Fundamentals of Artificial Intelligence
Chapter 12: **Knowledge Representation**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it

https://disi.unitn.it/rseba/DIDATTICA/fai_2024/

Teaching assistants:

Mauro Dragoni, dragoni@fbk.eu, https://www.maurodragoni.com/teaching/fai/

Paolo Morettin, paolo.morettin@unitn.it, https://paolomorettin.github.io/

M.S. Course "Artificial Intelligence Systems", academic year 2024-2025

Last update: Thursday 5th September, 2024, 19:00

Outline

Outline

Generalities

Q: What content do we put into an agent's KB?

- how do we organize such content?
- how do we represent facts about the world?

- A whole AI field: Knowledge Representation, KR
  - often combined with Automated Reasoning on KB

  ⟹ Knowledge Representation & Reasoning, KRR
- KR: use logics (e.g. FOL) to represent the most important aspects of the real world, such as: action, space, time, knowledge, belief
- Topics:
  - ontologies and ontological engineering
  - objects and categories, composite objects, measurements, ...
  - actions and change, events, temporal intervals, ...
  - reasoning about knowledge & beliefs
  - reasoning about categories
  - default reasoning
  - ...

Generalities

Q: What content do we put into an agent's KB?

- how do we organize such content?
- how do we represent facts about the world?

- A whole AI field: Knowledge Representation, KR
  - often combined with Automated Reasoning on KB

  $\Longrightarrow$ Knowledge Representation & Reasoning, KRR

- KR: use logics (e.g. FOL) to represent the most important aspects of the real world, such as: action, space, time, knowledge, belief
- Topics:
  - ontologies and ontological engineering
  - objects and categories, composite objects, measurements, ...
  - actions and change, events, temporal intervals, ...
  - reasoning about knowledge & beliefs
  - reasoning about categories
  - default reasoning
  - ...

Generalities

Q: What content do we put into an agent's KB?

- how do we organize such content?
- how do we represent facts about the world?

- A whole AI field: Knowledge Representation, KR
  - often combined with Automated Reasoning on KB

  $\Longrightarrow$ Knowledge Representation & Reasoning, KRR
- KR: use logics (e.g. FOL) to represent the most important aspects of the real world, such as: action, space, time, knowledge, belief
- Topics:
  - ontologies and ontological engineering
  - objects and categories, composite objects, measurements, ...
  - actions and change, events, temporal intervals, ...
  - reasoning about knowledge & beliefs
  - reasoning about categories
  - default reasoning
  - ...

Generalities

Q: What content do we put into an agent's KB?

- how do we organize such content?
- how do we represent facts about the world?

- A whole AI field: Knowledge Representation, KR
  - often combined with Automated Reasoning on KB

  $\Longrightarrow$ Knowledge Representation & Reasoning, KRR
- KR: use logics (e.g. FOL) to represent the most important aspects of the real world, such as: action, space, time, knowledge, belief
- Topics:
  - ontologies and ontological engineering
  - objects and categories, composite objects, measurements, ...
  - actions and change, events, temporal intervals, ...
  - reasoning about knowledge & beliefs
  - reasoning about categories
  - default reasoning
  - ...

Knowledge Engineering and Ontological Engineering

Knowledge Engineering

- The activity to formalize a specific problem or task domain
- Relevant questions to be addressed:
  - What are the relevant facts, objects, relations ... ?
  - Which is the right level of abstraction?
  - What are the queries to the KB (inferences)?

Ontological Engineering

- The activity to build general-purpose ontologies
  - should: (with the addition of domain-specific axioms)
  - In non-trivial domains, reasoning and problem solving could involve several areas of knowledge simultaneously
  - $\Longrightarrow$
- Several attempts to build general-purpose ontologies
  - CYC, DBpedia, TextRunner, ...
  - not very successful so far

Knowledge Engineering and Ontological Engineering

Knowledge Engineering

- The activity to formalize a specific problem or task domain
- Relevant questions to be addressed:
  - What are the relevant facts, objects, relations ... ?
  - Which is the right level of abstraction?
  - What are the queries to the KB (inferences)?

Ontological Engineering

- The activity to build general-purpose ontologies
  - should be applicable in any special-purpose domain (with the addition of domain-specific axioms)
  - In non trivial domains, reasoning and problem solving could involve several areas of knowledge simultaneously
    $\Longrightarrow$ different areas of knowledge must be combined
- Several attempts to build general-purpose ontologies
  - CYC, DBpedia, TextRunner, ...
  - not very successful so far

Knowledge Engineering and Ontological Engineering

Knowledge Engineering

- The activity to formalize a specific problem or task domain
- Relevant questions to be addressed:
  - What are the relevant facts, objects, relations ... ?
  - Which is the right level of abstraction?
  - What are the queries to the KB (inferences)?

Ontological Engineering

- The activity to build general-purpose ontologies
  - should be applicable in any special-purpose domain (with the addition of domain-specific axioms)
  - In non trivial domains, reasoning and problem solving could involve several areas of knowledge simultaneously
    $\Longrightarrow$ different areas of knowledge must be combined
- Several attempts to build general-purpose ontologies
  - CYC, DBpedia, TextRunner, ...
  - not very successful so far

Knowledge Engineering and Ontological Engineering

Knowledge Engineering

- The activity to formalize a specific problem or task domain
- Relevant questions to be addressed:
  - What are the relevant facts, objects, relations ... ?
  - Which is the right level of abstraction?
  - What are the queries to the KB (inferences)?

Ontological Engineering

- The activity to build general-purpose ontologies
  - should be applicable in any special-purpose domain (with the addition of domain-specific axioms)
  - In non trivial domains, reasoning and problem solving could involve several areas of knowledge simultaneously
    $\Longrightarrow$ different areas of knowledge must be combined
- Several attempts to build general-purpose ontologies
  - CYC, DBpedia, TextRunner, ...
  - not very successful so far

Knowledge Engineering and Ontological Engineering

Knowledge Engineering

- The activity to formalize a specific problem or task domain
- Relevant questions to be addressed:
  - What are the relevant facts, objects, relations ... ?
  - Which is the right level of abstraction?
  - What are the queries to the KB (inferences)?

Ontological Engineering

- The activity to build general-purpose ontologies
  - should be applicable in any special-purpose domain (with the addition of domain-specific axioms)
  - In non trivial domains, reasoning and problem solving could involve several areas of knowledge simultaneously
    $\Longrightarrow$ different areas of knowledge must be combined
- Several attempts to build general-purpose ontologies
  - CYC, DBpedia, TextRunner, ...
  - not very successful so far

Outline

Categories and Objects

Categories, Objects, Members and Subclasses

- KR requires the organisation of objects into categories
  - interaction at the level of the object
  - reasoning at the level of categories
  - ex: typically we want to buy a basketball, rather than a particular basketball instance
- Categories play a role in predictions about objects
  - agent infers the presence of certain objects from perceptual input
  - infers category from the perceived properties of the objects,
  - uses category information to make predictions about the objects
- Categories can be represented in two ways by FOL
  - predicates (ex Basketball(x)): relations
  - reification of categories into objects (ex Basketballs): sets
    $\Longrightarrow$ allows categories to be argument of predicates/functions
- Membership of a category as set membership
  - ex: $Member(b, Basketballs)$ (abbr. $b \in Basketballs$)
- Subcategories (aka subclasses) are (strict) subsets
  - ex: Subset(Basketballs, Balls) (abbr. $Basketballs \subset Balls$)

Categories and Objects

Categories, Objects, Members and Subclasses

- KR requires the organisation of objects into categories
  - interaction at the level of the object
  - reasoning at the level of categories
  - ex: typically we want to buy a basketball, rather than a particular basketball instance
- Categories play a role in predictions about objects
  - agent infers the presence of certain objects from perceptual input
  - infers category from the perceived properties of the objects,
  - uses category information to make predictions about the objects
- Categories can be represented in two ways by FOL
  - predicates (ex Basketball(x)): relations
  - reification of categories into objects (ex Basketballs): sets
    $\Longrightarrow$ allows categories to be argument of predicates/functions
- Membership of a category as set membership
  - ex: $Member(b, Basketballs)$ (abbr. $b \in Basketballs$)
- Subcategories (aka subclasses) are (strict) subsets
  - ex: Subset(Basketballs, Balls) (abbr. $Basketballs \subset Balls$)

Categories and Objects

Categories, Objects, Members and Subclasses

- KR requires the organisation of objects into categories
  - interaction at the level of the object
  - reasoning at the level of categories
  - ex: typically we want to buy a basketball, rather than a particular basketball instance
- Categories play a role in predictions about objects
  - agent infers the presence of certain objects from perceptual input
  - infers category from the perceived properties of the objects,
  - uses category information to make predictions about the objects
- Categories can be represented in two ways by FOL
  - predicates (ex Basketball(x)): relations
  - reification of categories into objects (ex Basketballs): sets

    $\Longrightarrow$ allows categories to be argument of predicates/functions
- Membership of a category as set membership
  - ex: $Member(b, Basketballs)$ (abbr. $b \in Basketballs$)
- Subcategories (aka subclasses) are (strict) subsets
  - ex: Subset(Basketballs, Balls) (abbr. $Basketballs \subset Balls$)

Categories and Objects

Categories, Objects, Members and Subclasses

- KR requires the organisation of objects into categories
  - interaction at the level of the object
  - reasoning at the level of categories
  - ex: typically we want to buy a basketball, rather than a particular basketball instance
- Categories play a role in predictions about objects
  - agent infers the presence of certain objects from perceptual input
  - infers category from the perceived properties of the objects,
  - uses category information to make predictions about the objects
- Categories can be represented in two ways by FOL
  - predicates (ex Basketball(x)): relations
  - reification of categories into objects (ex Basketballs): sets
    $\Longrightarrow$ allows categories to be argument of predicates/functions
- Membership of a category as set membership
  - ex: $Member(b, Basketballs)$ (abbr. $b \in Basketballs$)
- Subcategories (aka subclasses) are (strict) subsets
  - ex: Subset(Basketballs, Balls) (abbr. $Basketballs \subset Balls$)

Categories and Objects

Categories, Objects, Members and Subclasses

- KR requires the organisation of objects into categories
  - interaction at the level of the object
  - reasoning at the level of categories
  - ex: typically we want to buy a basketball, rather than a particular basketball instance
- Categories play a role in predictions about objects
  - agent infers the presence of certain objects from perceptual input
  - infers category from the perceived properties of the objects,
  - uses category information to make predictions about the objects
- Categories can be represented in two ways by FOL
  - predicates (ex Basketball(x)): relations
  - reification of categories into objects (ex Basketballs): sets

    $\Longrightarrow$ allows categories to be argument of predicates/functions
- Membership of a category as set membership
  - ex: $Member(b, Basketballs)$ (abbr. $b \in Basketballs$)
- Subcategories (aka subclasses) are (strict) subsets
  - ex: Subset(Basketballs, Balls) (abbr. $Basketballs \subset Balls$)

Categories and Objects [cont.]

