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

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
http://disi.unitn.it/rseba/DIDATTICA/fai_2020/

Teaching assistant: Mauro Dragoni – dragoni@fbk.eu
http://www.maurodragoni.com/teaching/fai/

M.S. Course "Artificial Intelligence Systems", academic year 2020-2021

Last update: Wednesday 9th December, 2020, 13:59

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

  $\Longrightarrow$ Knowledge Representation & Reasoning, KRR
- KR: use 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 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 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 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 be applicable for (many) specific purpose (tasks)
  - (with the addition of domain-specific axioms)
  - In non-trivial domains, reasoning and problem solving should involve several areas of knowledge simultaneously
  - The content of ontologies is independent of how it is represented
- 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)?

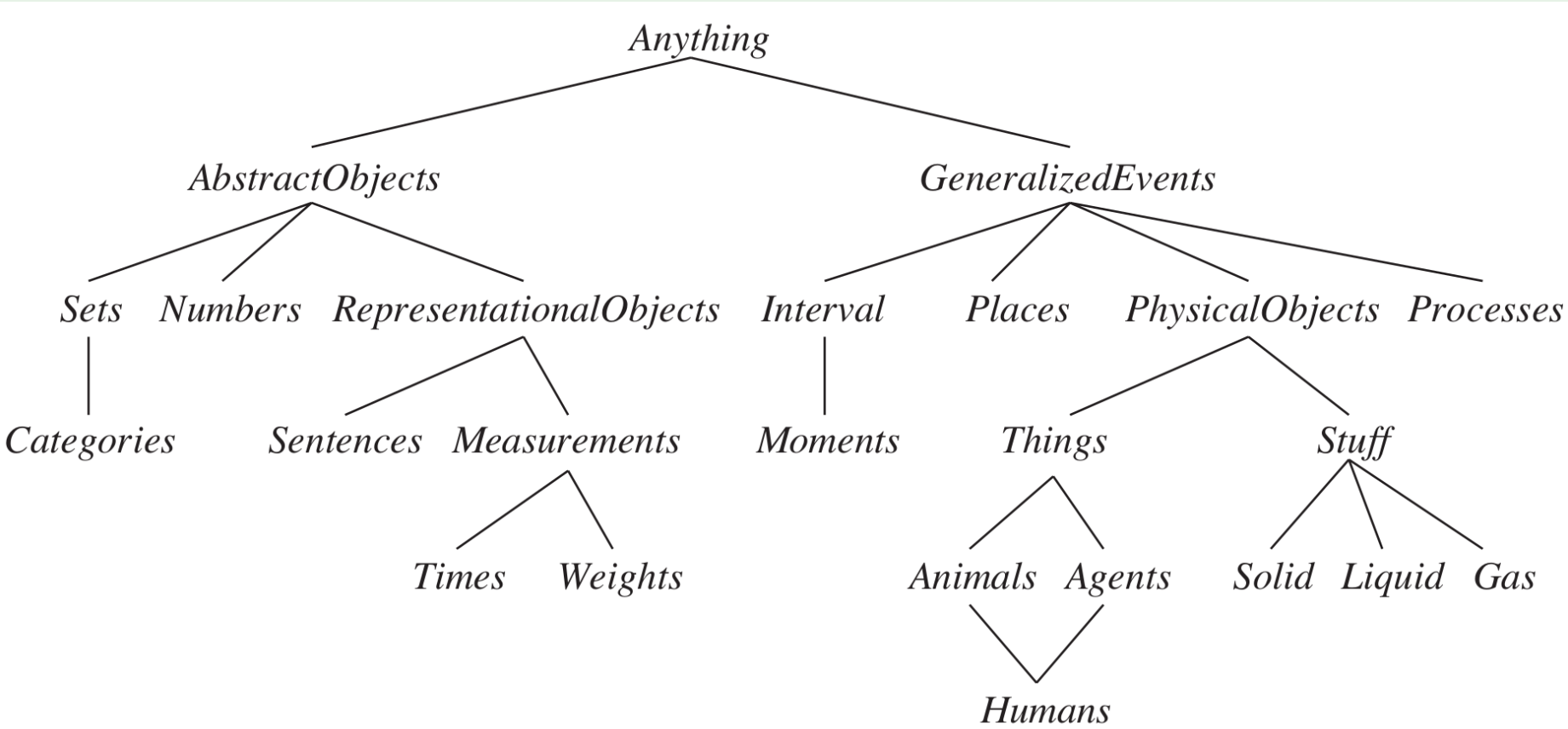
Ontological Engineering

- 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

A General-Purpose/Upper Ontology

An upper ontology of the world

- Link: the lower concept is a specialization of the upper one
  - Note: physical objects specialization of generalized events (see later)



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 $\forall x.(x \in \mathit{Food} \rightarrow \mathit{Edible}(x))$, $\mathit{Fruit} \subset \mathit{Food}$, $\mathit{Apples} \subset \mathit{Fruit}$
    then $\forall x.(x \in \mathit{Apple} \rightarrow \mathit{Edible}(x))$
- A member inherits the properties of the category
  - if $a \in \mathit{Apples}$, then $\mathit{Edible}(a)$
- 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 $\forall x.(x \in \mathit{Food} \rightarrow \mathit{Edible}(x))$, $\mathit{Fruit} \subset \mathit{Food}$, $\mathit{Apples} \subset \mathit{Fruit}$
    then $\forall x.(x \in \mathit{Apple} \rightarrow \mathit{Edible}(x))$
- A member inherits the properties of the category
  - if $a \in \mathit{Apples}$, then $\mathit{Edible}(a)$
- 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 $\forall x.(x \in \mathit{Food} \rightarrow \mathit{Edible}(x))$, $\mathit{Fruit} \subset \mathit{Food}$, $\mathit{Apples} \subset \mathit{Fruit}$
    then $\forall x.(x \in \mathit{Apple} \rightarrow \mathit{Edible}(x))$
- A member inherits the properties of the category
  - if $a \in \mathit{Apples}$, then $\mathit{Edible}(a)$
- 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
    $\forall x.((Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5'' \wedge x \in Balls) \rightarrow x \in Basketballs)$
  - 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
    $\forall x.((Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5'' \wedge x \in Balls)$
    $\rightarrow x \in Basketballs)$
  - 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.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \wedge c_2 \in s \wedge c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - 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
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \wedge i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \wedge ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{Males, Females\}, Animals)$

Categories and Objects [cont.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \wedge c_2 \in s \wedge c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - 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
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \wedge i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \wedge ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{Males, Females\}, Animals)$

Categories and Objects [cont.]

Derived relations

- Two or more categories in a set s are disjoint iff they have no members in common
  - $Disjoint(s) \leftrightarrow (\forall c_1 c_2.\ ((c_1 \in s \wedge c_2 \in s \wedge c_1 \neq c_2) \rightarrow Intersection(c_1, c_2) = \emptyset)$
  - 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
  - $ExaustiveDecomposition(s, c) \leftrightarrow \forall i.(i \in c \leftrightarrow (\exists c_2.(c_2 \in s \wedge i \in c_2)))$
  - ex: $E.D.(\{Americans, Canadians, Mexicans\}, NorthAmericans)$
- A disjoint exhaustive decomposition is a partition
  - $Partition(s, c) \leftrightarrow (Disjoint(s) \wedge ExhaustiveDecomposition(s, c))$
  - ex: $Partition(\{Males, Females\}, Animals)$

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: $\forall x.(x \in Typical(Tomatoes) \rightarrow (Red(x) \wedge Round(x)))$

$\Longrightarrow$ 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

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

- 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

- 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

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

(© S. Russell & P. Norwig, AIMA)

- 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

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

(© S. Russell & P. Norwig, AIMA)

- 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 $Length(L_1) = Inches(1.5) \wedge Inches(1.5) = Centimeters(3.81)$
- 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 $\mathit{Length}(L_1) = \mathit{Inches}(1.5) \wedge \mathit{Inches}(1.5) = \mathit{Centimeters}(3.81)$
- Conversion between units:
  - $\forall i.\ \mathit{Centimeters}(2.54 \times i) = \mathit{Inches}(i)$
- Measures can be used to describe objects:
  - ex: $\mathit{Diameter}(\mathit{Basketball}_{12}) = \mathit{Inches}(9.5)$
  - ex: $\mathit{ListPrice}(\mathit{Basketball}_{12}) = \$(19)$
  - ex: $\forall d.(d \in \mathit{Days} \rightarrow \mathit{Duration}(d) = \mathit{Hours}(24))$

