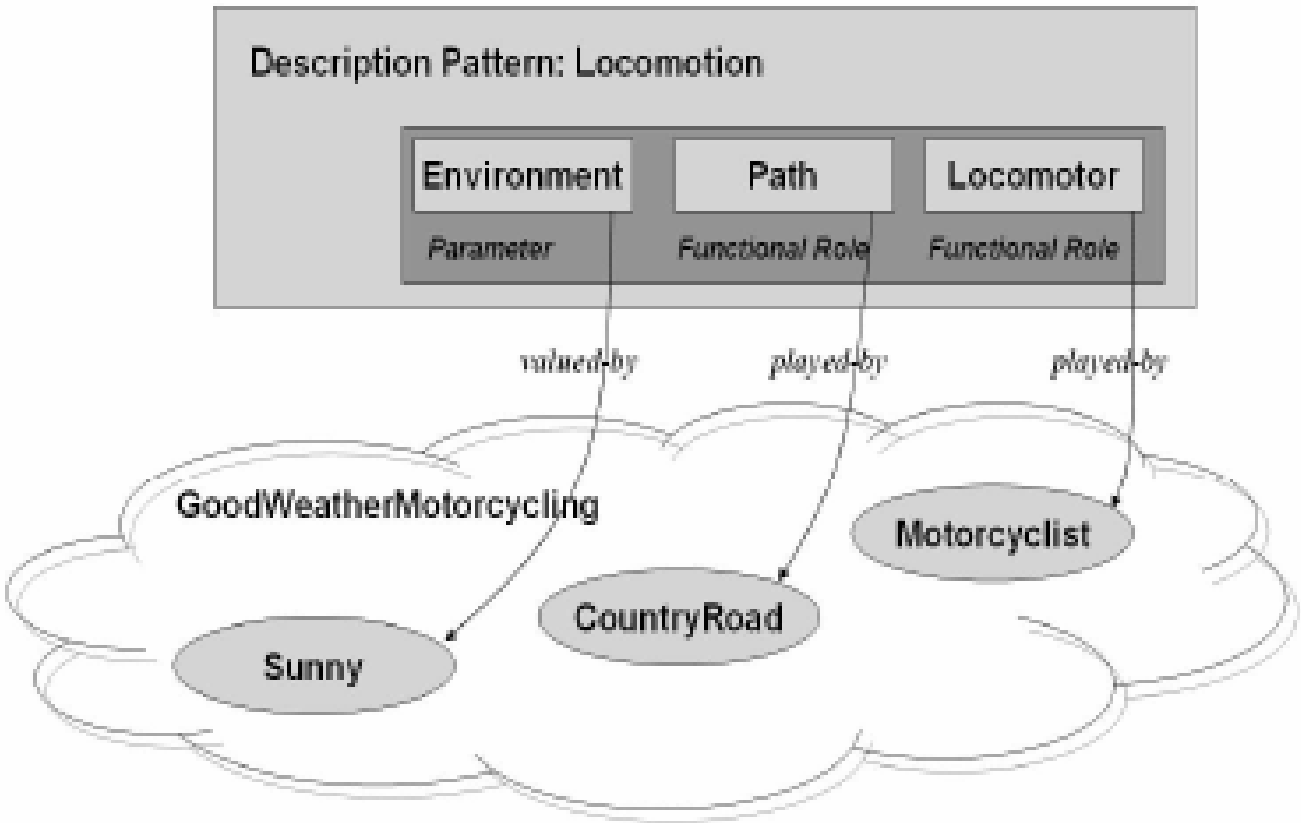
← Deskription

← Situation

Ein nicht physikalischer Begriff wird in einer Deskription beschrieben durch

- Rollen, die Gegenstände/Personen spielen
- Parameter
- Ggf. Abläufe, die stattfinden
- Festlegung wer die Rollen spielen darf; welches Wertebereich die Parameter haben

