Logics and Statistics for Language Modeling

Carlos Areces

areces@loria.fr
http://www.loria.fr/~areces/ls
INRIA Nancy Grand Est
Nancy, France

2009/2010

Today's Program

Today's Program

- Description Logics
- History and Applications
- Syntax and Semantics
- The Tableaux Method

Description Logics

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.
- They originate from the Quillian's Semantic Networks and Minsky's Frame paradigm.
- Their main characteristics are:
  - A simple to use language (an extension of the propositional language, without variables);

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.
- They originate from the Quillian's Semantic Networks and Minsky's Frame paradigm.
- Their main characteristics are:
  - A simple to use language (an extension of the propositional language, without variables);
  - But that includes a notion of quantification (guarded quantification);

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.
- They originate from the Quillian's Semantic Networks and Minsky's Frame paradigm.
- Their main characteristics are:
  - A simple to use language (an extension of the propositional language, without variables);
  - But that includes a notion of quantification (guarded quantification);
  - With special operators chosen to facilitate the enunciation of definitions;

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.
- They originate from the Quillian's Semantic Networks and Minsky's Frame paradigm.
- Their main characteristics are:
  - A simple to use language (an extension of the propositional language, without variables);
  - But that includes a notion of quantification (guarded quantification);
  - With special operators chosen to facilitate the enunciation of definitions;
  - With a good balance between expressivity and tractability;

Description Logics

- Description Logics (DL) are formal languages which are specially tailored for knowledge representation.
- They originate from the Quillian's Semantic Networks and Minsky's Frame paradigm.
- Their main characteristics are:
  - A simple to use language (an extension of the propositional language, without variables);
  - But that includes a notion of quantification (guarded quantification);
  - With special operators chosen to facilitate the enunciation of definitions;
  - With a good balance between expressivity and tractability;
  - With highly optimized inference systems.

Description Logics

Description Logics

In a DL we have operators to build definitions using individuals, concepts and roles:

- Individuals are "objects" in a given universe.

Description Logics

In a DL we have operators to build definitions using individuals, concepts and roles:

- Individuals are "objects" in a given universe.
- Concepts correspond to "classes of objects" and will be interpreted as sets in a given universe.

Description Logics

In a DL we have operators to build definitions using individuals, concepts and roles:

- Individuals are "objects" in a given universe.
- Concepts correspond to "classes of objects" and will be interpreted as sets in a given universe.
- Roles correspond to "links between objects" and will be interpreted as binary relations over a given universe.

Description Logics

In a DL we have operators to build definitions using individuals, concepts and roles:

- Individuals are "objects" in a given universe.
- Concepts correspond to "classes of objects" and will be interpreted as sets in a given universe.
- Roles correspond to "links between objects" and will be interpreted as binary relations over a given universe.

Example: The "Happy Father"

Concepts $= \{$ Man, Woman, Happy, Rich $\}$

Roles $= \{$ has-children $\}$

Individuals $= \{$ carlos $\}$

$$\begin{aligned}\text{HappyFather} \equiv \text{Man} \ \wedge & \ \exists\, \text{has-children.Man} \ \wedge \\ & \ \exists\, \text{has-children.Woman} \ \wedge \\ & \ \forall\, \text{has-children.(Happy} \vee \text{Rich)}\end{aligned}$$

carlos:$\neg$HappyFather

Application Areas

Application Areas

- Terminological Knowledge Bases and Ontologies
  - DLs were created exactly for this task
  - Specially useful as a language to define and maintain ontologies

Application Areas

- Terminological Knowledge Bases and Ontologies
  - DLs were created exactly for this task
  - Specially useful as a language to define and maintain ontologies
- Semantic Web
  - To add 'semantic markup' to the information in the web.
  - Such markup would use ontological repositories as a store of common definitions with clear semantics
  - DL inference systems would be used for the development, mantainment and merging of these ontologies, and for the dynamic evolution of resources (e.g. search).

