Methods of Artificial Intelligence: Lecture

7. Session: Knowledge Representation II

Kai-Uwe Kühnberger, Nohayr Muhammad

Winter Term 2022/2023

December 16th, 2022

Remarks

- Next week: no on-site lecture due to energy saving issues. Nevertheless, we will have an online lecture next Friday.
- Next week: no on-site seminar due to energy saving issues. Nevertheless, we will have an online seminar next Thursday.
- VIPS for Knowledge Representation I are online in the meantime.
  - As submission deadline I specified December 28th, 2022. There should be enough time to do the VIPS

Overview

- Representing Space
- Description Logics: Syntax and Semantics
- Description Logics: TBox and ABox
- Description Logics: Inferences

Representing Space

Regions instead of Metric Spaces:

The Region Connection Calculus RCC

The Idea of the RCC-8

- The RCC-8 is based on the concept of a region
  - Topological accounts as well as geometric accounts are usually based on the concept of point-sets
  - This is different in the RCC-8
    - Points are not crucial for spatial reasoning, but regions
- Axioms for the basic relation C (connection)
  - $\forall x$: C(x,x)
  - $\forall x,y$: C(x,y) $\rightarrow$ C(y,x)
    - Arguments of relation *C* are regions
- Using the connection predicate C to define an *is_part* relation P and an overlap relation O
  - P(x,y) := $\forall z$: (C(z,x) $\rightarrow$ C(z,y))
  - O(x,y) := $\exists z$: (P(z,x) $\wedge$ P(z,y))
- The *is_part* and overlap relations can be used to define other relations of regions

The Relations of the RCC-8

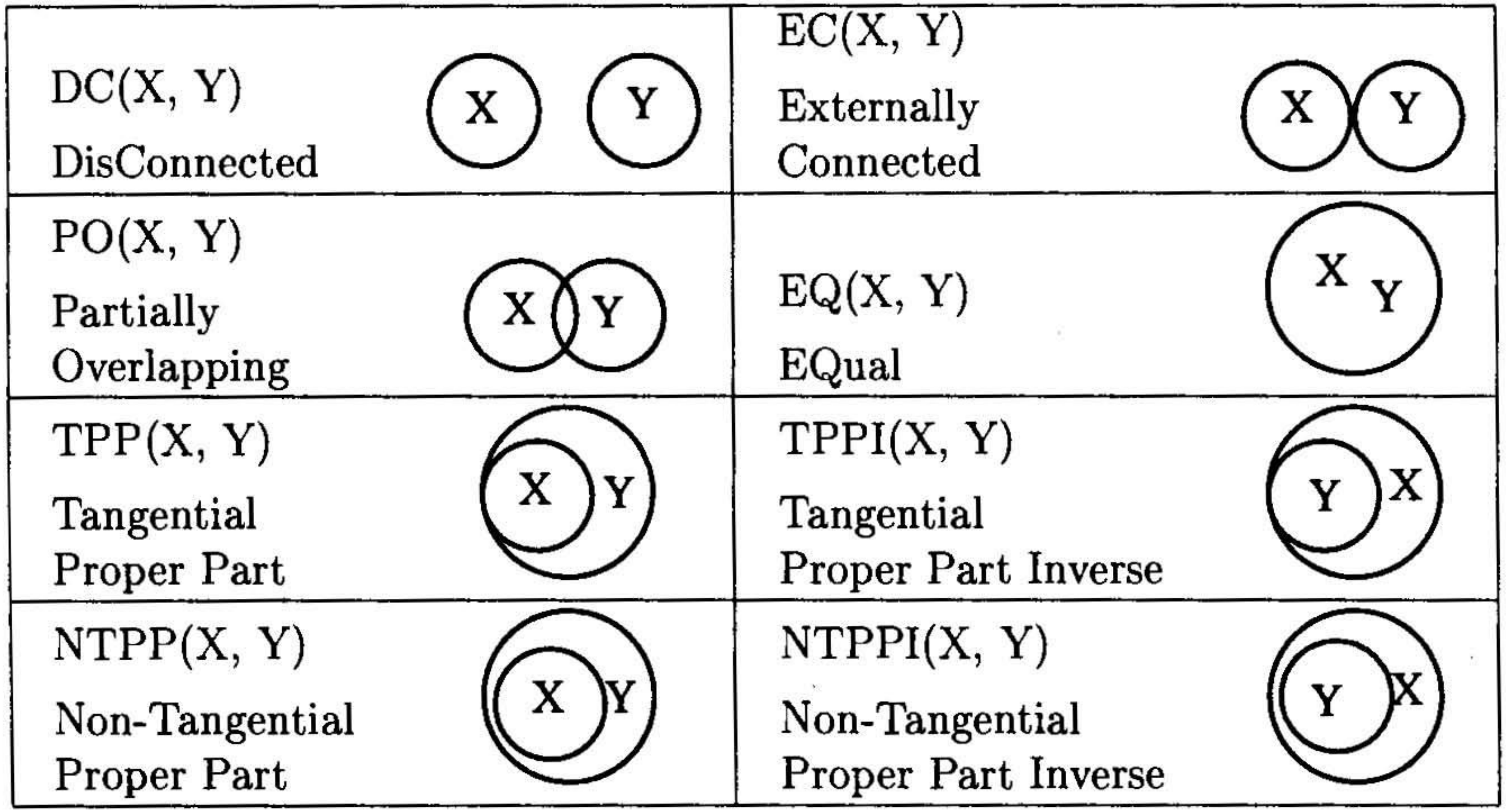
- What could these formulas mean?
  - $DC(x,y) := \neg C(x,y)$
  - $EC(x,y) := C(x,y) \wedge \neg O(x,y)$
  - $PO(x,y) := O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$
  - $EQ(x,y) := P(x,y) \wedge P(y,x)$
  - $[PP(x,y) := P(x,y) \wedge \neg P(y,x)]$
  - $TPP(x,y) := PP(x,y) \wedge \exists z(EC(z,x) \wedge EC(z,y))$
  - $TPPI(x,y) := PP(y,x) \wedge \exists z(EC(z,y) \wedge EC(z,x))$
  - $NTPP(x,y) := PP(x,y) \wedge \neg\exists z(EC(z,x) \wedge EC(z,y))$
  - $NTPPI(x,y) := PP(y,x) \wedge \neg\exists z(EC(z,y) \wedge EC(z,x))$

