Description Logics

Structural Description Logics: $\mathcal{FL}^-$

Enrico Franconi

franconi@cs.man.ac.uk

http://www.cs.man.ac.uk/~franconi

Department of Computer Science, University of Manchester

$\mathcal{FL}^-$

- In the following part of this lecture, we will concentrate ourself to the simplest conceivable *structural* description logic: $\mathcal{FL}^-$.
- We will consider $\mathcal{FL}^-$ as a logical language: speak about:
  - Syntax
  - Semantics
  - Reasoning problems
    - Decidability
    - Complexity
  - Reasoning procedures
    - Soundness
    - Completeness
    - Asymptotic complexity

The grammar

$$C, D \quad \rightarrow \quad A \mid C \sqcap D \mid \forall R.C \mid \exists R$$

$A \in$ *atomic-concept*

$R \in$ *atomic-role*

$C, D \in$ *concept*

$$\begin{aligned} \textit{concept} \quad ::= \quad & \langle \textit{atomic-concept} \rangle \mid \\ & \langle \textit{concept} \rangle \sqcap \langle \textit{concept} \rangle \mid \\ & \exists \langle \textit{atomic-role} \rangle \mid \\ & \forall \langle \textit{atomic-role} \rangle . \langle \textit{concept} \rangle \end{aligned}$$

Alternative grammar

$$
\begin{aligned}
\textit{concept} \quad ::= \quad & \langle \textit{atomic-concept} \rangle \mid \\
& (\ : \texttt{and}\ \langle \textit{concept} \rangle \ldots \langle \textit{concept} \rangle) \mid \\
& (: \texttt{some}\ \langle \textit{atomic-role} \rangle) \mid \\
& (: \texttt{all}\ \langle \textit{atomic-role} \rangle\ \langle \textit{concept} \rangle)
\end{aligned}
$$

Intuitive semantics

Intuitively (as we already know):

- *Concepts* represent classes, i.e., sets of individuals.
- *Roles* represent relations between pairs of individuals.

- Atomic concepts are the names of primitive (undefined) concepts.
- `:and` constructions represent conjoined concepts, so, for example, `(:and Adult Male Person)` would represent the concept of something that is at the same time an adult, a male, and a person.
- This allows us to put several properties (i.e. *super-concepts* or attribute restrictions) together in the definition of a concept.

Quantifiers

- The `:all` construct provides a concept restriction on the values of an attribute (x is an `(:all R C)` if and only if each `R` of x is a `C`. Thus, `(:all CHILD Doctor)` corresponds to the concept of something all of whose children are doctors. It is a way to *restrict* the value of a slot at a frame.
- The `:some` operator guarantees that there will be at least one value for the attribute named (x is a `(:some R)` if and only if x has at least one `R`. For instance, `(:and Person (:some CHILD))` would represent the concept of a parent. This is a way to introduce a slot at a frame.

Formal Semantics

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of:

- a nonempty set $\Delta^{\mathcal{I}}$ (the *domain*)
- a function $\cdot^{\mathcal{I}}$ (the *interpretation function*) that maps
  - every *concept* to a subset of $\Delta^{\mathcal{I}}$
  - every *role* to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Extension of concepts

An interpretation function $\cdot^{\mathcal{I}}$ is an extension function iff:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta \mid \forall y.\ (x,y) \in R^{\mathcal{I}} \Rightarrow \\
&\qquad\qquad y \in C^{\mathcal{I}}\} \\
(\exists R)^{\mathcal{I}} &= \{x \in \Delta \mid \exists y.\ (x,y) \in R^{\mathcal{I}}\}
\end{aligned}
$$

Recall that $C^{\mathcal{I}}$ is a set of all the individual in the extension of C: so, writing $x \in C^{\mathcal{I}}$ has the same truth value as $C(x)$. Analogously, $(x,y) \in R^{\mathcal{I}}$ is the same as $R(x,y)$.

Exercises

Choose a domain $\Delta^{\mathcal{I}}$ and an extension functions over $\Delta^{\mathcal{I}}$; compute the extensions of the following concepts:

- `(:and Adult Male)`
- `(:and Adult Male Rich)`
- `(:all CHILD (:and Adult Male))`
- `(:some CHILD)`
- $\exists \texttt{CHILD} \sqcap \forall \texttt{CHILD}.(\exists \texttt{CHILD} \sqcap \texttt{Adult})$

The subsumption problem

$\mathtt{C} \sqsubseteq \mathtt{D}$

$\mathtt{C}$ *is subsumed by* $\mathtt{D}$

iff

for any domain $\Delta^{\mathcal{I}}$

and any extension function $\cdot^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$:

$\mathtt{C}^{\mathcal{I}} \subseteq \mathtt{D}^{\mathcal{I}}$

i.e.,

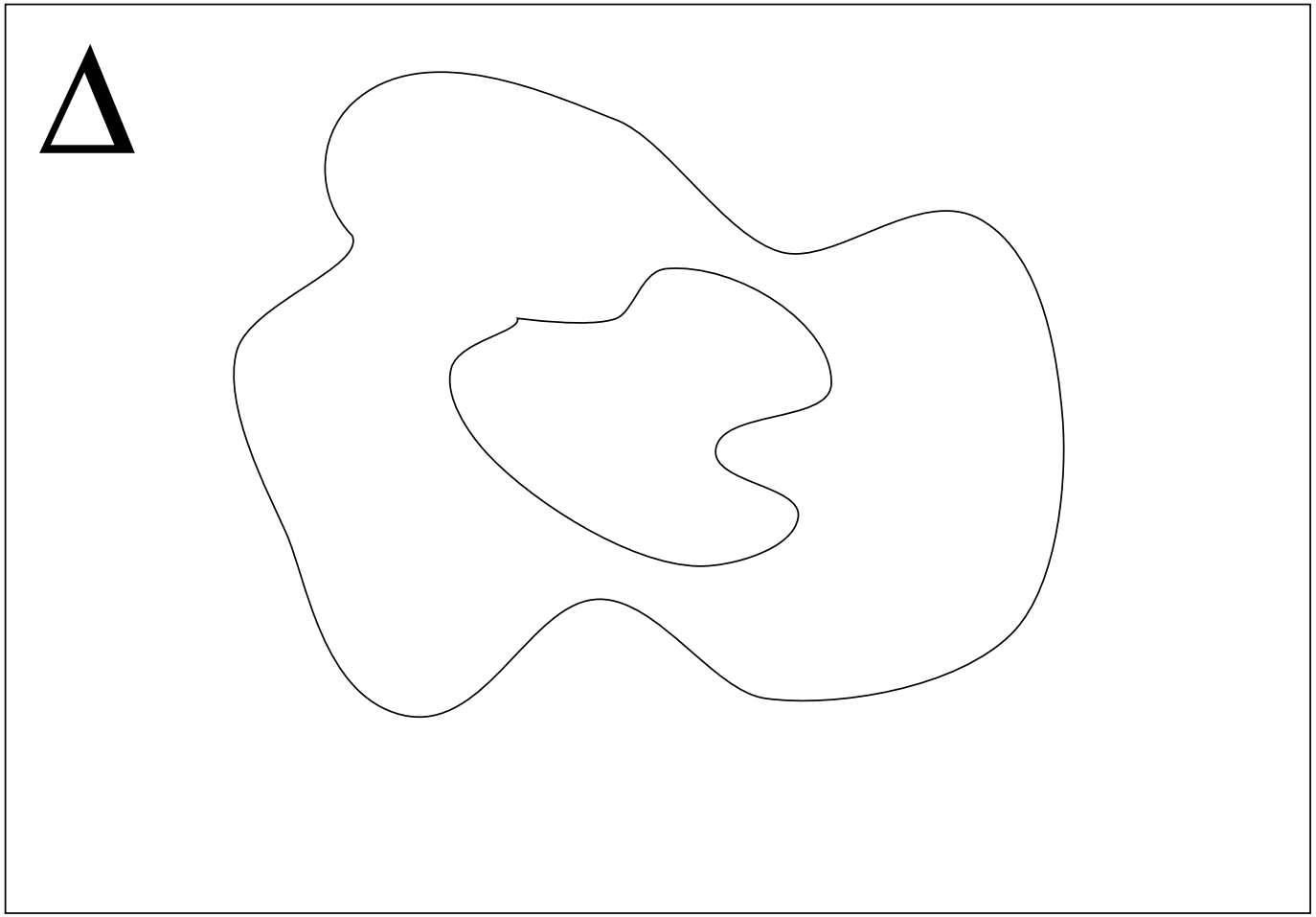
$\forall x.\ C(x) \rightarrow D(x)$

Simple examples

- `(:and Adult Male)` $\sqsubseteq$ `Adult`
- `(:and Adult Male Rich)` $\sqsubseteq$ `(:and Adult Male)`
- `(:all CHILD (:and Adult Male))` $\sqsubseteq$ `(:all CHILD Adult)`
- `(:and (:all CHILD Adult)(:some CHILD))` $\sqsubseteq$ `(:all CHILD Adult)`
- `(:all CHILD Adult)` $\not\sqsubseteq$ `(:some CHILD)`
- `(:some CHILD)` $\not\sqsubseteq$ `(:all CHILD Adult)`

The universal quantifier

$(\forall R.C)^{\mathcal{I}} =$



Computational properties

Subsumption for $\mathcal{FL}^-$ has the following computational properties, which will be proved constructively:

- Decidable
- in P

The subsumption *structural* algorithm

- The algorithm for computing Subsumption is based on structural comparisons between concept expressions.
- At the heart of structural comparison is the idea that if the two concept expressions to be compared are made of subexpressions, one can compare separately one subexpression of a concept with all those of the others.

The normal form

The algorithm works in two phases: first, concepts are rewritten in a normal form, then their structures are compared.