Application Areas

- Terminological Knowledge Bases and Ontologies
  - DLs were created exactly for this task
  - Specially useful as a language to define and maintain ontologies
- Semantic Web
  - To add 'semantic markup' to the information in the web.
  - Such markup would use ontological repositories as a store of common definitions with clear semantics
  - DL inference systems would be used for the development, mantainment and merging of these ontologies, and for the dynamic evolution of resources (e.g. search).
- Computational Linguistics
  - Many tasks in computational linguistics require inference and 'background knowledge': reference resolution, question/answering.
  - In some cases, the expressive power of DLs is enough and we don't need to move to FOL.

Short Story of DL

Short Story of DL

- 1st Stage:
  - Incomplete Systems (Back, Classic, Loom, ...)
  - Based in structural algorithms

Short Story of DL

- 1st Stage:
  - Incomplete Systems (Back, Classic, Loom, ...)
  - Based in structural algorithms
- 2nd Stage:
  - Development of tableaux algorithms and first complexity results
  - Tableaux based systems for PSpace complete logics (Kris, Crack)
  - Research in optimization techniques

Short Story of DL

- 1st Stage:
  - Incomplete Systems (BACK, CLASSIC, LOOM, ...)
  - Based in structural algorithms
- 2nd Stage:
  - Development of tableaux algorithms and first complexity results
  - Tableaux based systems for PSpace complete logics (KRIS, CRACK)
  - Research in optimization techniques
- 3rd Stage:
  - Tableaux algorithms for very expressive DLs
  - Tableau based systems with many optimizations
    for ExpTime Logics (FaCT, DLP, RACER, PELLET)
  - Relation with modal logics and fragments of FOL

Short Story of DL

- 1st Stage:
  - Incomplete Systems (BACK, CLASSIC, LOOM, . . .)
  - Based in structural algorithms
- 2nd Stage:
  - Development of tableaux algorithms and first complexity results
  - Tableaux based systems for PSpace complete logics (KRIS, CRACK)
  - Research in optimization techniques
- 3rd Stage:
  - Tableaux algorithms for very expressive DLs
  - Tableau based systems with many optimizations
    for ExpTime Logics (FACT, DLP, RACER, PELLET)
  - Relation with modal logics and fragments of FOL
- 4th Stage:
  - Mature implementations (Commercial!)
  - Applications and tools start to be widely used (e.g., Semantic Web).

Description Logics

Description Logics

$$
\begin{array}{l}
\text{HappyFather} \equiv \text{Man} \wedge \\
\quad \exists\ \text{has-children.Man}\ \wedge \\
\quad \exists\ \text{has-children.Woman}\ \wedge \\
\quad \forall\ \text{has-children.(Happy} \vee \text{Rich)} \\
\text{carlos:}\neg\text{HappyFather}
\end{array}
$$

Description Logics

- The language is defined in three steps.

HappyFather $\equiv$ Man$\wedge$
$\exists$ has-children.Man $\wedge$
$\exists$ has-children.Woman $\wedge$
$\forall$ has-children.(Happy $\vee$ Rich)
carlos:$\neg$HappyFather

Description Logics

$\text{HappyFather} \equiv \text{Man} \wedge$
$\exists\ \text{has-children.Man}\ \wedge$
$\exists\ \text{has-children.Woman}\ \wedge$
$\forall\ \text{has-children.(Happy} \vee \text{Rich)}$
carlos:$\neg$HappyFather

- The language is defined in three steps.
  - Concepts: we construct complex concepts using other concepts (atomics or introduced via definitions) and roles: E.g., $\exists$has-children.Man

Description Logics

$$\begin{aligned}&\text{HappyFather} \equiv \text{Man} \wedge \\ &\quad \exists\ \text{has-children.Man}\ \wedge \\ &\quad \exists\ \text{has-children.Woman}\ \wedge \\ &\quad \forall\ \text{has-children.(Happy} \vee \text{Rich)} \\ &\text{carlos:}\neg\text{HappyFather}\end{aligned}$$

- The language is defined in three steps.
  - Concepts: we construct complex concepts using other concepts (atomics or introduced via definitions) and roles: E.g., $\exists$has-children.Man
  - Definitions: we use concepts to build definitions (or relations between definitions): E.g., HappyFather $\equiv$ ...