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 $\mathit{Length}(L_1) = \mathit{Inches}(1.5) \wedge \mathit{Inches}(1.5) = \mathit{Centimeters}(3.81)$
- Conversion between units:
  - $\forall i.\ \mathit{Centimeters}(2.54 \times i) = \mathit{Inches}(i)$
- Measures can be used to describe objects:
  - ex: $\mathit{Diameter}(\mathit{Basketball}_{12}) = \mathit{Inches}(9.5)$
  - ex: $\mathit{ListPrice}(\mathit{Basketball}_{12}) = \$(19)$
  - ex: $\forall d.(d \in \mathit{Days} \rightarrow \mathit{Duration}(d) = \mathit{Hours}(24))$

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: $Difficulty(ProveP \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:

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

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

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

Then: $(Wrote(Norvig, E_1) \wedge Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$

- 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: $Difficulty(ProveP \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:
  $\forall e_1 e_2.((e_1 \in Exercises \wedge e_2 \in Exercises \wedge$
  $Wrote(Norvig, e_1) \wedge Wrote(Russell, e_2))$
  $\rightarrow Difficulty(e_1) > Difficulty(e_2))$
  $\forall e_1 e_2.((e_1 \in Exercises \wedge e_2 \in Exercises \wedge Difficulty(e_1) > Difficulty(e_2))$
  $\rightarrow ExpectedScore(e_1) < ExpectedScore(e_2))$
  $\forall e_1 e_2.(ExpectedScore(e_1) < ExpectedScore(e_2) \rightarrow Pick(e_1, e_2) = e_2$
  Then: $(Wrote(Norvig, E_1) \wedge Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$
- 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: $Difficulty(ProveP \neq NP) > Difficulty(SolvePuzzle)$
- Allow for reasoning by exploiting transitivity of monotonicity:
  $\forall e_1 e_2.((e_1 \in Exercises \wedge e_2 \in Exercises \wedge$
  $Wrote(Norvig, e_1) \wedge Wrote(Russell, e_2))$
  $\rightarrow Difficulty(e_1) > Difficulty(e_2))$
  $\forall e_1 e_2.((e_1 \in Exercises \wedge e_2 \in Exercises \wedge Difficulty(e_1) > Difficulty(e_2))$
  $\rightarrow ExpectedScore(e_1) < ExpectedScore(e_2))$
  $\forall e_1 e_2.(ExpectedScore(e_1) < ExpectedScore(e_2) \rightarrow Pick(e_1, e_2) = e_2$
  Then: $(Wrote(Norvig, E_1) \wedge Wrote(Russell, E_2)) \models Pick(E_1, E_2) = E_2$
- 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: $b \in \mathit{butter}$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in \mathit{Butter} \wedge \mathit{PartOf}(p, b)) \rightarrow p \in \mathit{Butter})$
- Can define sub-categories, which are stuff
  - ex: $\mathit{UnsaltedButter} \subset \mathit{Butter}$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in \mathit{Butter} \rightarrow \mathit{MeltingPoint}(b, \mathit{Centigrade}(30)))$
- 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: $b \in \mathit{butter}$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in \mathit{Butter} \wedge \mathit{PartOf}(p, b)) \rightarrow p \in \mathit{Butter})$
- Can define sub-categories, which are stuff
  - ex: $\mathit{UnsaltedButter} \subset \mathit{Butter}$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in \mathit{Butter} \rightarrow \mathit{MeltingPoint}(b, \mathit{Centigrade}(30)))$
- 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: $b \in \mathit{butter}$: "b is an amount/quantity of butter"
- Any part of stuff is still stuff:
  - ex: $\forall b, p.((b \in \mathit{Butter} \wedge \mathit{PartOf}(p, b)) \rightarrow p \in \mathit{Butter})$
- Can define sub-categories, which are stuff
  - ex: $\mathit{UnsaltedButter} \subset \mathit{Butter}$
- Stuff has a number of intrinsic properties, shared by its subparts
  - e.g., color, fat content, density ...
  - ex: $\forall b.(b \in \mathit{Butter} \rightarrow \mathit{MeltingPoint}(b, \mathit{Centigrade}(30)))$
- 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, ...

Outline

[Recall from Ch.10:] Situation Calculus

Basic concepts

- Situation:
  - the initial state is a situation
  - if s is a situation and a is an action, then $Result(s, a)$ is a situation
  - $Result()$ injective: $Result(s, a) = Result(s', a') \leftrightarrow (s = s' \wedge a = a')$
  - a solution is a situation that satisfies the goal
- Action preconditions: $\Phi(s) \rightarrow Poss(a, s)$
  - $\Phi(s)$ describes preconditions
  - ex: $(Alive(Agent, s) \wedge Have(Agent, Arrow, s)) \rightarrow Poss(Shoot, s)$
- Successor-state axioms (similar to propositional case):

$$[\text{Action is possible}] \rightarrow \left[\begin{array}{l} [\text{Fluent is true in result state}] \leftrightarrow \\ ([\text{Action's effect made it true}] \vee \\ \ ([\text{It was true before}] \wedge [\text{action left it alone}])) \end{array}\right]$$

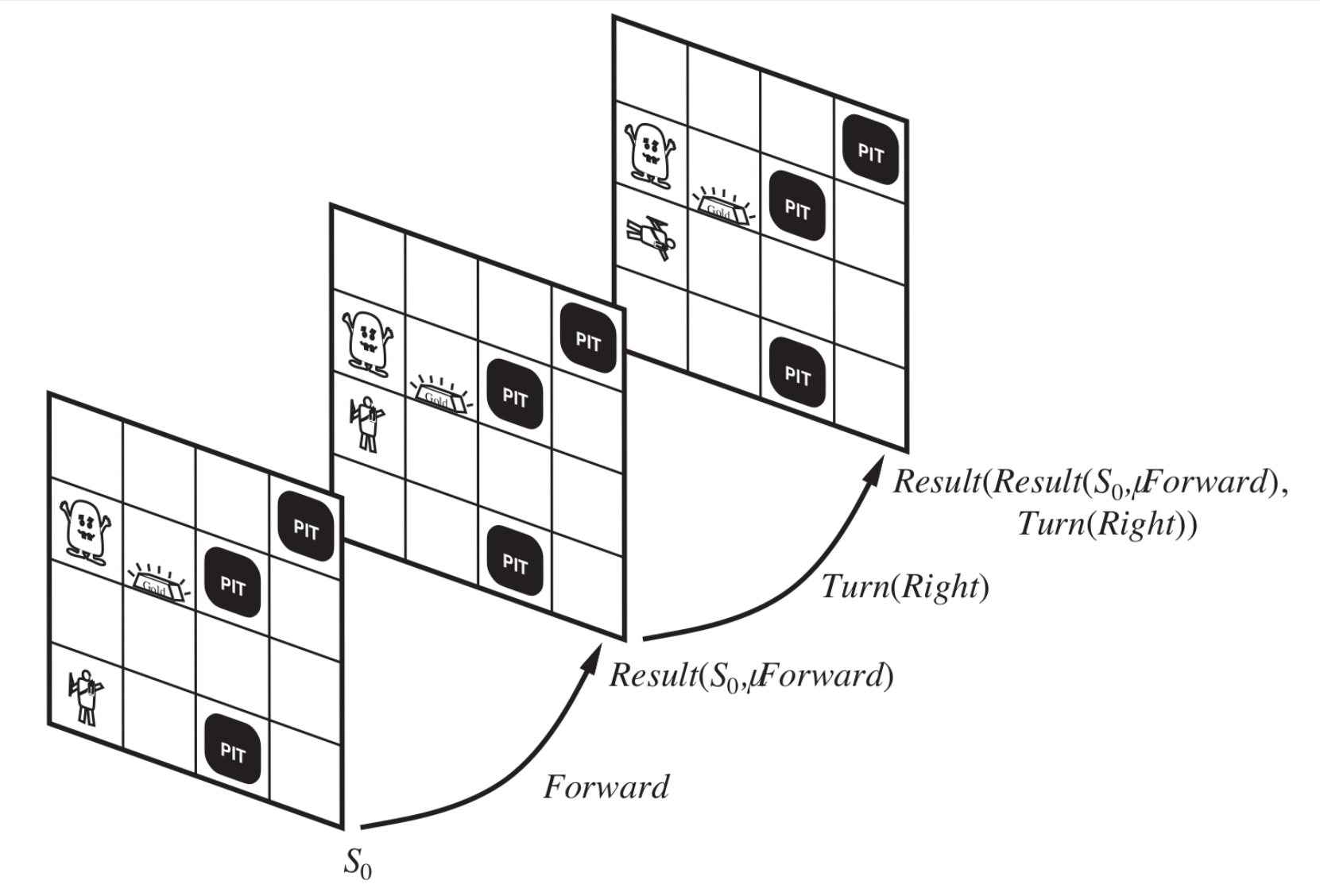
  - ex: $Poss(a, s) \rightarrow$

$$\left[\begin{array}{l} Holding(Agent, g, Result(a, s)) \leftrightarrow \\ a = Grab(g) \vee (Holding(Agent, g, s) \wedge a \neq Release(g)) \end{array}\right]$$

- Unique action axioms: $A_i(x, ...) \neq A_j(y, ...)$; A_i injective
  - ex $Shoot(x) \neq Grab(y)$

[Recall from Ch.10:] Situation Calculus: Example

Situations as the results of actions in the Wumpus world



Limitation of Situation Calculus

- Situation calculus is limited in its applicability:
  - single agent
  - actions are discrete and instantaneous (no duration in time)
  - actions happen one at a time:
    $\Longrightarrow$ no concurrency, no simultaneous actions
  - only primitive actions: no way to combine actions
    (no conditionals, no iterations, ...)

