Logical Aspects of Artificial Intelligence
Introduction to DLs & Properties

Stéphane Demri demri@lsv.fr
https://cv.archives-ouvertes.fr/stephane-demri

September 21th, 2022

Plan of the lecture

- Recapitulation of the previous lecture.
- More $\mathcal{ALC}$ extensions.
- Relationships with first-order logic.
- Tree interpretation property.
- A decision procedure using model-theoretical properties.
- Tableaux proof system for $\mathcal{ALC}$ (Part I).
- Exercises session.

Recapitulation of the Previous Lecture

$\mathcal{ALC}$ in a nutshell

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

- Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$.
- TBox $\mathcal{T} = \{C \sqsubseteq D, \ldots\}$.
- ABox $\mathcal{A} = \{a : C, (b, b') : r, \ldots\}$.
- Knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. (a.k.a. ontology)
- Decision problems include concept satisfiability, knowledge base consistency, and other problems for classification.

$$
\begin{aligned}
\top^{\mathcal{I}} &\stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &\stackrel{\text{def}}{=} \emptyset \\
(\neg C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &\stackrel{\text{def}}{=} C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &\stackrel{\text{def}}{=} C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(\mathfrak{a}) \cap C^{\mathcal{I}} \neq \emptyset\} \\
(\forall r.C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(\mathfrak{a}) \subseteq C^{\mathcal{I}}\}
\end{aligned}
$$

A few properties about $\mathcal{ALC}$

- Concept satisfiability problem is PSPACE-complete.
- Knowledge base consistency problem is EXPTIME-complete.
- $\mathcal{ALC}$ has many well-known fragments and extensions, some of them to deal with
  - inverse roles,
  - number restrictions,
  - etc..

Extensions of $\mathcal{ALC}$ (Part II)

Naming individuals in complex concepts

- Concepts in $\mathcal{ALC}$

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

- ... but individual names a (as in the concept assertion $a : C$) are not concepts in $\mathcal{ALC}$.

Naming individuals in complex concepts

- Concepts in $\mathcal{ALC}$

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

- ... but individual names a (as in the concept assertion $a : C$) are not concepts in $\mathcal{ALC}$.

- How to express the concept “the courses taught by Mary” ?

$$\mathtt{Course} \sqcap \exists \mathtt{Teaches}^{-}.\mathtt{Mary}$$

Naming individuals in complex concepts

- Concepts in $\mathcal{ALC}$

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

- …but individual names a (as in the concept assertion $a : C$) are not concepts in $\mathcal{ALC}$.
- How to express the concept "the courses taught by Mary" ?

$$\texttt{Course} \sqcap \exists \texttt{Teaches}^{-}.\texttt{Mary}$$

- Nominals in hybrid (modal) logics: propositional variables true in only one world of the domain.

Naming individuals in complex concepts

- Concepts in $\mathcal{ALC}$

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

- ...but individual names a (as in the concept assertion $a : C$) are not concepts in $\mathcal{ALC}$.

- How to express the concept "the courses taught by Mary" ?

$$\texttt{Course} \sqcap \exists \texttt{Teaches}^{-}.\texttt{Mary}$$

- Nominals in hybrid (modal) logics: propositional variables true in only one world of the domain.

- **Nominals** in DLs: individual names inside concepts, written $\{a\}$, where $a \in \mathsf{N_I}$ with $\{a\}^{\mathcal{I}} \overset{\text{def}}{=} \{a^{\mathcal{I}}\}$.

⚠ Syntactic trick $\{\cdot\}$: $\texttt{Course} \sqcap \exists \texttt{Teaches}^{-}.\{\texttt{Mary}\}$.

Nominals in DLs

- Given a logic $\mathfrak{L}$, $\mathfrak{L}\mathcal{O}$ is defined as $\mathfrak{L}$ except that nominals are added.

Nominals in DLs

- Given a logic $\mathfrak{L}$, $\mathfrak{L}\mathcal{O}$ is defined as $\mathfrak{L}$ except that nominals are added.
- Concept satisfiability for $\mathcal{ALCOQ}$ is PSPACE-complete and knowledge base consistency is EXPTIME-complete.
- ... but concept satisfiability for $\mathcal{ALCOI}$ is EXPTIME-complete.

Role $\mathcal{H}$ierarchies

- $\mathcal{ALC}$ is not able to express complex role constraints such that a relation is included in another relation.
  (GCIs are concept inclusions though!)
- Typically, the interpretation of `Attends` should include the interpretation of `AttendsActively`.

Role $\mathcal{H}$ierarchies

- $\mathcal{ALC}$ is not able to express complex role constraints such that a relation is included in another relation.
  (GCIs are concept inclusions though!)

- Typically, the interpretation of `Attends` should include the interpretation of `AttendsActively`.

- **Role inclusion axiom** (RIA) of the form $r \sqsubseteq s$ with

$$\mathcal{I} \models r \sqsubseteq s \quad \overset{\text{def}}{\Leftrightarrow} \quad r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$$

- Given a logic $\mathfrak{L}$, $\mathfrak{L}\mathcal{H}$ is defined as $\mathfrak{L}$ except that role inclusion axioms are added (in the TBox).

or add an ex

Role $\mathcal{H}$ierarchies

- $\mathcal{ALC}$ is not able to express complex role constraints such that a relation is included in another relation.
  (GCIs are concept inclusions though!)

- Typically, the interpretation of `Attends` should include the interpretation of `AttendsActively`.

- **Role inclusion axiom** (RIA) of the form $r \sqsubseteq s$ with

$$\mathcal{I} \models r \sqsubseteq s \quad \stackrel{\text{def}}{\Leftrightarrow} \quad r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$$

- Given a logic $\mathfrak{L}$, $\mathfrak{L}\mathcal{H}$ is defined as $\mathfrak{L}$ except that role inclusion axioms are added (in the TBox $\wedge$).

  or add an ex

- Concept satisfiability for $\mathcal{ALCH}$ is PSPACE-complete and knowledge base consistency is EXPTIME-complete.

A local version

- A **role value map** is an atomic concept of the form $r \sqsubseteq s$:

$$(r \sqsubseteq s)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(\mathfrak{a}) \subseteq s^{\mathcal{I}}(\mathfrak{a})\}$$

A local version

- A **role value map** is an atomic concept of the form $r \sqsubseteq s$:

$$(r \sqsubseteq s)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(\mathfrak{a}) \subseteq s^{\mathcal{I}}(\mathfrak{a})\}$$

⚠ Role value maps are local variants of role inclusion axioms (RIAs).

A local version

- A **role value map** is an atomic concept of the form $r \sqsubseteq s$:

$$(r \sqsubseteq s)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(\mathfrak{a}) \subseteq s^{\mathcal{I}}(\mathfrak{a})\}$$

⚠ Role value maps are local variants of role inclusion axioms (RIAs).

- The RIA $r \sqsubseteq s$ can be encoded by the GCI $\top \sqsubseteq (r \sqsubseteq s)$.

(doe

Transitive roles

- Many natural relations are transitive (`AncestorOf`, `HasPart`, etc.) but this cannot be expressed in $\mathcal{ALCH}$.

- **Transitivity axioms** are of the form $\mathtt{Trans}(r)$:

$$\mathcal{I} \models \mathtt{Trans}(r) \quad \overset{\text{def}}{\Leftrightarrow} \quad r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}}$$

Transitive roles

- Many natural relations are transitive (`AncestorOf`, `HasPart`, etc.) but this cannot be expressed in $\mathcal{ALCH}$.

- **Transitivity axioms** are of the form $\texttt{Trans}(r)$:

$$\mathcal{I} \models \texttt{Trans}(r) \quad \overset{\text{def}}{\Leftrightarrow} \quad r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}}$$

- Extensions of $\mathcal{ALC}$ with transitivity axioms in TBoxes is obtained by replacing $\mathcal{ALC}$ by $\mathcal{S}$. (new naming rule)

Transitive roles

- Many natural relations are transitive (`AncestorOf`, `HasPart`, etc.) but this cannot be expressed in $\mathcal{ALCH}$.

- **Transitivity axioms** are of the form $\mathtt{Trans}(r)$:

$$\mathcal{I} \models \mathtt{Trans}(r) \quad \stackrel{\text{def}}{\Leftrightarrow} \quad r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}}$$

- Extensions of $\mathcal{ALC}$ with transitivity axioms in TBoxes is obtained by replacing $\mathcal{ALC}$ by $\mathcal{S}$. (new naming rule)

- Concept satisfiability for $\mathcal{S}$ is PSPACE-complete and EXPTIME-complete for knowledge base consistency.

