

Knowledge Representation and Processing

Florian Rabe (for a course given with Michael Kohlhase)

Computer Science, University Erlangen-Nürnberg, Germany

Summer 2020

Administrative Information

Format

Zoom

- ▶ lectures and exercises via zoom
- ▶ participants muted by default for simplicity
- ▶ interaction strongly encouraged We don't want to lecture —
we want to have a conversation during which you learn
- ▶ let's try out zoom
 - ▶ use reactions to say yes no, ask for break etc.
 - ▶ feel free to annotate my slides
 - ▶ talk in the chat

Recordings

- ▶ maybe prerecorded video lectures or recorded zoom meeting
- ▶ to be decided along the way

Background

Instructors

- ▶ Prof. Dr. Michael Kohlhase
Professor of Knowledge Representation and Processing
- ▶ PD Dr. Florian Rabe
same research group

Course

- ▶ This course is given for the first time
- ▶ Always a little bit of an experiment cutting edge vs. unpolished
- ▶ Could become signature course of our research group same name!

Prerequisites

Required

- ▶ basic knowledge about formal languages, context-free grammars
but we'll do a quick revision here

Helpful

- ▶ Algorithms and Data Structures mostly as a contrast to this lecture
- ▶ Basic logic we'll revise it slightly differently here
- ▶ all other courses as examples of how knowledge pervades all of CS

General

- ▶ Curiosity this course is a bit unusual
- ▶ Interest in big picture
this course touches on lots of things from all over CS

Examination and Grading

Suggestion

- ▶ grade determined by single exam
- ▶ written or oral depends on number of students
- ▶ some acknowledgment for practical exercises

to be finalized next week

Exam-relevant

- ▶ anything mentioned in notes
- ▶ anything discussed in lectures

neither is a superset of the other!

Materials and Exam-Relevance

Textbook

- ▶ does not exist
- ▶ normal for research-near specialization courses

Notes

- ▶ textbook-style but not as comprehensive
- ▶ developed along the way

Slides

- ▶ not comprehensive
- ▶ used as visual aid, conversation starters

Communication

Open for questions

- ▶ open door policy in our offices if the lockdown ever ends
- ▶ always room for questions during lectures
- ▶ for personal questions, contact me during/after lecture or by email
- ▶ forum at <https://fsi.cs.fau.de/forum/154-Wissensrepraesentation-und-Verarbeitung>

Materials

- ▶ official notes and slides as pdf:
<https://kwarc.info/teaching/WuV/>
will be updated from time to time
- ▶ watch me prepare the materials: <https://github.com/florian-rabe/Teaching/tree/master/WuV>
pull requests and issues welcome

Exercises

Learning Goals

- ▶ Get acquainted with state of the art of practice
- ▶ Try out real tools

Homeworks

- ▶ one major project as running example
- ▶ homeworks building on each other

build one large knowledge-based system
details on later slides

Overview and Essential Concepts

Representation and Processing

Common pairs of concepts:

| Representation | Processing |
|-----------------|------------|
| Static | Dynamic |
| Situation | Change |
| Be | Become |
| Data Structures | Algorithms |
| Set | Function |
| State | Transition |
| Space | Time |

Data and Knowledge

2×2 key concepts

| Syntax | Data |
|-----------|-----------|
| Semantics | Knowledge |

- ▶ Data: any object that can be stored in a computer
Example: $((49.5739143, 11.0264941), "2020 - 04 - 21 T16 : 15 : 00 CEST")$
- ▶ Syntax: a system of rules that describes which data is **well-formed**
Example: "a pair of (a pair of two IEEE double precision floating point numbers) and a string encoding of a time stamp"
- ▶ Semantics: system of rules that determines the meaning of well-formed data
- ▶ Knowledge: combination of some data with its syntax and semantics

Knowledge is Elusive

Representation of key concepts

- ▶ Data: using primitive objects
implemented as bits, bytes, strings, records, arrays, ...
- ▶ Syntax: (context-free) grammars, (context-sensitive) type systems
implemeted as inductive data structures
- ▶ Semantics: functions for evaluation, interpretation, of well-formed data
implemented as recursive algorithms on the syntax
- ▶ Knowledge: elusive
emerges from applying and interacting with the semantics

Semantics as Translation

- ▶ Knowledge can be captured by a higher layer of syntax
- ▶ Then semantics is translation into syntax

| Data syntax | Semantics function | Knowledge syntax |
|--------------------|---------------------------|---------------------|
| SPARQL query | evaluation | result set |
| SQL query | evaluation | result table |
| program | compiler | binary code |
| program expression | interpreter | result value |
| logical formula | interpretation in a model | mathematical object |
| HTML document | rendering | graphics context |

