Engineering of Logics for the Content-based Representation of Information

Franz Baader
Theoretical Computer Science
TU Dresden
Germany

- Content-based representation of information
- The role of logics and why they must be engineered
- Description Logics as a successful instance of this approach
- Two applications of DL: Semantic Web and Databases

Dresden

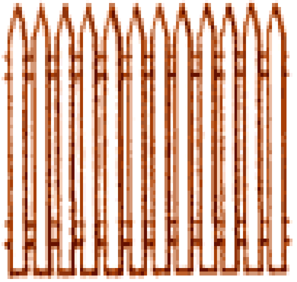
Content-based representation

- representation of the "meaning" of the information
- shared understanding of this meaning among all agents (human users, search engines, ...) using the information

- understanding of meaning should result in
  - ability to draw conclusions from the represented information
  - ability to determine semantic equivalence of syntactically different representations

Example

searching for information on the WWW



- looking for garden centers offering palisades for my new garden
  - search engine should know that paling is a similar notion
  - and that fence subsumes both

- use of an ontology:
  - defines the important notions (classes, relations, objects) of the domain
  - states constraints on the way these notions can be interpreted
  - information about synonyms, subsumption, etc. can automatically be deduced from the definitions and constraints

Dresden

Semantics of the representation formalism

- Need for a formal, well-defined semantics since otherwise there cannot be a shared understanding and reliable reasoning
  - not just "intuitive" or purely "procedural" semantics

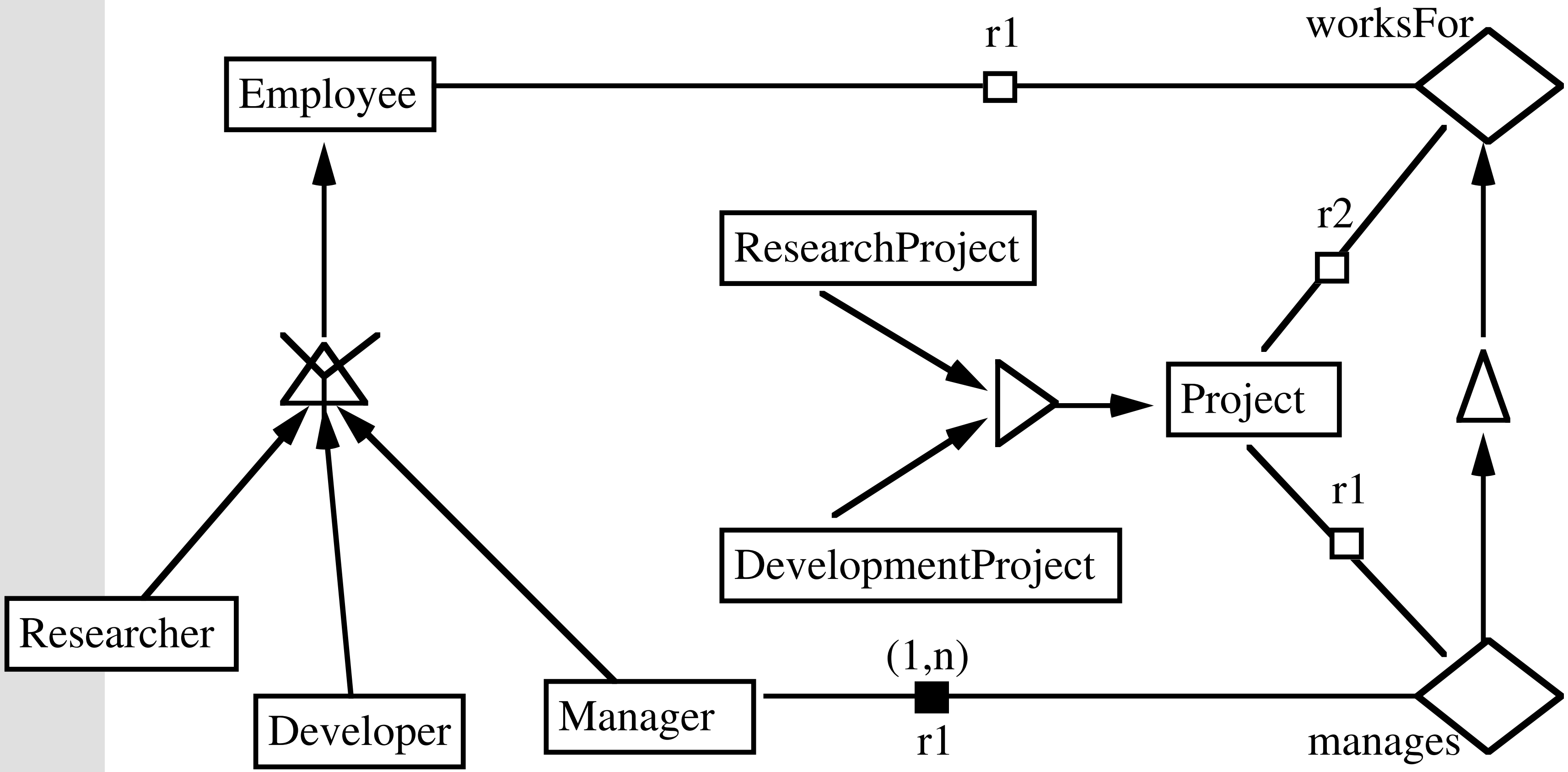
- comprehensible to human users
- usable by machines (e.g. in reasoning)