Normal Form:

1. All nested conjunctions are flattened, i.e. $A \sqcap (B \sqcap C) \rightsquigarrow A \sqcap B \sqcap C$.
2. All conjunctions of universal quantifications are factorized, i.e. $\forall R.C \sqcap \forall R.D \rightsquigarrow \forall R.(C \sqcap D)$.

The rewritten concepts are logically equivalent to the previous ones, hence subsumption is preserved by this transformation.

(Exercise: prove it)

The core algorithm: SUBS?[C,D]

Let $C = C_1 \sqcap \cdots \sqcap C_n$ and $D = D_1 \sqcap \cdots \sqcap D_m$
(in normal form).

Then SUBS?[C,D] returns TRUE if and only if for all C_i:

1. if C_i is either an atomic concept, or is a concept of the form $\exists R$, then there exists a D_j such that $C_i = D_j$;
2. if C_i is a concept of the form $\forall R.C'$, then there exists a D_j of the form $\forall R.D'$ (same atomic role R) such that SUBS?[C',D'].

Simple exercises

Check the following subsumption using the structural algorithm:

- `(:and Adult Male)` $\sqsubseteq$ `Adult`
- `(:and Adult Male Rich)` $\sqsubseteq$ `(:and Adult Male)`
- `(:all CHILD (:and Adult Male))` $\sqsubseteq$ `(:all CHILD Adult)`
- `(:and (:all CHILD Adult)(:some CHILD))` $\sqsubseteq$ `(:all CHILD Adult)`
- `(:all CHILD Adult)` $\not\sqsubseteq$ `(:some CHILD)`
- `(:some CHILD)` $\not\sqsubseteq$ `(:all CHILD Adult)`

Asymptotic complexity

- By induction on the nesting of $\forall$-quantifiers, one can prove that the complexity of the above algorithm is $O(|C| \times |D|)$ (i.e., quadratic in the length of the longest argument).
- If subexpressions of each concept are ordered (e.g. lexicographically), then it can be shown that the complexity is only linear, so the dominant factor becomes the complexity of ordering concepts.

Soundness

Remember: whenever a **sound** reasoning procedure claims to have found a solution for a given instance of the problem, then this is actually a solution.

The structural subsumption algorithm is sound since, when it says that a concept $\mathtt{C}$ subsumes a concept $\mathtt{D}$ – i.e., SUBS?[C, D] returns true – then it holds that $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$ for all interpretations.

Observation: the part of the algorithm computing the normal form does not change the extension of the concepts for any interpretation; thus it does not affect the soundness (and the completeness) of the algorithm.

Informal proof

- Suppose that SUBS?[C, D] returns TRUE and consider one of the conjuncts of C – call it C_i.
- Either C_i is among the D_j, or it is of the form $\forall R.C'$.
- In the latter case, there is a $\forall R.D'$ among the D_j, where SUBS?[C',D'].
- Then, by induction, any extension of D' must be a subset of C', and so any extension of D_j must be a subset of C_i's.
- So, no matter what C_i is, the extension of D – which is the conjunction of all the D_j's – must be a subset of C_i.
- Since this is true for every C_i, the extension of D must also be a subset of the extension of C – which is the intersection of all the extensions of C_i.
- So, whenever SUBS?[C, D] returns TRUE, C subsumes D, i.e., $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$.

Completeness

Remember: whenever an instance of the problem has a solution, a **complete** reasoning procedure computes the solution for that instance.

The structural subsumption algorithm is complete since, whenever it holds that $\mathtt{C}^{\mathcal{I}} \subseteq \mathtt{D}^{\mathcal{I}}$ for all interpretations, then the algorithm says that $\mathtt{C}$ subsumes $\mathtt{D}$.

Observation: The proof is done by showing that anytime SUBS?[C, D] returns FALSE, there exists an interpretation assigning an element to D but not to C, i.e., in that interpretation the extension on C is not a superset of the extension of D.

Idea of the proof

- The proof is done by showing that anytime SUBS?[C, D] returns FALSE, there exists an interpretation assigning an element to D but not to C, i.e., in that interpretation the extension on C is not a superset of the extension of D.
- This shows a counter-example, i.e., anytime SUBS?[C, D] returns FALSE it is not true that $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$ for that particular interpretation, and so $C \not\sqsubseteq D$.

Idea of the proof *(cont.)*

- The proof relies on the fact that anytime SUBS?[C, D] returns FALSE it is possible to find a conjunct C_i of C which has no correspondent conjunct in D.
- It is shown that in this case there exists an interpretation which assigns an object to any primitive concept, but not to the factorized one C_i.
- Thus, it is not possible that $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$.

Structural algorithms (or normalize-and-compare)

What happens if we enrich the expressivity?

- CLASSIC
- BACK
- LOOM

These (old) systems have incomplete algorithms, due to the interactions between constructors, which can not be taken into account by structural algorithms, which is based on syntactical comparisons between subexpressions of the concepts.

Example: $A \sqcup \neg A$ subsumes every concept, even if such a concept does not mention at all the atomic concept A in its definition.