Inheritance and Taxonomies

- A subcategory inherits the properties of the category
  - ex:
    if $\forall x.(x \in Food \rightarrow Edible(x))$, $Fruit \subset Food$, $Apples \subset Fruit$
    then $\forall x.(x \in Apple \rightarrow Edible(x))$
- A member inherits the properties of the category
  - if $a \in Apples$, then $Edible(a)$
- Subclass relation organize categories into taxonomies (aka taxonomic hierarchies)
  - ex: taxonomy of >10M living&extinct species
  - ex: Dewey Decimal System: taxonomy of all fields of knowledge

Categories and Objects [cont.]

Inheritance and Taxonomies

- A subcategory **inherits** the properties of the category
  - ex:
    if $\forall x.(x \in Food \rightarrow Edible(x))$, $Fruit \subset Food$, $Apples \subset Fruit$
    then $\forall x.(x \in Apple \rightarrow Edible(x))$
- A member **inherits** the properties of the category
  - if $a \in Apples$, then $Edible(a)$
- Subclass relation organize categories into taxonomies (aka taxonomic hierarchies)
  - ex: taxonomy of >10M living&extinct species
  - ex: Dewey Decimal System: taxonomy of all fields of knowledge

Categories and Objects [cont.]

Inheritance and Taxonomies

- A subcategory inherits the properties of the category
  - ex:
    if $\forall x.(x \in Food \rightarrow Edible(x))$, $Fruit \subset Food$, $Apples \subset Fruit$
    then $\forall x.(x \in Apple \rightarrow Edible(x))$
- A member inherits the properties of the category
  - if $a \in Apples$, then $Edible(a)$
- Subclass relation organize categories into taxonomies (aka taxonomic hierarchies)
  - ex: taxonomy of >10M living&extinct species
  - ex: Dewey Decimal System: taxonomy of all fields of knowledge

Categories and Objects [cont.]

FOL Reasoning about Categories

- FOL allows to state facts about categories:
  - an object is a member of a category
    $BB_9 \in Basketballs$
  - a category is a subclass of another category
    $Basketballs \subset Balls$
  - all members of a category have some properties
    $\forall x.(x \in Basketballs \rightarrow Spherical(x))$
  - members of a category can be recognized by some properties
    $\forall x.((Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5'' \wedge x \in Balls)$
    $\rightarrow x \in Basketballs)$
  - category as a whole has some properties
    $Dogs \in DomesticatedSpecies$
- New categories can be defined by providing necessary and sufficient conditions for membership
  - $\forall x.(x \in Bachelors \leftrightarrow (Unmarried(x) \wedge x \in Adults \wedge x \in Males))$

Categories and Objects [cont.]

FOL Reasoning about Categories

- FOL allows to state facts about categories:
  - an object is a member of a category
    $BB_9 \in Basketballs$
  - a category is a subclass of another category
    $Basketballs \subset Balls$
  - all members of a category have some properties
    $\forall x.(x \in Basketballs \rightarrow Spherical(x))$
  - members of a category can be recognized by some properties
    $\forall x.((Orange(x) \land Round(x) \land Diameter(x) = 9.5'' \land x \in Balls)$
    $\rightarrow x \in Basketballs)$
  - category as a whole has some properties
    $Dogs \in DomesticatedSpecies$
- New categories can be defined by providing necessary and sufficient conditions for membership
  - $\forall x.(x \in Bachelors \leftrightarrow (Unmarried(x) \land x \in Adults \land x \in Males))$

Categories and Objects [cont.]

FOL Reasoning about Categories

- FOL allows to state facts about categories:
  - an object is a member of a category
    $BB_9 \in Basketballs$
  - a category is a subclass of another category
    $Basketballs \subset Balls$
  - all members of a category have some properties
    $\forall x.(x \in Basketballs \rightarrow Spherical(x))$
  - members of a category can be recognized by some properties
    $\forall x.((Orange(x) \land Round(x) \land Diameter(x) = 9.5'' \land x \in Balls)$
    $\rightarrow x \in Basketballs)$
  - category as a whole has some properties
    $Dogs \in DomesticatedSpecies$
- New categories can be defined by providing necessary and sufficient conditions for membership
  - $\forall x.(x \in Bachelors \leftrightarrow (Unmarried(x) \land x \in Adults \land x \in Males))$

Categories and Objects [cont.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \land c_2 \in s \land c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - ex: $Disjoint(\{Animals, Vegetables\})$, $Disjoint(\{Insects, Birds, Mammals, Reptiles\})$,
- A set of categories s is an exhaustive decomposition of a category c iff all members of c are covered by categories in s
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \land i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \land ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{NorthernItalians, CentralItalians, SouthernItalians, InsularItalians\}, Italians)$

Categories and Objects [cont.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \land c_2 \in s \land c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - ex: $Disjoint(\{Animals, Vegetables\})$, $Disjoint(\{Insects, Birds, Mammals, Reptiles\})$,
- A set of categories s is an exhaustive decomposition of a category c iff all members of c are covered by categories in s
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \land i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \land ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{NorthernItalians, CentralItalians, SouthernItalians, InsularItalians\}, Italians)$

Categories and Objects [cont.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \land c_2 \in s \land c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - ex: $Disjoint(\{Animals, Vegetables\})$, $Disjoint(\{Insects, Birds, Mammals, Reptiles\})$,
- A set of categories s is an exhaustive decomposition of a category c iff all members of c are covered by categories in s
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \land i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \land ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{NorthernItalians, CentralItalians, SouthernItalians, InsularItalians\}, Italians)$

Digression: Natural Kinds

- Many categories have no clear-cut definition (ex: chair, bush, ...)
  - Ex: tomatoes are sometimes green, red, yellow, black; they are mostly round
- One useful solution: category "Typical(.)", s.t. $Typical(c) \subseteq c$

$\Longrightarrow$ most knowledge about natural kinds will actually be about their typical instances
  - ex: $\forall x.(x \in Typical(Tomatoes) \rightarrow (Red(x) \wedge Round(x)))$

$\Longrightarrow$ We can write down useful facts about categories without providing exact definitions

Note

Quine (1953) challenged the utility of the notion of strict definition.

- Ex: "bachelor": is the Pope a bachelor?

  $\Longrightarrow$ technically yes, but misleading

Digression: Natural Kinds

- Many categories have no clear-cut definition (ex: chair, bush, ...)
  - Ex: tomatoes are sometimes green, red, yellow, black; they are mostly round
- One useful solution: category "Typical(.)", s.t. $Typical(c) \subseteq c$

  $\Longrightarrow$ most knowledge about natural kinds will actually be about their typical instances
  - ex: $\forall x.(x \in Typical(Tomatoes) \rightarrow (Red(x) \wedge Round(x)))$

$\Longrightarrow$ We can write down useful facts about categories without providing exact definitions

Note

Quine (1953) challenged the utility of the notion of strict definition.

- Ex: "bachelor": is the Pope a bachelor?

  $\Longrightarrow$ technically yes, but misleading

Digression: Natural Kinds

- Many categories have no clear-cut definition (ex: chair, bush, ...)
  - Ex: tomatoes are sometimes green, red, yellow, black; they are mostly round
- One useful solution: category "Typical(.)", s.t. $Typical(c) \subseteq c$

  $\Longrightarrow$ most knowledge about natural kinds will actually be about their typical instances
  - ex: $\forall x.(x \in Typical(Tomatoes) \rightarrow (Red(x) \land Round(x)))$

$\Longrightarrow$ We can write down useful facts about categories without providing exact definitions

Note

Quine (1953) challenged the utility of the notion of strict definition.

- Ex: "bachelor": is the Pope a bachelor?

  $\Longrightarrow$ technically yes, but misleading

Digression: Natural Kinds

- Many categories have no clear-cut definition (ex: chair, bush, ...)
  - Ex: tomatoes are sometimes green, red, yellow, black; they are mostly round
- One useful solution: category "Typical(.)", s.t. $Typical(c) \subseteq c$

  $\Longrightarrow$ most knowledge about natural kinds will actually be about their typical instances
  - ex: $\forall x.(x \in Typical(Tomatoes) \rightarrow (Red(x) \land Round(x)))$

$\Longrightarrow$ We can write down useful facts about categories without providing exact definitions

Note

Quine (1953) challenged the utility of the notion of strict definition.

- Ex: "bachelor": is the Pope a bachelor?

  $\Longrightarrow$ technically yes, but misleading

Physical Composition

- $PartOf(.,.)$ relation: One object may be part of another
  - $PartOf(Bucharest, Romania)$
  - $PartOf(Romania, EasternEurope)$
  - $PartOf(EasternEurope, Europe)$
- $PartOf(.,.)$ is reflexive and transitive:
  - $\forall x.PartOf(x,x)$
  - $\forall x,y,z.((PartOf(x,y) \land PartOf(y,z)) \rightarrow PartOf(x,z))$

  $\Longrightarrow$ $PartOf(Bucharest, Europe)$
- Categories of composite objects are often characterized by structural relations among parts. Ex: Biped

- Other concepts & relations: PartPartition, BunchOf...