- Other properties are included in knowledge bases such as reflexivity, irreflexivity, symmetry, functionality, …

Syntactic sugar

Syntax	DL syntax	Equivalent axioms
DisjointWith	$Dis(C, D)$	$C \sqcap D \sqsubseteq \bot$
DisjointUnionOf		$C \equiv C_1 \sqcup \cdots \sqcup C_n$ + $Dis(C_i, C_j)$
EquivalentTo	$r \equiv s$	$r \sqsubseteq s$, $s \sqsubseteq r$
Domain	$Dom(r) \sqsubseteq C$	$\exists r.\top \sqsubseteq C$
Range	$Ran(r) \sqsubseteq C$	$\top \sqsubseteq \forall r.C$

A selection of complexity results

http://www.cs.man.ac.uk/~ezolin/dl/

cs.man.ac.uk

Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and updated often

Base description logic: $\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements

$\mathcal{ALC}$ ::= $\bot$ | $\top$ | A | $\neg C$ | $C \sqcap D$ | $C \sqcup D$ | $\exists R.C$ | $\forall R.C$

Concept constructors:	**Role constructors:**
☐ $\mathcal{F}$ – functionality[2]: $(\leq 1\ R)$	☐ $\mathcal{I}$ – role inverse: R^-
☐ $\mathcal{N}$ – (unqualified) number restrictions: $(\geq n\ R)$, $(\leq n\ R)$	☐ $\sqcap$ – role intersection[3]: $R \sqcap S$
☐ $\mathcal{Q}$ – qualified number restrictions: $(\geq n\ R.C)$, $(\leq n\ R.C)$	☐ $\sqcup$ – role union: $R \sqcup S$
☐ $\mathcal{O}$ – nominals: $\{a\}$ or $\{a_1, \ldots, a_n\}$ ("one-of")	☐ $\neg$ – role complement: $\neg R$ [full]
☐ μ – least fixpoint operator: $\mu X.C$	☐ $\circ$ – role chain (composition): $R \circ S$
[Forbid] complex roles[5] in number restrictions[6]	☐ $*$ – reflexive-transitive closure[4]: R^*
	☐ id – concept identity: $id(C)$

TBox (concept axioms):	**RBox (role axioms):**
◉ empty TBox	☐ $\mathcal{S}$ – role transitivity: $Tr(R)$
○ acyclic TBox ($A \equiv C$, A is a concept name; no cycles)	☐ $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
○ general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D)	☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
	☐ s – some additional features (click to see them)

[Reset] You have selected a Description Logic: $\mathcal{ALC}$

Complexity[7] of reasoning problems[8]		
Concept satisfiability	**PSpace-complete**	• Hardness for $\mathcal{ALC}$: see [80]. • Upper bound for $\mathcal{ALCQ}$: see [12, Theorem 4.6].
ABox consistency	**PSpace-complete**	• Hardness follows from that for concept satisfiability. • Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A].
Important properties of the Description Logic		
Finite model property	**Yes**	$\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the finite model property can be f Sect. 2.3].
Tree model property	**Yes**	$\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the tree model property can be fo Proposition 2.15].

Maintained by: Evgeny Zolin

Any comments are we

A queen logic $\mathcal{SROIQ}$

- $\mathcal{SROIQ}$ agrees with the ontology language OWL 2 DL.
- $\mathcal{SROIQ}$ contains more than one might think from its name.

A queen logic $\mathcal{SROIQ}$

- $\mathcal{SROIQ}$ agrees with the ontology language OWL 2 DL.
- $\mathcal{SROIQ}$ contains more than one might think from its name.
- Its knowledge bases contain a **role box (RBox)** to specify constraints about the interpretation of **role expressions.**

A queen logic $\mathcal{SROIQ}$

- $\mathcal{SROIQ}$ agrees with the ontology language OWL 2 DL.
- $\mathcal{SROIQ}$ contains more than one might think from its name.
- Its knowledge bases contain a **role box (RBox)** to specify constraints about the interpretation of **role expressions.**
- The set of roles **R** is made of role names r, its converses r^- and the **universal role** U with $U^\mathcal{I} \stackrel{\text{def}}{=} \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.
- New atomic concept $\exists R.\mathrm{Self}$ with

$$(\exists R.\mathrm{Self})^\mathcal{I} \stackrel{\text{def}}{=} \{\mathfrak{a} \mid (\mathfrak{a}, \mathfrak{a}) \in R^\mathcal{I}\}$$

A queen logic $\mathcal{SROIQ}$

- $\mathcal{SROIQ}$ agrees with the ontology language OWL 2 DL.
- $\mathcal{SROIQ}$ contains more than one might think from its name.
- Its knowledge bases contain a **role box (RBox)** to specify constraints about the interpretation of **role expressions.**
- The set of roles $\mathbf{R}$ is made of role names r, its converses r^- and the **universal role** U with $U^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.
- New atomic concept $\exists R.\mathrm{Self}$ with

$$(\exists R.\mathrm{Self})^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \mid (\mathfrak{a}, \mathfrak{a}) \in R^{\mathcal{I}}\}$$

- Role assertions $(a, b) : \neg R$ are allowed in the ABox.
- + ingredients from its name (nominals, qualified number restrictions, inverse).

Role axioms in the RBox

- **Complex role inclusion axioms (CRIA)** $R_1 \circ \cdots \circ R_n \sqsubseteq R$.

  (R instead r because R not nece

⚠ A regularity constraint is required on the set of CRIAs (unspecified here).

Role axioms in the RBox

- **Complex role inclusion axioms (CRIA)** $R_1 \circ \cdots \circ R_n \sqsubseteq R$.

  (R instead r because R not nece

⚠ A regularity constraint is required on the set of CRIAs (unspecified here).

- Role axioms specifying disjointness, transitivity, reflexivity, irreflexivity, symmetry, asymmetry.

- The knowledge base consistency problem for $\mathcal{SROIQ}$ is N2EXPTIME-complete.

How $\mathcal{SROIQ}$ is related to the semantic web

- Semantic web:
  - A vision of a computer-understandable Web.
  - Distributed knowledge and data in reusable form.
  - XML, RDF and OWL are part of the story.

How $\mathcal{SROIQ}$ is related to the semantic web

- Semantic web:
  - A vision of a computer-understandable Web.
  - Distributed knowledge and data in reusable form.
  - XML, RDF and OWL are part of the story.
- Principles towards a semantic Web of data
  - Give a name to everything.
  - Relationships form a graph between the entities.
  - The names are addresses on the Web.
  - Provide a formal semantics so that knowledge is encoded in a machine interpretable way.

OWL based on description logics

- OWL: Web Ontology Language.
- Motivated by semantic web activities: add meaning to web content by annotating it with terms defined in ontologies.
- It is a World Wide Web (W3C) standard.
- OWL has an explicit formal semantics ...
  ...based on description logics such as $\mathcal{SROIQ}$.

DLs and ontology languages

- W3C's OWL 2 is based on $\mathcal{SROIQ}$.
- OWL was based on $\mathcal{SHOIN}$.

(gue

DLs and ontology languages

- W3C's OWL 2 is based on $\mathcal{SROIQ}$.
- OWL was based on $\mathcal{SHOIN}$.

(gue

- An OWL ontology is a mixed set containing TBox axioms and ABox assertions.
- More on complexity/scalability:
  - OWL ($\mathcal{SHOIN}$) is NEXPTIME-complete.
  - OWL 2 EL is PTIME-complete.