- logic as an appropriate tool
  - yields formal semantics
  - reasoning about the information as logical inference problem
  - standard approaches for logical reasoning can be used

Dresden

Example

graph-based formalisms such as semantic networks (AI), ER diagrams (DB), UML diagrams (software engineering)



Pictures say more than 1000 words,
but they may tell 100 different stories, depending on the viewer.

Why engineering of logics?

- Expressiveness vs. tractability issue:
  - application-relevant knowledge must be expressible
  - reasoning must still be "feasible"

  Requires logics that are tailored to the application problem

- Practical considerations, usability of logics:
  not just investigation of formal properties (axiomatization, interpolation, ...), but emphasis on algorithmic properties
  - (worst-case) complexity analysis
  - "practical" algorithms
  - optimization techniques
  - empirical evaluation

Own contributions

to this endeavour

- Designing expressive knowledge representation formalisms and practical reasoning tools, application in chemical process engineering, databases, and semantic Web

  - collaboration with E. Franconi, I. Horrocks (U. Manchester), M. Lenzerini (U. Rome), W. Marquardt (RWTH Aachen)

- Combination of logics and reasoners: equational theories (word problem and unification), modal and description logics

  - collaboration with K. Schulz (U. Munich), C. Tinelli (U. Iowa), F. Wolter (U. Leipzig)

Description Logics

class of knowledge representation formalisms

Descended from structured inheritance networks [Brachman 78] via the system KL-ONE [Brachman&Schmolze 85]. Emphasis on well-defined basic inference procedures: subsumption and instance problem.

Phase 1:

- implementation of incomplete systems (Back, Classic, Loom, ...)
- based on structural subsumption algorithms

Phase 2:

- development of tableau-based algorithms and complexity results
- first implemented tableau-based systems (Kris, Crack)
- first formal investigation of optimization methods

Phase 3:

- tableau-based algorithms for very expressive DLs
- highly optimized tableau-based systems (FaCT, Racer)
- relationship to modal logic and decidable fragments of FOL

Description logic systems

structure

description language

- constructors for building complex concepts and roles out of atomic concepts and roles
- formal, logic-based semantics

TBox

defines terminology of the application domain

ABox

states facts about a specific "world"

knowledge base

reasoning component

- derive implicitly represented knowledge (e.g., subsumption)
- "practical" algorithms

Dresden

Description language

examples of typical constructors:

$C \sqcap D$, $\neg C$, $\forall r . C$, $\exists r . C$, $(\geq n\ r)$

A man	Human $\sqcap$ $\neg$ Female $\sqcap$
that is married to a doctor, and	$\exists$ married-to . Doctor $\sqcap$
has at least 5 children,	$(\geq 5$ child$)$ $\sqcap$
all of whom are professors.	$\forall$ child . Professor

TBox

definition of concepts

Happy-man = Human $\sqcap$ ...

statement of constraints

$\exists$ married-to . Doctor $\sqsubseteq$ Doctor

ABox

properties of individuals

Happy-Man(Franz)

child(Franz,Luisa)

child(Franz,Julian)

Dresden

Formal semantics

based on interpretations as in predicate logic

An interpretation I associates

- concepts C with sets C^I and
- roles r with binary relations r^I.