Physical Composition

- $PartOf(.,.)$ relation: One object may be part of another
  - $PartOf(Bucharest, Romania)$
  - $PartOf(Romania, EasternEurope)$
  - $PartOf(EasternEurope, Europe)$
- $PartOf(.,.)$ is reflexive and transitive:
  - $\forall x.PartOf(x,x)$
  - $\forall x,y,z.((PartOf(x,y) \land PartOf(y,z)) \rightarrow PartOf(x,z))$

  $\Longrightarrow$ $PartOf(Bucharest, Europe)$
- Categories of composite objects are often characterized by structural relations among parts. Ex: Biped

- Other concepts & relations: PartPartition, BunchOf...

Physical Composition

- *PartOf*$(.,.)$ relation: One object may be part of another
  - *PartOf*(*Bucharest*, *Romania*)
  - *PartOf*(*Romania*, *EasternEurope*)
  - *PartOf*(*EasternEurope*, *Europe*)
- *PartOf*$(.,.)$ is reflexive and transitive:
  - $\forall x.PartOf(x,x)$
  - $\forall x,y,z.((PartOf(x,y) \wedge PartOf(y,z)) \rightarrow PartOf(x,z))$

  $\Longrightarrow$ *PartOf*(*Bucharest*, *Europe*)
- Categories of composite objects are often characterized by structural relations among parts. Ex: Biped

$$
\begin{aligned}
Biped(a) \quad \Rightarrow \quad & \exists\, l_1, l_2, b \;\; Leg(l_1) \wedge Leg(l_2) \wedge Body(b) \;\wedge \\
& PartOf(l_1,a) \wedge PartOf(l_2,a) \wedge PartOf(b,a) \;\wedge \\
& Attached(l_1,b) \wedge Attached(l_2,b) \;\wedge \\
& l_1 \neq l_2 \wedge [\forall\, l_3 \;\; Leg(l_3) \wedge PartOf(l_3,a) \;\Rightarrow\; (l_3 = l_1 \vee l_3 = l_2)]
\end{aligned}
$$

- Other concepts & relations: PartPartition, BunchOf...

Physical Composition

- *PartOf*(., .) relation: One object may be part of another
  - *PartOf*(*Bucharest*, *Romania*)
  - *PartOf*(*Romania*, *EasternEurope*)
  - *PartOf*(*EasternEurope*, *Europe*)
- *PartOf*(., .) is reflexive and transitive:
  - $\forall x.PartOf(x, x)$
  - $\forall x, y, z.((PartOf(x, y) \land PartOf(y, z)) \rightarrow PartOf(x, z))$

  $\Longrightarrow$ *PartOf*(*Bucharest*, *Europe*)
- Categories of composite objects are often characterized by structural relations among parts. Ex: Biped

$$\begin{aligned} Biped(a) \;\Rightarrow\; & \exists\, l_1, l_2, b \;\; Leg(l_1) \land Leg(l_2) \land Body(b) \;\land \\ & PartOf(l_1, a) \land PartOf(l_2, a) \land PartOf(b, a) \;\land \\ & Attached(l_1, b) \land Attached(l_2, b) \;\land \\ & l_1 \neq l_2 \land [\forall\, l_3 \;\; Leg(l_3) \land PartOf(l_3, a) \;\Rightarrow\; (l_3 = l_1 \lor l_3 = l_2)] \end{aligned}$$

- Other concepts & relations: PartPartition, BunchOf...

Measurements

Quantitative Measurements

- Objects may have “quantitative” properties
  - e.g. height, mass, cost, ...
- Values that we assign to these properties are measures
- Can be represented by unit functions
  - ex $Length(L_1) = Inches(1.5) \land Inches(1.5) = Centimeters(3.81)$
- Conversion between units:
  - $\forall i.\ Centimeters(2.54 \times i) = Inches(i)$
- Measures can be used to describe objects:
  - ex: $Diameter(Basketball_{12}) = Inches(9.5)$
  - ex: $ListPrice(Basketball_{12}) = \$(19)$
  - ex: $\forall d.(d \in Days \rightarrow Duration(d) = Hours(24))$

Measurements

Quantitative Measurements

- Objects may have “quantitative” properties
  - e.g. height, mass, cost, ...
- Values that we assign to these properties are measures
- Can be represented by unit functions
  - ex $Length(L_1) = Inches(1.5) \wedge Inches(1.5) = Centimeters(3.81)$
- Conversion between units:
  - $\forall i.\ Centimeters(2.54 \times i) = Inches(i)$
- Measures can be used to describe objects:
  - ex: $Diameter(Basketball_{12}) = Inches(9.5)$
  - ex: $ListPrice(Basketball_{12}) = \$(19)$
  - ex: $\forall d.(d \in Days \rightarrow Duration(d) = Hours(24))$

Measurements

Quantitative Measurements

- Objects may have "quantitative" properties
  - e.g. height, mass, cost, ...
- Values that we assign to these properties are measures
- Can be represented by unit functions
  - ex $Length(L_1) = Inches(1.5) \wedge Inches(1.5) = Centimeters(3.81)$
- Conversion between units:
  - $\forall i.\ Centimeters(2.54 \times i) = Inches(i)$
- Measures can be used to describe objects:
  - ex: $Diameter(Basketball_{12}) = Inches(9.5)$
  - ex: $ListPrice(Basketball_{12}) = \$(19)$
  - ex: $\forall d.(d \in Days \rightarrow Duration(d) = Hours(24))$

Measurements

Quantitative Measurements

- Objects may have "quantitative" properties
  - e.g. height, mass, cost, ...
- Values that we assign to these properties are measures
- Can be represented by unit functions
  - ex $Length(L_1) = Inches(1.5) \wedge Inches(1.5) = Centimeters(3.81)$
- Conversion between units:
  - $\forall i.\ Centimeters(2.54 \times i) = Inches(i)$
- Measures can be used to describe objects:
  - ex: $Diameter(Basketball_{12}) = Inches(9.5)$
  - ex: $ListPrice(Basketball_{12}) = \$(19)$
  - ex: $\forall d.(d \in Days \rightarrow Duration(d) = Hours(24))$

Measurements

Quantitative Measurements

- Objects may have "quantitative" properties
  - e.g. height, mass, cost, ...
- Values that we assign to these properties are measures
- Can be represented by unit functions
  - ex $Length(L_1) = Inches(1.5) \wedge Inches(1.5) = Centimeters(3.81)$
- Conversion between units:
  - $\forall i.\ Centimeters(2.54 \times i) = Inches(i)$
- Measures can be used to describe objects:
  - ex: $Diameter(Basketball_{12}) = Inches(9.5)$
  - ex: $ListPrice(Basketball_{12}) = \$(19)$
  - ex: $\forall d.(d \in Days \rightarrow Duration(d) = Hours(24))$

Measurements [cont.]

Qualitative Measurements

- Some measures have no scale
  - ex: beauty, deliciousness, difficulty,...
- Most important aspect of measures: they are orderable
  - Ex: $Deliciousness(SacherTorte) > Deliciousness(BrussellSprout)$
  - Ex: $Beauty(PaulNewmann) > Beauty(MartyFeldman)$
  - Ex: $Difficulty(Prove\_P \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:
  $\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Wrote(Norvig, e_1) \land Wrote(Russell, e_2))$
  $\rightarrow Difficulty(e_1) > Difficulty(e_2))$
  $\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Difficulty(e_1) > Difficulty(e_2))$
  $\rightarrow ExpectedScore(e_1) < ExpectedScore(e_2))$
  $\forall e_1 e_2.(ExpectedScore(e_1) < ExpectedScore(e_2) \rightarrow Pick(e_1, e_2) = e_2$
  Then: $(Wrote(Norvig, E_1) \land Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$
- Qualitative physics: a subfield of AI that investigates how to reason about physical systems without numerical computations

Measurements [cont.]

Qualitative Measurements

- Some measures have no scale
  - ex: beauty, deliciousness, difficulty,...
- Most important aspect of measures: they are orderable
  - Ex: $Deliciousness(SacherTorte) > Deliciousness(BrussellSprout)$
  - Ex: $Beauty(PaulNewmann) > Beauty(MartyFeldman)$
  - Ex: $Difficulty(Prove\_P \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:

  $\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Wrote(Norvig, e_1) \land Wrote(Russell, e_2)) \rightarrow Difficulty(e_1) > Difficulty(e_2))$

  $\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Difficulty(e_1) > Difficulty(e_2)) \rightarrow ExpectedScore(e_1) < ExpectedScore(e_2))$

  $\forall e_1 e_2.(ExpectedScore(e_1) < ExpectedScore(e_2) \rightarrow Pick(e_1, e_2) = e_2$

  Then: $(Wrote(Norvig, E_1) \land Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$
- Qualitative physics: a subfield of AI that investigates how to reason about physical systems without numerical computations

Measurements [cont.]