Heterogeneity of Data and Knowledge

- ▶ Capturing knowledge is difficult
- ▶ Many different approaches to semantics
 - ▶ fundamental formal and methodological differences
 - ▶ often captured in different fields, conferences, courses, languages, tools
- ▶ Data formats equally heterogeneous
 - ▶ ontologies
 - ▶ programs
 - ▶ logical proofs
 - ▶ databases
 - ▶ documents

Challenges of Heterogeneity

Challenges

- ▶ collaboration across communities
- ▶ translation across languages
- ▶ conversion between data formats
- ▶ interoperability across tools

Sources of problems

- ▶ interoperability across formats/tools major source of
 - ▶ complexity
 - ▶ bugs
- ▶ friction in project team due to differing preferences, expertise
- ▶ difficult choice between languages/tools with competing advantages
 - ▶ reverting choices difficult, costly
 - ▶ maintaining legacy choices increases complexity

Aspects of Knowledge

- ▶ Tetrapod model of knowledge active research by our group
- ▶ classifies approaches to knowledge into five aspects

| Aspect | KRLs (examples) |
|----------------|--|
| ontologization | ontology languages (OWL), description logics (ALC) |
| concretization | relational databases (SQL, JSON) |
| computation | programming languages (C) |
| deduction | logics (HOL) |
| narration | document languages (HTML, LaTeX) |

Relations between the Aspects

Ontology is distinguished: capture the knowledge that the other four aspects share



Complementary Advantages of the Aspects

| Aspect | objects | characteristic | | |
|-----------------|---------------------------|----------------------------|--------------------------------------|--|
| | | advantage | joint advantage of the other aspects | application |
| ded. comp. | formal proofs programs | correctness efficiency | ease of use well-definedness | verification execution |
| concr. narr. | concrete objects texts | tangibility flexibility | abstraction formal semantics | storage/retrieval human understanding |

| Aspect pair | characteristic advantage |
|----------------------------|---|
| ded./comp. narr./concr. | rich meta-theory simple languages |
| ded./narr. comp./concr. | theorems and proofs normalization |
| ded./concr. comp./narr. | decidable well-definedness Turing completeness |

Structure of the Course

Aspect-independent parts

- ▶ general methods that are shared among the aspects
- ▶ to be discussed as they come up

Aspects-specific parts

- ▶ one part (about 2 weeks) for each aspect
- ▶ high-level overview of state of the art
- ▶ focus on comparison/evaluation of the aspect-specific results

Structure of the Exercises

One major project

- ▶ representative for a project that a CS graduate might be put in charge of
- ▶ challenging heterogeneous data and knowledge
- ▶ requires integrating/combining different languages, tools

unique opportunity in this course because knowledge is everywhere

Concrete project

- ▶ develop a univis-style system for a university
- ▶ lots of heterogeneous knowledge
 - ▶ course and program descriptions
 - ▶ legal texts
 - ▶ websites
 - ▶ grade tables
 - ▶ transcript generation code
- ▶ build a completely functional system applying the lessons of the course

Ontological Knowledge

Components of an Ontology

8 main declarations

- ▶ **individual** — concrete objects that exist in the real world, e.g., "Florian Rabe" or "WuV"
- ▶ **concept** — abstract groups of individuals, e.g., "instructor" or "course"
- ▶ **relation** — binary relations between two individuals, e.g., "teaches"
- ▶ **properties** — binary relations between an individuals and a concrete value (a number, a date, etc.), e.g., "has-credits"
- ▶ **concept assertions** — the statement that a particular individual is an instance of a particular concept
- ▶ **relation assertions** — the statement that a particular relation holds about two individuals
- ▶ **property assertions** — the statement that a particular individual has a particular value for a particular property
- ▶ **axioms** — statements about relations between concepts, e.g., "instructor" \sqsubseteq "person"

Divisions of an Ontology

Abstract vs. concrete

- ▶ TBox: concepts, relations, properties, axioms
everything that does not use individuals
- ▶ ABox: individuals and assertions

Named vs. unnamed

- ▶ Signature: individuals, concepts, relations, properties
together called entities or resources
- ▶ Theory: assertions, axioms

Comparison of Terminology

| Here | OWL | Description logics | ER model | UML | semantics via logics |
|------------|-----------------|--------------------|-------------|--------------------|----------------------|
| individual | instance | individual | entity | object, instance | constant |
| concept | class | concept | entity-type | class | unary predicate |
| relation | object property | role | role | association | binary predicate |
| property | data property | (not common) | attribute | field of base type | binary predicate |

| | | |
|-------------------------|----------------|-------------|
| domain | individual | concept |
| type theory, logic | constant, term | type |
| set theory | element | set |
| database | row | table |
| philosophy ¹ | object | property |
| grammar | proper noun | common noun |

¹as in <https://plato.stanford.edu/entries/object/>

Ontologies as Sets of Triples

| Assertion | Triple | | |
|--------------------|----------------|---------------|--------------|
| | Subject | Predicate | Object |
| concept assertion | "Florian Rabe" | is-a | "instructor" |
| relation assertion | "Florian Rabe" | "teaches" | "WuV" |
| property assertion | "WuV" | "has credits" | 7.5 |

Efficient representation of ontologies using RDF and RDFS
standardized special entities.