DnS



← Deskription

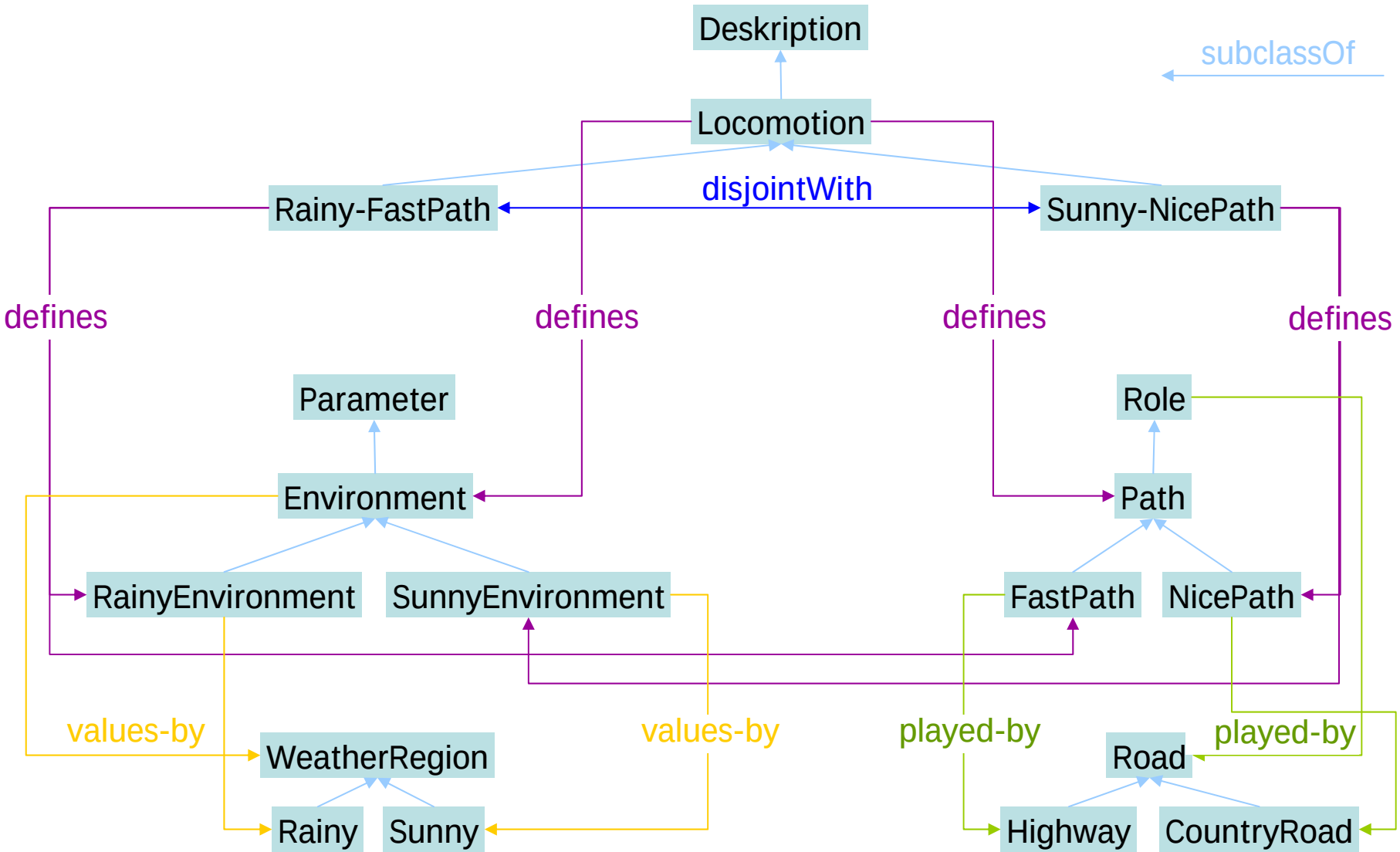
← Situation

Situation ist eine Menge konkreter ontologischer Entitäten (Instanzen)

Eine Situation kann einer Deskription genügen, wenn sie diese (teilweise) „instanziiert“

DnS

Problem: vorgegebene Deskriptionenmodellierung war misslungen



TODO

Eingabe: XML-Snippet, der die Benutzeraussage/Anfrage mit in DOLCE (und Domänenontologien) definierten Mitteln repräsentiert

```
< Statement >
    ...
    < WeatherRegion ? / >
    < Road r / >
    ...
< / Statement >
```