Qualitative Measurements

- Some measures have no scale
  - ex: beauty, deliciousness, difficulty,...
- Most important aspect of measures: they are orderable
  - Ex: $Deliciousness(SacherTorte) > Deliciousness(BrussellSprout)$
  - Ex: $Beauty(PaulNewmann) > Beauty(MartyFeldman)$
  - Ex: $Difficulty(Prove\_P \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:

  $$\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Wrote(Norvig, e_1) \land Wrote(Russell, e_2)) \rightarrow Difficulty(e_1) > Difficulty(e_2))$$

  $$\forall e_1 e_2.((e_1 \in Exercises \land e_2 \in Exercises \land Difficulty(e_1) > Difficulty(e_2)) \rightarrow ExpectedScore(e_1) < ExpectedScore(e_2))$$

  $$\forall e_1 e_2.(ExpectedScore(e_1) < ExpectedScore(e_2) \rightarrow Pick(e_1, e_2) = e_2$$

  Then: $(Wrote(Norvig, E_1) \land Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$
- Qualitative physics: a subfield of AI that investigates how to reason about physical systems without numerical computations

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning "an amount/quantity of..."
  - ex: $b \in butter$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \wedge PartOf(p, b)) \rightarrow p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \rightarrow MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning "an amount/quantity of..."
  - ex: $b \in butter$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \wedge PartOf(p, b)) \rightarrow p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \rightarrow MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning "an amount/quantity of..."
  - ex: $b \in butter$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \land PartOf(p, b)) \rightarrow p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \rightarrow MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning “an amount/quantity of...”
  - ex: $b \in butter$: “b is an amount/quantity of butter”
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \land PartOf(p, b)) \rightarrow p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \rightarrow MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning “an amount/quantity of...”
  - ex: $b \in butter$: “b is an amount/quantity of butter”
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \land PartOf(p, b)) \rightarrow p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \rightarrow MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Objects vs Stuff

- There are countable objects
  - e,g, apples, holes, theorems, ...
- ... and mass objects, aka stuff or substances
  - e.g. butter, water, energy, ...

$\Longrightarrow$ Intuitive meaning "an amount/quantity of..."
  - ex: $b \in butter$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in Butter \wedge PartOf(p, b)) \to p \in Butter)$
- Can define sub-categories, which are stuff
  - ex: $UnsaltedButter \subset Butter$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in Butter \to MeltingPoint(b, Centigrade(30)))$
- Stuff has no extrinsic properties
  - e.g., weight, length, shape, ...

Outline

Agents' Attitudes

- Intelligence is intrinsically social: agents need to negotiate and coordinate with other agents
- In multi-agents scenarios, to predict what other agents will do, we need methods to model mental states of other agents
  - representations of other agents' knowledge (and beliefs, goals)
- Agent's Propositional attitudes: Knows, Believes, Wants,...
  - ex "Lois **Knows** that Superman can fly"

Problem

Propositional attitudes do not behave as regular predicates

- issue: Referential opacity vs. referential transparency

Agents' Attitudes

- Intelligence is intrinsically social: agents need to negotiate and coordinate with other agents
- In multi-agents scenarios, to predict what other agents will do, we need methods to model mental states of other agents
  - representations of other agents' knowledge (and beliefs, goals)
- Agent's Propositional attitudes: Knows, Believes, Wants,...
  - ex "Lois **Knows** that Superman can fly"

Problem

Propositional attitudes do not behave as regular predicates

- issue: Referential opacity vs. referential transparency

Agents' Attitudes

- Intelligence is intrinsically social: agents need to negotiate and coordinate with other agents
- In multi-agents scenarios, to predict what other agents will do, we need methods to model mental states of other agents
  - representations of other agents' knowledge (and beliefs, goals)
- Agent's Propositional attitudes: Knows, Believes, Wants,...
  - ex "Lois **Knows** that Superman can fly"

Problem

Propositional attitudes do not behave as regular predicates

- issue: Referential opacity vs. referential transparency

Agents' Attitudes

- Intelligence is intrinsically social: agents need to negotiate and coordinate with other agents
- In multi-agents scenarios, to predict what other agents will do, we need methods to model mental states of other agents
  - representations of other agents' knowledge (and beliefs, goals)
- Agent's Propositional attitudes: Knows, Believes, Wants,...
  - ex "Lois **Knows** that Superman can fly"

Problem

Propositional attitudes do not behave as regular predicates

- issue: Referential opacity vs. referential transparency

Referential opacity vs. Referential transparency

- Consider the assertion "Lois knows that Superman can fly"
- Consider the FOL formalization: $Knows(Lois, CanFly(Superman))$
- Minor Problem: $CanFly(Superman)$ is a formula

  $\Longrightarrow$ cannot occur as argument of a predicate
  $\Longrightarrow$ must apply reification $\Longrightarrow$ make it a term
- Major Problem (Referential Transparency of FOL):
  - since Superman is Clark Kent (but Lois doesn't know it!), FOL allows to conclude "Lois knows that Clark Kent can fly":
    $Superman = Clark \wedge Knows(Lois, CanFly(Superman))$
    $\models_{FOL} Knows(Lois, CanFly(Clark))$

  $\Longrightarrow$ Wrong inference! (Lois doesn't know Clark Kent can fly!)
- Hint: FOL predicates transparent to equality reasoning:

$$t = s \wedge P(s, ...) \models_{FOL} P(t, ...)$$

- Need a logic which is opaque to equality reasoning (aka Referential Opacity): Modal Logics

Referential opacity vs. Referential transparency

- Consider the assertion "Lois knows that Superman can fly"
- Consider the FOL formalization: $Knows(Lois, CanFly(Superman))$
- Minor Problem: $CanFly(Superman)$ is a formula

  $\Longrightarrow$ cannot occur as argument of a predicate

  $\Longrightarrow$ must apply reification $\Longrightarrow$ make it a term
- Major Problem (Referential Transparency of FOL):
  - since Superman is Clark Kent (but Lois doesn't know it!), FOL allows to conclude "Lois knows that Clark Kent can fly":

    $Superman = Clark \wedge Knows(Lois, CanFly(Superman))$

    $\models_{FOL} Knows(Lois, CanFly(Clark))$

  $\Longrightarrow$ Wrong inference! (Lois doesn't know Clark Kent can fly!)
- Hint: FOL predicates transparent to equality reasoning:

$$t = s \wedge P(s, ...) \models_{FOL} P(t, ...)$$

- Need a logic which is opaque to equality reasoning (aka Referential Opacity): Modal Logics

Referential opacity vs. Referential transparency

- Consider the assertion "Lois knows that Superman can fly"
- Consider the FOL formalization: $Knows(Lois, CanFly(Superman))$
- Minor Problem: $CanFly(Superman)$ is a formula

  $\Longrightarrow$ cannot occur as argument of a predicate

  $\Longrightarrow$ must apply reification $\Longrightarrow$ make it a term
- Major Problem (Referential Transparency of FOL):
  - since Superman is Clark Kent (but Lois doesn't know it!), FOL allows to conclude "Lois knows that Clark Kent can fly":
    $Superman = Clark \land Knows(Lois, CanFly(Superman))$
    $\models_{FOL} Knows(Lois, CanFly(Clark))$

  $\Longrightarrow$ Wrong inference! (Lois doesn't know Clark Kent can fly!)
- Hint: FOL predicates transparent to equality reasoning:

$$t = s \land P(s, ...) \models_{FOL} P(t, ...)$$

- Need a logic which is opaque to equality reasoning (aka Referential Opacity): Modal Logics

Referential opacity vs. Referential transparency

- Consider the assertion "Lois knows that Superman can fly"
- Consider the FOL formalization: $Knows(Lois, CanFly(Superman))$
- Minor Problem: $CanFly(Superman)$ is a formula

$\Longrightarrow$ cannot occur as argument of a predicate

$\Longrightarrow$ must apply reification $\Longrightarrow$ make it a term

- Major Problem (Referential Transparency of FOL):
  - since Superman is Clark Kent (but Lois doesn't know it!), FOL allows to conclude "Lois knows that Clark Kent can fly":
    $Superman = Clark \wedge Knows(Lois, CanFly(Superman))$
    $\models_{FOL} Knows(Lois, CanFly(Clark))$

$\Longrightarrow$ Wrong inference! (Lois doesn't know Clark Kent can fly!)

- Hint: FOL predicates transparent to equality reasoning:

$$t = s \wedge P(s, ...) \models_{FOL} P(t, ...)$$

- Need a logic which is opaque to equality reasoning (aka Referential Opacity): Modal Logics

Referential opacity vs. Referential transparency

- Consider the assertion “Lois knows that Superman can fly”
- Consider the FOL formalization: $Knows(Lois, CanFly(Superman))$
- Minor Problem: $CanFly(Superman)$ is a formula

  $\Longrightarrow$ cannot occur as argument of a predicate
  $\Longrightarrow$ must apply reification $\Longrightarrow$ make it a term
- Major Problem (Referential Transparency of FOL):
  - since Superman is Clark Kent (but Lois doesn’t know it!), FOL allows to conclude “Lois knows that Clark Kent can fly”:
    $Superman = Clark \wedge Knows(Lois, CanFly(Superman))$
    $\models_{FOL} Knows(Lois, CanFly(Clark))$

  $\Longrightarrow$ Wrong inference! (Lois doesn’t know Clark Kent can fly!)
- Hint: FOL predicates transparent to equality reasoning:

$$t = s \wedge P(s, ...) \models_{FOL} P(t, ...)$$

- Need a logic which is opaque to equality reasoning (aka Referential Opacity): Modal Logics