The Relations of the RCC-8

- Here are definitions of the relations two regions can have

  - $DC(x,y) := \neg C(x,y)$ — Disconnected
  - $EC(x,y) := C(x,y) \wedge \neg O(x,y)$ — Externally connected
  - $PO(x,y) := O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$ — Partially overlapping
  - $EQ(x,y) := P(x,y) \wedge P(y,x)$ — Equal
  - $TPP(x,y) := PP(x,y) \wedge \exists z(EC(z,x) \wedge EC(z,y))$ — Tangential proper part
  - $TPPI(x,y) := PP(y,x) \wedge \exists z(EC(z,y) \wedge EC(z,x))$ — Tangential proper part inverse
  - $NTPP(x,y) := PP(x,y) \wedge \neg\exists z(EC(z,x) \wedge EC(z,y))$ — Non-tangential proper part
  - $NTPPI(x,y) := PP(y,x) \wedge \neg\exists z(EC(z,y) \wedge EC(z,x))$ — Non-tangential proper part inverse

The Relations of the RCC-8: Canonical Model

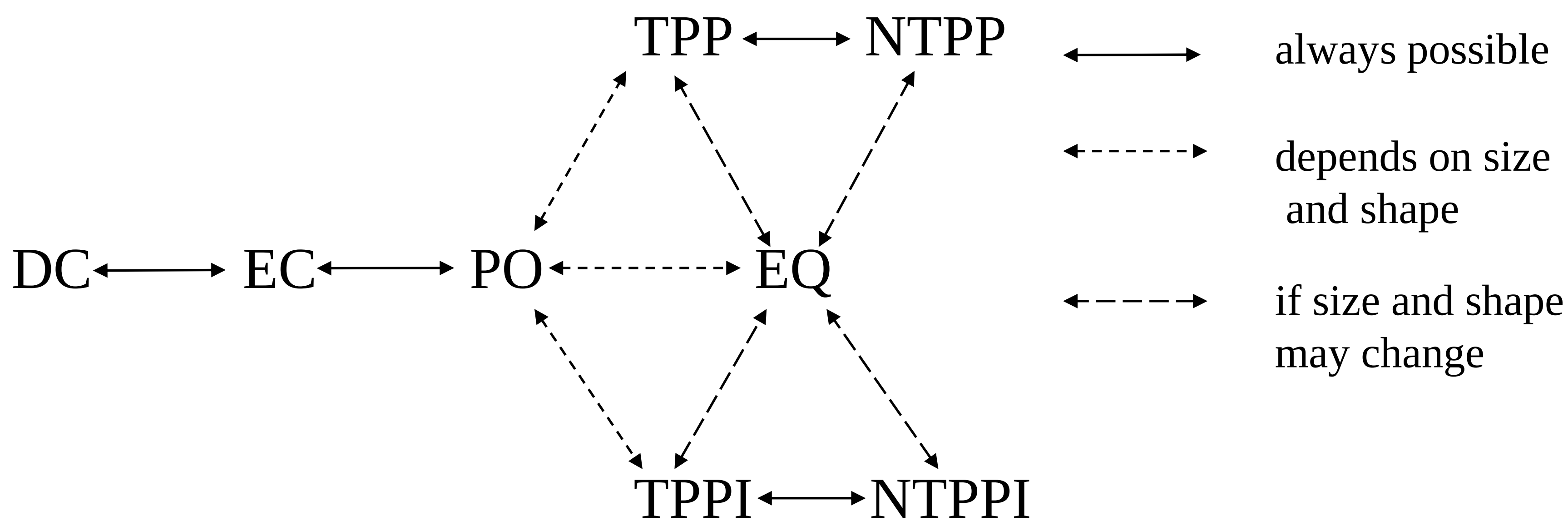


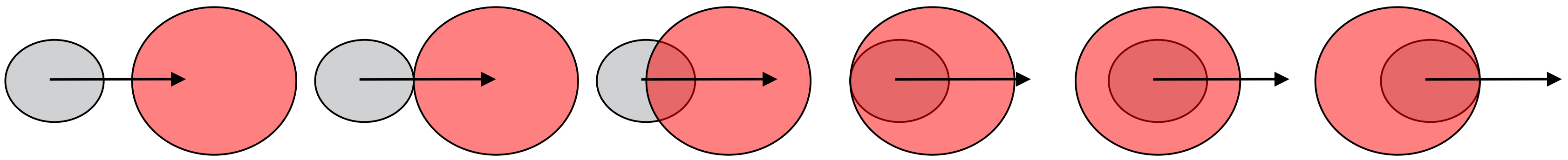
DC(X, Y) DisConnected	EC(X, Y) Externally Connected
PO(X, Y) Partially Overlapping	EQ(X, Y) EQual
TPP(X, Y) Tangential Proper Part	TPPI(X, Y) Tangential Proper Part Inverse
NTPP(X, Y) Non-Tangential Proper Part	NTPPI(X, Y) Non-Tangential Proper Part Inverse

Tabelle 10.2: Die Relationen des Region Connection Calculus

Tabble 10.2: The relations of the region connection calculus

- Motion corresponds to transitions:





Properties of RCC-8

- The RCC-8 is a generalization of the 13 interval relations of Allen's tense logic
  - Difference comes from the earlier / later relation (direction) in modeling time whereas in spatial reasoning there is nothing like an intrinsic direction of the dimensions

- In the space of closed discs:
  - The relations are exhaustive: There is no other basic relation possible between two regions
  - The relations are well-defined: Two regions are related in at most one way.

Variants of the RCC-8: RCC-7 and RCC-5

- RCC-5:
  - Collapsing:

DC	and EC	to DR	(discrete)
TPP	and NTPP	to PP	(proper part),
TPPI	and NTPPI	to PPI	(proper part inverse)