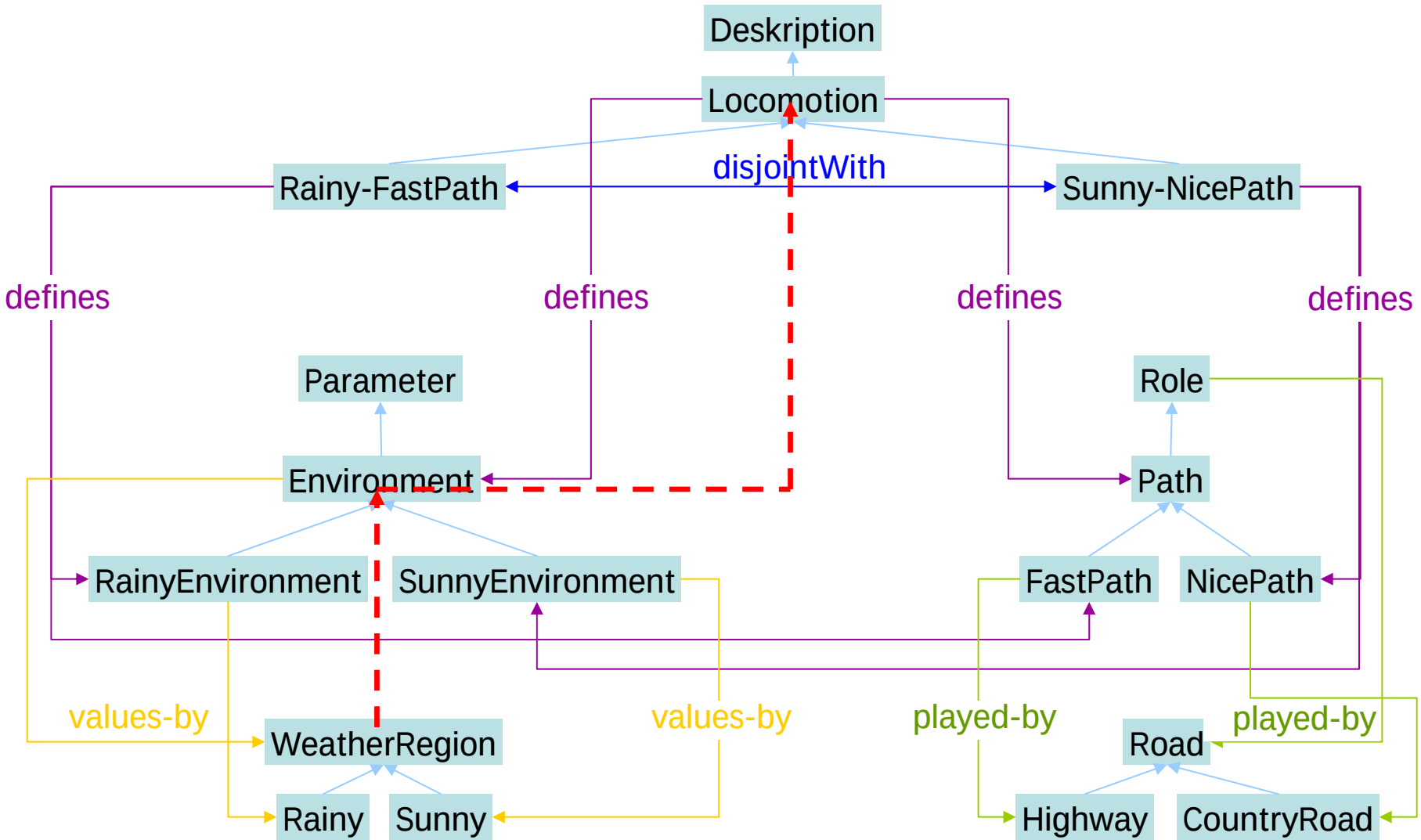
Ziel: Mit Kontextinformationen bereicherte Benutzeraussage/Anfrage

```
< Statement >
    ...
    < WeatherRegion >
        < Sunny s / >
    < / WeatherRegion >
    < Road >
        < CountryRoad c / >
    < / Road >
    ...
< / Statement >
```

Lösung

1. Finde Deskriptionen, denen gegebene Situation genügt

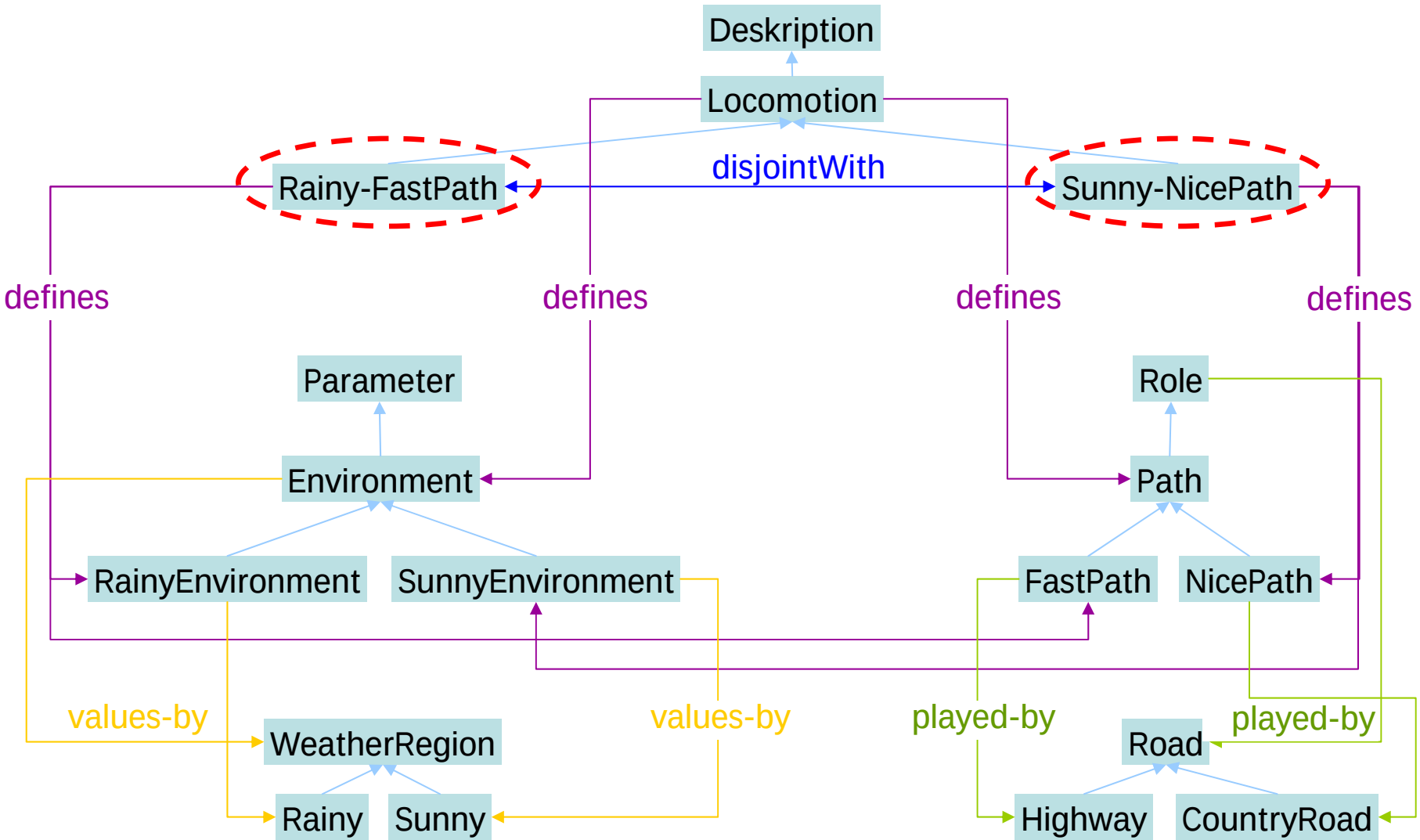
   DnS definiert satisfy-Relation, aber implementiert sie nicht ☹