Description Logics

HappyFather $\equiv$ Man$\wedge$
$\exists$ has-children.Man $\wedge$
$\exists$ has-children.Woman $\wedge$
$\forall$ has-children.(Happy $\vee$ Rich)
carlos:$\neg$HappyFather

- The language is defined in three steps.
  - Concepts: we construct complex concepts using other concepts (atomics or introduced via definitions) and roles: E.g., $\exists$has-children.Man
  - Definitions: we use concepts to build definitions (or relations between definitions): E.g., HappyFather $\equiv$ ...
  - Assertions: assign concepts and roles to particular elements in our model: E.g., carlos:$\neg$HappyFather

Concept Construction

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman
  - Boolean Operators: If C and D are concepts the the following are concepts

    $C \wedge D$ the conjunction of C and D Rich $\wedge$ Handsome

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman
  - Boolean Operators: If C and D are concepts the the following are concepts

$C \wedge D$	the conjunction of C and D	Rich $\wedge$ Handsome
$C \vee D$	the disjunction of C and D	Rich $\vee$ Handsome

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman
  - Boolean Operators: If C and D are concepts the the following are concepts

$C \wedge D$	the conjunction of C and D	Rich $\wedge$ Handsome
$C \vee D$	the disjunction of C and D	Rich $\vee$ Handsome
$\neg C$	the negation of C	$\neg$Rich

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman
  - Boolean Operators: If C and D are concepts the the following are concepts

$C \wedge D$	the conjunction of C and D	Rich $\wedge$ Handsome
$C \vee D$	the disjunction of C and D	Rich $\vee$ Handsome
$\neg C$	the negation of C	$\neg$Rich

  - Relational Operators: if C is a concept and R is a role, the following are concepts

$\forall R.C$	each element acc. through R is in C	$\forall$has-children.Woman

Concept Construction

- A concept can be
  - $\top$, the trivial concept, of which every element is a member.
  - An atomic concept: Man, Woman
  - Boolean Operators: If C and D are concepts the the following are concepts

$C \wedge D$	the conjunction of C and D	Rich $\wedge$ Handsome
$C \vee D$	the disjunction of C and D	Rich $\vee$ Handsome
$\neg C$	the negation of C	$\neg$Rich

  - Relational Operators: if C is a concept and R is a role, the following are concepts

$\forall R.C$	each element acc. through R is in C	$\forall$has-children.Woman
$\exists R.C$	some element acc. through R is in C	$\exists$has-children.Woman

Building Definitions

Building Definitions

Given two concepts C and D, there are two types of definitions:

- Partial Definitions: $C \sqsubseteq D$. conditions specified in C are sufficient to qualify elements in C as members of D D, but they are not necessary; or vice-versa.

Building Definitions

Given two concepts C and D, there are two types of definitions:

- Partial Definitions: $C \sqsubseteq D$. conditions specified in C are sufficient to qualify elements in C as members of D D, but they are not necessary; or vice-versa.

$$\exists\text{has-children}.\text{Man} \wedge \exists\text{has-children}.\text{Woman} \quad \sqsubseteq \quad \text{BusyFather} \qquad \text{(suff. condition)}$$

Building Definitions

Given two concepts C and D, there are two types of definitions:

- Partial Definitions: $C \sqsubseteq D$. conditions specified in C are sufficient to qualify elements in C as members of D D, but they are not necessary; or vice-versa.

$$\begin{array}{rcll} \exists \mathsf{has\text{-}children.Man} \wedge \exists \mathsf{has\text{-}children.Woman} & \sqsubseteq & \mathsf{BusyFather} & \text{(suff. condition)} \\ \mathsf{BusyFather} & \sqsubseteq & \exists \mathsf{has\text{-}children}.\top & \text{(nec. condition)} \end{array}$$