Variant of the RCC-8: RCC-5

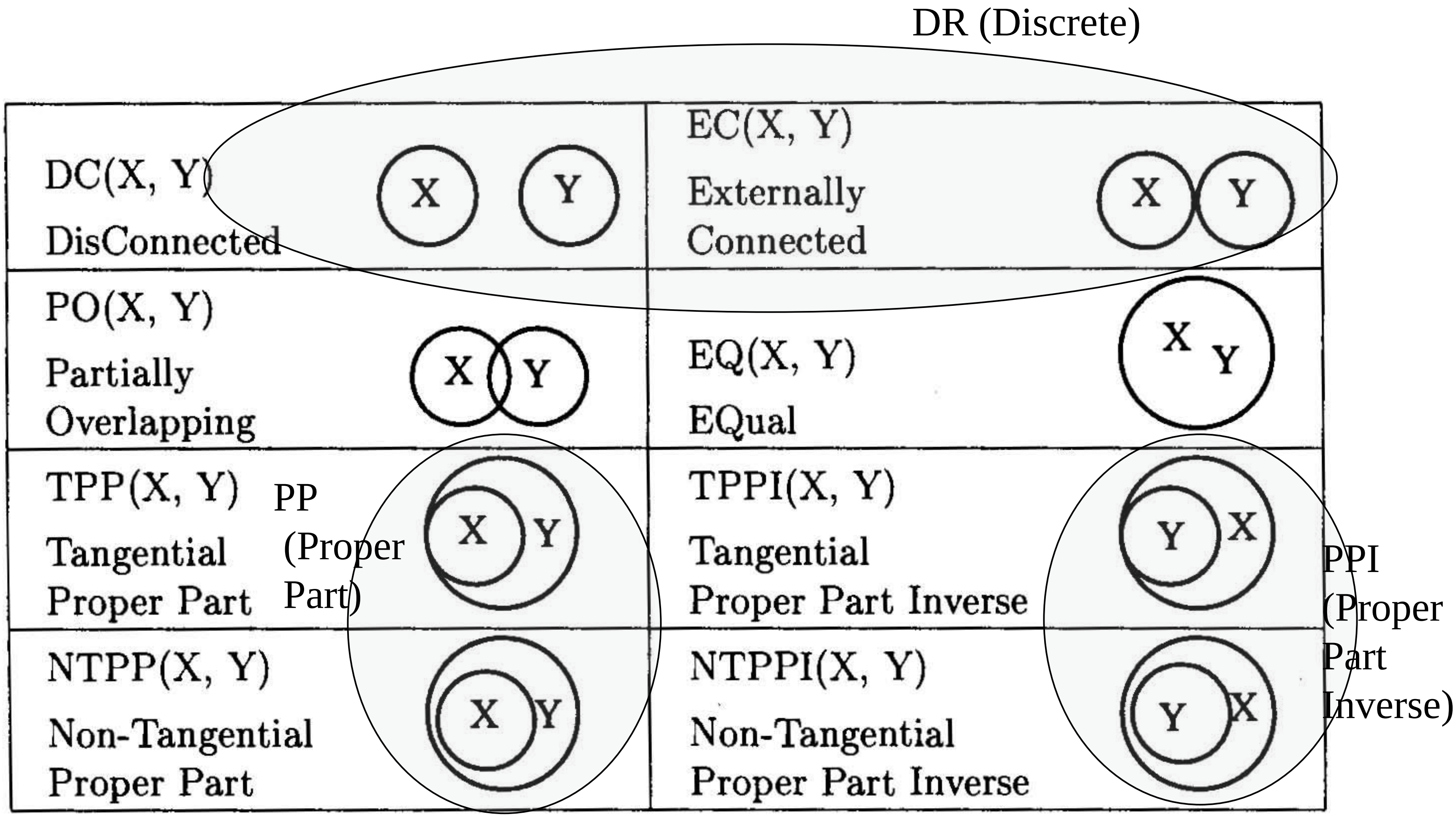


Tabelle 10.2: Die Relationen des Region Connection Calculus

Variant of the RCC-8: RCC-7

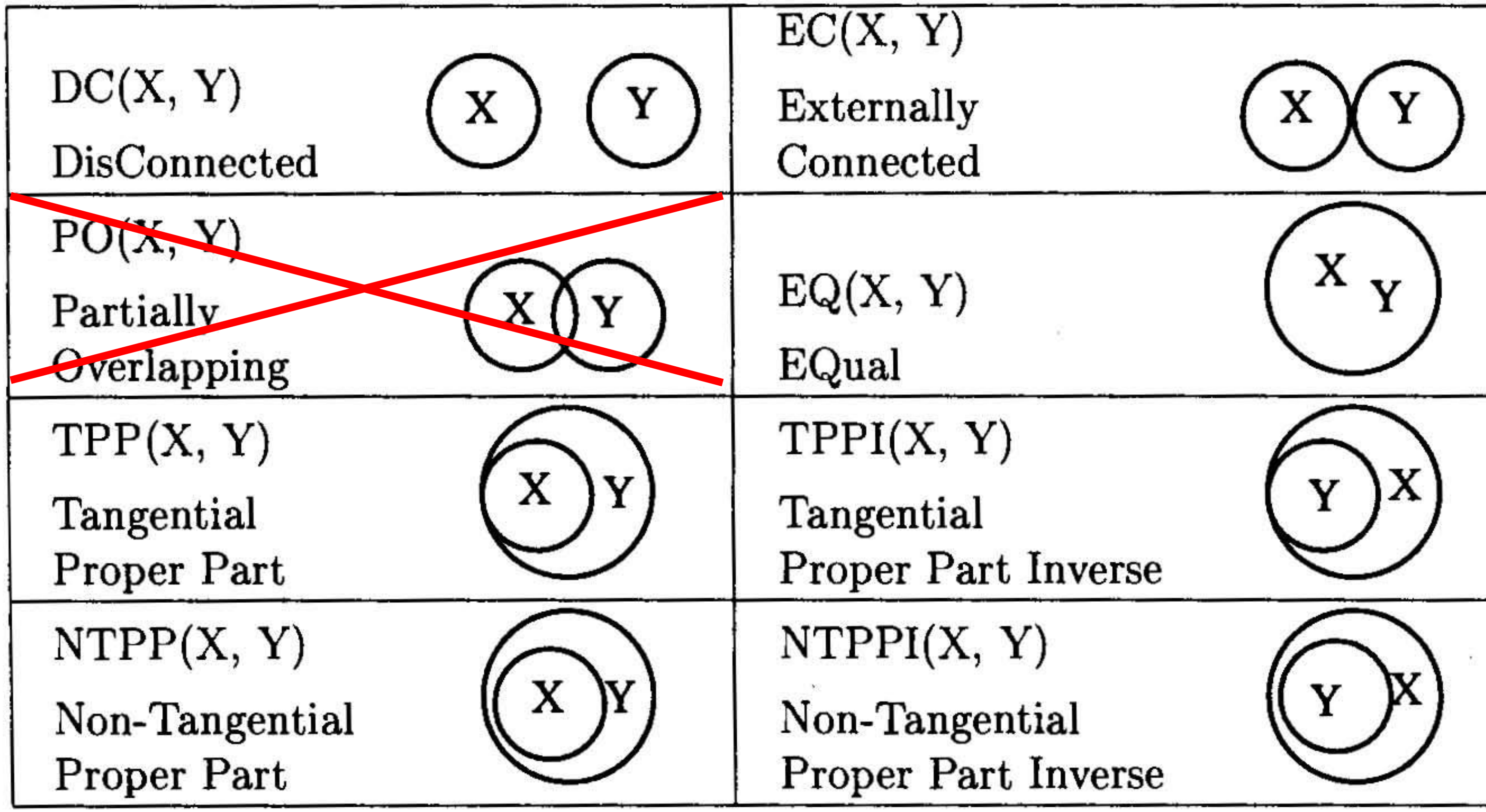
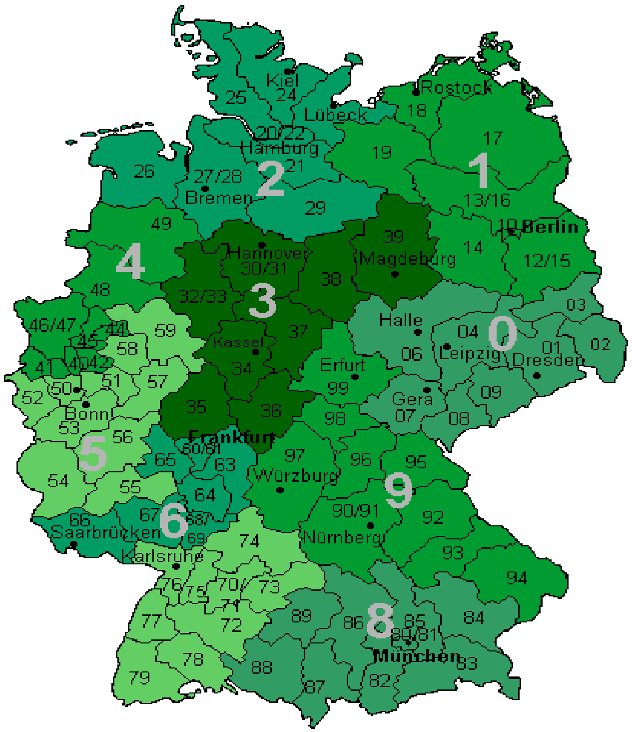
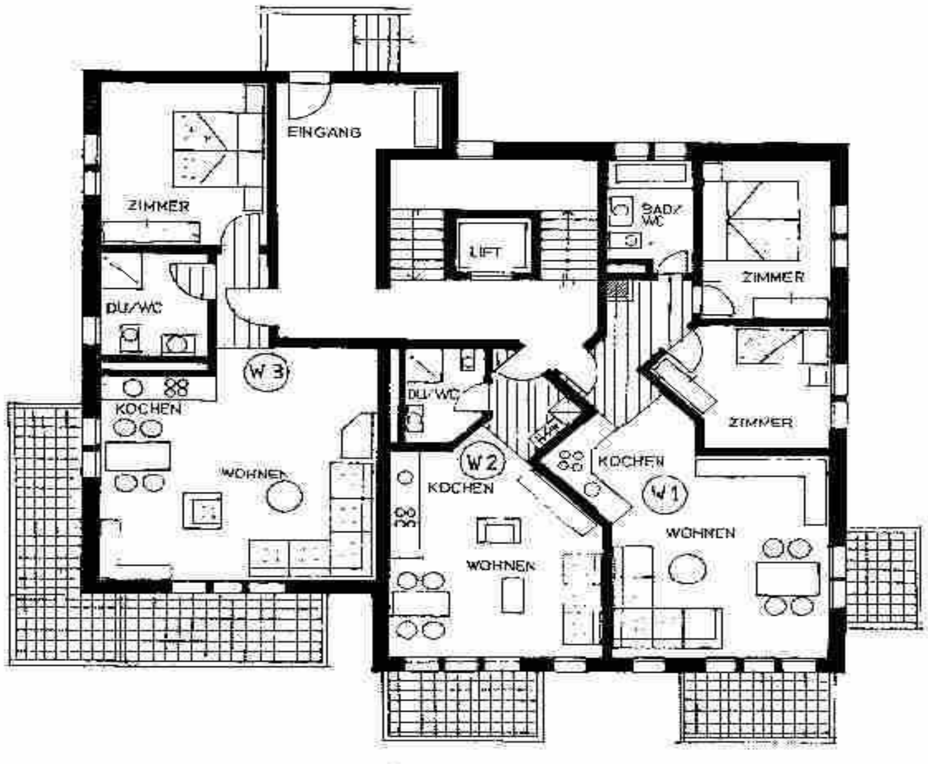


Tabelle 10.2: Die Relationen des Region Connection Calculus

Variant of the RCC-8: RCC-7

- Application: RCC-7 is suitable for applications in domains where spatial regions cannot partially overlap





Remarks

- Often we need directions for spatial reasoning
- (“Do we need to turn left before or after passing city X?”)
- Directions cannot be represented in RCC calculi.
- Prepositions like “under”, “before”, “left of” etc. cannot be represented in RCC-like systems
- What can be modeled by RCC calculi: *in*, *out* and *adjacent*

Questions

Description Logics: Syntax and Semantics

Representing Terminological Knowledge

Description Logics

- Description Logics (DLs) are a family of KR languages
- Classical FOL is designed to assert things about objects
  - DLs are designed to make it easier to describe definitions and properties of categories
- Certain knowledge representation formalisms, like semantic networks, miss a proper formal semantics
  - DLs are equipped with a logic-based semantics