Lösung

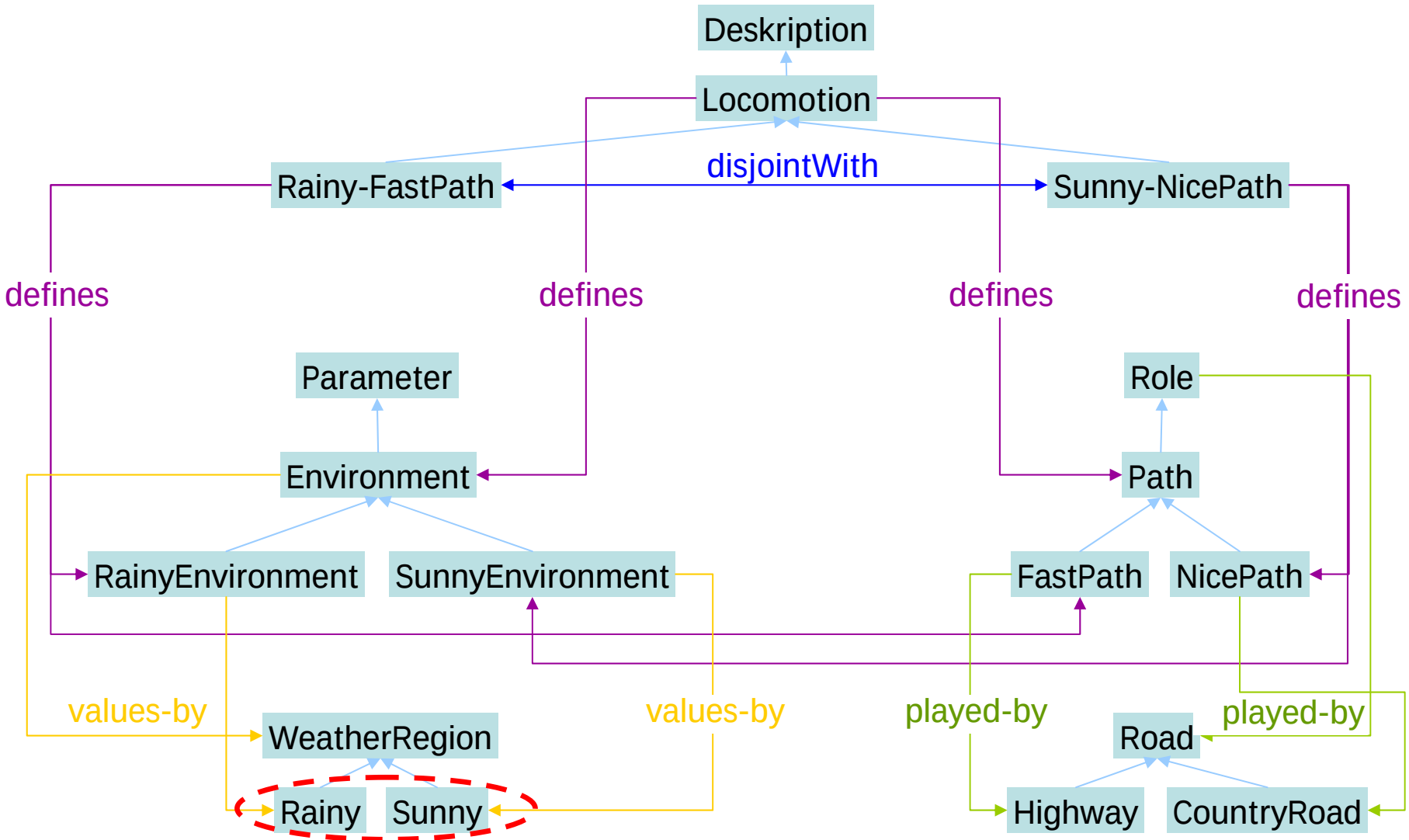
2. Finde spezifischere Deskriptionen, welche die obigen präzisieren