OWL RDF/XML exchange syntax

$$\texttt{Person} \sqcap \forall \texttt{Teaches.Course}$$

```
<owl:Class>
 <owl:intersectionOf rdf:parseType=" collection">
   <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#Teaches"/>
     <owl:allValuesFrom>
      <owl:Class rdf:about="#Course"/>
     </owl:allValuesFrom>
   </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

OWL reasoners and Protégé

- OWL reasoners: implement decision procedures for consistency and ontology classification.
- Open-source ontology editor Protégé.
  - Interaction with DL reasoners (FaCT++, Pellet, Racer).
  - Show results about ontology classification.
  - Helpful to work with toy ontologies.

More on OWL reasoners in S. Borgwardt's slides

OWL 2 Reasoners

OWL 2 DL:

- Konclude http://derivo.de/en/products/konclude/
- Pellet https://github.com/stardog-union/pellet/
- FaCT++ https://bitbucket.org/dtsarkov/factplusplus/
- HermiT https://github.com/phillord/hermit-reasoner/
- PAGOdA https://www.cs.ox.ac.uk/isg/tools/PAGOdA/

OWL 2 EL:

- ELK https://github.com/liveontologies/elk-reasoner/
- CEL https://lat.inf.tu-dresden.de/systems/cel/

OWL 2 QL:

- ontop https://github.com/ontop/ontop/
- Mastro http://www.dis.uniroma1.it/~mastro/

OWL 2 RL:

- RDFox http://www.cs.ox.ac.uk/isg/tools/RDFox/

Relationships with First-Order Logic

Foreign language for modal/classical logicians

- Description logics can be understood as fragments of first-order logics.
- Similarly, reasoning tasks for description logics can be understood as decision problems for modal logics.

Foreign language for modal/classical logicians

- Description logics can be understood as fragments of first-order logics.
- Similarly, reasoning tasks for description logics can be understood as decision problems for modal logics.
- This can be made precise and sometimes results for modal/first-oder logics can be used.
- Specificity of DLs: many fragments, many extensions and numerous original reasoning tasks (non-exhaustive presentation in this course).

Description logics and its shared history

- Late 80s: description logics developped as logical formalisms for semantic networks.

Description logics and its shared history

- Late 80s: description logics developped as logical formalisms for semantic networks.

- In the 1990s: relationships with first-order logic, modal logics, PDL-like logics, etc.
  (PDL = Propositional Dynamic Logic)

Description logics and its shared history

- Late 80s: description logics developped as logical formalisms for semantic networks.
- In the 1990s: relationships with first-order logic, modal logics, PDL-like logics, etc.
  (PDL = Propositional Dynamic Logic)
- Logical basis for the Web Ontology Language OWL.

Description logics and its shared history

- Late 80s: description logics developped as logical formalisms for semantic networks.
- In the 1990s: relationships with first-order logic, modal logics, PDL-like logics, etc.
  (PDL = Propositional Dynamic Logic)
- Logical basis for the Web Ontology Language OWL.
- Analogies between ontologies and databases lead also to relationships with query answering languages.

From concepts to first-order formulae

- Interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ understood as first-order models.

From concepts to first-order formulae

- Interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ understood as first-order models.
- Translation of non-logical symbols.

Description logics		First-order logic
individual name $a \in \mathsf{N_I}$	$\approx$	constant a
concept name $A \in \mathsf{N_C}$	$\approx$	unary predicate A
role name $r \in \mathsf{N_R}$	$\approx$	binary predicate r

- Translation of concepts, assertions and GCIs by internalising the DL semantics.

Example of translation

- A small knowledge base.

$$\begin{array}{lcl} \exists\texttt{Attends}.\top & \sqsubseteq & \texttt{Person} \\ \texttt{Teacher} & \equiv & \texttt{Person} \sqcap \exists\texttt{Teaches}.\texttt{Course} \\ \texttt{Alice} : \texttt{Teacher} & & \end{array}$$

Example of translation

- A small knowledge base.

$$\begin{array}{lcl} \exists \mathtt{Attends}.\top & \sqsubseteq & \mathtt{Person} \\ \mathtt{Teacher} & \equiv & \mathtt{Person} \sqcap \exists \mathtt{Teaches}.\mathtt{Course} \\ \mathtt{Alice} : \mathtt{Teacher} & & \end{array}$$

- Its translation in FOL:

$$\forall\, \mathtt{x}\, (\exists\, \mathtt{y}\, \mathtt{Attends}(\mathtt{x}, \mathtt{y}) \Rightarrow \mathtt{Person}(\mathtt{x})) \wedge$$

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

$$\mathtt{Teacher}(\mathtt{Alice})$$

Example of translation

- A small knowledge base.

$$
\begin{array}{lcl}
\exists \mathtt{Attends}.\top & \sqsubseteq & \mathtt{Person} \\
\mathtt{Teacher} & \equiv & \mathtt{Person} \sqcap \exists \mathtt{Teaches}.\mathtt{Course} \\
\mathtt{Alice} : \mathtt{Teacher} & &
\end{array}
$$

- Its translation in FOL:

$$
\begin{gathered}
\forall\, \mathtt{x}\, (\exists\, \mathtt{y}\, \mathtt{Attends}(\mathtt{x}, \mathtt{y}) \Rightarrow \mathtt{Person}(\mathtt{x})) \wedge \\
\forall\, \mathtt{x}\, (\mathtt{Teacher}(\mathtt{x}) \Leftrightarrow (\mathtt{Person}(\mathtt{x}) \wedge \exists\, \mathtt{y} (\mathtt{Teaches}(\mathtt{x}, \mathtt{y}) \wedge \mathtt{Course}(\mathtt{y})))) \wedge \\
\mathtt{Teacher}(\mathtt{Alice})
\end{gathered}
$$

- Concepts can be understood as first-order formulae with one free variable.

Internalisation of $\mathcal{ALC}$ semantics

$$
\begin{array}{lcl}
\mathfrak{t}(A, \mathrm{x}) & \stackrel{\text{def}}{=} & A(\mathrm{x}) \\
\mathfrak{t}(\top, \mathrm{x}) \;/\; \mathfrak{t}(\bot, \mathrm{x}) & \stackrel{\text{def}}{=} & \top \;/\; \bot \\
\mathfrak{t}(\neg C, \mathrm{x}) & \stackrel{\text{def}}{=} & \neg \mathfrak{t}(C, \mathrm{x}) \\
\mathfrak{t}(C_1 \sqcap C_2, \mathrm{x}) & \stackrel{\text{def}}{=} & \mathfrak{t}(C_1, \mathrm{x}) \wedge \mathfrak{t}(C_2, \mathrm{x}) \\
\mathfrak{t}(C_1 \sqcup C_2, \mathrm{x}) & \stackrel{\text{def}}{=} & \mathfrak{t}(C_1, \mathrm{x}) \vee \mathfrak{t}(C_2, \mathrm{x}) \\
\mathfrak{t}(\exists r.C, \mathrm{x}) & \stackrel{\text{def}}{=} & \exists\, \mathrm{y}\; r(\mathrm{x}, \mathrm{y}) \wedge \mathfrak{t}(C, \mathrm{y}) \\
\mathfrak{t}(\forall r.C, \mathrm{x}) & \stackrel{\text{def}}{=} & \forall\, \mathrm{y}\; r(\mathrm{x}, \mathrm{y}) \Rightarrow \mathfrak{t}(C, \mathrm{y})
\end{array}
$$

where y is a fresh variable.

Internalisation of $\mathcal{ALC}$ semantics

$$
\begin{array}{lcl}
\mathfrak{t}(A, \mathrm{x}) & \stackrel{\text{def}}{=} & A(\mathrm{x}) \\
\mathfrak{t}(\top, \mathrm{x}) \ / \ \mathfrak{t}(\bot, \mathrm{x}) & \stackrel{\text{def}}{=} & \top \ / \ \bot \\
\mathfrak{t}(\neg C, \mathrm{x}) & \stackrel{\text{def}}{=} & \neg \mathfrak{t}(C, \mathrm{x}) \\
\mathfrak{t}(C_1 \sqcap C_2, \mathrm{x}) & \stackrel{\text{def}}{=} & \mathfrak{t}(C_1, \mathrm{x}) \wedge \mathfrak{t}(C_2, \mathrm{x}) \\
\mathfrak{t}(C_1 \sqcup C_2, \mathrm{x}) & \stackrel{\text{def}}{=} & \mathfrak{t}(C_1, \mathrm{x}) \vee \mathfrak{t}(C_2, \mathrm{x}) \\
\mathfrak{t}(\exists r.C, \mathrm{x}) & \stackrel{\text{def}}{=} & \exists\, \mathrm{y}\ r(\mathrm{x}, \mathrm{y}) \wedge \mathfrak{t}(C, \mathrm{y}) \\
\mathfrak{t}(\forall r.C, \mathrm{x}) & \stackrel{\text{def}}{=} & \forall\, \mathrm{y}\ r(\mathrm{x}, \mathrm{y}) \Rightarrow \mathfrak{t}(C, \mathrm{y})
\end{array}
$$

where y is a fresh variable.

- Given $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\mathfrak{a} \in C^{\mathcal{I}}$ iff $\mathcal{I}, \rho[\mathrm{x} \leftarrow \mathfrak{a}] \models \mathfrak{t}(C, \mathrm{x})$.

- Many-one reduction from $\mathcal{ALC}$ satisfiability problem to FOL satisfiability. (why?)

Translating knowledge bases

$$\begin{array}{lcl}
\mathfrak{t}(C \sqsubseteq D) & \stackrel{\text{def}}{=} & \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Rightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(C \equiv D) & \stackrel{\text{def}}{=} & \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Leftrightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(a : C) & \stackrel{\text{def}}{=} & \mathfrak{t}(C, \mathrm{x})[a/\mathrm{x}] \\
\mathfrak{t}((a, b) : r) & \stackrel{\text{def}}{=} & r(a, b)
\end{array}$$

where $\varphi[a/\mathrm{x}]$ (also written $\varphi(a)$) is the formula obtained from φ by replacing the free occurrences of x by a.

Translating knowledge bases

$$