The semantics of the constructors is defined through identities:

- $(C \sqcap D)^I = C^I \cap D^I$
- $(\geq n\ r)^I = \{ d \mid \#\{e \mid (d,e) \in r^I\} \geq n \}$
- $(\forall r . C)^I = \{ d \mid \forall e: (d,e) \in r^I \Rightarrow e \in C^I \}$
- . . .

$I \models A = C$ iff $A^I = C^I$

$I \models C \sqsubseteq D$ iff $C^I \subseteq D^I$

$I \models C(a)$ iff $a^I \in C^I$

$I \models r(a,b)$ iff $(a^I,b^I) \in r^I$

model

Dresden

Reasoning

makes implicitly represented knowledge explicit,
is provided as service by the DL system, e.g.:

polynomial reductions

Subsumption: Is C a subconcept of D?

$C \sqsubseteq D$ iff $C^I \subseteq D^I$ for all interpretations I.

Satisfiability: Is the concept description C non-contradictory?

C is satisfiable iff there is an I such that $C^I \neq \emptyset$.

Consistency: Is the ABox $\mathcal{A}$ non-contradictory?

$\mathcal{A}$ is consistent iff it has a model.

Instantiation: Is e an instance of C w.r.t. the given ABox $\mathcal{A}$?

$\mathcal{A} \models C(e)$ iff $e^I \in C^I$ for all models I of $\mathcal{A}$.

in presence of negation

Dresden

Satisfiability algorithm

Idea

generate an interpretation I such that $C_0^I \neq \emptyset$

Data structure

for describing (partial) interpretations: ABoxes

(w.l.o.g. all concept descriptions in negation normal form)

Approach

ABox assertions are viewed as constraints;

propagate constraints.

- Starting with $\mathcal{A}_0 := \{C_0(x_0)\}$, the algorithm applies transformation rules until all constraints are satisfied or an obvious contradiction is detected.
- Every rule corresponds to one constructor.
- Disjunction requires non-deterministic rule: two alternatives.

