

Description Logics

Carsten Lutz and Ulrike Sattler

TU Dresden, Germany

Part Vb: What we left out.



I. Other means of expressivity

- n -ary relations
- role value maps: DLs without the tree model property
- concrete domains: numbers and arithmetics in concepts
- temporal extensions of Description Logics

II. Non-standard reasoning problems

- Least Common Subsumer (LCS)
- Most Specific Concept (MSC)
- Rewriting
- Approximation

For more: see upcoming DL handbook!!



Idea: replace binary roles by n -ary relations.

- + Nice for encoding ER-diagrams without reification of relations
- How can n -ary relations be handled without introducing variables?

The Description Logic \mathcal{DLR} :

Concepts C and relations R of arbitrary arity

Universal restrictions: $\forall[1](R \rightarrow 2 : C) \sqcap \forall[1](R \rightarrow 3 : D)$

Existential restrictions: $\exists[1](R \sqcap 2 : C \sqcap 3 : D)$

+ number restrictions, general TBoxes, etc.

$\text{Son} \doteq \text{Male} \sqcap \exists[3](\text{Parenthood} \sqcap 1 : \text{Mother} \sqcap 2 : \text{Father})$

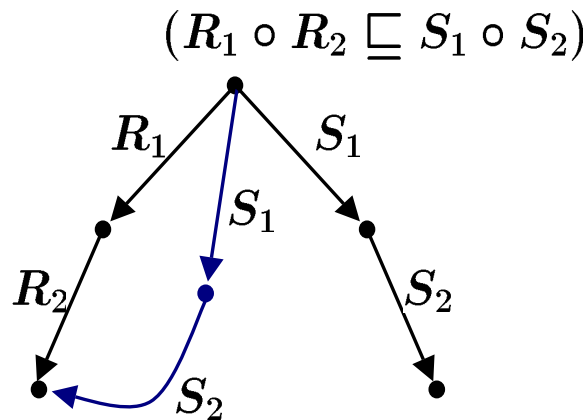
Reasoning with \mathcal{DLR} is EXPTIME -complete.



Role Value Maps

\mathcal{ALC} extended with constructor $(R_1 \circ \dots \circ R_k \sqsubseteq S_1 \circ \dots \circ S_\ell)$

Semantics:



Having no half brothers and sisters: $(\text{mother} \circ \text{child} \sqsubseteq \text{father} \circ \text{child})$
 $\sqcap (\text{father} \circ \text{child} \sqsubseteq \text{mother} \circ \text{child})$

Satisfiability of \mathcal{ALC} -concepts with role value maps is **undecidable** (without TBoxes).

Restricted variants:

- role chains of length 1: decidability simple
- role chains of equal length: open problem



Feature (Dis)Agreements

\mathcal{ALCF} : \mathcal{ALC} + another restriction of role value maps:

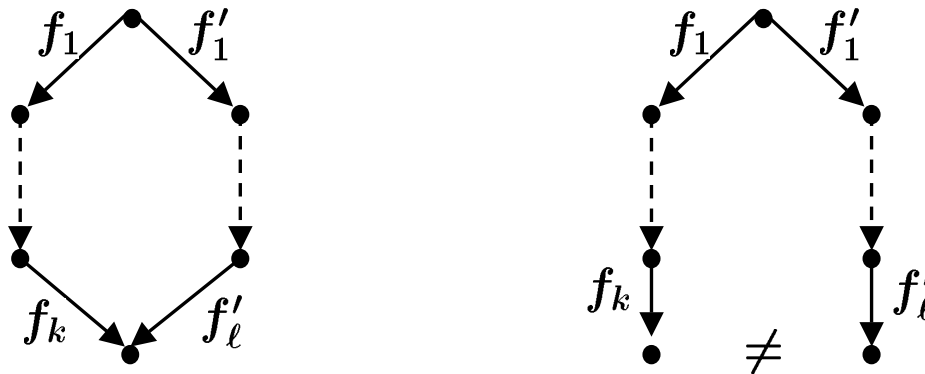
1. introduce a new kind of role called **feature**

features f are interpreted in **partial functions** $f^{\mathcal{I}}$:

$$\{(d, e_1), (d, e_2)\} \subseteq f^{\mathcal{I}} \text{ implies } e_1 = e_2$$

2. replace role value maps by **feature agreements** and **disagreements**:

$$(f_1 \circ \dots \circ f_k \downarrow f'_1 \circ \dots \circ f'_\ell) \quad (f_1 \circ \dots \circ f_k \uparrow f'_1 \circ \dots \circ f'_\ell)$$



Satisfiability of \mathcal{ALCF} -concepts is in PSPACE (without TBoxes).