\begin{aligned}
\mathfrak{t}(C \sqsubseteq D) &\stackrel{\text{def}}{=} \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Rightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(C \equiv D) &\stackrel{\text{def}}{=} \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Leftrightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(a : C) &\stackrel{\text{def}}{=} \mathfrak{t}(C, \mathrm{x})[a/\mathrm{x}] \\
\mathfrak{t}((a, b) : r) &\stackrel{\text{def}}{=} r(a, b)
\end{aligned}
$$

where $\varphi[a/\mathrm{x}]$ (also written $\varphi(a)$) is the formula obtained from φ by replacing the free occurrences of x by a.

- $\mathfrak{t}(\mathcal{K})$ is defined as the conjunction

$$
\bigwedge_{\alpha \in \mathcal{T}} \mathfrak{t}(\alpha) \wedge \bigwedge_{\alpha \in \mathcal{A}} \mathfrak{t}(\alpha)
$$

Translating knowledge bases

$$
\begin{array}{lcl}
\mathfrak{t}(C \sqsubseteq D) & \stackrel{\text{def}}{=} & \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Rightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(C \equiv D) & \stackrel{\text{def}}{=} & \forall\, \mathrm{x}\ \mathfrak{t}(C, \mathrm{x}) \Leftrightarrow \mathfrak{t}(D, \mathrm{x}) \\
\mathfrak{t}(a : C) & \stackrel{\text{def}}{=} & \mathfrak{t}(C, \mathrm{x})[a/\mathrm{x}] \\
\mathfrak{t}((a, b) : r) & \stackrel{\text{def}}{=} & r(a, b)
\end{array}
$$

where $\varphi[a/\mathrm{x}]$ (also written $\varphi(a)$) is the formula obtained from φ by replacing the free occurrences of x by a.

- $\mathfrak{t}(\mathcal{K})$ is defined as the conjunction

$$\bigwedge_{\alpha \in \mathcal{T}} \mathfrak{t}(\alpha) \wedge \bigwedge_{\alpha \in \mathcal{A}} \mathfrak{t}(\alpha)$$

- Given $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\mathcal{I} \models \mathcal{K}$ iff $\mathcal{I} \models \mathfrak{t}(\mathcal{K})$.

- $\mathcal{K}$ is consistent iff $\mathfrak{t}(\mathcal{K})$ is satisfiable.

Locating DLs within first-order logic fragments

- The definition of $\mathfrak{t}(C, \mathrm{x})$ can be optimised to recycle variables and to use only two variables x_0 and x_1.

Locating DLs within first-order logic fragments

- The definition of $\mathfrak{t}(C, \mathrm{x})$ can be optimised to recycle variables and to use only two variables x_0 and x_1.

$$
\begin{aligned}
\mathfrak{t}(\exists r.C, \mathrm{x}_i) &\stackrel{\text{def}}{=} \exists\, \mathrm{x}_{1-i}\; r(\mathrm{x}_i, \mathrm{x}_{1-i}) \wedge \mathfrak{t}(C, \mathrm{x}_{1-i}) \\
\mathfrak{t}(\forall r.C, \mathrm{x}_i) &\stackrel{\text{def}}{=} \forall\, \mathrm{x}_{1-i}\; r(\mathrm{x}_i, \mathrm{x}_{1-i}) \Rightarrow \mathfrak{t}(C, \mathrm{x}_{1-i})
\end{aligned}
$$

Locating DLs within first-order logic fragments

- The definition of $\mathfrak{t}(C, \mathtt{x})$ can be optimised to recycle variables and to use only two variables $\mathtt{x}_0$ and $\mathtt{x}_1$.

$$\begin{aligned}
\mathfrak{t}(\exists r.C, \mathtt{x}_i) &\stackrel{\text{def}}{=} \exists\, \mathtt{x}_{1-i}\; r(\mathtt{x}_i, \mathtt{x}_{1-i}) \wedge \mathfrak{t}(C, \mathtt{x}_{1-i}) \\
\mathfrak{t}(\forall r.C, \mathtt{x}_i) &\stackrel{\text{def}}{=} \forall\, \mathtt{x}_{1-i}\; r(\mathtt{x}_i, \mathtt{x}_{1-i}) \Rightarrow \mathfrak{t}(C, \mathtt{x}_{1-i})
\end{aligned}$$

- FO2 (FOL restricted to two individual variables) satisfiability is NExpTime-complete.

Locating DLs within first-order logic fragments

- The definition of $\mathfrak{t}(C, \mathrm{x})$ can be optimised to recycle variables and to use only two variables x_0 and x_1.

$$\begin{aligned}
\mathfrak{t}(\exists r.C, \mathrm{x}_i) &\stackrel{\text{def}}{=} \exists\, \mathrm{x}_{1-i}\; r(\mathrm{x}_i, \mathrm{x}_{1-i}) \wedge \mathfrak{t}(C, \mathrm{x}_{1-i}) \\
\mathfrak{t}(\forall r.C, \mathrm{x}_i) &\stackrel{\text{def}}{=} \forall\, \mathrm{x}_{1-i}\; r(\mathrm{x}_i, \mathrm{x}_{1-i}) \Rightarrow \mathfrak{t}(C, \mathrm{x}_{1-i})
\end{aligned}$$

- FO2 (FOL restricted to two individual variables) satisfiability is NEXPTIME-complete.

- Actually, recycling of variables in $\mathfrak{t}(C, \mathrm{x})$ leads to the guarded fragment GF restricted to two variables (GF2), whose satisfiability is EXPTIME-complete.

Locating DLs within first-order logic fragments

- The definition of $\mathfrak{t}(C, \mathtt{x})$ can be optimised to recycle variables and to use only two variables $\mathtt{x}_0$ and $\mathtt{x}_1$.

$$\begin{aligned}
\mathfrak{t}(\exists r.C, \mathtt{x}_i) &\stackrel{\text{def}}{=} \exists\, \mathtt{x}_{1-i}\; r(\mathtt{x}_i, \mathtt{x}_{1-i}) \wedge \mathfrak{t}(C, \mathtt{x}_{1-i}) \\
\mathfrak{t}(\forall r.C, \mathtt{x}_i) &\stackrel{\text{def}}{=} \forall\, \mathtt{x}_{1-i}\; r(\mathtt{x}_i, \mathtt{x}_{1-i}) \Rightarrow \mathfrak{t}(C, \mathtt{x}_{1-i})
\end{aligned}$$

- FO2 (FOL restricted to two individual variables) satisfiability is NEXPTIME-complete.
- Actually, recycling of variables in $\mathfrak{t}(C, \mathtt{x})$ leads to the guarded fragment GF restricted to two variables (GF2), whose satisfiability is EXPTIME-complete.
- Which additional DL features can be translated
  - into FOL?
  - into a decidable fragment of FOL?

More translations into FOL

$$\mathfrak{t}(\exists r^{-}.C, \mathrm{x}) \quad \stackrel{\text{def}}{=} \quad \exists\, \mathrm{y}\; r(\mathrm{y}, \mathrm{x}) \wedge \mathfrak{t}(C, \mathrm{y})$$

More translations into FOL

$$
\begin{aligned}
\mathfrak{t}(\exists r^{-}.C, \mathrm{x}) &\stackrel{\text{def}}{=} \exists\, \mathrm{y}\; r(\mathrm{y}, \mathrm{x}) \wedge \mathfrak{t}(C, \mathrm{y}) \\
\mathfrak{t}(\{a\}, \mathrm{x}) &\stackrel{\text{def}}{=} \;?!? \\
\mathfrak{t}((\geq n\; r \cdot C), \mathrm{x}) &\stackrel{\text{def}}{=} \exists^{\geq n}\, \mathrm{y}\; r(\mathrm{x}, \mathrm{y}) \wedge \mathfrak{t}(C, \mathrm{y}) \\
\mathfrak{t}(\exists r.\mathsf{Self}, \mathrm{x}) &\stackrel{\text{def}}{=} \;?!?
\end{aligned}
$$

More translations into FOL

$$
\begin{array}{lcl}
\mathfrak{t}(\exists r^{-}.C, \mathtt{x}) & \stackrel{\text{def}}{=} & \exists\, \mathtt{y}\; r(\mathtt{y}, \mathtt{x}) \land \mathfrak{t}(C, \mathtt{y}) \\
\mathfrak{t}(\{a\}, \mathtt{x}) & \stackrel{\text{def}}{=} & \;\; ?!? \\
\mathfrak{t}((\geq n\; r \cdot C), \mathtt{x}) & \stackrel{\text{def}}{=} & \exists^{\geq n}\, \mathtt{y}\; r(\mathtt{x}, \mathtt{y}) \land \mathfrak{t}(C, \mathtt{y}) \\
\mathfrak{t}(\exists r.\mathsf{Self}, \mathtt{x}) & \stackrel{\text{def}}{=} & \;\; ?!? \\
\mathfrak{t}(r \sqsubseteq s) & \stackrel{\text{def}}{=} & \forall\, \mathtt{x}, \mathtt{y}\; r(\mathtt{x}, \mathtt{y}) \Rightarrow s(\mathtt{x}, \mathtt{y}) \\
\mathfrak{t}(\mathtt{Trans}(r)) & \stackrel{\text{def}}{=} & \forall \mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3\; r(\mathtt{x}_1, \mathtt{x}_2) \land r(\mathtt{x}_2, \mathtt{x}_3) \Rightarrow r(\mathtt{x}_1, \mathtt{x}_3) \\
\mathfrak{t}(r_1 \circ \cdots \circ r_n \sqsubseteq r) & \stackrel{\text{def}}{=} & \forall \mathtt{x}_1, \cdots, \mathtt{x}_{n+1} \\
 & & r_1(\mathtt{x}_1, \mathtt{x}_2) \land \cdots \land r_n(\mathtt{x}_n, \mathtt{x}_{n+1}) \Rightarrow r(\mathtt{x}_1, \mathtt{x}_{n+1})
\end{array}
$$

$$
\exists^{\geq k}\, \mathtt{x}\; \varphi(\mathtt{x}) \stackrel{\text{def}}{=} \exists\, \mathtt{x}_1, \ldots, \mathtt{x}_k\; \Big(\bigwedge_{i \neq j} \mathtt{x}_i \neq \mathtt{x}_j\Big) \land \bigwedge_i \varphi(\mathtt{x}_i)
$$

Properties About Interpretations

About tree interpretations

- $\mathcal{I}$ is a tree interpretation for C with respect to $\mathcal{T}$ iff the conditions below hold:
  - $\mathbf{t}_{\mathcal{I}} = (\Delta^{\mathcal{I}}, \bigcup_r r^{\mathcal{I}})$ is a tree,
  - the root of $\mathbf{t}_{\mathcal{I}}$ belongs to $C^{\mathcal{I}}$,
  - $\mathcal{I} \models \mathcal{T}$.

About tree interpretations

- $\mathcal{I}$ is a tree interpretation for C with respect to $\mathcal{T}$ iff the conditions below hold:
  - $\mathbf{t}_{\mathcal{I}} = (\Delta^{\mathcal{I}}, \bigcup_r r^{\mathcal{I}})$ is a tree,
  - the root of $\mathbf{t}_{\mathcal{I}}$ belongs to $C^{\mathcal{I}}$,
  - $\mathcal{I} \models \mathcal{T}$.

- $\mathcal{ALC}$ has the tree interpretation property and the finite interpretation property.

About tree interpretations

- $\mathcal{I}$ is a tree interpretation for C with respect to $\mathcal{T}$ iff the conditions below hold:
  - $\mathbf{t}_{\mathcal{I}} = (\Delta^{\mathcal{I}}, \bigcup_r r^{\mathcal{I}})$ is a tree,