Wann ist eine Deskription spezifischer als ein andere?



Lösung

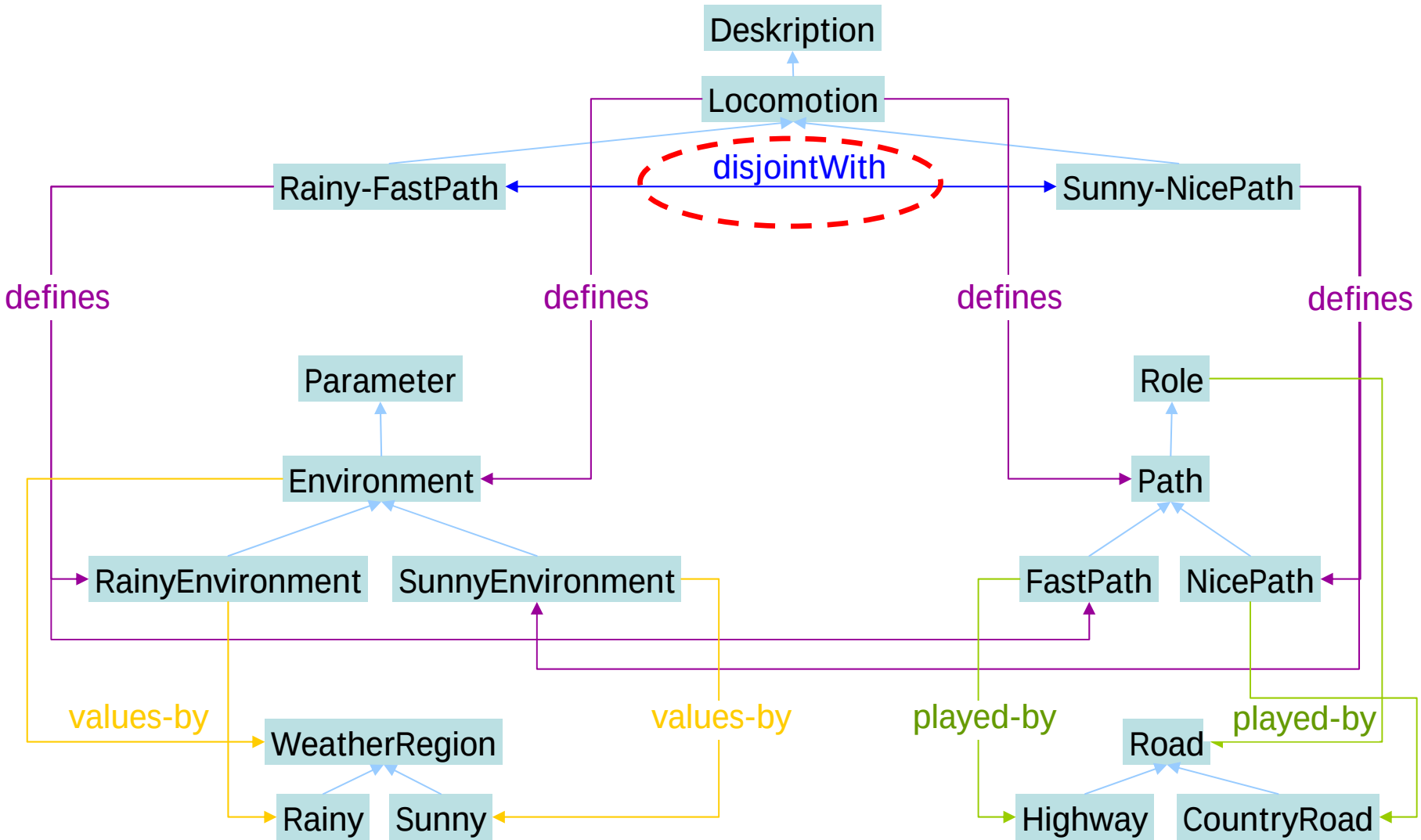
3. Bestimme notwendige Kontextinformationen