Decidability / complexity are **fragile** w.r.t. the addition of TBoxes:

1. Satisfiability of \mathcal{ALCF} -concepts w.r.t. general TBoxes is undecidable.
2. Satisfiability of \mathcal{ALCF} -concepts w.r.t. acyclic TBoxes is NEXP TIME -complete.

Only very few DLs exhibit such behaviour: in the majority of cases,

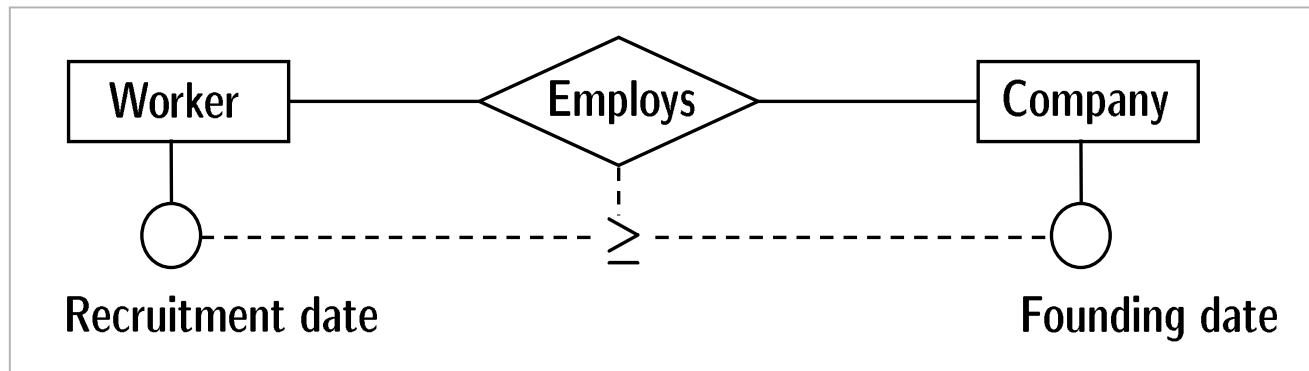
1. adding general TBoxes preserves decidability
2. adding acyclic TBoxes does not even change the complexity class

Referring to numerical data in concepts is important issue:

- In knowledge representation:

“Frank earns €500 per month which is less than his boss earns.”

- For reasoning about ER diagrams:



How can such numerical database constraints be expressed in DLs?

A concrete domain $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$ consists of

- a set $\Delta_{\mathcal{D}}$ and
- a set $\Phi_{\mathcal{D}}$ of predicate names; each $P \in \Phi_{\mathcal{D}}$ is equipped with an arity n
a **fixed extension** $P^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$.

Examples:

1. the natural numbers \mathbb{N} and predicates $=_n, <, >, =$
2. the real numbers \mathbb{R} and predicates $=_r, <, >, =, +, *$
3. the subsets of \mathbb{R}^2 and “spatial” predicates $=_{\text{polygon}}, \text{overlaps}, \text{etc.}$

The Description Logic $\mathcal{ALC}(\mathcal{D})$

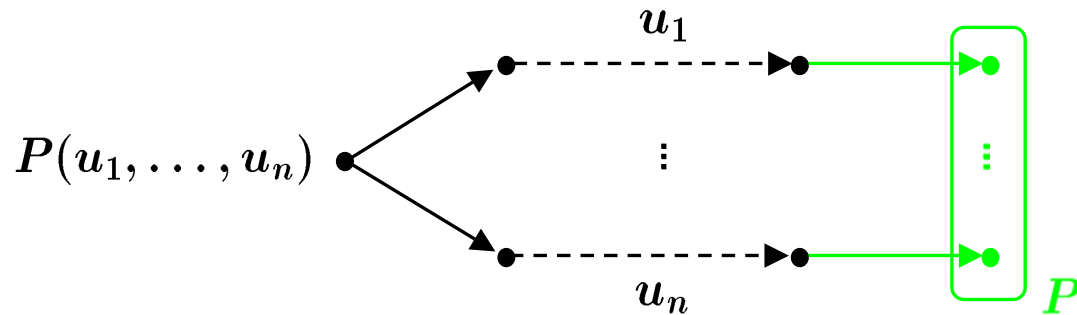
$\mathcal{ALC}(\mathcal{D})$: extension of \mathcal{ALC} with concrete domain \mathcal{D}