  - the root of $\mathbf{t}_{\mathcal{I}}$ belongs to $C^{\mathcal{I}}$,
  - $\mathcal{I} \models \mathcal{T}$.

- $\mathcal{ALC}$ has the tree interpretation property and the finite interpretation property.

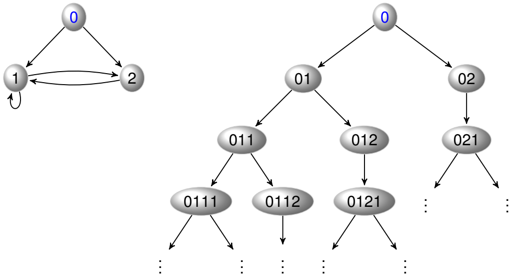
- **Path** in $\mathcal{I}$: finite sequence $(\mathfrak{a}_1, \ldots, \mathfrak{a}_n) \in (\Delta^{\mathcal{I}})^+$ such that for all $i \in [1, n-1]$, we have $(\mathfrak{a}_i, \mathfrak{a}_{i+1}) \in \bigcup_r r^{\mathcal{I}}$.

- $\mathfrak{a}$**-path**: path such that $\mathfrak{a}_1 = \mathfrak{a}$.

Unravelling an interpretation with a single role

- Unravelling of $\mathcal{I}$ at $\mathfrak{a} \in \Delta^{\mathcal{I}}$: $\mathcal{U} = (\Delta^{\mathcal{U}}, \cdot^{\mathcal{U}})$ with
  - $\Delta^{\mathcal{U}}$ is the set of $\mathfrak{a}$-paths in $\mathcal{I}$.
  - For all A, $A^{\mathcal{U}} \stackrel{\text{def}}{=} \{(\mathfrak{a}_1, \ldots, \mathfrak{a}_n) \in \Delta^{\mathcal{U}} \mid \mathfrak{a}_n \in A^{\mathcal{I}}\}$,
  - For all role names r, we have

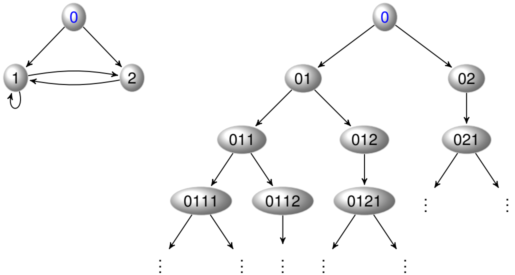
$$r^{\mathcal{U}} \stackrel{\text{def}}{=} \{((\mathfrak{a}_1, \ldots, \mathfrak{a}_n), (\mathfrak{a}_1, \ldots, \mathfrak{a}_n, \mathfrak{a}_{n+1})) \mid (\mathfrak{a}_n, \mathfrak{a}_{n+1}) \in r^{\mathcal{I}}\}$$



Unravelling an interpretation with a single role

- Unravelling of $\mathcal{I}$ at $\mathfrak{a} \in \Delta^{\mathcal{I}}$: $\mathcal{U} = (\Delta^{\mathcal{U}}, \cdot^{\mathcal{U}})$ with
  - $\Delta^{\mathcal{U}}$ is the set of $\mathfrak{a}$-paths in $\mathcal{I}$.
  - For all A, $A^{\mathcal{U}} \overset{\text{def}}{=} \{(\mathfrak{a}_1, \ldots, \mathfrak{a}_n) \in \Delta^{\mathcal{U}} \mid \mathfrak{a}_n \in A^{\mathcal{I}}\}$,
  - For all role names r, we have

$$r^{\mathcal{U}} \overset{\text{def}}{=} \{((\mathfrak{a}_1, \ldots, \mathfrak{a}_n), (\mathfrak{a}_1, \ldots, \mathfrak{a}_n, \mathfrak{a}_{n+1})) \mid (\mathfrak{a}_n, \mathfrak{a}_{n+1}) \in r^{\mathcal{I}}\}$$



- C is satisfiable with respect to a TBox $\mathcal{T}$ implies C has a tree interpretation with respect to $\mathcal{T}$.

Subconcepts (towards small interpretation pro

- Set of **subconcepts** $\mathsf{sub}(C)$ and **size** $\mathsf{size}(C)$:

Concept C	$\mathsf{sub}(C)$	$\mathsf{size}(C)$
A	$\{A\}$	1
$\top/\bot$	$\{\top\}/\{\bot\}$	1
$\neg C_1$	$\mathsf{sub}(C_1) \cup \{\neg C_1\}$	$1 + \mathsf{size}(C_1)$
$C_1 \sqcap C_2$	$\mathsf{sub}(C_1) \cup \mathsf{sub}(C_2) \cup \{C_1 \sqcap C_2\}$	$1 + \mathsf{size}(C_1) + \mathsf{size}(C_2)$
$C_1 \sqcup C_2$	$\mathsf{sub}(C_1) \cup \mathsf{sub}(C_2) \cup \{C_1 \sqcup C_2\}$	$1 + \mathsf{size}(C_1) + \mathsf{size}(C_2)$
$\exists r.C_1$	$\mathsf{sub}(C_1) \cup \{\exists r.C_1\}$	$1 + \mathsf{size}(C_1)$
$\forall r.C_1$	$\mathsf{sub}(C_1) \cup \{\forall r.C_1\}$	$1 + \mathsf{size}(C_1)$

Subconcepts (towards small interpretation pro

- Set of **subconcepts** $\text{sub}(C)$ and **size** $\text{size}(C)$:

Concept C	$\text{sub}(C)$	$\text{size}(C)$
A	$\{A\}$	1
$\top/\bot$	$\{\top\}/\{\bot\}$	1
$\neg C_1$	$\text{sub}(C_1) \cup \{\neg C_1\}$	$1 + \text{size}(C_1)$
$C_1 \sqcap C_2$	$\text{sub}(C_1) \cup \text{sub}(C_2) \cup \{C_1 \sqcap C_2\}$	$1 + \text{size}(C_1) + \text{size}(C_2)$
$C_1 \sqcup C_2$	$\text{sub}(C_1) \cup \text{sub}(C_2) \cup \{C_1 \sqcup C_2\}$	$1 + \text{size}(C_1) + \text{size}(C_2)$
$\exists r.C_1$	$\text{sub}(C_1) \cup \{\exists r.C_1\}$	$1 + \text{size}(C_1)$
$\forall r.C_1$	$\text{sub}(C_1) \cup \{\forall r.C_1\}$	$1 + \text{size}(C_1)$

$$\text{sub}(\mathcal{T}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \text{sub}(C) \cup \text{sub}(D) \qquad \text{sub}(\mathcal{A}) = \bigcup_{a:C \in \mathcal{A}} \text{sub}(C)$$

- $\text{size}(\mathcal{T})$, $\text{size}(\mathcal{A})$: sum of the sizes of its elements.

Subconcepts (towards small interpretation pro

- Set of **subconcepts** $\mathrm{sub}(C)$ and **size** $\mathrm{size}(C)$:

Concept C	$\mathrm{sub}(C)$	$\mathrm{size}(C)$
A	$\{A\}$	1
$\top/\bot$	$\{\top\}/\{\bot\}$	1
$\neg C_1$	$\mathrm{sub}(C_1) \cup \{\neg C_1\}$	$1 + \mathrm{size}(C_1)$
$C_1 \sqcap C_2$	$\mathrm{sub}(C_1) \cup \mathrm{sub}(C_2) \cup \{C_1 \sqcap C_2\}$	$1 + \mathrm{size}(C_1) + \mathrm{size}(C_2)$
$C_1 \sqcup C_2$	$\mathrm{sub}(C_1) \cup \mathrm{sub}(C_2) \cup \{C_1 \sqcup C_2\}$	$1 + \mathrm{size}(C_1) + \mathrm{size}(C_2)$
$\exists r.C_1$	$\mathrm{sub}(C_1) \cup \{\exists r.C_1\}$	$1 + \mathrm{size}(C_1)$
$\forall r.C_1$	$\mathrm{sub}(C_1) \cup \{\forall r.C_1\}$	$1 + \mathrm{size}(C_1)$

$$\mathrm{sub}(\mathcal{T}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathrm{sub}(C) \cup \mathrm{sub}(D) \qquad \mathrm{sub}(\mathcal{A}) = \bigcup_{a:C \in \mathcal{A}} \mathrm{sub}(C)$$

- $\mathrm{size}(\mathcal{T})$, $\mathrm{size}(\mathcal{A})$: sum of the sizes of its elements.
- $\mathrm{card}(\mathrm{sub}(\mathcal{T}) \cup \mathrm{sub}(\mathcal{A})) \leq \mathrm{size}(\mathcal{T}) + \mathrm{size}(\mathcal{A})$.

Type

- A set of concepts X is **closed under subconcepts** iff $\text{sub}(X) = X$.
- $\text{sub}(C)$, $\text{sub}(\mathcal{T})$, $\text{sub}(\mathcal{A})$ are closed under subconcepts.

Type

- A set of concepts X is **closed under subconcepts** iff $\mathrm{sub}(X) = X$.
- $\mathrm{sub}(C)$, $\mathrm{sub}(\mathcal{T})$, $\mathrm{sub}(\mathcal{A})$ are closed under subconcepts.
- X**-type** of $\mathfrak{a}$ in $\mathcal{I}$:

$$\mathrm{type}_X(\mathfrak{a}) \stackrel{\text{def}}{=} \{C \in X \mid \mathfrak{a} \in C^{\mathcal{I}}\}$$

- If X is finite, then

$$\mathrm{card}(\{\mathrm{type}_X(\mathfrak{a}) \mid \mathfrak{a} \in \Delta^{\mathcal{I}}\}) \leq 2^{\mathrm{card}(X)}$$

Type

- A set of concepts X is **closed under subconcepts** iff $\mathsf{sub}(X) = X$.
- $\mathsf{sub}(C)$, $\mathsf{sub}(\mathcal{T})$, $\mathsf{sub}(\mathcal{A})$ are closed under subconcepts.
- X**-type** of $\mathfrak{a}$ in $\mathcal{I}$:

$$\mathsf{type}_X(\mathfrak{a}) \stackrel{\mathsf{def}}{=} \{C \in X \mid \mathfrak{a} \in C^{\mathcal{I}}\}$$

- If X is finite, then

$$\mathsf{card}(\{\mathsf{type}_X(\mathfrak{a}) \mid \mathfrak{a} \in \Delta^{\mathcal{I}}\}) \leq 2^{\mathsf{card}(X)}$$

- Small interpretation property is established by showing that no need to keep too many individuals with the same X-type with $X = \mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A})$.

The filtration technique

- Set X closed under subconcepts, interpretation $\mathcal{I}$.
- $\mathfrak{a} \approx_X \mathfrak{b} \stackrel{\text{def}}{\Leftrightarrow} \text{type}_X(\mathfrak{a}) = \text{type}_X(\mathfrak{b})$, and equivalence class $[\mathfrak{a}]_X$.

The filtration technique

- Set X closed under subconcepts, interpretation $\mathcal{I}$.

- $\mathfrak{a} \approx_X \mathfrak{b} \overset{\text{def}}{\Leftrightarrow} \text{type}_X(\mathfrak{a}) = \text{type}_X(\mathfrak{b})$, and equivalence class $[\mathfrak{a}]_X$.

- ***X*-filtration** $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$: (quotient structure)
  - $\Delta^{\mathcal{J}} \overset{\text{def}}{=} \{[\mathfrak{a}]_X \mid \mathfrak{a} \in \Delta^{\mathcal{I}}\}$.
  - $A^{\mathcal{J}} \overset{\text{def}}{=} \{[\mathfrak{a}]_X \mid$ there is $\mathfrak{a}' \in [\mathfrak{a}]_X$ such that $\mathfrak{a}' \in A^{\mathcal{I}}\}$. $(A \in X)$
  - $r^{\mathcal{J}} \overset{\text{def}}{=} \{([\mathfrak{a}]_X, [\mathfrak{b}]_X) \mid$ there is $\mathfrak{a}' \in [\mathfrak{a}]_X, \mathfrak{b}' \in [\mathfrak{b}]_X$ such that $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}\}$. (Is $\mathcal{J}$ well-defined?)

- $\text{card}(\Delta^{\mathcal{J}}) \leq 2^{\text{card}(X)}$.

The filtration technique

- Set X closed under subconcepts, interpretation $\mathcal{I}$.

- $\mathfrak{a} \approx_X \mathfrak{b} \overset{\text{def}}{\Leftrightarrow} \text{type}_X(\mathfrak{a}) = \text{type}_X(\mathfrak{b})$, and equivalence class $[\mathfrak{a}]_X$.

- ***X*-filtration** $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$: (quotient structure)
  - $\Delta^{\mathcal{J}} \overset{\text{def}}{=} \{[\mathfrak{a}]_X \mid \mathfrak{a} \in \Delta^{\mathcal{I}}\}$.
  - $A^{\mathcal{J}} \overset{\text{def}}{=} \{[\mathfrak{a}]_X \mid$ there is $\mathfrak{a}' \in [\mathfrak{a}]_X$ such that $\mathfrak{a}' \in A^{\mathcal{I}}\}$. $(A \in X)$