Event Calculus

- Based on events, points in time, intervals rather than situations
- Reification of fluents
  - a fluent is an object represented by a term
    (ex: $At(Shankar, Berkeley)$)
    $\Longrightarrow$ does not say if it is true
  - T (True): asserts that a fluent is true at some point in time t
    ex: $T(At(Shankar, Berkeley), t)$
- Reification of events
  - events are described as instances of event categories
  - ex: event E_1: Shankar flies from San Francisco to WashingtonDC:
    $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge$
    $Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - reification allows for adding arbitrary information about fluents
  - ex: Shankar's flight was bumpy: $Bumpy(E_1)$

Event Calculus

- Based on events, points in time, intervals rather than situations
- Reification of fluents
  - a fluent is an object represented by a term
    (ex: $At(Shankar, Berkeley)$)
    $\Longrightarrow$ does not say if it is true
  - T (True): asserts that a fluent is true at some point in time t
    ex: $T(At(Shankar, Berkeley), t)$
- Reification of events
  - events are described as instances of event categories
  - ex: event E_1: Shankar flies from San Francisco to WashingtonDC:
    $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge$
    $Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - reification allows for adding arbitrary information about fluents
  - ex: Shankar's flight was bumpy: $Bumpy(E_1)$

Event Calculus

- Based on events, points in time, intervals rather than situations
- Reification of fluents
  - a fluent is an object represented by a term
    (ex: $At(Shankar, Berkeley)$)
    $\Longrightarrow$ does not say if it is true
  - T (True): asserts that a fluent is true at some point in time t
    ex: $T(At(Shankar, Berkeley), t)$
- Reification of events
  - events are described as instances of event categories
  - ex: event E_1: Shankar flies from San Francisco to WashingtonDC:
    $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge$
    $Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - reification allows for adding arbitrary information about fluents
  - ex: Shankar's flight was bumpy: $Bumpy(E_1)$

Event Calculus

- Based on events, points in time, intervals rather than situations
- Reification of fluents
  - a fluent is an object represented by a term
    (ex: $At(Shankar, Berkeley)$)
    $\Longrightarrow$ does not say if it is true
  - T (True): asserts that a fluent is true at some point in time t
    ex: $T(At(Shankar, Berkeley), t)$
- Reification of events
  - events are described as instances of event categories
  - ex: event E_1: Shankar flies from San Francisco to WashingtonDC:
    $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge$
    $Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - reification allows for adding arbitrary information about fluents
  - ex: Shankar's flight was bumpy: $Bumpy(E_1)$

Event Calculus

- Based on events, points in time, intervals rather than situations
- Reification of fluents
  - a fluent is an object represented by a term
    (ex: $At(Shankar, Berkeley)$)
    $\Longrightarrow$ does not say if it is true
  - T (True): asserts that a fluent is true at some point in time t
    ex: $T(At(Shankar, Berkeley), t)$
- Reification of events
  - events are described as instances of event categories
  - ex: event E_1: Shankar flies from San Francisco to WashingtonDC:
    $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge$
    $Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - reification allows for adding arbitrary information about fluents
  - ex: Shankar's flight was bumpy: $Bumpy(E_1)$

Event Calculus: Intervals

- A time interval i.e. a pair of times (*start*, *end*)
  - i.e., $i = (t_1, t_2)$ is the time interval that starts at t_1 and ends at t_2
- The list of predicates for (one version of) the event calculus:

  $T(f, t)$: fluent f is true at time t

  $Happens(e, i)$: event e happens over the time interval i

  $Initiates(e, f, t)$: event e causes fluent f to start to hold at time t

  $Terminates(e, f, t)$: event e causes fluent f to cease to hold at time t

  $Clipped(f, i)$: fluent f ceases to be true at some point during time interval i

  $Restored(f, i)$: fluent f becomes true sometime during time interval i
- A distinguished event, Start, describes the initial state

Event Calculus: Intervals

- A time interval i.e. a pair of times $(start, end)$
  - i.e., $i = (t_1, t_2)$ is the time interval that starts at t_1 and ends at t_2
- The list of predicates for (one version of) the event calculus:

  $T(f, t)$: fluent f is true at time t

  $Happens(e, i)$: event e happens over the time interval i

  $Initiates(e, f, t)$: event e causes fluent f to start to hold at time t

  $Terminates(e, f, t)$: event e causes fluent f to cease to hold at time t

  $Clipped(f, i)$: fluent f ceases to be true at some point during time interval i

  $Restored(f, i)$: fluent f becomes true sometime during time interval i
- A distinguished event, Start, describes the initial state

Event Calculus: Intervals

- A time interval i.e. a pair of times $(start, end)$
  - i.e., $i = (t_1, t_2)$ is the time interval that starts at t_1 and ends at t_2
- The list of predicates for (one version of) the event calculus:

  $T(f, t)$: fluent f is true at time t

  $Happens(e, i)$: event e happens over the time interval i

  $Initiates(e, f, t)$: event e causes fluent f to start to hold at time t

  $Terminates(e, f, t)$: event e causes fluent f to cease to hold at time t

  $Clipped(f, i)$: fluent f ceases to be true at some point during time interval i

  $Restored(f, i)$: fluent f becomes true sometime during time interval i
- A distinguished event, Start, describes the initial state

Event Calculus: Intervals [cont.]

Some definitions of the predicates (universal quantifications omitted)

- Definition of T:
  - a fluent f holds at time t if the fluent was initiated by an event e at some time t_1 in the past and was not made false (clipped) by an intervening event
    $(Happens(e,(t_1,t_2)) \wedge Initiates(e,f,t_1) \wedge \neg Clipped(f,(t_1,t)) \wedge t_1 < t) \rightarrow T(f,t)$
  - a fluent f does not hold at time t if the fluent was terminated by an event at some time t_2 in the past and was not restored by an event occurring at a later time
    $(Happens(e,(t_1,t_2)) \wedge Terminates(e,f,t_1) \wedge \neg Restored(f,(t_1,t)) \wedge t_1 < t) \rightarrow \neg T(f,t)$
- Extension of T to intervals:
  - a fluent f holds over an interval (t_1,t_2) if it holds on every point within the interval:
    $T(f,(t_1,t_2)) \leftrightarrow [\forall t.(t_1 \leq t \wedge t < t_2) \rightarrow T(f,t)]$
- ...

Event Calculus: Intervals [cont.]

Some definitions of the predicates (universal quantifications omitted)

- Definition of T:
  - a fluent f holds at time t if the fluent was initiated by an event e at some time t_1 in the past and was not made false (clipped) by an intervening event
    $(Happens(e,(t_1,t_2)) \wedge Initiates(e,f,t_1) \wedge \neg Clipped(f,(t_1,t)) \wedge t_1 < t) \rightarrow T(f,t)$
  - a fluent f does not hold at time t if the fluent was terminated by an event at some time t_2 in the past and was not restored by an event occurring at a later time
    $(Happens(e,(t_1,t_2)) \wedge Terminates(e,f,t_1) \wedge \neg Restored(f,(t_1,t)) \wedge t_1 < t) \rightarrow \neg T(f,t)$
- Extension of T to intervals:
  - a fluent f holds over an interval (t_1,t_2) if it holds on every point within the interval:
    $T(f,(t_1,t_2)) \leftrightarrow [\forall t.(t_1 \leq t \wedge t < t_2) \rightarrow T(f,t)]$
- ...

Actions in the Event Calculus

- Fluents and actions are related with domain-specific axioms
  - similar to successor-state axioms
- Ex: “the only way to use up an arrow is to shoot it”, assuming the agent has an arrow in the initial situation:
  - $Initiates(e, HaveArrow(a), t) \leftrightarrow e = Start$
  - $Terminates(e, HaveArrow(a), t) \leftrightarrow e \in Shootings(a)$
- We can extend event calculus to make it possible to represent
  - simultaneous events (e.g. two people needed to ride a seesaw)
  - exogenous events (e.g. the wind moves an object)
  - continuous events (e.g. bathtub water level continuously rising)
  - ...