New atomic types:

- (abstract) features f are functional roles
- concrete features g are mapped to partial functions $g^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta_{\mathcal{D}}$

Path: sequence of features $u = f_1 \cdots f_n g$ with f_1, \dots, f_n abstract and g concrete

New concept constructor:



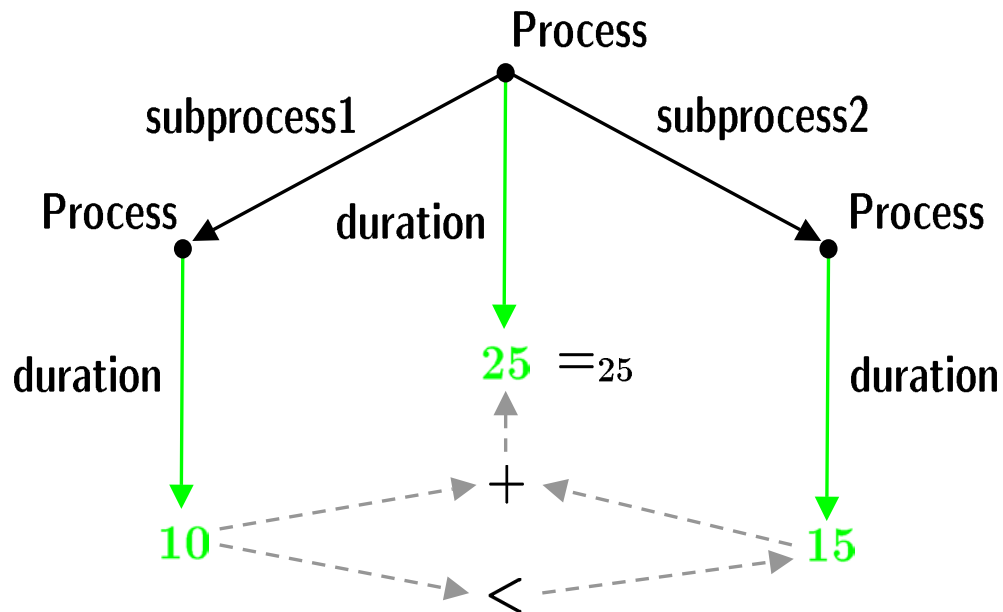
An Example $\mathcal{ALC}(\mathcal{D})$ Concept

Process $\sqcap \exists \text{subprocess1}.\text{Process} \sqcap \exists \text{subprocess2}.\text{Process}$

$\sqcap =_{25}(\text{duration})$

$\sqcap ((\text{subprocess1} \circ \text{duration}) < (\text{subprocess2} \circ \text{duration}))$

$\sqcap +((\text{subprocess1} \circ \text{duration}), (\text{subprocess2} \circ \text{duration}), \text{duration})$



Complexity and decidability depend on the concrete domain used!

- concrete domain \mathbb{N} with predicates $=_n, =, +, *$
 $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability is undecidable (without TBoxes)
- concrete domain \mathbb{N} with predicates $=_n, =, +$
 $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability PSPACE-complete.
 $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability w.r.t. general TBoxes undecidable.
- concrete domain \mathbb{N} with predicates $=_n, =, <, >$
 $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability PSPACE-complete.
 $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability w.r.t. general TBoxes EXPTIME-complete.

General decidability and complexity results:

\mathcal{D} -satisfiability: satisfiability of finite conjunctions $P_1(\overline{x_1}) \wedge \dots \wedge P_k(\overline{x_k})$

● Decidability without TBoxes:

If \mathcal{D} -satisfiability is decidable, then $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability is decidable

● Complexity without TBoxes:

If \mathcal{D} -satisfiability is in PSPACE, then $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability is PSPACE-complete.