Building Definitions

Given two concepts C and D, there are two types of definitions:

- Partial Definitions: $C \sqsubseteq D$. conditions specified in C are sufficient to qualify elements in C as members of D D, but they are not necessary; or vice-versa.

$$\exists \mathsf{has\text{-}children.Man} \wedge \exists \mathsf{has\text{-}children.Woman} \sqsubseteq \mathsf{BusyFather} \quad \text{(suff. condition)}$$
$$\mathsf{BusyFather} \sqsubseteq \exists \mathsf{has\text{-}children}.\top \quad \text{(nec. condition)}$$

- Total Definitions: $C \equiv D$. Conditions indicated in D are both necessary and sufficient to qualify elements of D as elements of C (and vice-versa). Concepts C and D are equivalent.

Building Definitions

Given two concepts C and D, there are two types of definitions:

- Partial Definitions: $C \sqsubseteq D$. conditions specified in C are sufficient to qualify elements in C as members of D D, but they are not necessary; or vice-versa.

$$\exists\text{has-children}.\text{Man} \wedge \exists\text{has-children}.\text{Woman} \sqsubseteq \text{BusyFather} \quad \text{(suff. condition)}$$

$$\text{BusyFather} \sqsubseteq \exists\text{has-children}.\top \quad \text{(nec. condition)}$$

- Total Definitions: $C \equiv D$. Conditions indicated in D are both necessary and sufficient to qualify elements of D as elements of C (and vice-versa). Concepts C and D are equivalent.

$$\text{GrandMother} \equiv \text{Woman} \wedge \exists\text{has-children}.\exists\text{has-children}.\top$$

(Equivalent to say both $C \sqsubseteq D$ and $D \sqsubseteq C$)

Building Assertions

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.

Given elements a and b, a concept C and a relation R

- Assigning elements to concepts: $a{:}C$. Indicates that C is true of a. I.e., all conditions indicated in C apply to a.

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.
Given elements a and b, a concept C and a relation R

- Assigning elements to concepts: $a{:}C$. Indicates that C is true of a. I.e., all conditions indicated in C apply to a.

carlos:Argentine

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.

Given elements a and b, a concept C and a relation R

- Assigning elements to concepts: $a{:}C$. Indicates that C is true of a. I.e., all conditions indicated in C apply to a.

$$\text{carlos}{:}\text{Argentine}$$

$$\text{carlos}{:}(\text{Argentine} \wedge \exists\text{Lives-in}.\text{Europe})$$

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.
Given elements a and b, a concept C and a relation R

- Assigning elements to concepts: $a{:}C$. Indicates that C is true of a. I.e., all conditions indicated in C apply to a.

$$\text{carlos:Argentine}$$

$$\text{carlos:}(\text{Argentine} \wedge \exists\text{Lives-in.Europe})$$

- Assigning elements elements to relations: $(a, b){:}R$. Indicates that the elements a and b are related via the role R.

Building Assertions

We can "assign assertions" to particular elements in the situation we are describing.

Given elements a and b, a concept C and a relation R

- Assigning elements to concepts: $a{:}C$. Indicates that C is true of a. I.e., all conditions indicated in C apply to a.

$$\text{carlos:Argentine}$$

$$\text{carlos:}(\text{Argentine} \wedge \exists\text{Lives-in.Europe})$$

- Assigning elements elements to relations: $(a, b){:}R$. Indicates that the elements a and b are related via the role R.