Actions in the Event Calculus

- Fluents and actions are related with domain-specific axioms
  - similar to successor-state axioms
- Ex: “the only way to use up an arrow is to shoot it”, assuming the agent has an arrow in the initial situation:
  - $\mathit{Initiates}(e, \mathit{HaveArrow}(a), t) \leftrightarrow e = \mathit{Start}$
  - $\mathit{Terminates}(e, \mathit{HaveArrow}(a), t) \leftrightarrow e \in \mathit{Shootings}(a)$
- We can extend event calculus to make it possible to represent
  - simultaneous events (e.g. two people needed to ride a seesaw)
  - exogenous events (e.g. the wind moves an object)
  - continuous events (e.g. bathtub water level continuously rising)
  - ...

Actions in the Event Calculus

- Fluents and actions are related with domain-specific axioms
  - similar to successor-state axioms
- Ex: “the only way to use up an arrow is to shoot it”, assuming the agent has an arrow in the initial situation:
  - $\mathit{Initiates}(e, \mathit{HaveArrow}(a), t) \leftrightarrow e = \mathit{Start}$
  - $\mathit{Terminates}(e, \mathit{HaveArrow}(a), t) \leftrightarrow e \in \mathit{Shootings}(a)$
- We can extend event calculus to make it possible to represent
  - simultaneous events (e.g. two people needed to ride a seesaw)
  - exogenous events (e.g. the wind moves an object)
  - continuous events (e.g. bathtub water level continuously rising)
  - ...

Processes (aka Liquid Events)

- Events s.t., if they happen over an interval, they also happen over any subinterval:
  $((e \in \mathit{Processes}) \wedge \mathit{Happens}(e, (t_1, t_4)) \wedge (t_1 < t_2 < t_3 < t_4)) \rightarrow \mathit{Happens}(e, (t_2, t_3))$
- Distinction between liquid and nonliquid events is analogous to that between substances and individual objects

"$(t_1 < t_2 < t_3 < t_4 < ...)$" shortcut for $(t_1 < t_2) \wedge (t_2 < t_3) \wedge (t_3 < t_4) \wedge ...$

Processes (aka Liquid Events)

- Events s.t., if they happen over an interval, they also happen over any subinterval:
  $((e \in \mathit{Processes}) \wedge \mathit{Happens}(e, (t_1, t_4)) \wedge (t_1 < t_2 < t_3 < t_4)) \rightarrow \mathit{Happens}(e, (t_2, t_3))$
- Distinction between liquid and nonliquid events is analogous to that between substances and individual objects

"$(t_1 < t_2 < t_3 < t_4 < ...)$" shortcut for $(t_1 < t_2) \wedge (t_2 < t_3) \wedge (t_3 < t_4) \wedge ...$

Time Intervals

- Two kinds of time intervals:
  - Extended intervals
  - Moments, zero duration:
    $Partition(\{Moments, ExtendedIntervals\}, Intervals)$
    $i \in Moments \leftrightarrow Duration(i) = Seconds(0)$
- Some more vocabulary:
  - $Time(x)$: points in a time scale, give absolute times in seconds
  - $Begin(i), End(i)$: the earliest and latest moments in an interval
  - $Duration(i)$: the duration of an interval
- Examples: (Start at midnight (GMT) on January 1, 1900):
  $Interval(i) \rightarrow Duration(i) = (Time(End(i)) - Time(Begin(i)))$
  $Time(Begin(AD1900)) = Seconds(0)$
  $Time(Begin(AD2001)) = Seconds(3187324800)$
  $Time(End(AD2001)) = Seconds(3218860800)$
  $Duration(AD2001) = Seconds(31536000)$
  $Time(Begin(AD2001)) = Date(0, 0, 0, 1, Jan, 2001)$
  $Date(0, 20, 21, 24, 1, 1995) = Seconds(3000000000)$

Time Intervals

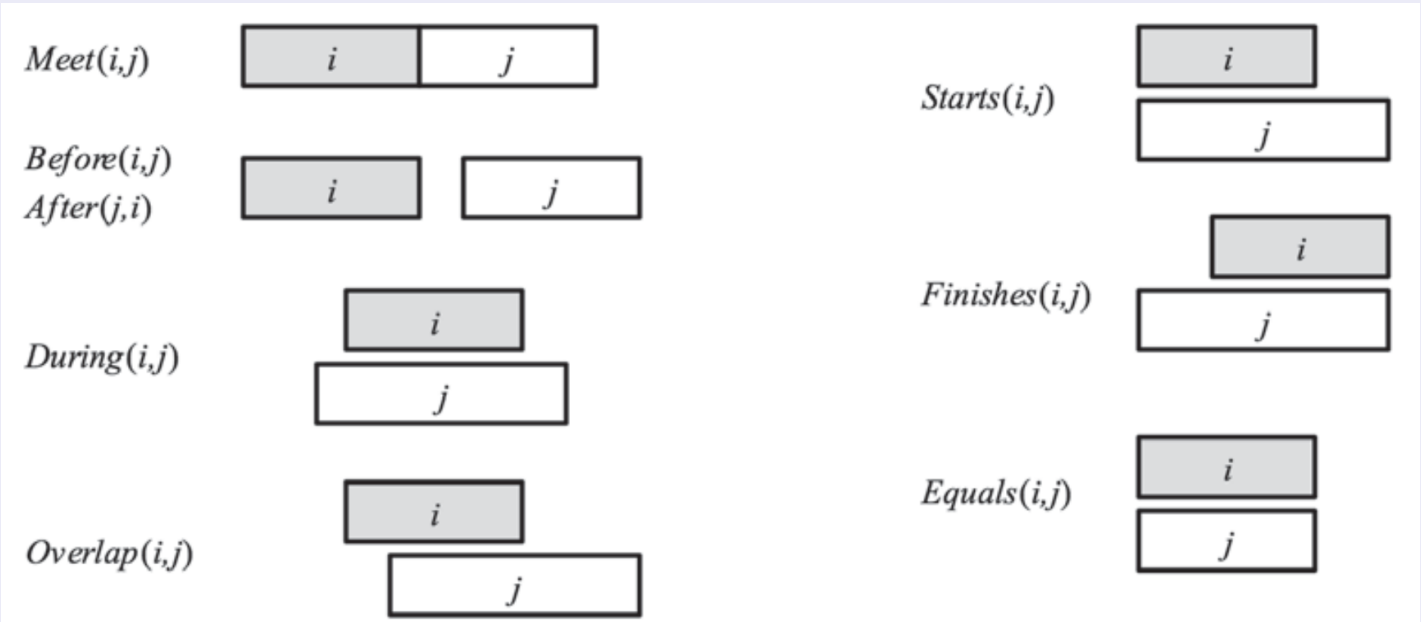
- Two kinds of time intervals:
  - Extended intervals
  - Moments, zero duration:
    $Partition(\{Moments, ExtendedIntervals\}, Intervals)$
    $i \in Moments \leftrightarrow Duration(i) = Seconds(0)$
- Some more vocabulary:
  - $Time(x)$: points in a time scale, give absolute times in seconds
  - $Begin(i), End(i)$: the earliest and latest moments in an interval
  - $Duration(i)$: the duration of an interval
- Examples: (Start at midnight (GMT) on January 1, 1900):
  $Interval(i) \rightarrow Duration(i) = (Time(End(i)) - Time(Begin(i)))$
  $Time(Begin(AD1900)) = Seconds(0)$
  $Time(Begin(AD2001)) = Seconds(3187324800)$
  $Time(End(AD2001)) = Seconds(3218860800)$
  $Duration(AD2001) = Seconds(31536000)$
  $Time(Begin(AD2001)) = Date(0, 0, 0, 1, Jan, 2001)$
  $Date(0, 20, 21, 24, 1, 1995) = Seconds(3000000000)$

Time Intervals

- Two kinds of time intervals:
  - Extended intervals
  - Moments, zero duration:
    $Partition(\{Moments, ExtendedIntervals\}, Intervals)$
    $i \in Moments \leftrightarrow Duration(i) = Seconds(0)$
- Some more vocabulary:
  - $Time(x)$: points in a time scale, give absolute times in seconds
  - $Begin(i), End(i)$: the earliest and latest moments in an interval
  - $Duration(i)$: the duration of an interval
- Examples: (Start at midnight (GMT) on January 1, 1900):
  $Interval(i) \rightarrow Duration(i) = (Time(End(i)) - Time(Begin(i)))$
  $Time(Begin(AD1900)) = Seconds(0)$
  $Time(Begin(AD2001)) = Seconds(3187324800)$
  $Time(End(AD2001)) = Seconds(3218860800)$
  $Duration(AD2001) = Seconds(31536000)$
  $Time(Begin(AD2001)) = Date(0, 0, 0, 1, Jan, 2001)$
  $Date(0, 20, 21, 24, 1, 1995) = Seconds(3000000000)$