● Complexity with acyclic TBoxes:

If \mathcal{D} -satisfiability is in NP, then $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability w.r.t. acyclic TBoxes is in NEXPTIME.

and very often NEXPTIME-hard

Observations:

- Knowledge is time dependent:

“Lucy lives in Paris now but will live in Bombay next year.”

- Time is important for defining concepts:

“A designated minister is someone who will become minister in the future.”

Idea:

Introduce temporal logic operators \bigcirc , \square , \diamond , and \mathcal{U} as concept constructors

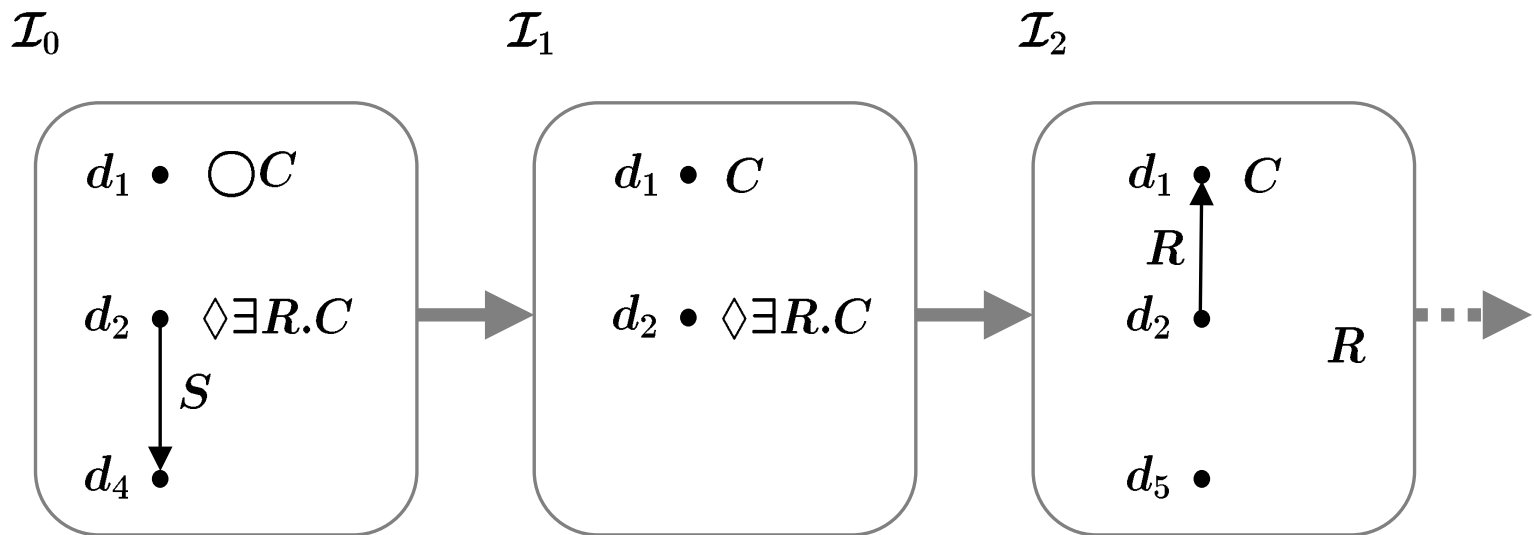
$$\text{Lucy} \rightarrow \forall \text{livesin.Paris} \sqcap \bigcirc(\forall \text{livesin.Bombay})$$
$$\text{DesignatedMinister} \doteq \text{Human} \sqcap \diamond \text{Minister}$$

Semantics of Temporal DLs

A temporal interpretation for \mathcal{ALC} is an infinite sequence

$$\mathcal{I}_0, \mathcal{I}_1, \dots$$

of standard interpretations.



- Apply temporal operators to concepts, roles, or concept equations?

$\text{EternalHusband} \doteq \text{Male} \sqcap \exists \square \text{married-to.Female}$

$\bigcirc \square (\top \doteq \text{Country} \sqcap \exists \text{location.Euroland} \rightarrow \forall \text{currency.Euro})$

- It is often natural to constrain interpretation domains:

- **varying domains:** no restriction
- **increasing domains:** $\Delta^{\mathcal{I}_0} \subseteq \Delta^{\mathcal{I}_1} \subseteq \dots$
- **decreasing domains:** $\Delta^{\mathcal{I}_0} \supseteq \Delta^{\mathcal{I}_1} \supseteq \dots$
- **constant domains:** $\Delta^{\mathcal{I}_0} = \Delta^{\mathcal{I}_1} = \dots$

constant domains are the most general case!