Lösung

4. Behandele evtl. Inkonsistenzen (zw. spezifischeren Deskriptionen)

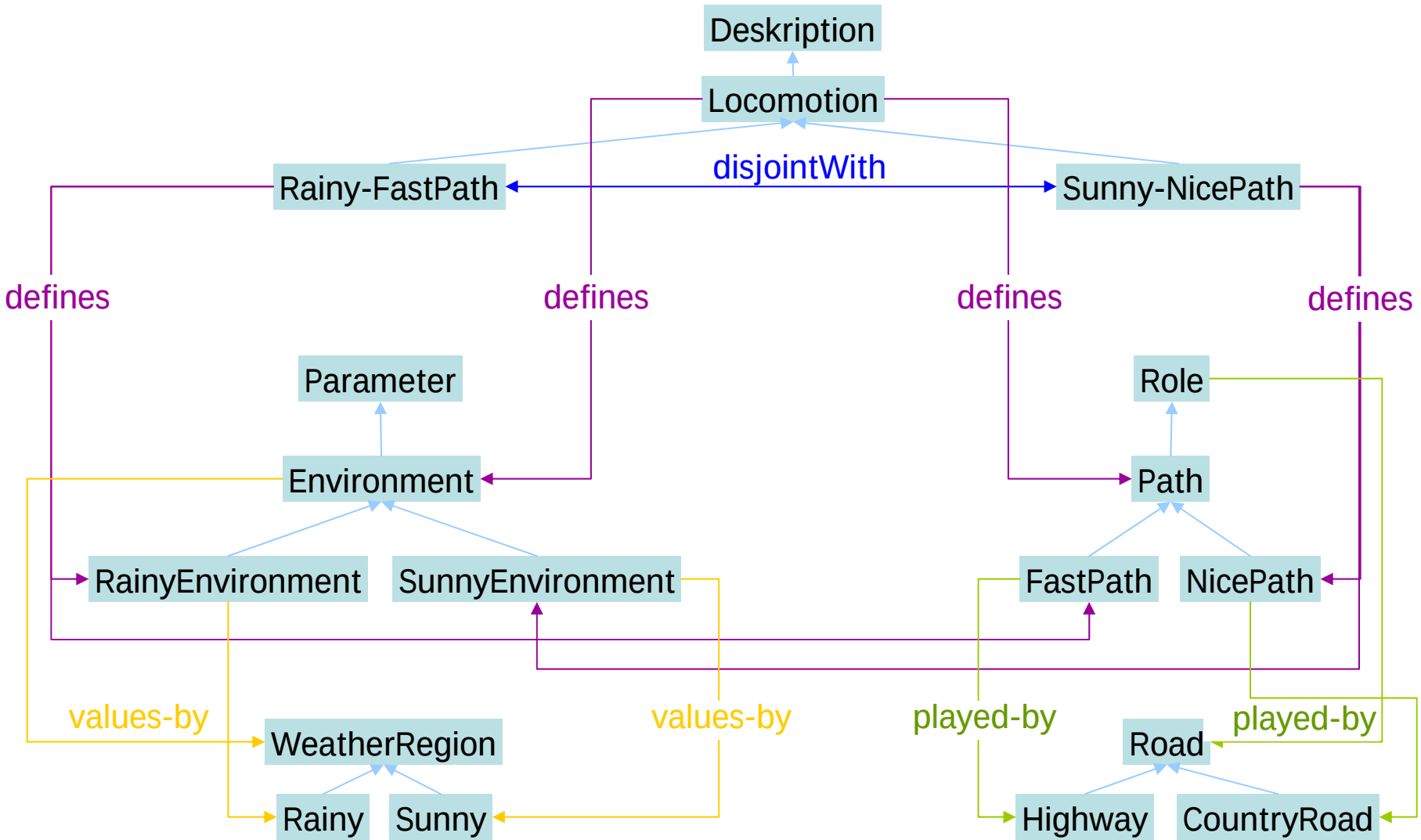
Wann ergibt sich Inkonsistenz?