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not": $K_{Lois}(K_{Clark} Identity(Superman, Clark) \vee K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \wedge Q) \iff K_A P \wedge K_A Q$
  - $K_A P \vee K_A Q \models K_A(P \vee Q)$, but $K_A(P \vee Q) \not\models K_A P \vee K_A Q$ (e.g. $K_A(P \vee \neg P) \not\models K_A P \vee K_A \neg P$)
- The following axiom holds in all (normal) modal logics: $K : (K_A \phi \wedge K_A(\phi \to \psi)) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \wedge K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not":
    $K_{Lois}(K_{Clark} Identity(Superman, Clark) \lor K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \land Q) \iff K_A P \land K_A Q$
  - $K_A P \lor K_A Q \models K_A(P \lor Q)$, but $K_A(P \lor Q) \not\models K_A P \lor K_A Q$ (e.g. $K_A(P \lor \neg P) \not\models K_A P \lor K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \land K_A(\phi \to \psi) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \land K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not":
    $K_{Lois}(K_{Clark} Identity(Superman, Clark) \vee K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \wedge Q) \Longleftrightarrow K_A P \wedge K_A Q$
  - $K_A P \vee K_A Q \models K_A(P \vee Q)$, but $K_A(P \vee Q) \not\models K_A P \vee K_A Q$ (e.g. $K_A(P \vee \neg P) \not\models K_A P \vee K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \wedge K_A(\phi \rightarrow \psi)) \rightarrow K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \rightarrow \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \rightarrow K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \rightarrow K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \wedge K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not":
    $K_{Lois}(K_{Clark} Identity(Superman, Clark) \lor K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \land Q) \iff K_A P \land K_A Q$
  - $K_A P \lor K_A Q \models K_A(P \lor Q)$, but $K_A(P \lor Q) \not\models K_A P \lor K_A Q$ (e.g. $K_A(P \lor \neg P) \not\models K_A P \lor K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \land K_A(\phi \to \psi)) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \land K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not":
    $K_{Lois}(K_{Clark} Identity(Superman, Clark) \vee K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \wedge Q) \Longleftrightarrow K_A P \wedge K_A Q$
  - $K_A P \vee K_A Q \models K_A(P \vee Q)$, but $K_A(P \vee Q) \not\models K_A P \vee K_A Q$ (e.g. $K_A(P \vee \neg P) \not\models K_A P \vee K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \wedge K_A(\phi \to \psi)) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \wedge K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not": $K_{Lois}(K_{Clark} Identity(Superman, Clark) \lor K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \land Q) \iff K_A P \land K_A Q$
  - $K_A P \lor K_A Q \models K_A(P \lor Q)$, but $K_A(P \lor Q) \not\models K_A P \lor K_A Q$ (e.g. $K_A(P \lor \neg P) \not\models K_A P \lor K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \land K_A(\phi \to \psi)) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \land K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not": $K_{Lois}(K_{Clark} Identity(Superman, Clark) \lor K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \land Q) \iff K_A P \land K_A Q$
  - $K_A P \lor K_A Q \models K_A(P \lor Q)$, but $K_A(P \lor Q) \not\models K_A P \lor K_A Q$ (e.g. $K_A(P \lor \neg P) \not\models K_A P \lor K_A \neg P$)
- The following axiom holds in all (normal) modal logics: $K : (K_A \phi \land K_A(\phi \rightarrow \psi)) \rightarrow K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \rightarrow \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \rightarrow K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \rightarrow K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \land K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $K_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $K_{Lois} CanFly(Superman)$
  - ex: "Lois knows Clark Kent knows if he is Superman or not": $K_{Lois}(K_{Clark} Identity(Superman, Clark) \lor K_{Clark} \neg Identity(Superman, Clark))$
- Properties in all modal logics:
  - $K_A(P \land Q) \iff K_A P \land K_A Q$
  - $K_A P \lor K_A Q \models K_A(P \lor Q)$, but $K_A(P \lor Q) \not\models K_A P \lor K_A Q$ (e.g. $K_A(P \lor \neg P) \not\models K_A P \lor K_A \neg P$)
- The following axiom holds in all (normal) modal logics:
  $K : (K_A \phi \land K_A(\phi \to \psi) \to K_A \psi$ (distribution axiom): "A is able to perform propositional inference"
- The following axioms hold in some (normal) modal logics:
  $T : K_A \varphi \to \varphi$ (knowledge axiom): "A knows only true facts"
  $4 : K_A \varphi \to K_A K_A \varphi$ (positive-introspection axiom): "If A knows fact φ, then [s]he knows [s]he knows it"
  $5 : \neg K_A \varphi \to K_A \neg K_A \varphi$ (negative-introspection axiom):
  "If A doesn't know φ, then [s]he knows [s]he doesn't know it"
- Referential Opacity: $Superman = Clark \land K_{Lois} CanFly(Superman) \not\models K_{Lois} CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard (most often even PSPACE-hard)

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible world states w_i (aka worlds, states)
  - possible states are connected in a graph by accessibility relations
  - one relation for each distinct modal operator K_A
- w_1 is accessible from w_0 wrt. K_A if everything which holds in w_1 is consistent with what A knows in w_0 (written "$Acc(K_A, w_0, w_1)$" or "$w_0 \xmapsto{K_A} w_1$")

  $\Longrightarrow$ $K_A\varphi$ holds in w_o iff φ holds in every state w_i accessible from w_0
  - the more is known in w_0, the less states are accessible from w_0
  - remark: two possible states may differ also for what an agent knows there
- Different modal logics differ by different properties of $Acc(K_A, ...)$
  - $T : K_A\varphi \rightarrow \varphi$ holds iff $Acc(K_A, ...)$ reflexive: $w \xmapsto{K_A} w$
  - $4 : K_A\varphi \rightarrow K_A K_A\varphi$ holds iff $Acc(K_A, ...)$ transitive: $w_0 \xmapsto{K_A} w_1$ and $w_1 \xmapsto{K_A} w_2 \Longrightarrow w_0 \xmapsto{K_A} w_2$
  - $5 : \neg K_A\varphi \rightarrow K_A\neg K_A\varphi$ holds iff $Acc(K_A, ...)$ euclidean: $w_0 \xmapsto{K_A} w_1$ and $w_0 \xmapsto{K_A} w_2 \Longrightarrow w_1 \xmapsto{K_A} w_2$

Semantics of Modal Logics

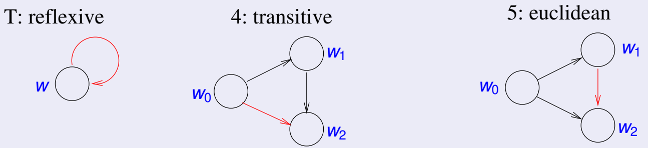
- A model (Kripke model) is a collection of possible world states w_i (aka worlds, states)
  - possible states are connected in a graph by accessibility relations
  - one relation for each distinct modal operator K_A
- w_1 is accessible from w_0 wrt. K_A if everything which holds in w_1 is consistent with what A knows in w_0 (written "$Acc(K_A, w_0, w_1)$" or "$w_0 \xmapsto{K_A} w_1$")

  $\Longrightarrow$ $K_A\varphi$ holds in w_o iff φ holds in every state w_i accessible from w_0
  - the more is known in w_0, the less states are accessible from w_0
  - remark: two possible states may differ also for what an agent knows there
- Different modal logics differ by different properties of $Acc(K_A, ...)$
  - $T : K_A\varphi \rightarrow \varphi$ holds iff $Acc(K_A, ...)$ reflexive: $w \xmapsto{K_A} w$
  - $4 : K_A\varphi \rightarrow K_AK_A\varphi$ holds iff $Acc(K_A, ...)$ transitive: $w_0 \xmapsto{K_A} w_1$ and $w_1 \xmapsto{K_A} w_2 \Longrightarrow w_0 \xmapsto{K_A} w_2$
  - $5 : \neg K_A\varphi \rightarrow K_A\neg K_A\varphi$ holds iff $Acc(K_A, ...)$ euclidean: $w_0 \xmapsto{K_A} w_1$ and $w_0 \xmapsto{K_A} w_2 \Longrightarrow w_1 \xmapsto{K_A} w_2$

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible world states w_i (aka worlds, states)
  - possible states are connected in a graph by accessibility relations
  - one relation for each distinct modal operator K_A
- w_1 is accessible from w_0 wrt. K_A if everything which holds in w_1 is consistent with what A knows in w_0 (written "$Acc(K_A, w_0, w_1)$" or "$w_0 \stackrel{K_A}{\longmapsto} w_1$")

  $\Longrightarrow$ $K_A\varphi$ holds in w_o iff φ holds in every state w_i accessible from w_0
  - the more is known in w_0, the less states are accessible from w_0
  - remark: two possible states may differ also for what an agent knows there
- Different modal logics differ by different properties of $Acc(K_A, ...)$
  - $T : K_A\varphi \rightarrow \varphi$ holds iff $Acc(K_A, ...)$ reflexive: $w \stackrel{K_A}{\longmapsto} w$
  - $4 : K_A\varphi \rightarrow K_AK_A\varphi$ holds iff $Acc(K_A, ...)$ transitive: $w_0 \stackrel{K_A}{\longmapsto} w_1$ and $w_1 \stackrel{K_A}{\longmapsto} w_2 \Longrightarrow w_0 \stackrel{K_A}{\longmapsto} w_2$
  - $5 : \neg K_A\varphi \rightarrow K_A\neg K_A\varphi$ holds iff $Acc(K_A, ...)$ euclidean: $w_0 \stackrel{K_A}{\longmapsto} w_1$ and $w_0 \stackrel{K_A}{\longmapsto} w_2 \Longrightarrow w_1 \stackrel{K_A}{\longmapsto} w_2$



Semantics of Modal Logics: Some Remarks

Assume the knowledge of A is correct: $T : K_A\varphi \rightarrow \varphi$ ("Everything which A knows holds")

- $\not\models \varphi \rightarrow K_A\varphi$: A does not know everything which holds!
- The less states are accessible, the more precise is the knowledge of A
  - uncertainty on some information makes accessible states different $\Longrightarrow$ A does not know the state [s]he is
  - complete knowledge: current state is the only successor of itself $\Longrightarrow$ A knows exactly the state [s]he is

Notice the difference:

- $K_A\neg P$: agent A knows that P does not hold (in all accessible states P is false)
- $\neg K_A P$: agent A does not know if P holds (in some accessible states P is false)

$\Longrightarrow$ $K_A\neg P \models \neg K_A P$, but $\neg K_A P \not\models K_A\neg P$

Semantics of Modal Logics: Some Remarks

Assume the knowledge of A is correct: $T : K_A\varphi \rightarrow \varphi$ ("Everything which A knows holds")

- $\not\models \varphi \rightarrow K_A\varphi$: A does not know everything which holds!
- The less states are accessible, the more precise is the knowledge of A
  - uncertainty on some information makes accessible states different $\Longrightarrow$ A does not know the state [s]he is
  - complete knowledge: current state is the only successor of itself $\Longrightarrow$ A knows exactly the state [s]he is

Notice the difference:

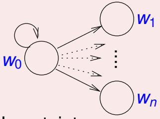
- $K_A\neg P$: agent A knows that P does not hold (in all accessible states P is false)
- $\neg K_A P$: agent A does not know if P holds (in some accessible states P is false)

$\Longrightarrow$ $K_A\neg P \models \neg K_A P$, but $\neg K_A P \not\models K_A\neg P$

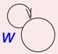
Semantics of Modal Logics: Some Remarks

Assume the knowledge of A is correct: $T : K_A\varphi \rightarrow \varphi$ ("Everything which A knows holds")

- $\not\models \varphi \rightarrow K_A\varphi$: A does not know everything which holds!
- The less states are accessible, the more precise is the knowledge of A
  - uncertainty on some information makes accessible states different
    $\Longrightarrow$ A does not know the state [s]he is
  - complete knowledge: current state is the only successor of itself
    $\Longrightarrow$ A knows exactly the state [s]he is



Uncertainty



Complete knowledge

Notice the difference:

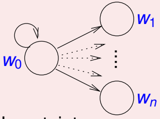
- $K_A\neg P$: agent A knows that P does not hold (in all accessible states P is false)
- $\neg K_A P$: agent A does not know if P holds (in some accessible states P is false)

$\Longrightarrow$ $K_A\neg P \models \neg K_A P$, but $\neg K_A P \not\models K_A\neg P$

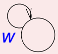
Semantics of Modal Logics: Some Remarks

Assume the knowledge of A is correct: $T: K_A\varphi \rightarrow \varphi$ ("Everything which A knows holds")

- $\not\models \varphi \rightarrow K_A\varphi$: A does not know everything which holds!
- The less states are accessible, the more precise is the knowledge of A
  - uncertainty on some information makes accessible states different $\Longrightarrow$ A does not know the state [s]he is
  - complete knowledge: current state is the only successor of itself $\Longrightarrow$ A knows exactly the state [s]he is



Uncertainty



Complete knowledge

Notice the difference:

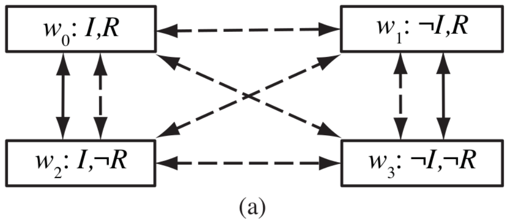
- $K_A\neg P$: agent A knows that P does not hold (in all accessible states P is false)
- $\neg K_A P$: agent A does not know if P holds (in some accessible states P is false)

$\Longrightarrow$ $K_A\neg P \models \neg K_A P$, but $\neg K_A P \not\models K_A\neg P$

Semantics of Modal Logics: Example

Accessibility relations: $K_{Superman}$ (solid arrows) and K_{Lois} (dotted arrows).

- Legenda:
  - R: "the weather report says tomorrow will rain"
  - I: "Superman's secret identity is Clark Kent."
  - Ex: $K_{Lois}(K_{Clark} I \lor K_{Clark} \neg I)$: "Lois Knows that Clark Knows if he is Superman or not."
- Superman knows his own identity: $K_{Superman} I \lor K_{Superman} \neg I$, and
  (a) neither Superman nor Lois has seen the weather report, she knows Superman knows if he is Clark
  $(\neg K_{Lois} R \land \neg K_{Lois} \neg R) \land (\neg K_{Superman} R \land \neg K_{Superman} \neg R) \land K_{Lois}(K_{Superman} I \lor K_{Superman} \neg I)$



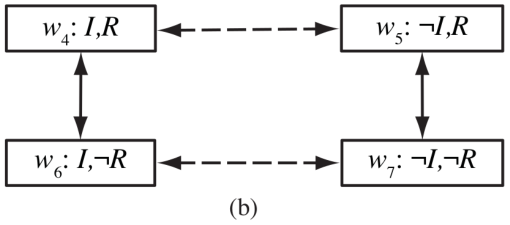
(self-loop arrows not reported)

(© S. Russell & P. Norwig, AIMA)

Semantics of Modal Logics: Example

Accessibility relations: $K_{Superman}$ (solid arrows) and K_{Lois} (dotted arrows).

- Legenda:
  - R: "the weather report says tomorrow will rain"
  - I: "Superman's secret identity is Clark Kent."
  - Ex: $K_{Lois}(K_{Clark} I \vee K_{Clark} \neg I)$: "Lois Knows that Clark Knows if he is Superman or not."
- Superman knows his own identity: $K_{Superman} I \vee K_{Superman} \neg I$, and
  (b) Lois has seen the weather report, Superman has not, but he knows that Lois has seen it
  $(K_{Lois} R \vee K_{Lois} \neg R) \wedge (\neg K_{Superman} R \wedge \neg K_{Superman} \neg R)$
  $K_{Lois}(K_{Superman} I \vee K_{Superman} \neg I) \wedge K_{Superman}(K_{Lois} R \vee K_{Lois} \neg R)$



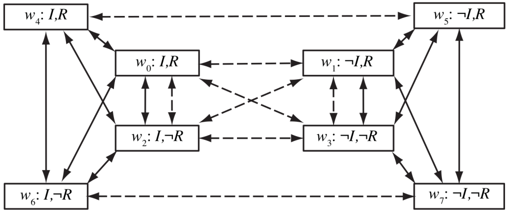
(© S. Russell & P. Norwig, AIMA)

(self-loop arrows not reported)

Semantics of Modal Logics: Example

Accessibility relations: $K_{Superman}$ (solid arrows) and K_{Lois} (dotted arrows).

- Legenda:
  - R: “the weather report says tomorrow will rain”
  - I: “Superman’s secret identity is Clark Kent.”
  - Ex: $K_{Lois}(K_{Clark} I \vee K_{Clark} \neg I)$: “Lois Knows that Clark Knows if he is Superman or not.”
- Superman knows his own identity: $K_{Superman} I \vee K_{Superman} \neg I$, and
  (c) Lois may or may not have seen the weather report, Superman has not:
  $((\neg K_{Lois} R \wedge \neg K_{Lois} \neg R) \vee (K_{Lois} R \vee K_{Lois} \neg R)) \wedge (\neg K_{Sup.} R \wedge \neg K_{Sup.} \neg R)$
  $K_{Lois}(K_{Superman} I \vee K_{Superman} \neg I)$



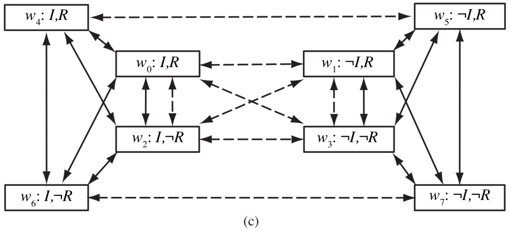
(c)

(self-loop arrows not reported)

Semantics of Modal Logics: Example

Accessibility relations: $K_{Superman}$ (solid arrows) and K_{Lois} (dotted arrows).

- Legenda:
  - R: "the weather report says tomorrow will rain"
  - I: "Superman's secret identity is Clark Kent."
  - Ex: $K_{Lois}(K_{Clark} I \vee K_{Clark} \neg I)$: "Lois Knows that Clark Knows if he is Superman or not."
- Superman knows his own identity: $K_{Superman} I \vee K_{Superman} \neg I$, and
  (c) Lois may or may not have seen the weather report, Superman has not:
  $((\neg K_{Lois} R \wedge \neg K_{Lois} \neg R) \vee (K_{Lois} R \vee K_{Lois} \neg R)) \wedge (\neg K_{Sup.} R \wedge \neg K_{Sup.} \neg R)$
  $K_{Lois}(K_{Superman} I \vee K_{Superman} \neg I)$



(self-loop arrows not reported)

(© S. Russell & P. Norwig, AIMA)

Exercise

Consider the previous example.

- For each scenario (a), (b) and (c) define doubly-nested knowledge in terms of

  $[\neg]K_{Lois}[\neg]K_{Lois}[\neg]I$,
  $[\neg]K_{Lois}[\neg]K_{Lois}[\neg]R$,
  $[\neg]K_{Sup.}[\neg]K_{Sup.}[\neg]I$,
  $[\neg]K_{Sup.}[\neg]K_{Sup.}[\neg]R$

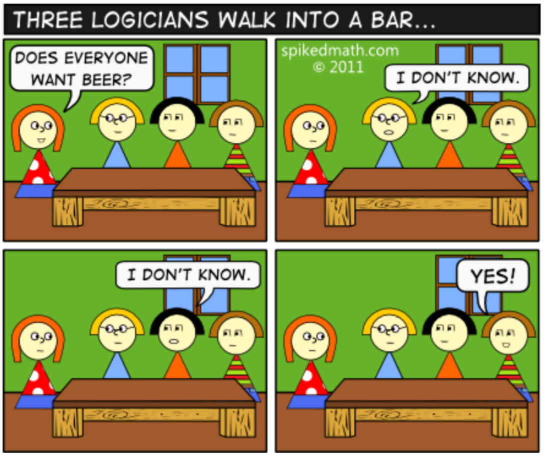
Exercise

Consider (normal) modal logics (i.e., axioms K, T, 4 and 5 hold).
Let IsRed(Pen), IsOnTable(Pen) be possible facts, let *Mary*, *John* be agents and let K_{Mary}, K_{John} denote the modal operators "Mary knows that..." and "John knows that..." respectively.
For each of the following facts, say if it is true or false.