Special Entities

RDF and RDFS define special entities for use in ontologies:

- ▶ "rdfs:Resource": concept of which all individuals are an instance and thus of which every concept is a subconcept
- ▶ "rdf:type": relates an entity to its type:
 - ▶ an individual to its concept (corresponding to is-a above)
 - ▶ other entities to their special type (see below)
- ▶ "rdfs:Class": special class for the type of classes
- ▶ "rdf:Property": special class for the type of properties
- ▶ "rdfs:subClassOf": a special relation that relates a subconcept to a superconcept
- ▶ "rdfs:domain": a special relation that relates a relation to the concepts of its subjects
- ▶ "rdfs:range": a special relation that relates a relation/property to the concept/type of its objects

Goal/effect: capture as many parts as possible as RDF triples.

Declarations as Triples using Special Entities

| Assertion | Triple | | |
|-----------------------------|------------|-------------------|-----------------|
| | Subject | Predicate | Object |
| individual | individual | "rdf:type" | "rdfs:Resource" |
| concept | concept | "rdf:type" | "rdf:Class" |
| relation | relation | "rdf:type" | "rdf:Property" |
| property | property | "rdf:type" | "rdf:Property" |
| concept assertion | individual | "rdf:type" | concept |
| relation assertion | individual | relation | individual |
| property assertion | individual | property | value |
| for special forms of axioms | | | |
| $c \sqsubseteq d$ | c | "rdfs:subClassOf" | d |
| $\text{dom } r \equiv c$ | r | "rdfs:domain" | c |
| $\text{rng } r \equiv c$ | r | "rdfs:range" | c |

An Example Ontology Language

see syntax and semantics in the lecture notes

Representing Formal Languages

Example: Syntax of Arithmetic Language

Syntax: represented as formal grammar

Numbers

| | | | |
|-----------|--|---------|----------|
| $N ::= 0$ | | 1 | literals |
| | | $N + N$ | sum |
| | | $N * N$ | product |

Formulas

| | |
|--------------------|------------------|
| $F ::= N \doteq N$ | equality |
| $N \leq N$ | ordering by size |

Implementation as inductive data type

Example: Semantics of Arithmetic Language

Semantics: represented as translation into known language

Problem: Need to choose a known language first

Here: unary numbers represented as strings

Built-in data (strings and booleans):

| | |
|------------------------------|------------|
| $S ::= \varepsilon$ | empty |
| $\quad \text{ (Unicode) }$ | characters |
| $B ::= \text{true}$ | truth |
| $\quad \text{ false}$ | falsity |

Built-in operations to work on the data:

- ▶ concatenation of strings $S ::= \text{conc}(S, S)$
- ▶ replacing all occurrences of c in S_1 with S_2
 $S ::= \text{replace}(S_1, c, S_2)$
- ▶ equality test: $B ::= S_1 == S_2$
- ▶ prefix test: $B ::= \text{startsWith}(S_1, S_2)$

Example: Semantics of Arithmetic Language

Represented as function from syntax to semantics

- ▶ mutually recursive, inductive functions for each non-terminal symbol
- ▶ compositional: recursive call on immediate subterms of argument

For numbers n : semantics $\llbracket n \rrbracket$ is a string

- ▶ $\llbracket 0 \rrbracket = \varepsilon$
- ▶ $\llbracket 1 \rrbracket = \text{"|"}$
- ▶ $\llbracket m + n \rrbracket = \text{conc}(\llbracket m \rrbracket, \llbracket n \rrbracket)$
- ▶ $\llbracket m * n \rrbracket = \text{replace}(\llbracket m \rrbracket, \text{"|"}, \llbracket n \rrbracket)$

For formulas f : semantics $\llbracket f \rrbracket$ is a boolean

- ▶ $\llbracket m \dot{=} n \rrbracket = \llbracket m \rrbracket == \llbracket n \rrbracket$
- ▶ $\llbracket m \leq n \rrbracket = \text{startsWith}(\llbracket n \rrbracket, \llbracket m \rrbracket)$