Allen's Interval Algebra

$$
\begin{aligned}
Meet(i,j) &\Leftrightarrow End(i) = Begin(j) \\
Before(i,j) &\Leftrightarrow End(i) < Begin(j) \\
After(j,i) &\Leftrightarrow Before(i,j) \\
During(i,j) &\Leftrightarrow Begin(j) < Begin(i) < End(i) < End(j) \\
Overlap(i,j) &\Leftrightarrow Begin(i) < Begin(j) < End(i) < End(j) \\
\text{Starts} \quad \sout{Begins(i,j)} &\Leftrightarrow Begin(i) = Begin(j) \\
Finishes(i,j) &\Leftrightarrow End(i) = End(j) \\
Equals(i,j) &\Leftrightarrow Begin(i) = Begin(j) \wedge End(i) = End(j)
\end{aligned}
$$



Allen's Interval Algebra: Example

$Meets(ReignOf(GeorgeVI), ReignOf(ElizabethII))$
$Overlap(Fifties, ReignOf(Elvis))$
$Begin(Fifties) = Begin(AD1950)$
$End(Fifties) = End(AD1959)$

Note

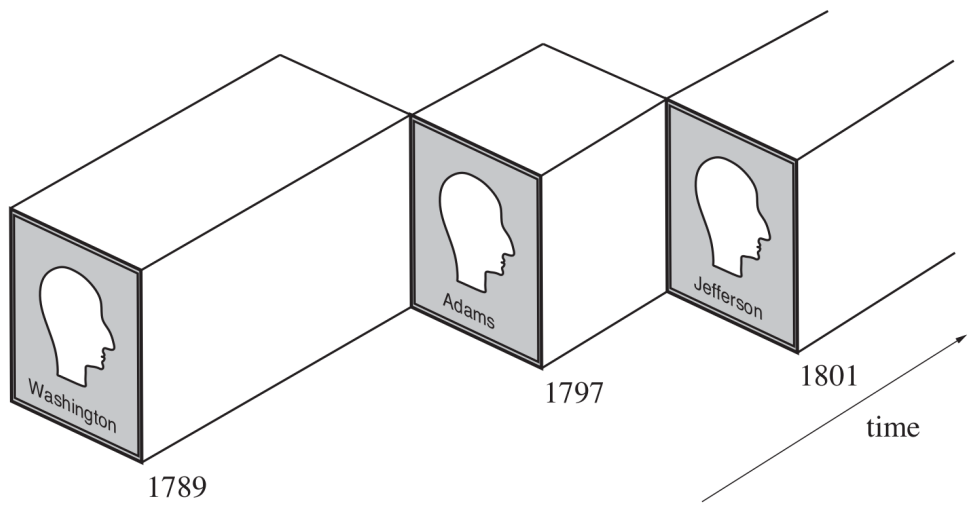
$Overlap(.,.)$ is not symmetric: $Overlap(i,j) \not\Longleftrightarrow Overlap(j,i)$

Physical Objects as Generalized Event

- Physical objects, when their properties change in time, are better represented as events with a duration
- Ex: $President(USA)$ have different properties in different periods
- Proposed solution: President(USA) denotes a single (abstract) object that consists of different people at different times
  - $T(Equals(President(USA), GeorgeWashington), AD1790)$
  - $T(Equals(President(USA), JohnAdams), AD1800)$
- ”Equals”, not “=”:
  a predicate cannot be
  the argument of another
  predicate in FOL
- Not “$President(USA, t)$”:
  time separate from fluents

Physical Objects as Generalized Event

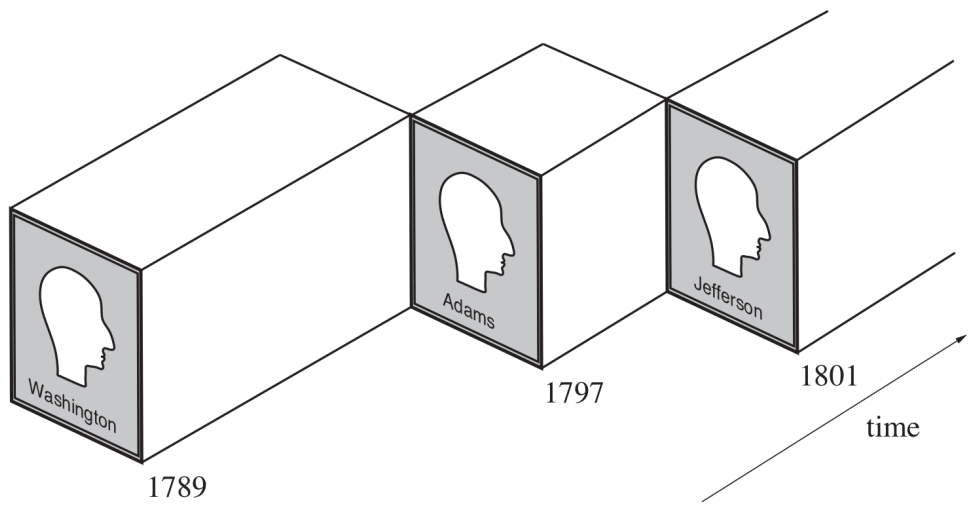
- Physical objects, when their properties change in time, are better represented as events with a duration
- Ex: $President(USA)$ have different properties in different periods
- Proposed solution: President(USA) denotes a single (abstract) object that consists of different people at different times
  - $T(Equals(President(USA), GeorgeWashington), AD1790)$
  - $T(Equals(President(USA), JohnAdams), AD1800)$
- ”Equals”, not “=”:
  a predicate cannot be
  the argument of another
  predicate in FOL
- Not “$President(USA, t)$”:
  time separate from fluents



(© S. Russell & P. Norwig, AIMA)

Physical Objects as Generalized Event

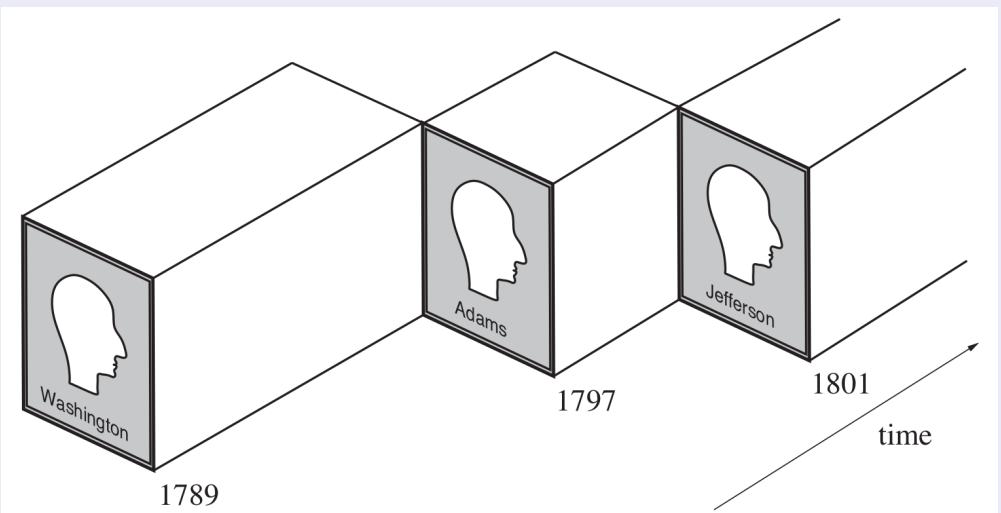
- Physical objects, when their properties change in time, are better represented as events with a duration
- Ex: $President(USA)$ have different properties in different periods
- Proposed solution: President(USA) denotes a single (abstract) object that consists of different people at different times
  - $T(Equals(President(USA), GeorgeWashington), AD1790)$
  - $T(Equals(President(USA), JohnAdams), AD1800)$
- ”Equals”, not “=”:
  a predicate cannot be
  the argument of another
  predicate in FOL
- Not “$President(USA, t)$”:
  time separate from fluents



(© S. Russell & P. Norwig, AIMA)