Dresden

Exists-restriction rule

$\{ \ \ldots \ \exists r.C(a) \ \ldots \ \}$

↓

$\{ \ \ldots \ \exists r.C(a), \ C(b), \ r(a,b) \ \ldots \ \}$

↑ *new individual name*

Condition

there is no c with C(c) and r(a,c) present

Dresden

Disjunction rule

$\{ \ \ldots \ (C \sqcup D)(a) \ \ldots \ \}$

Condition

neither
C(a) nor D(a)
is present

$\{ \ldots (C \sqcup D)(a), C(a) \ldots \}$

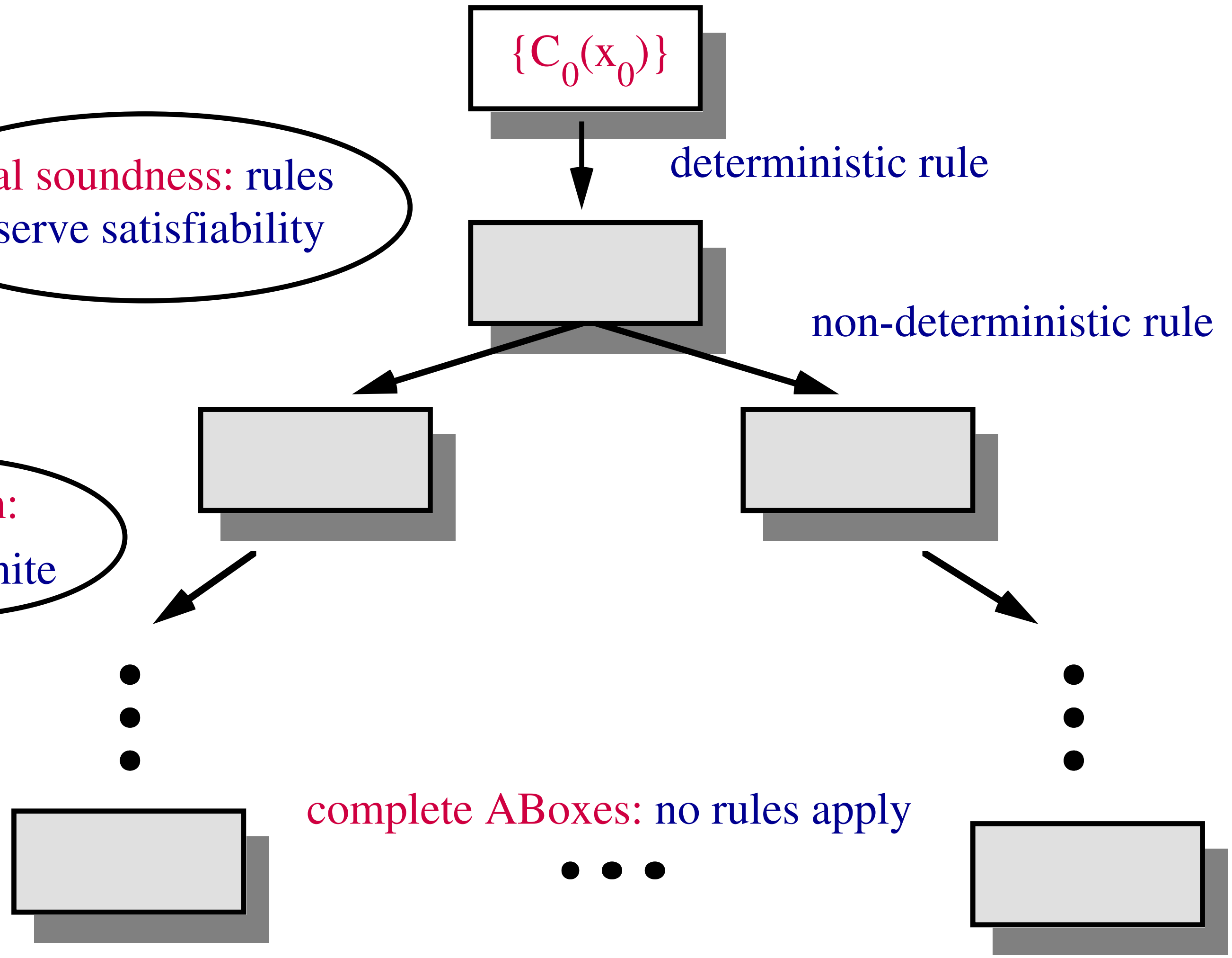
$\{ \ldots (C \sqcup D)(a), D(a) \ldots \}$

Dresden

search tree

local soundness: rules preserve satisfiability

termination: all paths finite



satisfiable iff one of the complete ABoxes is open, i.e., does not contain an obvious contradiction (clash) → $A(x), \neg A(x)$

Ontologies

for the Semantic Web

"An ontology is a specification of a conceptualization." (Tom Gruber, Stanford)

- An abstract, simplified view of the world, expressed in an appropriate formal language with well-defined semantics.
- Facilitates shared understanding: common ontologies for a set of agents allow them to communicate about a domain of discourse without necessarily operating on a globally shared theory.