  - $r^{\mathcal{J}} \overset{\text{def}}{=} \{([\mathfrak{a}]_X, [\mathfrak{b}]_X) \mid$ there is $\mathfrak{a}' \in [\mathfrak{a}]_X, \mathfrak{b}' \in [\mathfrak{b}]_X$ such that $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}\}$. (Is $\mathcal{J}$ well-defined?)

- $\text{card}(\Delta^{\mathcal{J}}) \leq 2^{\text{card}(X)}$.

- For all $C \in X$ and $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we have $\mathfrak{a} \in C^{\mathcal{I}}$ iff $[\mathfrak{a}]_X \in C^{\mathcal{J}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

- Suppose that $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, there is $[\mathfrak{b}]_X$ such that $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$ and $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

- Suppose that $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, there is $[\mathfrak{b}]_X$ such that $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$ and $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $r^{\mathcal{J}}$, there is $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}$ such that $\mathfrak{a}' \in [\mathfrak{a}]_X$ and $\mathfrak{b}' \in [\mathfrak{b}]_X$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

- Suppose that $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, there is $[\mathfrak{b}]_X$ such that $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$ and $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $r^{\mathcal{J}}$, there is $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}$ such that $\mathfrak{a}' \in [\mathfrak{a}]_X$ and $\mathfrak{b}' \in [\mathfrak{b}]_X$.
  - As X closed under subconcepts and by (IH), $\mathfrak{b} \in C^{\mathcal{I}}$ and as $\mathfrak{b}' \in [\mathfrak{b}]_X$, $\mathfrak{b}' \in C^{\mathcal{I}}$ too.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

- Suppose that $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, there is $[\mathfrak{b}]_X$ such that $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$ and $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $r^{\mathcal{J}}$, there is $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}$ such that $\mathfrak{a}' \in [\mathfrak{a}]_X$ and $\mathfrak{b}' \in [\mathfrak{b}]_X$.
  - As X closed under subconcepts and by (IH), $\mathfrak{b} \in C^{\mathcal{I}}$ and as $\mathfrak{b}' \in [\mathfrak{b}]_X$, $\mathfrak{b}' \in C^{\mathcal{I}}$ too.
  - By definition of $\cdot^{\mathcal{I}}$, we get $\mathfrak{a}' \in (\exists r.C)^{\mathcal{I}}$.

Induction step for $\exists r.C \in X$

- First, suppose that $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.
  - By definition of $\cdot^{\mathcal{I}}$, there is $\mathfrak{b}$ such that $(\mathfrak{a}, \mathfrak{b}) \in r^{\mathcal{I}}$ and $\mathfrak{b} \in C^{\mathcal{I}}$.
  - By definition of $\mathcal{J}$, $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$.
  - As X closed under subconcepts and by (IH), $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, we get $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.

- Suppose that $[\mathfrak{a}]_X \in (\exists r.C)^{\mathcal{J}}$.
  - By definition of $\cdot^{\mathcal{J}}$, there is $[\mathfrak{b}]_X$ such that $([\mathfrak{a}]_X, [\mathfrak{b}]_X) \in r^{\mathcal{J}}$ and $[\mathfrak{b}]_X \in C^{\mathcal{J}}$.
  - By definition of $r^{\mathcal{J}}$, there is $(\mathfrak{a}', \mathfrak{b}') \in r^{\mathcal{I}}$ such that $\mathfrak{a}' \in [\mathfrak{a}]_X$ and $\mathfrak{b}' \in [\mathfrak{b}]_X$.
  - As X closed under subconcepts and by (IH), $\mathfrak{b} \in C^{\mathcal{I}}$ and as $\mathfrak{b}' \in [\mathfrak{b}]_X$, $\mathfrak{b}' \in C^{\mathcal{I}}$ too.
  - By definition of $\cdot^{\mathcal{I}}$, we get $\mathfrak{a}' \in (\exists r.C)^{\mathcal{I}}$.
  - As $\mathfrak{a}' \in [\mathfrak{a}]_X$, $\mathfrak{a} \in (\exists r.C)^{\mathcal{I}}$.

Small interpretation property

- C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, X closed under subconcepts with $(\mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A})) \subseteq X$.

Small interpretation property

- C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, X closed under subconcepts with $(\text{sub}(C) \cup \text{sub}(\mathcal{T}) \cup \text{sub}(\mathcal{A})) \subseteq X$.
- If C is satisfiable w.r.t. $\mathcal{K}$, then there is an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{K}$, $C^{\mathcal{J}} \neq \emptyset$ and $\text{card}(\Delta^{\mathcal{J}}) \leq 2^{\text{card}(X)}$.

Small interpretation property

- C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, X closed under subconcepts with $(\mathrm{sub}(C) \cup \mathrm{sub}(\mathcal{T}) \cup \mathrm{sub}(\mathcal{A})) \subseteq X$.
- If C is satisfiable w.r.t. $\mathcal{K}$, then there is an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{K}$, $C^{\mathcal{J}} \neq \emptyset$ and $\mathrm{card}(\Delta^{\mathcal{J}}) \leq 2^{\mathrm{card}(X)}$.
- $\mathcal{J}$ is an X-filtration based on some interpretation $\mathcal{I}$ such that for all $a : C \in \mathcal{A}$, we have $a^{\mathcal{J}} \stackrel{\text{def}}{=} [a^{\mathcal{I}}]_X$.
- The equivalence "$\mathfrak{a} \in C^{\mathcal{I}}$ iff $[\mathfrak{a}]_X \in C^{\mathcal{J}}$" leads to the satisfaction of $\mathcal{K}$ in $\mathcal{J}$. (to be checked)

Small interpretation property

- C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, X closed under subconcepts with $(\text{sub}(C) \cup \text{sub}(\mathcal{T}) \cup \text{sub}(\mathcal{A})) \subseteq X$.
- If C is satisfiable w.r.t. $\mathcal{K}$, then there is an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{K}$, $C^{\mathcal{J}} \neq \emptyset$ and $\text{card}(\Delta^{\mathcal{J}}) \leq 2^{\text{card}(X)}$.
- $\mathcal{J}$ is an X-filtration based on some interpretation $\mathcal{I}$ such that for all $a : C \in \mathcal{A}$, we have $a^{\mathcal{J}} \stackrel{\text{def}}{=} [a^{\mathcal{I}}]_X$.
- The equivalence "$\mathfrak{a} \in C^{\mathcal{I}}$ iff $[\mathfrak{a}]_X \in C^{\mathcal{J}}$" leads to the satisfaction of $\mathcal{K}$ in $\mathcal{J}$. (to be checked)
- $\mathcal{K}$ consistent implies there is $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $\text{card}(\Delta^{\mathcal{I}}) \leq 2^{\text{size}(\mathcal{K})}$.

Generalities about decision procedures

- A **decision problem** P is a subset of Σ^*.
  (Σ is a finite alphabet)

- Alternatively, given $\mathfrak{w} \in \Sigma^*$, is $\mathfrak{w}$ in P?

Generalities about decision procedures

- A **decision problem** P is a subset of Σ^*.
  (Σ is a finite alphabet)

- Alternatively, given $\mathfrak{w} \in \Sigma^*$, is $\mathfrak{w}$ in P?

- An algorithm for P is **sound** if whenever it answers "$\mathfrak{w} \in P$", then $\mathfrak{w} \in P$.

- An algorithm for P is **complete** if whenever $\mathfrak{w} \in P$, it answers "$\mathfrak{w} \in P$".

- An algorithm for P is **terminating** if it stops after finitely many steps for all $\mathfrak{w} \in \Sigma^*$.

Generalities about decision procedures

- A **decision problem** P is a subset of Σ^*.
  (Σ is a finite alphabet)
- Alternatively, given $\mathfrak{w} \in \Sigma^*$, is $\mathfrak{w}$ in P?
- An algorithm for P is **sound** if whenever it answers "$\mathfrak{w} \in P$", then $\mathfrak{w} \in P$.
- An algorithm for P is **complete** if whenever $\mathfrak{w} \in P$, it answers "$\mathfrak{w} \in P$".
- An algorithm for P is **terminating** if it stops after finitely many steps for all $\mathfrak{w} \in \Sigma^*$.
- **Decision procedure**: sound, complete and terminating.

A brute force decision procedure

- Concept consistency with respect to a knowledge base.
- Input: C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

A brute force decision procedure

- Concept consistency with respect to a knowledge base.
- Input: C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.
- Guess an interpretation $\mathcal{I}$ such that $\mathrm{card}(\Delta^{\mathcal{I}}) \leq 2^{\mathrm{card}(X)}$ with $X = \mathrm{sub}(C) \cup \mathrm{sub}(\mathcal{T}) \cup \mathrm{sub}(\mathcal{A})$.

A brute force decision procedure

- Concept consistency with respect to a knowledge base.
- Input: C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.
- Guess an interpretation $\mathcal{I}$ such that $\mathrm{card}(\Delta^{\mathcal{I}}) \leq 2^{\mathrm{card}(X)}$ with $X = \mathrm{sub}(C) \cup \mathrm{sub}(\mathcal{T}) \cup \mathrm{sub}(\mathcal{A})$.