$$(\text{carlos,nancy})\text{:Lives-in}$$

A Complete Example

A Complete Example

$$
\begin{aligned}
\mathsf{Woman} &\sqsubseteq \mathsf{Person} \wedge \exists \mathsf{sex.Female} \\
\mathsf{Man} &\sqsubseteq \mathsf{Person} \wedge \exists \mathsf{sex.Male} \\
\mathsf{FatherOrMother} &\equiv \mathsf{Person} \wedge \exists \mathsf{has\text{-}children.Person} \\
\mathsf{Mother} &\equiv \mathsf{Woman} \wedge \mathsf{FatherOrMother} \\
\mathsf{Father} &\equiv \mathsf{Man} \wedge \mathsf{FatherOrMother}
\end{aligned}
$$

$$
\begin{gathered}
\mathsf{alice}{:}\mathsf{Mother} \\
(\mathsf{alice},\mathsf{betty}){:}\mathsf{has\text{-}children} \\
(\mathsf{alice},\mathsf{carlos}){:}\mathsf{has\text{-}children}
\end{gathered}
$$

Reasoning with Description Logics

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?
    Equivalently, can we find a model where C is not empty?

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?
    Equivalently, can we find a model where C is not empty?
  - Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A\rangle$ makes a a C?

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?
    Equivalently, can we find a model where C is not empty?
  - Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A \rangle$ makes a a C?
    Equivalently, does every model where $\langle T, A \rangle$ is true, also makes $a : C$ true?

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?
    Equivalently, can we find a model where C is not empty?
  - Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A\rangle$ makes a a C?
    Equivalently, does every model where $\langle T, A\rangle$ is true, also makes $a : C$ true?
  - Concept Equivalence: Given some definitions T, some assertions A, and two concepts C_1 and C_2, does the information in $\langle T, A\rangle$ makes the concept $C_1 and C_2$ equivalent?

Reasoning with Description Logics

- There are different Reasoning Task that we might be interested in, when using Description Logics.
- For example:
  - Concept Inconsistency: Given a concept C, is C always empty in every model?
    Equivalently, can we find a model where C is not empty?
  - Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A \rangle$ makes a a C?
    Equivalently, does every model where $\langle T, A \rangle$ is true, also makes $a : C$ true?
  - Concept Equivalence: Given some definitions T, some assertions A, and two concepts C_1 and C_2, does the information in $\langle T, A \rangle$ makes the concept $C_1 and C_2$ equivalent?
    Equivalently, does every model where $\langle T, A \rangle$ is true, also makes $C_1 \equiv C_2$ true?

The Tableaux Method

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.
  - Each branch of the tree represent a (partial) model of the root formula.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.
  - Each branch of the tree represent a (partial) model of the root formula.
  - Branches are expanded via tableaux rules.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.
  - Each branch of the tree represent a (partial) model of the root formula.
  - Branches are expanded via tableaux rules.
  - If a branch contains a contradition it is closed.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.
  - Each branch of the tree represent a (partial) model of the root formula.
  - Branches are expanded via tableaux rules.
  - If a branch contains a contradition it is closed.
  - If no further rule can be applied and there is at least a branch which is not closed, then we have found a model for the root.

The Tableaux Method

- We will use the Tableaux Method to solve the inference tasks we introduced in the previous slide.
- What is a tableaux? It's a method to search for models
  - It's a collection of formulas (assertions) organized as a tree.
  - Each branch of the tree represent a (partial) model of the root formula.
  - Branches are expanded via tableaux rules.
  - If a branch contains a contradition it is closed.
  - If no further rule can be applied and there is at least a branch which is not closed, then we have found a model for the root.
- A branch is closed if for some C and some a, both $a : C$ and $a : \neg C$ are in the branch; or if $a : \neg\top$ is in the branch.

Tableaux Rules

Tableaux Rules

For Conjunction:

$$\frac{a : C_1 \wedge C_2}{\begin{array}{c} a : C_2 \\ a : C_1 \end{array}} \; (\wedge)$$

Tableaux Rules

For Conjunction:

$$\frac{a : C_1 \wedge C_2}{\begin{array}{c} a : C_2 \\ a : C_1 \end{array}}\ (\wedge) \qquad \frac{a : \neg(C_1 \wedge C_2)}{a : \neg C_1 \quad \mid \quad a : \neg C_2}\ (\neg\wedge)$$