- If $K_{Mary}\neg$IsRed(Pen) holds, then $\neg K_{Mary}$IsRed(Pen) holds
- If $\neg K_{Mary}$IsRed(Pen) holds, then $K_{Mary}\neg$IsRed(Pen) holds
- If K_{John}IsRed(Pen) and IsRed(Pen) $\leftrightarrow$ IsOnTable(Pen) hold, then K_{John}IsOnTable(Pen) holds
- If K_{Mary}IsRed(Pen) and K_{Mary}(IsRed(Pen) $\rightarrow$ $\mathrm{K}_{\mathrm{John}}$IsRed(Pen)) hold, then $K_{Mary}K_{John}$IsRed(Pen)) holds

Exercise

- Why does the third logician answers "Yes"?
- Formalize and solve the problem by means of modal logic (K+T+4+5)



(Courtesy of Maria Simi, UniPI)

Outline

Outline

Reasoning Systems for Categories

Q. How to organize and reason with categories?

- Semantic Networks
  - allow to visualize knowledge bases
  - efficient algorithms for category membership inference
  - limited expressivity
  - many variants
- Description Logics (DLs)
  - formal language for constructing and combining category definitions
  - (relatively) efficient algorithms to decide subset and superset relationships between categories
  - many DLs
    - up to very high expressivity
    - up to very high complexity (e.g., DOUBLY-EXPTIME)

Reasoning Systems for Categories

Q. How to organize and reason with categories?

- Semantic Networks
  - allow to visualize knowledge bases
  - efficient algorithms for category membership inference
  - limited expressivity
  - many variants
- Description Logics (DLs)
  - formal language for constructing and combining category definitions
  - (relatively) efficient algorithms to decide subset and superset relationships between categories
  - many DLs
    - up to very high expressivity
    - up to very high complexity (e.g., DOUBLY-EXPTIME)

Reasoning Systems for Categories

Q. How to organize and reason with categories?

- Semantic Networks
  - allow to visualize knowledge bases
  - efficient algorithms for category membership inference
  - limited expressivity
  - many variants
- Description Logics (DLs)
  - formal language for constructing and combining category definitions
  - (relatively) efficient algorithms to decide subset and superset relationships between categories
  - many DLs
    - up to very high expressivity
    - up to very high complexity (e.g., DOUBLY-EXPTIME)

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

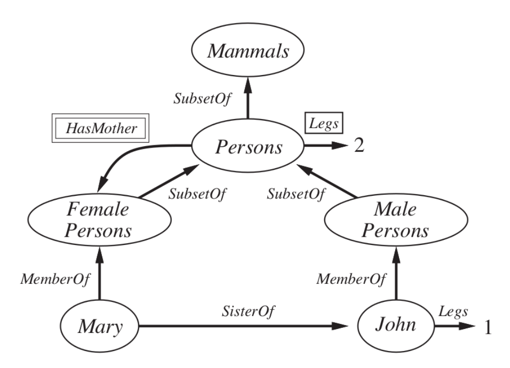
- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks

- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - Generic concepts, corresponding to categories/classes
  - Individual concepts, corresponding to individuals
- Two special relations are always present, with different names
  - IS-A, aka SubsetOf/SubclassOf (subclass)
  - InstanceOf aka MemberOf (membership)
- Inheritance detection straightforward
- Ability to represent default values for categories
- Limited expressive power: cannot represent negation, disjunction, nested function symbols, existential quantification

Semantic Networks: Example

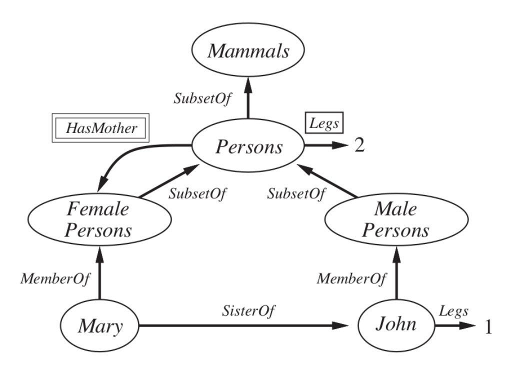
- Notice
  - "HasMother" is a relation between persons (individuals) (categories do not have mothers)
  - "HasMother" (double-boxed notation) means
    $\forall x.(x \in Persons \rightarrow [\forall y.(HasMother(x,y) \rightarrow y \in FemalePersons)])$
  - "Legs" is a property of single persons (individuals)
  - "Legs" (single-boxed notation) means:
    $\forall x.(x \in Persons \rightarrow Legs(x, 2))$



(© S. Russell & P. Norwig, AIMA)

Semantic Networks: Example

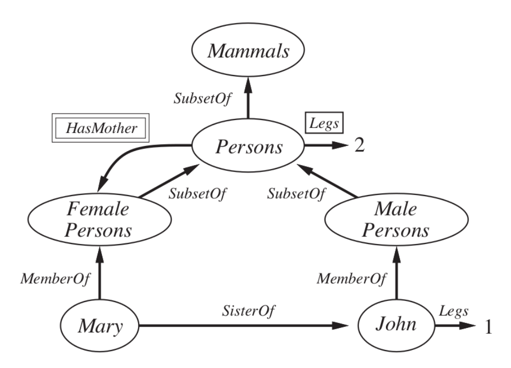
- Notice
  - "HasMother" is a relation between persons (individuals) (categories do not have mothers)
  - "HasMother" (double-boxed notation) means
    $\forall x.(x \in Persons \rightarrow [\forall y.(HasMother(x, y) \rightarrow y \in FemalePersons)])$
  - "Legs" is a property of single persons (individuals)
  - "Legs" (single-boxed notation) means:
    $\forall x.(x \in Persons \rightarrow Legs(x, 2))$



(© S. Russell & P. Norwig, AIMA)

Semantic Networks: Example

- Notice
  - "HasMother" is a relation between persons (individuals) (categories do not have mothers)
  - "HasMother" (double-boxed notation) means
    $\forall x.(x \in Persons \rightarrow [\forall y.(HasMother(x, y) \rightarrow y \in FemalePersons)])$
  - "Legs" is a property of single persons (individuals)
  - "Legs" (single-boxed notation) means:
    $\forall x.(x \in Persons \rightarrow Legs(x, 2))$

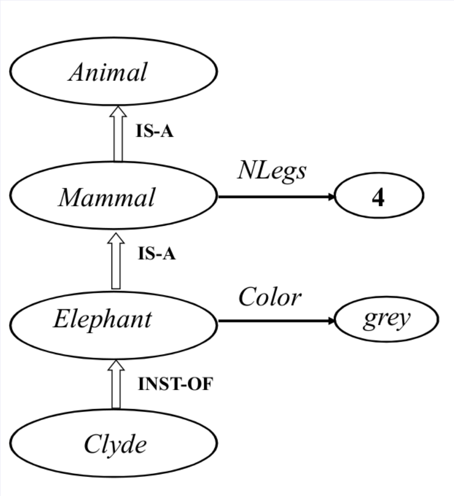


Inheritance in Semantic Networks

- Inheritance conveniently implemented as link traversal

Q. How many legs has Clyde?

$\Longrightarrow$ follow the INST-OF/IS-A chain until find the property NLegs



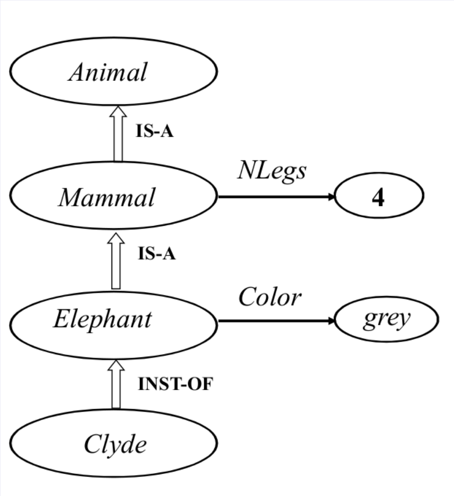
(Courtesy of Maria Simi, UniPI)

Inheritance in Semantic Networks

- Inheritance conveniently implemented as link traversal

Q. How many legs has Clyde?

⟹ follow the INST-OF/IS-A chain until find the property NLegs



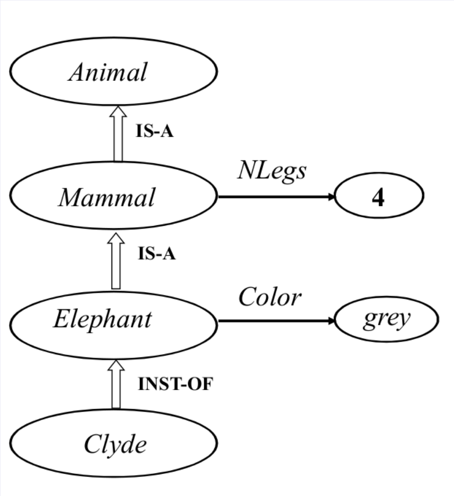
(Courtesy of Maria Simi, UniPI)

Inheritance in Semantic Networks

- Inheritance conveniently implemented as link traversal

Q. How many legs has Clyde?

$\Longrightarrow$ follow the INST-OF/IS-A chain until find the property NLegs



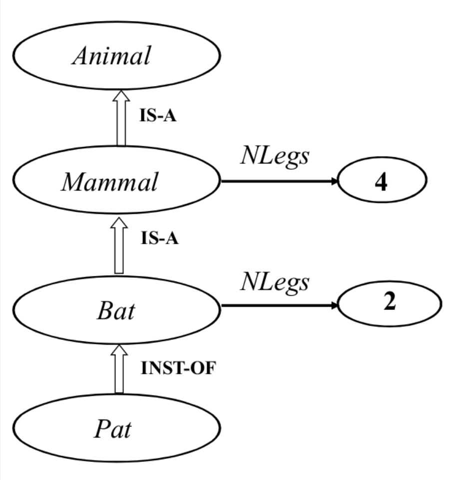
(Courtesy of Maria Simi, UniPI)

Inheritance with Exceptions

The presence of exceptions does not create any problem with S.N.