DAML+OIL

joint proposal by EU/US initiatives for a W3C ontology standard

- RDF (schema) based syntax
- semantics defined by translation into an expressive DL
- reasoning employs highly optimized DL reasoner (FaCT)

SHIQ

DL used to define the semantics of DAML+OIL
depends on last 10 years of DL research

- very expressive DL:
  - Boolean operators ($\sqcap$, $\sqcup$, $\neg$)
  - value and existential restrictions ($\forall r.C$, $\exists r.C$)
  - qualified number restrictions
  - general inclusion axioms
  - transitive roles, inverse roles, and role hierarchies

} *ALC* [Schmidt-Schauß& Smolka 88/91]

- implemented systems: FaCT [Horrocks 98] and Racer [Haarslev,Moeller 01]
  - tableau-based subsumption algorithm
    building on experience of Kris [B.&Hollunder 91]
  - highly optimized implementation
    building on experience with optimizing Kris
    [B.,Franconi,Hollunder,Nebel,Profitlich 92]

Dresden

Qualified number restrictions

extend the simple number restrictions of early DL systems

- Can not only express "At least 3 children"

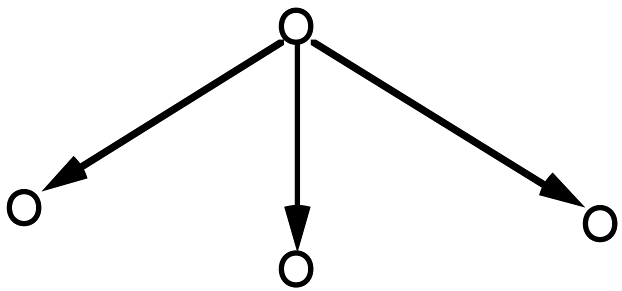
  $(\geq 3\ \text{child})$

- but also "At most 1 daughter and at most 1 son"

  $(\leq 1\ \text{child}.\text{Female}) \sqcap (\leq 1\ \text{child}.\neg\text{Female})$

- First algorithm that can handle qualified number restrictions proposed in [Hollunder&B. 91]:
  - Introduces a nondeterministic "choose-rule"
  - necessary to detect inconsistencies:

    $(\leq 1\ \text{child}.\text{Female}) \sqcap (\leq 1\ \text{child}.\neg\text{Female}) \sqcap (\geq 3\ \text{child})$



Dresden

General inclusion axioms

extend the simple concept definitions of early DL systems

- Can be used to formulate complex constraints, e.g.,
  - domain and range constraints on roles:

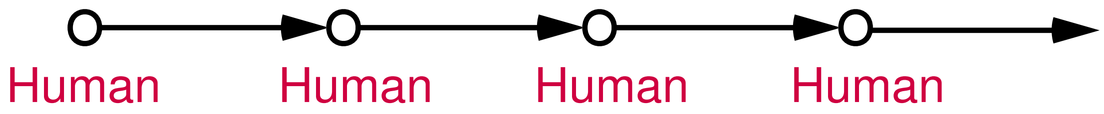
    $\exists\,\text{child}\,.\,\text{Human} \sqsubseteq \text{Human}$

    $\text{Human} \sqsubseteq \forall\,\text{child}\,.\,\text{Human}$

- Make reasoning considerably harder (for $\mathcal{ALC}$, complexity jumps from PSpace to ExpTime).

- First algorithm that can handle general inclusion axioms proposed in [B., Bürckert,Hollunder,Nutt,Siekmann 90]:
  - termination requires "blocking":

    $\text{Human} \sqsubseteq \exists\,\text{parent}\,.\,\text{Human}$



Complex roles

extend the simple atomic roles
of early DL systems

- Transitive roles can express partonomies, causality, ..., e.g.,
  $\exists$ part . (Reactor $\sqcap$ $\exists$ part . Heater) implies $\exists$ part . Heater
  - Transitive roles in DLs first treated in [Sattler 96]:
    $\mathcal{ALC}$ with transitive roles still in PSpace.

- Role hierarchies can (e.g.) express that son is a subrole of child
  - Transitive roles and role hierarchies can simulate general inclusion axioms [Horrocks,Sattler 98].

- Inverse roles: e.g., parent is the inverse of child
  - Because of the combination of general inclusion axioms, inverse roles, and number restrictions, $\mathcal{SHIQ}$ does not have the finite model property.
  - First algorithm for $\mathcal{SHIQ}$ presented in [Horrocks,Sattler,Tobies 99/00]
  - requires a very sophisticated blocking condition.