Tableaux Rules

For Conjunction:

$$\frac{a : C_1 \wedge C_2}{\begin{array}{c} a : C_2 \\ a : C_1 \end{array}} \ (\wedge) \qquad \frac{a : \neg(C_1 \wedge C_2)}{a : \neg C_1 \quad | \quad a : \neg C_2} \ (\neg\wedge)$$

For Disjunction

$$\frac{a : C_1 \vee C_2}{a : C_1 \quad | \quad a : C_2} \ (\vee)$$

Tableaux Rules

For Conjunction:

$$\frac{a : C_1 \wedge C_2}{\begin{array}{c} a : C_2 \\ a : C_1 \end{array}} \ (\wedge) \qquad \frac{a : \neg(C_1 \wedge C_2)}{a : \neg C_1 \quad | \quad a : \neg C_2} \ (\neg\wedge)$$

For Disjunction

$$\frac{a : C_1 \vee C_2}{a : C_1 \quad | \quad a : C_2} \ (\vee) \qquad \frac{a : \neg(C_1 \vee C_2)}{\begin{array}{c} a : \neg C_2 \\ a : \neg C_1 \end{array}} \ (\neg\vee)$$

Tableaux Rules

Tableaux Rules

For Existential:

$$\frac{a : \exists R.C}{\begin{array}{c} b : C \\ (a, b) : R \end{array}} \ (\exists)$$

for b a new individual

Tableaux Rules

For Existential:

$$\frac{a : \exists R.C}{\begin{array}{c} b : C \\ (a, b) : R \end{array}} \; (\exists) \qquad \frac{\begin{array}{c} a : \neg(\exists R.C) \\ (a, b) : R \end{array}}{b : \neg C} \; (\neg\exists)$$

for b a new individual

Tableaux Rules

For Existential:

$$\frac{a : \exists R.C}{\begin{array}{c} b : C \\ (a,b) : R \end{array}} \ (\exists) \qquad \frac{\begin{array}{c} a : \neg(\exists R.C) \\ (a,b) : R \end{array}}{b : \neg C} \ (\neg\exists)$$

for b a new individual

For Universal:

$$\frac{\begin{array}{c} a : \forall R.C \\ (a,b) : R \end{array}}{b : C} \ (\forall)$$

Tableaux Rules

For Existential:

$$\frac{a : \exists R.C}{\begin{array}{c} b : C \\ (a, b) : R \end{array}} \; (\exists) \qquad \frac{\begin{array}{c} a : \neg(\exists R.C) \\ (a, b) : R \end{array}}{b : \neg C} \; (\neg\exists)$$

for b a new individual

For Universal:

$$\frac{\begin{array}{c} a : \forall R.C \\ (a, b) : R \end{array}}{b : C} \; (\forall) \qquad \frac{a : \neg(\forall R.C)}{\begin{array}{c} b : \neg C \\ (a, b) : R \end{array}} \; (\neg\forall)$$

for b a new individual

Tableaux Rules

Tableaux Rules

For a set T of Definitions

$$\frac{C_1 \sqsubseteq C_2 \in T}{a : \neg C_1 \vee C_2}\ (\sqsubseteq) \qquad\qquad \frac{C_1 \equiv C_2 \in T}{\begin{array}{c} a : \neg C_2 \vee C_1 \\ a : \neg C_1 \vee C_2 \end{array}}\ (\equiv)$$

for a any individual in the tableaux

Tableaux Rules

For a set T of Definitions

$$\frac{C_1 \sqsubseteq C_2 \in T}{a : \neg C_1 \vee C_2}\ (\sqsubseteq) \qquad \frac{C_1 \equiv C_2 \in T}{\begin{array}{c} a : \neg C_2 \vee C_1 \\ a : \neg C_1 \vee C_2 \end{array}}\ (\equiv)$$

for a any individual in the tableaux
For a set A of Assertions

$$\frac{a : C \in A}{a : C}\ (a :) \qquad \frac{(a, b) : R \in A}{(a, b) : R}\ ((a, b) :)$$

Using the Tableaux Rules

Using the Tableaux Rules

- Concept Inconsistency: Given a concept C, is C always empty in every model?