- How many legs has Pat?
- Just take the most specific information: the first that is found going up the hierarchy

⟹ ability to represent default values for categories



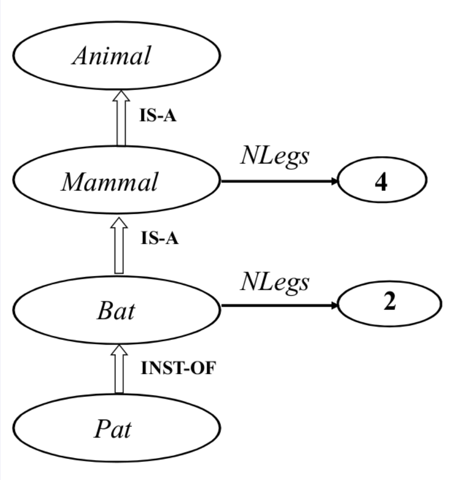
(Courtesy of Maria Simi, UniPI)

Inheritance with Exceptions

The presence of exceptions does not create any problem with S.N.

- How many legs has Pat?
- Just take the most specific information: the first that is found going up the hierarchy

⟹ ability to represent default values for categories



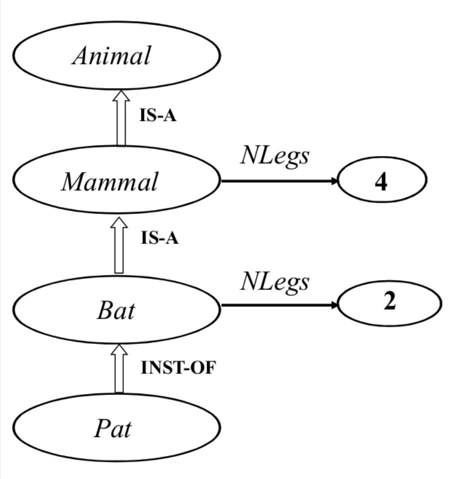
(Courtesy of Maria Simi, UniPI)

Inheritance with Exceptions

The presence of exceptions does not create any problem with S.N.

- How many legs has Pat?
- Just take the most specific information: the first that is found going up the hierarchy

$\Longrightarrow$ ability to represent default values for categories



(Courtesy of Maria Simi, UniPI)

Encoding N-Ary Relations

- Semantic networks allow only binary relations

Q. How to represent n-ary relations?

$\Longrightarrow$ Reify the proposition as an event belonging to an appropriate event category

- ex "Fly_{17}" for $Fly(Shankar, NewYork, NewDelhi, Yesterday)$

Encoding N-Ary Relations

- Semantic networks allow only binary relations

Q. How to represent n-ary relations?

⟹ Reify the proposition as an event belonging to an appropriate event category

- ex "Fly_{17}" for $Fly(Shankar, NewYork, NewDelhi, Yesterday)$

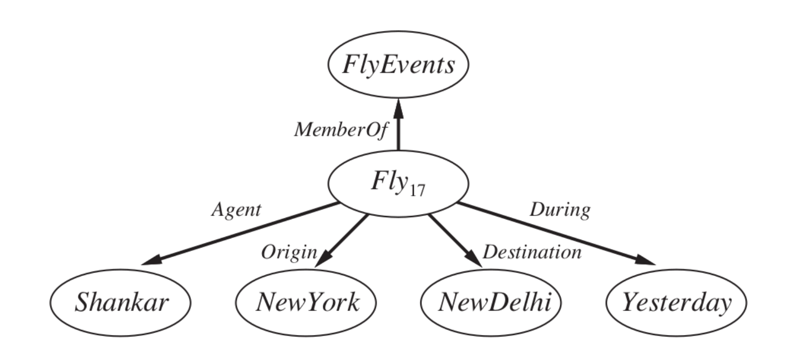
Encoding N-Ary Relations

- Semantic networks allow only binary relations

Q. How to represent n-ary relations?

⟹ Reify the proposition as an event belonging to an appropriate event category

  - ex "Fly_{17}" for *Fly*(*Shankar*, *NewYork*, *NewDelhi*, *Yesterday*)



(© S. Russell & P. Norwig, AIMA)

Outline

1. Ontologies and Ontological Engineering
2. Categories and Objects
3. Reasoning about Knowledge
4. Reasoning about Categories
   - Semantic Networks (hints)
   - Description Logics

Description Logics

- Designed to describe definitions and properties about categories
- Principal inference tasks:
  - Subsumption: check if one category is a subset (sub-category) of another
  - Classification: check whether an object belongs to a category
  - Consistency: check if category membership criteria are satisfiable
- Defaults and exceptions are lost

Description Logics

- Designed to describe definitions and properties about categories
- Principal inference tasks:
  - Subsumption: check if one category is a subset (sub-category) of another
  - Classification: check whether an object belongs to a category
  - Consistency: check if category membership criteria are satisfiable
- Defaults and exceptions are lost

Description Logics

- Designed to describe definitions and properties about categories
- Principal inference tasks:
  - Subsumption: check if one category is a subset (sub-category) of another
  - Classification: check whether an object belongs to a category
  - Consistency: check if category membership criteria are satisfiable
- Defaults and exceptions are lost

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts: and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children: $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists: $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top$, $\bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts: and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children: $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists: $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - $\top, \bot$: universal and empty concepts
  - atomic concepts: ex: *Female*, *Male*, *Article*, *Journalist*,...
  - operators for the construction of complex concepts:
    and ($\sqcap$), or ($\sqcup$), not ($\neg$), all ($\forall$), some ($\exists$), atleast ($\geq n$), atmost ($\leq n$), ...
  - ex: mothers (i.e., women who have children) of at least three female children:
    $Woman \sqcap \exists hasChildren.Person \sqcap \geq 3\ hasChild.Female$
  - ex: articles that have authors and whose authors are all journalists:
    $Article \sqcap \exists hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $hasChildren \equiv hasSon \sqcup hasDaughter$
- Individuals (used in assertions only)
  - ex: Mary, John

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Boxes and A-Boxes

- Terminologies (T-Boxes): sets of
  - concepts definitions ($C_1 \equiv C_2$)
    ex: $Father \equiv Man \sqcap \exists hasChild.Person$
  - or concept generalizations ($C_1 \sqsubseteq C_2$)
    ex: $Woman \sqsubseteq Person$
- Assertions (A-Boxes): assert
  - individuals as concept members $i : C$,
    where i is an individual and C is a concept
    ex: $mary : Person$, $john : Father$
  - individual pairs as relation members $\langle i, j \rangle : R$,
    where i,j are individuals and R is a relation
    ex: $\langle john, mary \rangle : hasChild$

T-Box: Example (Logic $\mathcal{ALCN}$)

$$
\begin{aligned}
\mathsf{Woman} &\equiv \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} &\equiv \mathsf{Person} \sqcap \neg\, \mathsf{Woman} \\
\mathsf{Mother} &\equiv \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} &\equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} &\equiv \mathsf{Father} \sqcup \mathsf{Mother} \\
\mathsf{Grandmother} &\equiv \mathsf{Mother} \sqcap \exists \mathsf{hasChild}.\mathsf{Parent} \\
\mathsf{MotherWithManyChildren} &\equiv \mathsf{Mother} \sqcap \geqslant 3\, \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{MotherWithoutDaughter} &\equiv \mathsf{Mother} \sqcap \forall \mathsf{hasChild}.\neg\, \mathsf{Woman} \\
\mathsf{Wife} &\equiv \mathsf{Woman} \sqcap \exists \mathsf{hasHusband}.\mathsf{Man}
\end{aligned}
$$

(Courtesy of Maria Simi, UniPI)

Reasoning Services for DLs

- Design and management of ontologies
  - consistency checking of concepts, creation of hierarchies
- Ontology integration
  - Relations between concepts of different ontologies
  - Consistency of integrated hierarchies
- Queries
  - Determine whether facts are consistent wrt ontologies
  - Determine if individuals are instances of concepts
  - Retrieve individuals satisfying a query (concept)
  - Verify if a concept is more general than another (subsumption)

Reasoning Services for DLs

- Design and management of ontologies
  - consistency checking of concepts, creation of hierarchies
- Ontology integration
  - Relations between concepts of different ontologies
  - Consistency of integrated hierarchies
- Queries
  - Determine whether facts are consistent wrt ontologies
  - Determine if individuals are instances of concepts
  - Retrieve individuals satisfying a query (concept)
  - Verify if a concept is more general than another (subsumption)

Reasoning Services for DLs

- Design and management of ontologies
  - consistency checking of concepts, creation of hierarchies
- Ontology integration
  - Relations between concepts of different ontologies
  - Consistency of integrated hierarchies
- Queries
  - Determine whether facts are consistent wrt ontologies
  - Determine if individuals are instances of concepts
  - Retrieve individuals satisfying a query (concept)
  - Verify if a concept is more general than another (subsumption)

Querying a DL Ontology: Example

All the children of John are females. Mary is a child of John.
Tim is a friend of professor Blake. Prove that Mary is a female.

- $\mathcal{A} \stackrel{\text{def}}{=} \{john : \forall hasChild.female, (john, mary) : hasChild,$
  $(blake, tim) : hasFriend, blake : professor\}$
- Query: $mary : female$ (or: is $\mathcal{A} \sqcap mary : \neg female$ unsatisfiable?)
- Yes

Querying a DL Ontology: Example

All the children of John are females. Mary is a child of John.
Tim is a friend of professor Blake. Prove that Mary is a female.

- $\mathcal{A} \stackrel{\text{def}}{=} \{john : \forall hasChild.female, (john, mary) : hasChild, (blake, tim) : hasFriend, blake : professor\}$
- Query: $mary : female$ (or: is $\mathcal{A} \sqcap mary : \neg female$ unsatisfiable?)
- Yes

Exercise

Given:

- a set of basic concepts: $\{\text{Person}, \text{Male}, \text{Doctor}, \text{Engineer}\}$
- a set of relations: $\{\text{hasChild}\}$

with their obvious meaning. Write a $\mathcal{T}$-box in $\mathcal{ALCN}$ defining the following concepts

(*a*) Female, Man, Woman (with their standard meaning)

(*b*) femaleDoctorWithoutChildren: female doctor with no children

(*c*) fatherOfFemaleDoctor: father of at least two female doctors

(*d*) motherOfDoctorsOrEngineers: woman whose children are all engineers or [a] doctors

[a] non-exclusive or.