Example

- When is FOL not the most natural choice?
  - E.g.,

$$\forall x\,(\mathtt{Teacher}(x) \Leftrightarrow \mathtt{Person}(x) \wedge \exists y\,(\mathtt{Teaches}(x, y) \wedge \mathtt{Course}(y)))$$

- In DL, it can be more easily represented as:

$$\mathtt{Person} \sqcap \exists \mathtt{teaches}.\mathtt{Course}$$

- DL is a user-friendly language for knowledge representation
  - Good for representing and reasoning on ontologies

KL-ONE (The Roots of DL)

- Brachman & Levesque (1985) came up with the idea of discriminating a T-Box and an A-Box for representing ontological knowledge
  - The T-box (terminology box):
    - Logical representation of conceptual knowledge (categories)
  - The A-box (assertion box):
    - Logical representation of facts about individuals (objects)
- They implemented their concept resulting in the language KL-ONE
- DLs are a modern variant of KL-ONE

A basic description logic: $\mathcal{ALC}$

- $\mathcal{ALC}$: Attributive Concept Language with Complements
- Concept descriptions, e.g.,

$$\texttt{Person} \sqcap \exists \texttt{teaches.Course}$$

- Main ingredients
  - Concept names, e.g., `Person` representing sets of elements, also called their extensions
  - Role names interpreted by binary relations between objects, e.g., `employedBy`
  - Concept constructors to build complex concepts, e.g., $\neg$, $\sqcap$, $\sqcup$, $\exists$, $\forall$

The Basic Language

- **Concept Names**

$$\mathrm{N}_{\mathbf{C}} = \{A_1, A_2, \ldots\}$$

  - Examples: `Parent`, `Sister`, `Student`

- **Role Names**

$$\mathrm{N}_{\mathbf{R}} = \{r_1, r_2, \ldots\}$$

  - Examples: `employedBy`, `motherOf`

- **Individual Names**

$$\mathrm{N}_{\mathbf{I}} = \{a_1, a_2, \ldots\}$$

  - Examples: `Mary`, `Alice`, `John`

The Basic Language

- Boolean constructors
  - **Concept negation** $\neg$ (class complement)
  - **Concept conjunction** $\sqcap$ (class intersection)
  - **Concept disjunction** $\sqcup$ (class union)
- Role restrictions
  - **Existential restriction** $\exists$ (at least one related individual)
  - **Value restriction** $\forall$ (all related individuals)

- Many more constructors exist in variants of DL logics

The Basic Language

- We introduce the syntax of the basic description language $\mathcal{ALC}$
- $C, D \quad \rightarrow \quad A \quad |$ (atomic concept)

A	$\vert$	(atomic concept)
$\top$	$\vert$	(universal concept)
$\bot$	$\vert$	(bottom concept)
$\neg C$	$\vert$	(negation)
$C \sqcap D$	$\vert$	(intersection)
$C \sqcup D$	$\vert$	(union)
$\forall R.C$	$\vert$	(value restriction)
$\exists R.C$	$\vert$	(existential quantification)

- $\mathcal{ALC}$ (attributive language with complement) is considered as a relatively minimal language of practical interest.

The Basic Language

An interpretation is a pair $<\Delta^I, \cdot^I>$ where Δ^I is a set of individuals and $\cdot^I$ is a function mapping concepts to subsets of Δ^I and roles to subsets of Δ^I x Δ^I.