Using the Tableaux Rules

- Concept Inconsistency: Given a concept C, is C always empty in every model?
  Run the tableaux rules on $a : C$ for an arbitrary a. If all the branches are closed, then C is always empty in every model.

Using the Tableaux Rules

- Concept Inconsistency: Given a concept C, is C always empty in every model?
  Run the tableaux rules on $a : C$ for an arbitrary a. If all the branches are closed, then C is always empty in every model.
- We prove that $C \wedge \neg(D \vee C)$ is inconsistent.

$$
\begin{array}{c}
a : C \wedge \neg(D \vee C) \\
a : C \\
a : \neg(D \vee C) \\
a : \neg D \\
a : \neg C \\
\otimes
\end{array}
$$

Using the Tableaux Rules

Using the Tableaux Rules

- Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A \rangle$ makes a a C?

Using the Tableaux Rules

- Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A\rangle$ makes a a C?
  Run the tableaux rules on $a : \neg C$. If all the branches are closed, then in every model $a : C$.

Using the Tableaux Rules

- Concept Membership: Given some definitions T, some assertions A, a concept C and an individual a, does the information in $\langle T, A \rangle$ makes a a C?
  Run the tableaux rules on $a : \neg C$. If all the branches are closed, then in every model $a : C$.
- We prove that given $T = \{\text{Father} \equiv \text{Man} \wedge \exists\text{has-child}.\top\}$ and $A = \{\text{a} : \text{Father}\}$ it follows that a : Man.

$$
\begin{array}{c}
\text{a} : \neg\text{Man} \\
\text{a} : \text{Father} \\
\text{a} : \neg\text{Father} \vee (\text{Man} \wedge \exists\text{has-child}.\top) \\
\text{a} : \text{Father} \vee \neg(\text{Man} \wedge \exists\text{has-child}.\top)
\end{array}
$$

$$
\begin{array}{cc}
\text{a} : \neg\text{Father} & \text{a} : (\text{Man} \wedge \exists\text{has-child}.\top) \\
\otimes & \text{a} : \text{Man} \\
 & \text{a} : \exists\text{has-child}.\top \\
 & \otimes
\end{array}
$$

Using the Tableaux Rules

Using the Tableaux Rules

- Concept Equivalence: Given some definitions T, some assertions A, and two concepts C_1 and C_2, does the information in $\langle T, A\rangle$ makes the concept $C_1 and C_2$ equivalent?

Using the Tableaux Rules

- Concept Equivalence: Given some definitions T, some assertions A, and two concepts C_1 and C_2, does the information in $\langle T, A \rangle$ makes the concept $C_1 and C_2$ equivalent? Run the tableaux rules on $a : C_1 \wedge \neg C_2$. If all the branches are closed, then in every model $C_1 \sqsubseteq C_2$.
  Do the same for $a : C_1 \wedge \neg C_2$.

Exercises

Prove that, with respect to the following definitions,

$$
\begin{aligned}
\mathsf{Man} &\equiv \mathsf{Male} \wedge \mathsf{Human} \\
\mathsf{Parent} &\equiv \exists \mathsf{children}.\top \\
\mathsf{Father} &\equiv \mathsf{Man} \wedge \mathsf{Parent} \\
\mathsf{Father\text{-}with\text{-}only\text{-}male\text{-}children} &\equiv \mathsf{Father} \wedge \mathsf{Human} \wedge (\forall \mathsf{children}.\mathsf{Male}) \\
\mathsf{Father\text{-}with\text{-}only\text{-}sons} &\equiv \mathsf{Man} \wedge (\exists \mathsf{children}.\top) \wedge (\forall \mathsf{children}.\mathsf{Man})
\end{aligned}
$$

the concept Father-with-only-sons and Father-with-only-male-children are **not** equivalent.