Physical Objects as Generalized Event

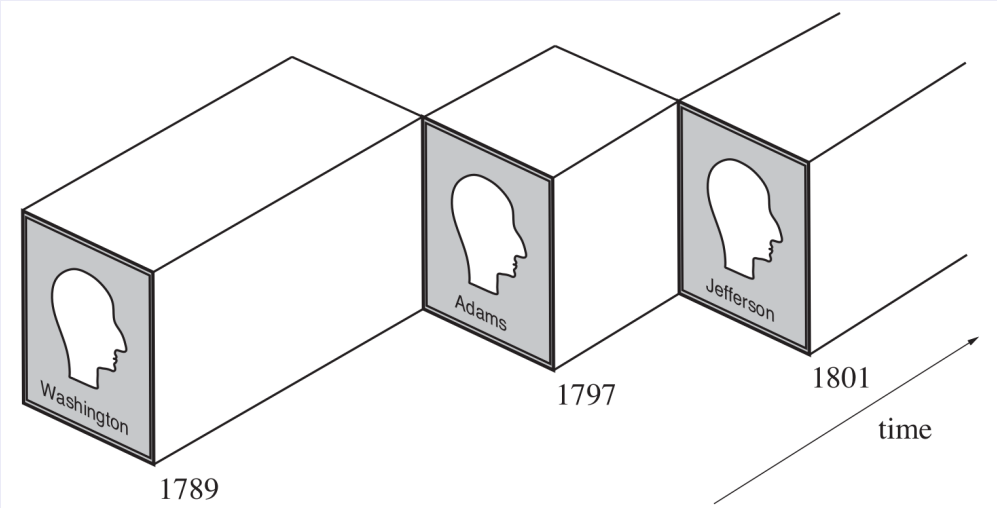
- Physical objects, when their properties change in time, are better represented as events with a duration
- Ex: *President*(*USA*) have different properties in different periods
- Proposed solution: President(USA) denotes a single (abstract) object that consists of different people at different times
  - $T(Equals(President(USA), GeorgeWashington), AD1790)$
  - $T(Equals(President(USA), JohnAdams), AD1800)$
- ”Equals”, not “=”:
  a predicate cannot be the argument of another predicate in FOL
- Not “$President(USA, t)$”: time separate from fluents



(© S. Russell & P. Norwig, AIMA)

Physical Objects as Generalized Event

- Physical objects, when their properties change in time, are better represented as events with a duration
- Ex: *President*(*USA*) have different properties in different periods
- Proposed solution: President(USA) denotes a single (abstract) object that consists of different people at different times
  - $T(Equals(President(USA), GeorgeWashington), AD1790)$
  - $T(Equals(President(USA), JohnAdams), AD1800)$
- ”Equals”, not “=”:
  a predicate cannot be the argument of another predicate in FOL
- Not “$President(USA, t)$”:
  time separate from fluents



(© S. Russell & P. Norwig, AIMA)

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. 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 to make it a term (as with event calculus)
- 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. 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 to make it a term (as with event calculus)
- 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. 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 to make it a term (as with event calculus)
- 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. 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 to make it a term (as with event calculus)
- 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. 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 to make it a term (as with event calculus)
- 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. 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 to make it a term (as with event calculus)
- 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 $\mathbf{K}_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $\mathbf{K}_{Lois}\,CanFly(Superman)$
  - ex: "Lois knows Klark Kent knows if he is Superman or not": $\mathbf{K}_{Lois}(\mathbf{K}_{Clark}\,Identity(Superman, Clark) \vee \mathbf{K}_{Clark}\neg Identity(Superman, Clark))$
- The following axiom holds in all (normal) modal logics:
  $K : (\mathbf{K}_A\phi \wedge K_A(\phi \rightarrow \psi) \rightarrow \mathbf{K}_A\psi$ (distribution axiom)
  $\Longrightarrow$ A is able to perform propositional inference
  - note: $\mathbf{K}_A(P \vee Q) \not\models \mathbf{K}_A P \vee \mathbf{K}_A Q$ (e.g. $\mathbf{K}_A(P \vee \neg P) \not\models \mathbf{K}_A P \vee \mathbf{K}_A \neg P$)
- The following axioms holds in some (normal) modal logics:
  $T : \mathbf{K}_A\varphi \rightarrow \varphi$ (knowledge axiom)
  $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ (positive-introspection axiom)
  $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ (negative-introspection axiom)
  ...
- Referential Opacity of modal logics:
  $Superman = Clark \wedge \mathbf{K}_{Lois}\,CanFly(Superman) \not\models \mathbf{K}_{Lois}\,CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $\mathbf{K}_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $\mathbf{K}_{Lois}\, CanFly(Superman)$
  - ex: "Lois knows Klark Kent knows if he is Superman or not": $\mathbf{K}_{Lois}(\mathbf{K}_{Clark}\, Identity(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identity(Superman, Clark))$
- The following axiom holds in all (normal) modal logics:
  $K : (\mathbf{K}_A\phi \wedge K_A(\phi \rightarrow \psi) \rightarrow \mathbf{K}_A\psi$ (distribution axiom)
  $\Longrightarrow$ A is able to perform propositional inference
  - note: $\mathbf{K}_A(P \vee Q) \not\models \mathbf{K}_A P \vee \mathbf{K}_A Q$ (e.g. $\mathbf{K}_A(P \vee \neg P) \not\models \mathbf{K}_A P \vee \mathbf{K}_A \neg P$)
- The following axioms holds in some (normal) modal logics:
  $T : \mathbf{K}_A\varphi \rightarrow \varphi$ (knowledge axiom)
  $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ (positive-introspection axiom)
  $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ (negative-introspection axiom)
  ...
- Referential Opacity of modal logics:
  $Superman = Clark \wedge \mathbf{K}_{Lois}\, CanFly(Superman) \not\models \mathbf{K}_{Lois}\, CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $\mathbf{K}_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $\mathbf{K}_{Lois}\, CanFly(Superman)$
  - ex: "Lois knows Klark Kent knows if he is Superman or not": $\mathbf{K}_{Lois}(\mathbf{K}_{Clark}\, Identity(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identity(Superman, Clark))$
- The following axiom holds in all (normal) modal logics:
  $K : (\mathbf{K}_A\phi \wedge K_A(\phi \rightarrow \psi) \rightarrow \mathbf{K}_A\psi$ (distribution axiom)
  $\Longrightarrow$ A is able to perform propositional inference
  - note: $\mathbf{K}_A(P \vee Q) \not\models \mathbf{K}_A P \vee \mathbf{K}_A Q$ (e.g. $\mathbf{K}_A(P \vee \neg P) \not\models \mathbf{K}_A P \vee \mathbf{K}_A \neg P$)
- The following axioms holds in some (normal) modal logics:
  $T : \mathbf{K}_A\varphi \rightarrow \varphi$ (knowledge axiom)
  $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ (positive-introspection axiom)
  $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ (negative-introspection axiom)
  ...
- Referential Opacity of modal logics:
  $Superman = Clark \wedge \mathbf{K}_{Lois}\, CanFly(Superman) \not\models \mathbf{K}_{Lois}\, CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $\mathbf{K}_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $\mathbf{K}_{Lois}\, CanFly(Superman)$
  - ex: "Lois knows Klark Kent knows if he is Superman or not": $\mathbf{K}_{Lois}(\mathbf{K}_{Clark}\, Identity(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identity(Superman, Clark))$
- The following axiom holds in all (normal) modal logics:
  $K : (\mathbf{K}_A\phi \wedge K_A(\phi \rightarrow \psi) \rightarrow \mathbf{K}_A\psi$ (distribution axiom)
  $\Longrightarrow$ A is able to perform propositional inference
  - note: $\mathbf{K}_A(P \vee Q) \not\models \mathbf{K}_A P \vee \mathbf{K}_A Q$ (e.g. $\mathbf{K}_A(P \vee \neg P) \not\models \mathbf{K}_A P \vee \mathbf{K}_A \neg P$)
- The following axioms holds in some (normal) modal logics:
  $T : \mathbf{K}_A\varphi \rightarrow \varphi$ (knowledge axiom)
  $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ (positive-introspection axiom)
  $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ (negative-introspection axiom)
  ...
- Referential Opacity of modal logics:
  $Superman = Clark \wedge \mathbf{K}_{Lois}\, CanFly(Superman) \not\models \mathbf{K}_{Lois}\, CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard

Modal Logics

- Modal logics include special modal operators that take formulas (not terms!) as arguments
  - "A knows P" is represented with $\mathbf{K}_A P$ (P formula, not term!)
  - ex: "Lois knows that Superman can fly": $\mathbf{K}_{Lois}\, CanFly(Superman)$
  - ex: "Lois knows Klark Kent knows if he is Superman or not": $\mathbf{K}_{Lois}(\mathbf{K}_{Clark}\, Identity(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identity(Superman, Clark))$
- The following axiom holds in all (normal) modal logics:
  $K : (\mathbf{K}_A\phi \wedge K_A(\phi \rightarrow \psi) \rightarrow \mathbf{K}_A\psi$ (distribution axiom)
  $\Longrightarrow$ A is able to perform propositional inference
  - note: $\mathbf{K}_A(P \vee Q) \not\models \mathbf{K}_A P \vee \mathbf{K}_A Q$ (e.g. $\mathbf{K}_A(P \vee \neg P) \not\models \mathbf{K}_A P \vee \mathbf{K}_A \neg P$)
- The following axioms holds in some (normal) modal logics:
  $T : \mathbf{K}_A\varphi \rightarrow \varphi$ (knowledge axiom)
  $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ (positive-introspection axiom)
  $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ (negative-introspection axiom)
  ...
- Referential Opacity of modal logics:
  $Superman = Clark \wedge \mathbf{K}_{Lois}\, CanFly(Superman) \not\models \mathbf{K}_{Lois}\, CanFly(Clark)$
- Reasoning in (propositional) Modal logics is NP-hard

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible worlds w_i
  - worlds are connected in a graph by accessibility relations
  - one relation for each distinct modal operator $\mathbf{K}_A$
- w_1 is accessible from w_0 wrt. $\mathbf{K}_A$ if everything which holds in w_1 is consistent with what A knows in w_0

  (written "$Acc(\mathbf{K}_A, w_0, w_1)$" or "$w_0 \overset{\mathbf{K}_A}{\longmapsto} w_1$")

  $\Longrightarrow$ $\mathbf{K}_A\varphi$ holds in w_o iff φ holds in every world w_i accessible from w_0
  - the more is known in w_0, the less worlds are accessible from w_0
  - two worlds may differ also for what is an agent knows there
- Different modal logics differ by different properties of $Acc(\mathbf{K}_A, ...)$
  - $T : \mathbf{K}_A\varphi \rightarrow \varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ reflexive
  - $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ transitive
  - $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ euclidean
  - ...

Notice the difference:

- $\mathbf{K}_A\neg P$: agent A knows that P does not hold
- $\neg\mathbf{K}_A P$: agent A does not know if P holds (or not)

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible worlds w_i
  - worlds are connected in a graph by accessibility relations
  - one relation for each distinct modal operator $\mathbf{K}_A$
- w_1 is accessible from w_0 wrt. $\mathbf{K}_A$ if everything which holds in w_1 is consistent with what A knows in w_0

  (written "$Acc(\mathbf{K}_A, w_0, w_1)$" or "$w_0 \overset{\mathbf{K}_A}{\longmapsto} w_1$")

  $\Longrightarrow$ $\mathbf{K}_A\varphi$ holds in w_o iff φ holds in every world w_i accessible from w_0
  - the more is known in w_0, the less worlds are accessible from w_0
  - two worlds may differ also for what is an agent knows there
- Different modal logics differ by different properties of $Acc(\mathbf{K}_A, ...)$
  - $T : \mathbf{K}_A\varphi \rightarrow \varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ reflexive
  - $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ transitive
  - $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ euclidean
  - ...

Notice the difference:

- $\mathbf{K}_A\neg P$: agent A knows that P does not hold
- $\neg\mathbf{K}_A P$: agent A does not know if P holds (or not)

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible worlds w_i
  - worlds are connected in a graph by accessibility relations
  - one relation for each distinct modal operator $\mathbf{K}_A$
- w_1 is accessible from w_0 wrt. $\mathbf{K}_A$ if everything which holds in w_1 is consistent with what A knows in w_0

  (written "$Acc(\mathbf{K}_A, w_0, w_1)$" or "$w_0 \overset{\mathbf{K}_A}{\longmapsto} w_1$")

  $\Longrightarrow$ $\mathbf{K}_A\varphi$ holds in w_o iff φ holds in every world w_i accessible from w_0
  - the more is known in w_0, the less worlds are accessible from w_0
  - two worlds may differ also for what is an agent knows there
- Different modal logics differ by different properties of $Acc(\mathbf{K}_A, ...)$
  - $T : \mathbf{K}_A\varphi \rightarrow \varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ reflexive
  - $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ transitive
  - $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ euclidean
  - ...

Notice the difference:

- $\mathbf{K}_A\neg P$: agent A knows that P does not hold
- $\neg\mathbf{K}_A P$: agent A does not know if P holds (or not)

Semantics of Modal Logics

- A model (Kripke model) is a collection of possible worlds w_i
  - worlds are connected in a graph by accessibility relations
  - one relation for each distinct modal operator $\mathbf{K}_A$
- w_1 is accessible from w_0 wrt. $\mathbf{K}_A$ if everything which holds in w_1 is consistent with what A knows in w_0

  (written "$Acc(\mathbf{K}_A, w_0, w_1)$" or "$w_0 \overset{\mathbf{K}_A}{\longmapsto} w_1$")

  $\Longrightarrow$ $\mathbf{K}_A\varphi$ holds in w_o iff φ holds in every world w_i accessible from w_0
  - the more is known in w_0, the less worlds are accessible from w_0
  - two worlds may differ also for what is an agent knows there
- Different modal logics differ by different properties of $Acc(\mathbf{K}_A, ...)$
  - $T : \mathbf{K}_A\varphi \rightarrow \varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ reflexive
  - $4 : \mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ transitive
  - $5 : \neg\mathbf{K}_A\varphi \rightarrow \mathbf{K}_A\neg\mathbf{K}_A\varphi$ holds iff $Acc(\mathbf{K}_A, ...)$ euclidean
  - ...

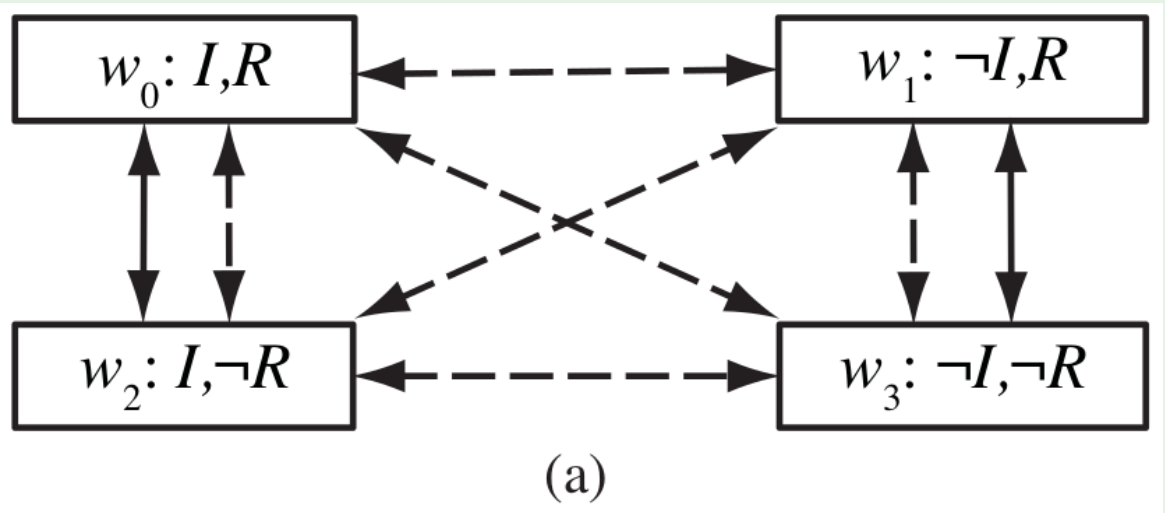
Notice the difference:

- $\mathbf{K}_A\neg P$: agent A knows that P does not hold
- $\neg\mathbf{K}_A P$: agent A does not know if P holds (or not)

Semantics of Modal Logics: Example

Accessibility relations: $\mathbf{K}_{Superman}$ (solid arrows) and $\mathbf{K}_{Lois}$ (dotted arrows).