- Compute the set $C^{\mathcal{I}}$ using a labelling algorithm based on the definition of $\cdot^{\mathcal{I}}$ for complex concepts.

A brute force decision procedure

- Concept consistency with respect to a knowledge base.
- Input: C, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.
- Guess an interpretation $\mathcal{I}$ such that $\mathrm{card}(\Delta^{\mathcal{I}}) \leq 2^{\mathrm{card}(X)}$ with $X = \mathrm{sub}(C) \cup \mathrm{sub}(\mathcal{T}) \cup \mathrm{sub}(\mathcal{A})$.
- Compute the set $C^{\mathcal{I}}$ using a labelling algorithm based on the definition of $\cdot^{\mathcal{I}}$ for complex concepts.
- Check the satisfaction of $\mathcal{I} \models \mathcal{K}$ using again a labelling algorithm.
- Guessing $\mathcal{I}$ and checking $C^{\mathcal{I}} \neq \emptyset$ and $\mathcal{I} \models \mathcal{K}$ can be done in NEXPTIME.

Labelling algorithm

- Given $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with finite $\Delta^{\mathcal{I}}$ and a concept C, let us compute $C^{\mathcal{I}}$ in polynomial time in size$(\mathcal{I})$ + size(C).

Labelling algorithm

- Given $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with finite $\Delta^{\mathcal{I}}$ and a concept C, let us compute $C^{\mathcal{I}}$ in polynomial time in size($\mathcal{I}$) + size(C).
- Let $C_1, \ldots, C_k$ be the subconcepts of C ordered by increasing size.
- In case of conflict, we make an arbitrary choice for the formulae of identical sizes.
  - $C_k = C$,
  - C_1 is a concept name (or $\perp/\top$),
  - if C_i is a strict subconcept of C_j, then $i < j$,
  - $k \leq$ size(C).

Labelling algorithm

- Given $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with finite $\Delta^{\mathcal{I}}$ and a concept C, let us compute $C^{\mathcal{I}}$ in polynomial time in $\text{size}(\mathcal{I}) + \text{size}(C)$.

- Let $C_1, \ldots, C_k$ be the subconcepts of C ordered by increasing size.

- In case of conflict, we make an arbitrary choice for the formulae of identical sizes.
  - $C_k = C$,
  - C_1 is a concept name (or $\bot/\top$),
  - if C_i is a strict subconcept of C_j, then $i < j$,
  - $k \leq \text{size}(C)$.

- **Labelling algorithm**: construction of **label** $l(\mathfrak{a})$ for each $\mathfrak{a}$.

Labelling algorithm: principles

- For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we build a set of concepts $l(\mathfrak{a})$ such that
  1. for every $i \in [1, k]$, either $C_i \in l(\mathfrak{a})$, or $\neg C_i \in l(\mathfrak{a})$, but not both at the same time,
  2. for every $D \in \{C_1, \ldots, C_k, \neg C_1, \ldots, \neg C_k\}$, $D \in l(\mathfrak{a})$ iff $\mathfrak{a} \in D^{\mathcal{I}}$.

Labelling algorithm: principles

- For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we build a set of concepts $l(\mathfrak{a})$ such that
  1. for every $i \in [1, k]$, either $C_i \in l(\mathfrak{a})$, or $\neg C_i \in l(\mathfrak{a})$, but not both at the same time,
  2. for every $D \in \{C_1, \ldots, C_k, \neg C_1, \ldots, \neg C_k\}$, $D \in l(\mathfrak{a})$ iff $\mathfrak{a} \in D^{\mathcal{I}}$.
- For all $i \in [1, k]$ and $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we insert either C_i in $l(\mathfrak{a})$ or $\neg C_i$ dans $l(\mathfrak{a})$, following the above ordering of subformulae.
- For each $i \in [1, k]$, the insertion requires time quadratic in size($\mathcal{I}$) (worst-case).

Labelling algorithm: principles

- For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we build a set of concepts $l(\mathfrak{a})$ such that
  1. for every $i \in [1, k]$, either $C_i \in l(\mathfrak{a})$, or $\neg C_i \in l(\mathfrak{a})$, but not both at the same time,
  2. for every $D \in \{C_1, \ldots, C_k, \neg C_1, \ldots, \neg C_k\}$, $D \in l(\mathfrak{a})$ iff $\mathfrak{a} \in D^{\mathcal{I}}$.
- For all $i \in [1, k]$ and $\mathfrak{a} \in \Delta^{\mathcal{I}}$, we insert either C_i in $l(\mathfrak{a})$ or $\neg C_i$ dans $l(\mathfrak{a})$, following the above ordering of subformulae.
- For each $i \in [1, k]$, the insertion requires time quadratic in size($\mathcal{I}$) (worst-case).
- Each set $l(\mathfrak{a})$ is initialized to the empty set.

Labelling algorithm: definition

We run the algorithm with increasing $i \in [1, k]$.

Case 1: C_i is a concept name.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, if $\mathfrak{a} \in C_i^{\mathcal{I}}$ by definition of $\mathcal{I}$, then insert C_i in $l(\mathfrak{a})$ otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

Labelling algorithm: definition

We run the algorithm with increasing $i \in [1, k]$.

Case 1: C_i is a concept name.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, if $\mathfrak{a} \in C_i^{\mathcal{I}}$ by definition of $\mathcal{I}$, then insert C_i in $l(\mathfrak{a})$ otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

Case 2: $C_i = C_{i_1} \sqcap C_{i_2}$ for some $i_1, i_2 < i$.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, insert C_i in $l(\mathfrak{a})$ if $\{C_{i_1}, C_{i_2}\} \subseteq l(\mathfrak{a})$ otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

Labelling algorithm: definition

We run the algorithm with increasing $i \in [1, k]$.

Case 1: C_i is a concept name.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, if $\mathfrak{a} \in C_i^{\mathcal{I}}$ by definition of $\mathcal{I}$, then insert C_i in $l(\mathfrak{a})$ otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

Case 2: $C_i = C_{i_1} \sqcap C_{i_2}$ for some $i_1, i_2 < i$.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, insert C_i in $l(\mathfrak{a})$ if $\{C_{i_1}, C_{i_2}\} \subseteq l(\mathfrak{a})$ otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

Case 3: $C_i = \exists r.C_{i_1}$ for some $i_1 < i$.
For every $\mathfrak{a} \in \Delta^{\mathcal{I}}$, if there is $\mathfrak{a}' \in r^{\mathcal{I}}(\mathfrak{a})$ such that C_{i_1} is in $l(\mathfrak{a}')$, then insert C_i in $l(\mathfrak{a})$, otherwise insert $\neg C_i$ in $l(\mathfrak{a})$.

This is a standard labelling algorithm as used for instance for the branching-time temporal logic CTL.

Tableaux Proof System for $\mathcal{ALC}$

Automated reasoning for non-classical logics

- Direct methods:
  - Analytical calculi: tableaux, sequents ($\Gamma \vdash \Delta$), etc.
  - Resolution. ($\frac{L \vee C_1 \quad \neg L' \vee C_2}{C_1\sigma \vee C_2\sigma}$ with $L\sigma = L'\sigma$)
  - Automata-based decision procedures. (φ sat. iff $\mathrm{L}(\mathbb{A}_\varphi) \neq \emptyset$)

Automated reasoning for non-classical logics

- Direct methods:
  - Analytical calculi: tableaux, sequents ($\Gamma \vdash \Delta$), etc.
  - Resolution. $\left(\frac{L \vee C_1 \quad \neg L' \vee C_2}{C_1\sigma \vee C_2\sigma}\right.$ with $L\sigma = L'\sigma$)
  - Automata-based decision procedures. (φ sat. iff $\mathrm{L}(\mathbb{A}_\varphi) \neq \emptyset$)
- Translation into
  - other modal logics (PDL, modal μ-calculus, ...)
  - decidable fragments of first-order logic (FO2, GF, CGF, ...)
  - second-order monadic logics (S2S, SωS, ...)

Tableaux-based proof systems

- Analytical method: the rules of the calculi perform a syntactic decomposition of the concepts (formulae, etc.).

Tableaux-based proof systems

- Analytical method: the rules of the calculi perform a syntactic decomposition of the concepts (formulae, etc.).
- Method developped initially by R. Smullyan for first-order logic (60's).
- Method developped by M. Fitting for intuitionistic and modal logics (circa 1975-1985).

Tableaux-based proof systems

- Analytical method: the rules of the calculi perform a syntactic decomposition of the concepts (formulae, etc.).
- Method developped initially by R. Smullyan for first-order logic (60's).
- Method developped by M. Fitting for intuitionistic and modal logics (circa 1975-1985).
- Close relationships with sequent-style proof systems.

Tableaux-based proof systems

- Analytical method: the rules of the calculi perform a syntactic decomposition of the concepts (formulae, etc.).
- Method developped initially by R. Smullyan for first-order logic (60's).
- Method developped by M. Fitting for intuitionistic and modal logics (circa 1975-1985).
- Close relationships with sequent-style proof systems.
- Modular approach as new conditions or new ingredients may correspond to the addition of new rules.

Tableaux-based proof systems

- Analytical method: the rules of the calculi perform a syntactic decomposition of the concepts (formulae, etc.).