Dresden

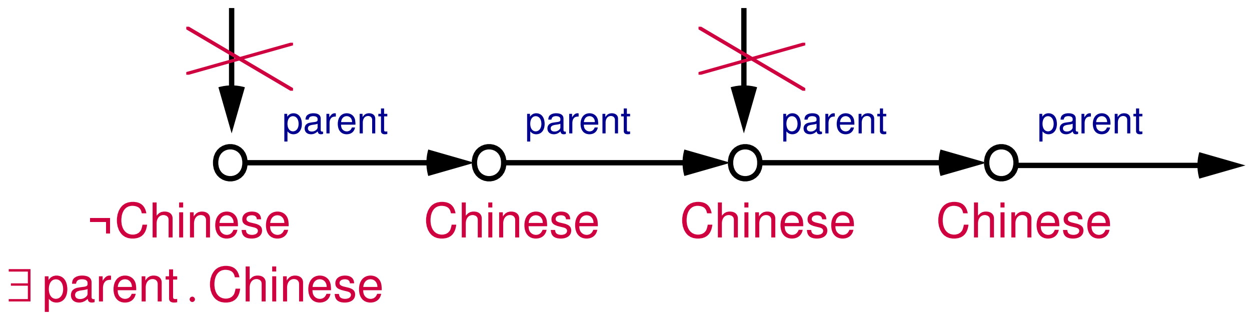
$\mathcal{SHIQ}$ does not have the finite model property

Finite model property: if a subsumption relationship does not hold, then there is a finite counter-model showing this.

Axioms:

Chinese $\sqsubseteq$ $\exists$parent.Chinese $\sqcap$ ($\leq$ 1 child)
parent is the inverse of child

Subsumption question: $\exists$parent.Chinese $\sqsubseteq$ Chinese ?



Conceptual modelling of data sources

- Semantic data model describes the "universe of discourse" about which the database will contain information by
  - introducing the terms to be used in talking about the domain, and
  - capturing their meaning by their inter-relationships and constraints.

- (Extended) entity-relationship diagrams (EER) are a semantic modelling formalism that allows to define such models.

- Semantic data models are usually employed in the design phase
  - to specify the requirements on the database
  - to generate the logical schema (e.g., in the relational model)

- Semantic data models can also be used
  - when integrating different data sources (schema integration)
  - for semantic query optimization

Description logics

for conceptual modelling

- The DL DLR with n-ary relations [Calvanese et al. 99] can express many semantic modelling languages such as EER diagrams.

- The DL $\mathcal{SHIQ}$ can express the relevant parts of DLR, and thus reasoners for $\mathcal{SHIQ}$ (like FaCT and Racer) can
  - check satisfiability of models expressed in EER
  - support schema integration by checking satisfiability of the integrated model

- ICOM (Intelligent Conceptual Modelling Tool) [Franconi and Ng 00] realizes this idea.

Dresden

Conclusion

- Expressive Description Logics can express ontology languages for the Semantic Web and semantic modelling languages for DBs, and provide useful reasoning tools.
- Reasoning in these DLs depends on the last 10 years of DL research
  - justifies our "proactive" research on foundations of DLs
  - which is responsible for the fact that we now have a significant technological lead
- Future directions:
  - even more expressive DLs (e.g., practical algorithms for $\mathcal{SHIQ}$ with individuals)
  - nonstandard inferences in DLs (least common subsumer, matching) that support building and maintaining large ontologies

Dresden

Overall goal

a warehouse of logics and inference tools

- Offer a rich palette of logics with good computational properties.
- Flexible and semantically well-founded schemes for combining logics and reasoners.
- Highly optimized implementations of reasoning tools.
- Scientifically well-founded evaluations in different application domains.

- Achieved by
  - comparing and combining different reasoning approaches (automata, tableaux, resolution, BDD, ...)
  - from different research fields (automated deduction, knowledge representation, mathematical logic, philosophical logic, verification, ...)