- Which temporal operators should be admitted? \bigcirc ? \diamond ? \square ? \mathcal{U} ?
- Do we want past operators? Future operators? Both?

Some Results

- Assumed setting:
- constant domains
 - temporal operators \bigcirc , \diamond , \square , and \mathcal{U} .
 - temporal operators for both past and future.

Decidability and complexity results:

- Temporal operators applicable to concepts only
 $\text{PTL}_{\mathcal{ALC}}$ -concept satisfiability w.r.t. general TBoxes is EXPTIME -complete.
- Temporal operators applicable to concepts and concept equations
 $\text{PTL}_{\mathcal{ALC}}$ -concept satisfiability w.r.t. general TBoxes is EXPSpace -complete.
- Temporal operators applicable to concepts and roles
 $\text{PTL}_{\mathcal{ALC}}$ -concept satisfiability w.r.t. general TBoxes is undecidable.



Least Common Subsumer (LCS)

Intuition:

The LCS of two concepts C and D is a concept describing
the **commonalities** of C and D .

Helps knowledge engineers to build up large knowledge bases.

Definition:

E is the **LCS** of C and D if

1. $C \sqsubseteq E$ and $D \sqsubseteq E$
2. for every F with $C \sqsubseteq F$ and $D \sqsubseteq F$, we have $E \sqsubseteq F$.

Only meaningful in DLs without disjunction:

\mathcal{FL}_0 : \sqcap and \forall

\mathcal{EL} : \sqcap and \exists

$\mathcal{AL}\mathcal{E}$: (\neg) , \sqcap , \forall , and \exists

$\mathcal{AL}\mathcal{E}\mathcal{Q}\mathcal{I}$: (\neg) , \sqcap , \forall , \exists , QNR , R^-

LCS usually exists and is computable, but can be hard to find

(e.g. EXPTIME for $\mathcal{AL}\mathcal{E}$)



Most Specific Concept (MSC)

Idea:

Allow the knowledge engineer to define concepts by giving examples:

1. examples given by ABox individuals
2. for each example, compute the most specific concept describing the individual
3. extract their commonalities by computing the LCS

Definition:

The **MSC** of an individual a in an ABox \mathcal{A} is the concept C such that

1. $\mathcal{A} \models a : C$
2. for each D with $\mathcal{A} \models a : D$, we have $C \sqsubseteq D$

Does not always exist (for example in \mathcal{EL} and \mathcal{ALN})

\implies consider only acyclic ABoxes or the “ k -approximation”



Motivation:

Concepts computed by LCS and MSC algorithms can become exceedingly long.

Idea:

Rewrite long concepts into smaller one using concept names defined in a TBox.

Results:

Finding the **smallest** rewriting:

e.g. NP-complete for \mathcal{AL} , PSPACE-complete for \mathcal{ALC}

But approximations do a very good job!

Note:

in the worst case, the steps

MSC \rightarrow LCS \rightarrow Rewriting

may yield an exponentially large concept.

Observation:

- It can be difficult to read complex concepts.
- especially disjunction and full negation are hard for the average user

Idea:

Approximate an \mathcal{ALC} -concept C with an $\mathcal{AL}\mathcal{E}$ -concept D preserving as much information as possible.

Definition:

An $\mathcal{AL}\mathcal{E}$ -concept D is the **approximation** of an \mathcal{ALC} -concept C if

1. $C \sqsubseteq D$
2. for each $\mathcal{AL}\mathcal{E}$ -concept E with $C \sqsubseteq E$, we have $D \sqsubseteq E$



Description Logics...

...are a lively and versatile area of research

...provide lots of research opportunities for logicians

...have many exciting applications

Most importantly: **Description Logics are fun!**

More information:

Our slides (available on the web at <http://lat.inf.tu-dresden.de/~clu/>)

The Description Logic Handbook

Send us a mail: {lutz,sattler}@tcs.inf.tu-dresden.de