- Method developped initially by R. Smullyan for first-order logic (60's).
- Method developped by M. Fitting for intuitionistic and modal logics (circa 1975-1985).
- Close relationships with sequent-style proof systems.
- Modular approach as new conditions or new ingredients may correspond to the addition of new rules.
- Labels are sometimes used in such proof systems.
  - Labels are interpreted as entities of the domain under construction.
  - Expressions external to the original logical language.
  - Labels can be also used as control data structures.

Methodology

- Satisfiability of C by checking consistency of $\mathcal{A} = \{a : C\}$.

Methodology

- Satisfiability of C by checking consistency of $\mathcal{A} = \{a : C\}$.
- First part dedicated to a tableaux calculus for checking the consistency status of ABoxes $\mathcal{A}$.

Methodology

- Satisfiability of C by checking consistency of $\mathcal{A} = \{a : C\}$.

- First part dedicated to a tableaux calculus for checking the consistency status of ABoxes $\mathcal{A}$.

- Second part presents an extension for checking the consistency status of a knowledge base $(\mathcal{T}, \mathcal{A})$.

Methodology

- Satisfiability of C by checking consistency of $\mathcal{A} = \{a : C\}$.
- First part dedicated to a tableaux calculus for checking the consistency status of ABoxes $\mathcal{A}$.
- Second part presents an extension for checking the consistency status of a knowledge base $(\mathcal{T}, \mathcal{A})$.
- This leads to a decision procedure (terminating) for knowledge base consistency in which a few optimisations leads to EXPTIME (not presented herein).

Methodology

- Satisfiability of C by checking consistency of $\mathcal{A} = \{a : C\}$.
- First part dedicated to a tableaux calculus for checking the consistency status of ABoxes $\mathcal{A}$.
- Second part presents an extension for checking the consistency status of a knowledge base $(\mathcal{T}, \mathcal{A})$.
- This leads to a decision procedure (terminating) for knowledge base consistency in which a few optimisations leads to ExpTime (not presented herein).
- The proof system works by rewriting ABoxes. $(\mathcal{A} \xrightarrow{*} \mathcal{A}')$
- Our presentation follows the usual way to present tableaux-style proof systems for description logics.

Negation normal form

- C is in **negation normal form (NNF)** $\stackrel{\text{def}}{\Leftrightarrow}$ the concept negation $\neg$ occurs only in front of concept names.
- Every concept has an equivalent concept in NNF:

$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D \qquad \neg(C \sqcap D) \equiv \neg C \sqcup \neg D$$

$$\neg\exists r.C \equiv \forall r.\neg C \qquad \neg\forall r.C \equiv \exists r.\neg C \qquad \neg\neg C \equiv C \qquad \neg\top \equiv \bot \qquad \neg\bot \equiv \top$$

Negation normal form

- C is in **negation normal form (NNF)** $\overset{\text{def}}{\Leftrightarrow}$ the concept negation $\neg$ occurs only in front of concept names.

- Every concept has an equivalent concept in NNF:

$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D \qquad \neg(C \sqcap D) \equiv \neg C \sqcup \neg D$$

$$\neg \exists r.C \equiv \forall r.\neg C \quad \neg \forall r.C \equiv \exists r.\neg C \quad \neg\neg C \equiv C \quad \neg \top \equiv \bot \quad \neg \bot \equiv \top$$

- Transforming a concept into an equivalent concept in NNF takes polynomial time (only).

(each connective has it

- NNFs are not a must but this simplifies forthcoming developments.

Principles of the tableaux-style proof systems

- The calculus is made of rewriting rules that transform an ABox $\mathcal{A}$ into another ABox $\mathcal{A}'$ nondeterministically.
  (ex: $\sqcap$-rule)

- ABoxes $\mathcal{A}$ are understood as partial description of interpretations.

  (individual name

- To guarantee termination, provisos are added to the application of the rules. (see the blocking technique)

Principles of the tableaux-style proof systems

- The calculus is made of rewriting rules that transform an ABox $\mathcal{A}$ into another ABox $\mathcal{A}'$ nondeterministically.
  (ex: $\sqcap$-rule)

- ABoxes $\mathcal{A}$ are understood as partial description of interpretations.

  (individual name

- To guarantee termination, provisos are added to the application of the rules. (see the blocking technique)

- ABoxes with no contradiction and for which no rule application adds value correspond to interpretations.

Example: the $\sqcap$-rule

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

- Applying the rule can be viewed as repairing a defect.

Example: the $\sqcap$-rule

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

- Applying the rule can be viewed as repairing a defect.
- Condition $\{a : C, a : D\} \not\subseteq \mathcal{A}$ rules out void rule applications.

Example: the $\sqcap$-rule

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

- Applying the rule can be viewed as repairing a defect.
- Condition $\{a : C, a : D\} \not\subseteq \mathcal{A}$ rules out void rule applications.
- The other rules are designed on the same pattern.

(gue $\sqcup$-rule)

Expansion rules for $\mathcal{ALC}$ ABox consistency

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

$\sqcup$-rule: If $a : C \sqcup D \in \mathcal{A}$ and $\{a : C, a : D\} \cap \mathcal{A} = \emptyset$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : E\} \qquad \text{for some } E \in \{C, D\}$$

Expansion rules for $\mathcal{ALC}$ ABox consistency

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

$\sqcup$-rule: If $a : C \sqcup D \in \mathcal{A}$ and $\{a : C, a : D\} \cap \mathcal{A} = \emptyset$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : E\} \qquad \text{for some } E \in \{C, D\}$$

$\exists$-rule: If $a : \exists r.C \in \mathcal{A}$ and there is no b such that $\{(a, b) : r, b : C\} \subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a, c) : r, c : C\} \qquad \text{where } c \text{ is fresh}$$

Expansion rules for $\mathcal{ALC}$ ABox consistency

$\sqcap$-rule: If $a : C \sqcap D \in \mathcal{A}$ and $\{a : C, a : D\} \not\subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C, a : D\}$$

$\sqcup$-rule: If $a : C \sqcup D \in \mathcal{A}$ and $\{a : C, a : D\} \cap \mathcal{A} = \emptyset$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : E\} \qquad \text{for some } E \in \{C, D\}$$

$\exists$-rule: If $a : \exists r.C \in \mathcal{A}$ and there is no b such that $\{(a, b) : r, b : C\} \subseteq \mathcal{A}$ then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a, c) : r, c : C\} \qquad \text{where } c \text{ is fresh}$$

$\forall$-rule: If $\{(a, b) : r, a : \forall r.C\} \subseteq \mathcal{A}$ and $b : C \notin \mathcal{A}$, then

$$\mathcal{A} \longrightarrow \mathcal{A} \cup \{b : C\}$$

Complete and clash-free ABox

- An ABox $\mathcal{A}$ contains a **clash** if $\{a : A, a : \neg A\} \subseteq \mathcal{A}$ or $a : \perp \in \mathcal{A}$.

⚠ Notion of clash to be extended if the concepts are not in NNF.

- An ABox $\mathcal{A}$ is **clash-free** if it contains no clash.

Complete and clash-free ABox

- An ABox $\mathcal{A}$ contains a **clash** if $\{a : A, a : \neg A\} \subseteq \mathcal{A}$ or $a : \perp \in \mathcal{A}$.

⚠ Notion of clash to be extended if the concepts are not in NNF.

- An ABox $\mathcal{A}$ is **clash-free** if it contains no clash.

- An ABox $\mathcal{A}$ is **complete** if it contains a clash or if no rule is applicable (saturation).

Complete and clash-free ABox

- An ABox $\mathcal{A}$ contains a **clash** if $\{a : A, a : \neg A\} \subseteq \mathcal{A}$ or $a : \perp \in \mathcal{A}$.

⚠ Notion of clash to be extended if the concepts are not in NNF.

- An ABox $\mathcal{A}$ is **clash-free** if it contains no clash.

- An ABox $\mathcal{A}$ is **complete** if it contains a clash or if no rule is applicable (saturation).

Objective: to show that $\mathcal{A}$ is consistent iff $\mathcal{A} \xrightarrow{*} \mathcal{A}'$ for some complete and clash-free ABox $\mathcal{A}'$.

- The only nondeterministic rule is the $\sqcup$-rule. (is it true?)

Conclusion

- Today lecture:
  - Translation into first-order logic.
  - Basic model-theoretical properties.
  - Tableaux proof system for $\mathcal{ALC}$ (Part I).
- Next week lecture: Tableaux proof system for $\mathcal{ALC}$