Lösung

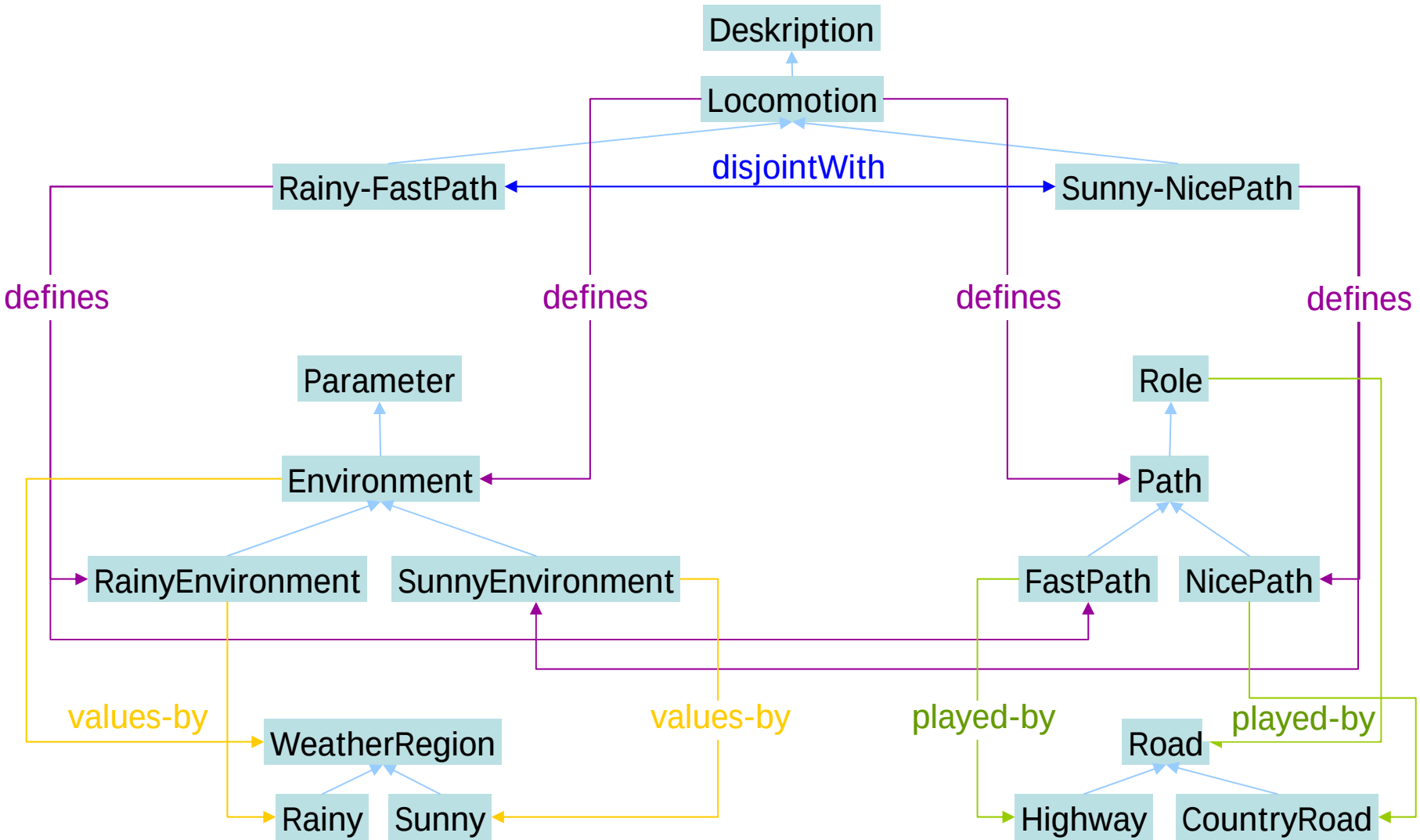
5. Sortiere ggf. spezifischere Deskriptionen nach Relevanz

   Was ist hier Relevanz?



Lösung

6. Ergänze die Eingabe



Herangehensweise

Vom Ziel heraus:

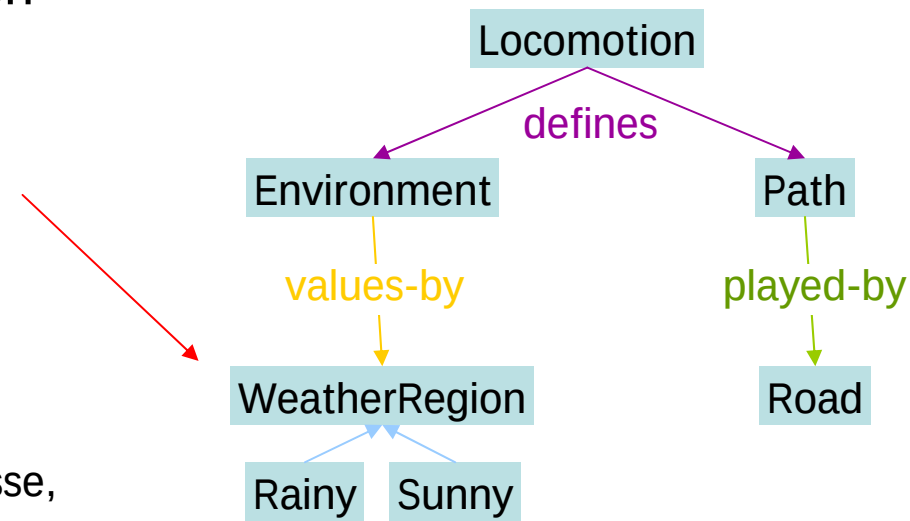
man will eine Aussage präzisieren

→Wann ist sie präzisierbar?

- Wenn in ihr etwas vorkommt, was mit einer Kontextinformationsquelle verknüpft ist

→Was kann sie präzisieren?