Meaning of concept descriptions is inductively defined

$$\top^I = \Delta^I$$

$$\bot^I = \emptyset$$

$$(\neg C)^I = \Delta^I \setminus C^I$$

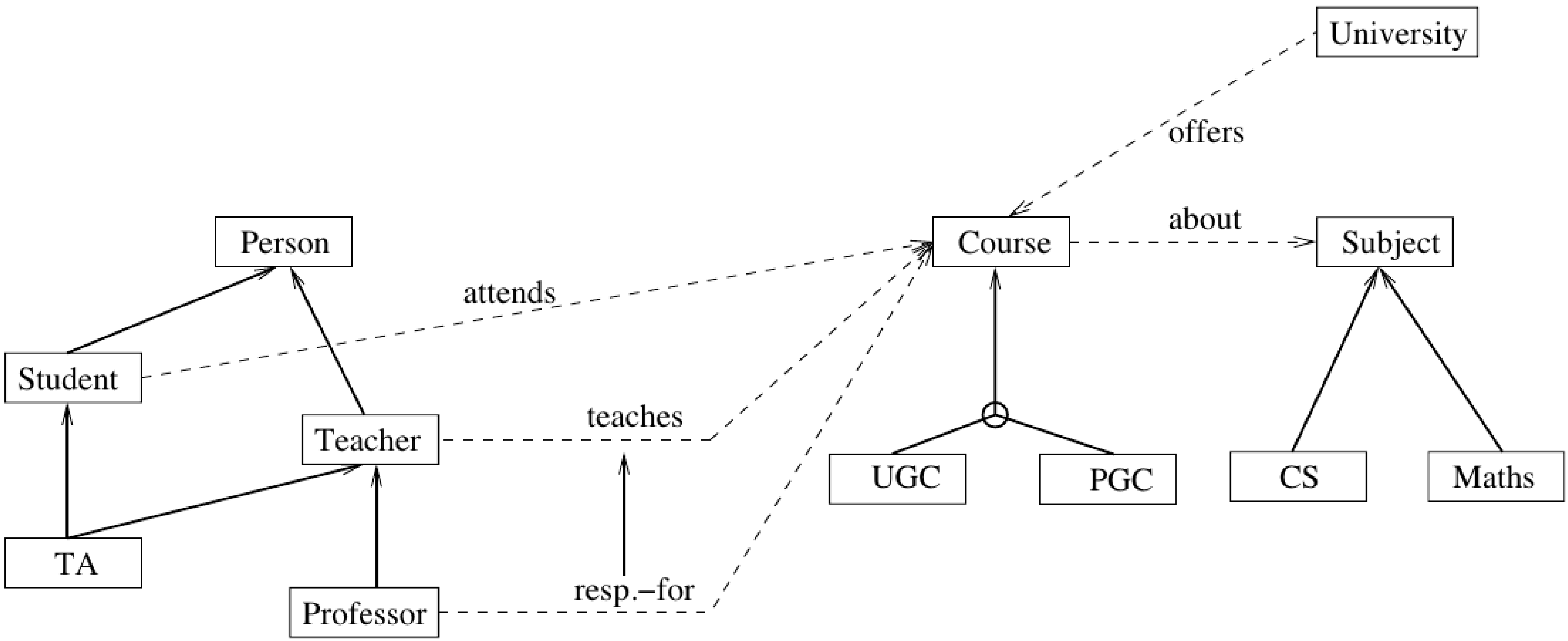
$$(C \sqcap D)^I = C^I \cap D^I$$

$$(C \sqcup D)^I = C^I \cup D^I$$

$$(\forall R.C)^I = \{a \in \Delta^I \mid \forall b{:}(a,b) \in R^I \rightarrow b \in C^I\}$$

$$(\exists R.C)^I = \{a \in \Delta^I \mid \exists b{:}(a,b) \in R^I \wedge b \in C^I\}$$

Example: Concepts and Roles



Rectangles represent concepts
Solid lines represent the subsumption relation (x is subconcept of y)
Dotted lines represent binary relations between individuals
In the diagram there is even a hierarchy of relations between “teaches” and “resp.-for”

Description Logics: TBox and ABox

Knowledge about Concepts and
Knowledge about Individuals

DL Knowledge Base

- When using a DL-based system in an application, we typically build concept descriptions for the domain of interest
- Then we create a *knowledge base* by:
  1. Defining the meaning of a concept in terms of a concept description:

     `UG-Student` corresponds to `Student` $\sqcap$ $\forall$`attends.UGC`
  2. Expressing background knowledge

     `UGC` is included in $\neg$`PGC`
  3. Asserting that individuals are instances of concept descriptions:

     `Mary` is an instance of `Teacher` $\sqcap$ $\exists$`teaches.PGC`
  4. Relating individual names by roles

     `Mary` teaches *`CS600`*

DL Knowledge Base

- Traditionally, we distinguish two parts
  1. Terminological Part (*TBox*)
     - Contain statements of the form 1 and 2 of previous slide
     - Corresponds to the *schema* of a database
  2. Assertion Part (*ABox*)
     - Contain statements of the form 3 and 4 of previous slide
     - Corresponds to the *data* of a database

General Concept Inclusion (GCI)

- Expressions of the form $C \sqsubseteq D$ are called **general concept inclusions**.
- Intuitive meaning:
  - C subsumes D
  - C is more specific than D (i.e. D is more general than C)
- Example: `Employee` $\sqsubseteq \exists$`WorksFor.T`
- Satisfaction relation: $I \vDash C \sqsubseteq D$ iff $C^I \subseteq D^I$

Concept Equivalence

- $C \sqsubseteq D$ and $D \sqsubseteq C$ abbreviated by $C \equiv D$ is called **concept equivalence**.
- Satisfaction relation:

$$I \models C \equiv D \qquad \text{iff} \qquad C^I = D^I$$

- Example:

$$\top \quad \equiv \quad (\neg \mathtt{Student} \sqcup \mathtt{Student})$$

TBox

- A finite set of general concept inclusions is called a TBox
- An interpretation that satisfies all general concept inclusions in a TBox is a *model* for it
- In practice we can use a TBox to restrict to those interpretations that fit our intuitions about the domain
  - E.g., if we believe that a course cannot be a person, we should include the following definition in our TBox:

$$\texttt{Course} \sqsubseteq \neg\texttt{Person}$$

ABox

- **Concept assertion**: stating that an individual *a* is an instance of a concept *C*:

$$a : C$$

- Satisfaction relation: $I \vDash a : C$ iff $a^I \in C^I$
- **Role assertion**: stating that two individuals *a* and *b* stand in the *r*-relation: $(a,b) : r$
- Satisfaction relation: $I \vDash (a,b) : r$ iff $(a^I,b^I) \in r^I$
- Example: `Alice: Student ⊓ ¬∃Pays.Tax`

ABox

- A finite set of concept and role assertions is called an ABox
- An interpretation that satisfies all assertions in an ABox is a *model* for it

- Remark: The set of individual names is disjoint from the sets of concept names and role names

DL Knowledge Base (formally)

- A knowledge base K = (T, A) consists of
  - a Tbox T and
  - an Abox A.
- An interpretation that is both a model of A and of T is called a model of K.

- A knowledge base like this can be seen as an ontology.

Description Logics: Inferences

The Tableaux Algorithm

DL Knowledge Base (formally)

- Two types of algorithms
  - Structural subsumption algorithms (for weak DLs)
  - Tableau-based algorithms (general technique)
- Remarks:
  - Relation of DLs to 2-variable logic
  - Most DLs can be reduced to 2-variable logic
  - Problematic cases are role composition and number restrictions: these operations cannot be expressed by 2-variable logic in general (why?)

DL Knowledge Base (formally)

- Structural subsumption algorithms try to test subsumption of concept descriptions
  - This works only if no disjunction is available
  - Compare Baader & Nutt: “Basic Description Logic”

- Tableau-based algorithms reduce subsumption to the unsatisfiability of concept descriptions:

  $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable
  (by checking if a (finite) model exists)

- We explain Tableau-based algorithms using an example
  - Assume we want to know whether $(\exists R.A) \sqcap (\exists R.B)$ is subsumed by $\exists R.(A \sqcap B)$
  - We must check whether
    $$C = (\exists R.A) \sqcap (\exists R.B) \sqcap \neg(\exists R.(A \sqcap B))$$
    is unsatisfiable

DL Knowledge Base (formally)

- Tableau-based algorithms: an example
  - Check for $(\exists R.A) \sqcap (\exists R.B) \sqsubseteq \exists R.(A \sqcap B)$
  - We must check whether

$$C = (\exists R.A) \sqcap (\exists R.B) \sqcap \neg(\exists R.(A \sqcap B))$$

  is unsatisfiable

- Push all negations as far as possible into the description (negation normal form)
  - $C' = (\exists R.A) \sqcap (\exists R.B) \sqcap (\forall R.(\neg A \sqcup \neg B))$
  - Assume that there exists a $b \in (C')^{\mathcal{I}}$
  - This corresponds to finding a model for an A-box: $\{b \in C'\}$

Inference Algorithms

Try to construct a finite interpretation $\mathcal{I}$ such that $(C')^{\mathcal{I}} \neq \emptyset$

1.	$b \in (C')^{\mathcal{I}}$	
2.	$b \in ((\exists R.A) \sqcap (\exists R.B) \sqcap (\forall R.(\neg A \sqcup \neg B)))^{\mathcal{I}}$	(1., def)
3.	$b \in (\exists R.A)^{\mathcal{I}}$	(2., $\sqcap$)
4.	$b \in (\exists R.B)^{\mathcal{I}}$	(2., $\sqcap$)
5.	$b \in (\forall R.(\neg A \sqcup \neg B))^{\mathcal{I}}$	(2., $\sqcap$)
6.	$\langle b,c\rangle \in R^{\mathcal{I}}$	(3., skolemization)
7.	$c \in A^{\mathcal{I}}$	
8.	$\langle b,d\rangle \in R^{\mathcal{I}}$	(4., skolemization)
9.	$d \in B^{\mathcal{I}}$	
10.	$c \in (\neg A \sqcup \neg B)^{\mathcal{I}}$	(5., 6., $\forall$)
11.	$d \in (\neg A \sqcup \neg B)^{\mathcal{I}}$	(5., 8., $\forall$)
12.	$c \in (\neg B)^{\mathcal{I}}$	(10., $c \in (\neg A)^{\mathcal{I}}$ clashes with 7.: $c \in A^{\mathcal{I}}$)
13.	$d \in (\neg A)^{\mathcal{I}}$	(11., $d \in (\neg B)^{\mathcal{I}}$ clashes with 9.: $d \in B^{\mathcal{I}}$)

$\Delta^{\mathcal{I}} = \{b,c,d\}$, $R^{\mathcal{I}} = \{\langle b,c\rangle, \langle b,d\rangle\}$, $A^{\mathcal{I}} = \{c\}$, $B^{\mathcal{I}} = \{d\}$, $I(b')=b$ is a finite model for for $\{b' \in C'\}$

Inference Algorithms

- We found a model for $\{b \in C'\}$
- $C' = (\exists R.A) \sqcap (\exists R.B) \sqcap (\forall R.(\neg A \sqcup \neg B))$ is satisfiable
- $C = (\exists R.A) \sqcap (\exists R.B) \sqcap \neg(\exists R.(A \sqcap B))$ is not unsatisfiable
- $(\exists R.A) \sqcap (\exists R.B) \sqsubseteq \exists R.(A \sqcap B)$ does not hold!

QUESTIONS?