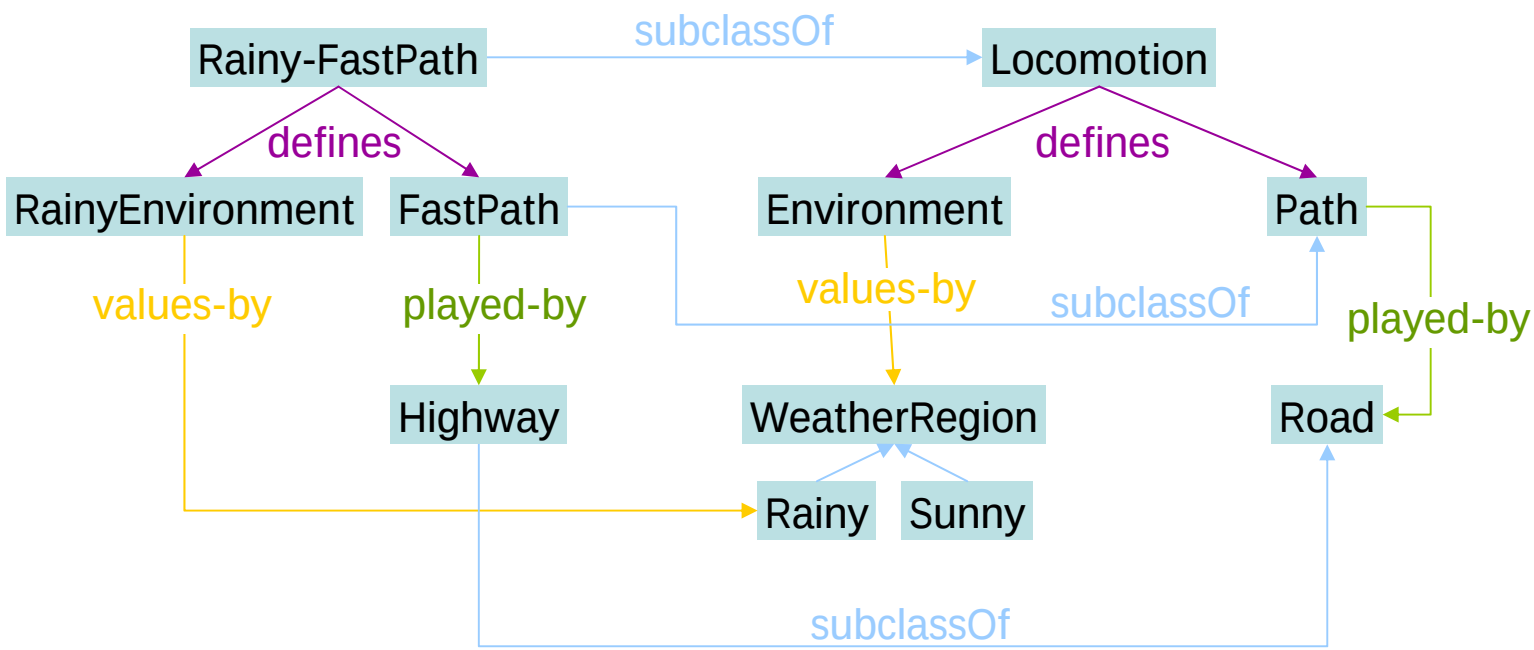
- Eine ontologische Unterklasse, die eine (konkrete) Kontextinformation aus dieser Quelle repräsentiert



Herangehensweise

→ Was kann sie präzisieren?

- Eine ontologische Unterklasse, die eine (konkrete) Kontextinformation aus dieser Quelle repräsentiert und zu einer Unterdeskription „gehört“
- Andere Elemente, die dieser Unterdeskription instanziieren, wenn sie Unterklassen der Elemente zu der Oberdeskription sind



Anfragen: Beispiele

„Ich will von Karlsruhe nach Berlin“

„Wann war Brasilien Weltmeister“

„Was ist morgen in Stuttgart los“

„Wer hat das 1 zu 0 geschossen“

„Wie ist die Verkehrslage zwischen Karlsruhe und Berlin“

„Wo gibt es hier Restaurants“

Schwerpunkt / Komplikationen

- Was ist das erwünschte Ergebnis?
- Wie erreicht man ihn:
  - Was hat man schon?
    - Was und wie ist bereits implementiert/modelliert?
      - Welche Repräsentationsformalismen? Was ermöglichen sie?
  - Welche Vorgaben sind zu beachten?
  - Was und wie soll noch modelliert werden?
  - Welche Tools gibt es (Erstellen, Editieren von Ontologien; sie in Java manipulieren; Inferenz)?
  - Welche Tools sind hilfreich:
    - Was ermöglichen sie?
    - Wie aufwändig ist es...
      - Sich mit diesen Tools vertraut zu machen (Dokumentation)?
      - Tools zu benutzen?
    - Wie effizient sind sie?
  - Was Tools verwenden und wann selbst implementieren?
  - Wie löst man die Aufgabe algorithmisch?