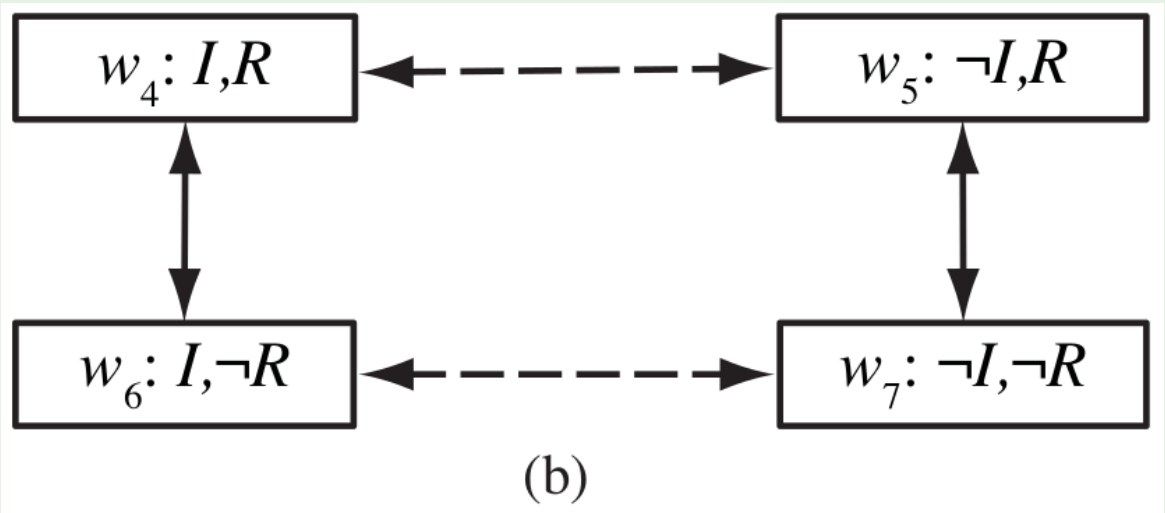
- Legenda:
  - R: “the weather report says tomorrow will rain”
  - I: “Superman’s secret identity is Clark Kent.”
  - all worlds are self-accessible (self-loop arrows not reported)
- Common knowledge:
  - Superman knows his own identity: $\mathbf{K}_{Superman} I$, and
    (a) neither he nor Lois has seen the weather report:
    $(\neg \mathbf{K}_{Lois} R \wedge \neg \mathbf{K}_{Lois} \neg R) \wedge (\neg \mathbf{K}_{Superman} R \wedge \neg \mathbf{K}_{Superman} \neg R)$
    $\mathbf{K}_{Lois}(\mathbf{K}_{Superman} I \vee \mathbf{K}_{Superman} \neg I)$



Semantics of Modal Logics: Example

Accessibility relations: $\mathbf{K}_{Superman}$ (solid arrows) and $\mathbf{K}_{Lois}$ (dotted arrows).

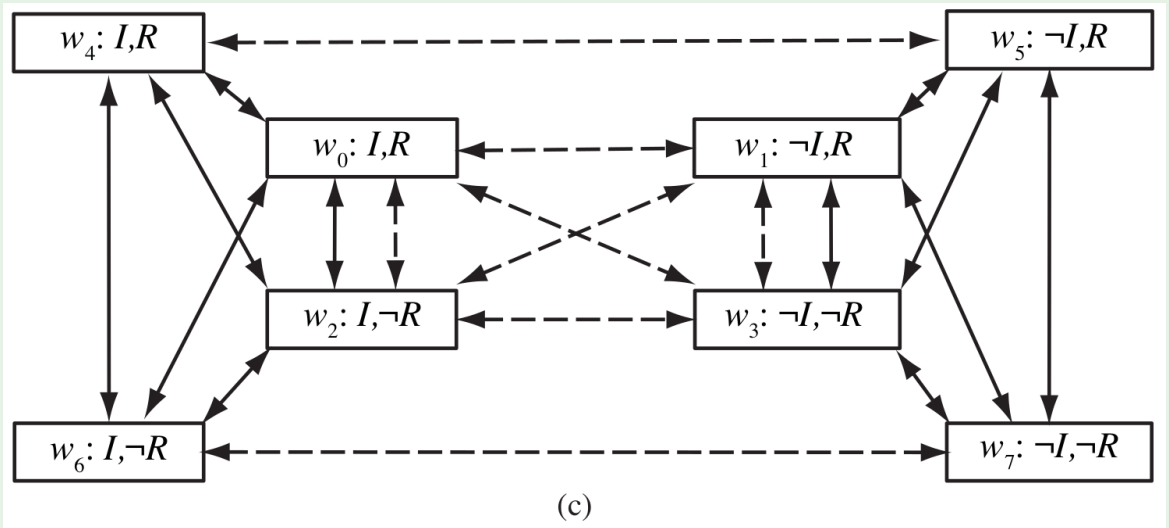
- Legenda:
  - R: “the weather report says tomorrow will rain”
  - I: “Superman’s secret identity is Clark Kent.”
  - all worlds are self-accessible (self-loop arrows not reported)
- Common knowledge:
  - Superman knows his own identity: $\mathbf{K}_{Superman} I$, and
    (b) Lois has seen the weather report, Superman has not:
    $(\mathbf{K}_{Lois} R \vee \mathbf{K}_{Lois} \neg R) \wedge (\neg \mathbf{K}_{Superman} R \wedge \neg \mathbf{K}_{Superman} \neg R)$
    $\mathbf{K}_{Lois}(\mathbf{K}_{Superman} I \vee \mathbf{K}_{Superman} \neg I) \wedge \mathbf{K}_{Superman}(\mathbf{K}_{Lois} R \vee \mathbf{K}_{Lois} \neg R)$



Semantics of Modal Logics: Example

Accessibility relations: $\mathbf{K}_{Superman}$ (solid arrows) and $\mathbf{K}_{Lois}$ (dotted arrows).

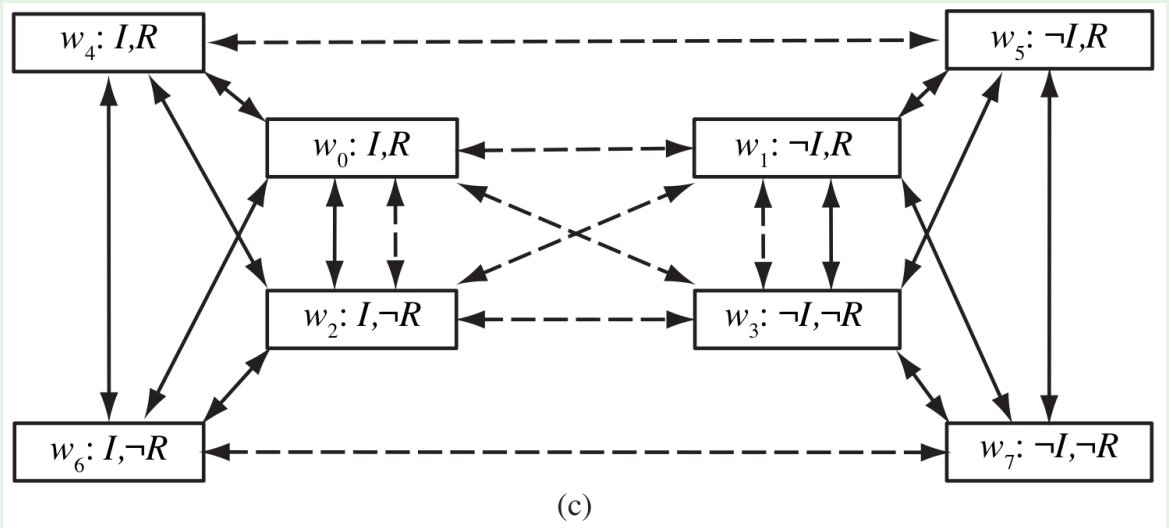
- Legenda:
  - R: “the weather report says tomorrow will rain”
  - I: “Superman’s secret identity is Clark Kent.”
  - all worlds are self-accessible (self-loop arrows not reported)
- Common knowledge:
  - Superman knows his own identity: $\mathbf{K}_{Superman} I$, and
    (c) Lois may or may not have seen the weather report, S. has not:
    $((\neg\mathbf{K}_{Lois} R \wedge \neg\mathbf{K}_{Lois}\neg R) \vee (\mathbf{K}_{Lois} R \vee \mathbf{K}_{Lois}\neg R)) \wedge (\neg\mathbf{K}_{Sup.} R \wedge \neg\mathbf{K}_{Sup.}\neg R)$
    $\mathbf{K}_{Lois}(\mathbf{K}_{Superman} I \vee \mathbf{K}_{Superman}\neg I)$



Semantics of Modal Logics: Example

Accessibility relations: $\mathbf{K}_{Superman}$ (solid arrows) and $\mathbf{K}_{Lois}$ (dotted arrows).

- Legenda:
  - R: “the weather report says tomorrow will rain”
  - I: “Superman’s secret identity is Clark Kent.”
  - all worlds are self-accessible (self-loop arrows not reported)
- Common knowledge:
  - Superman knows his own identity: $\mathbf{K}_{Superman} I$, and
    (c) Lois may or may not have seen the weather report, S. has not:
    $((\neg\mathbf{K}_{Lois} R \wedge \neg\mathbf{K}_{Lois}\neg R) \vee (\mathbf{K}_{Lois} R \vee \mathbf{K}_{Lois}\neg R)) \wedge (\neg\mathbf{K}_{Sup.} R \wedge \neg\mathbf{K}_{Sup.}\neg R)$
    $\mathbf{K}_{Lois}(\mathbf{K}_{Superman} I \vee \mathbf{K}_{Superman}\neg I)$



Exercise

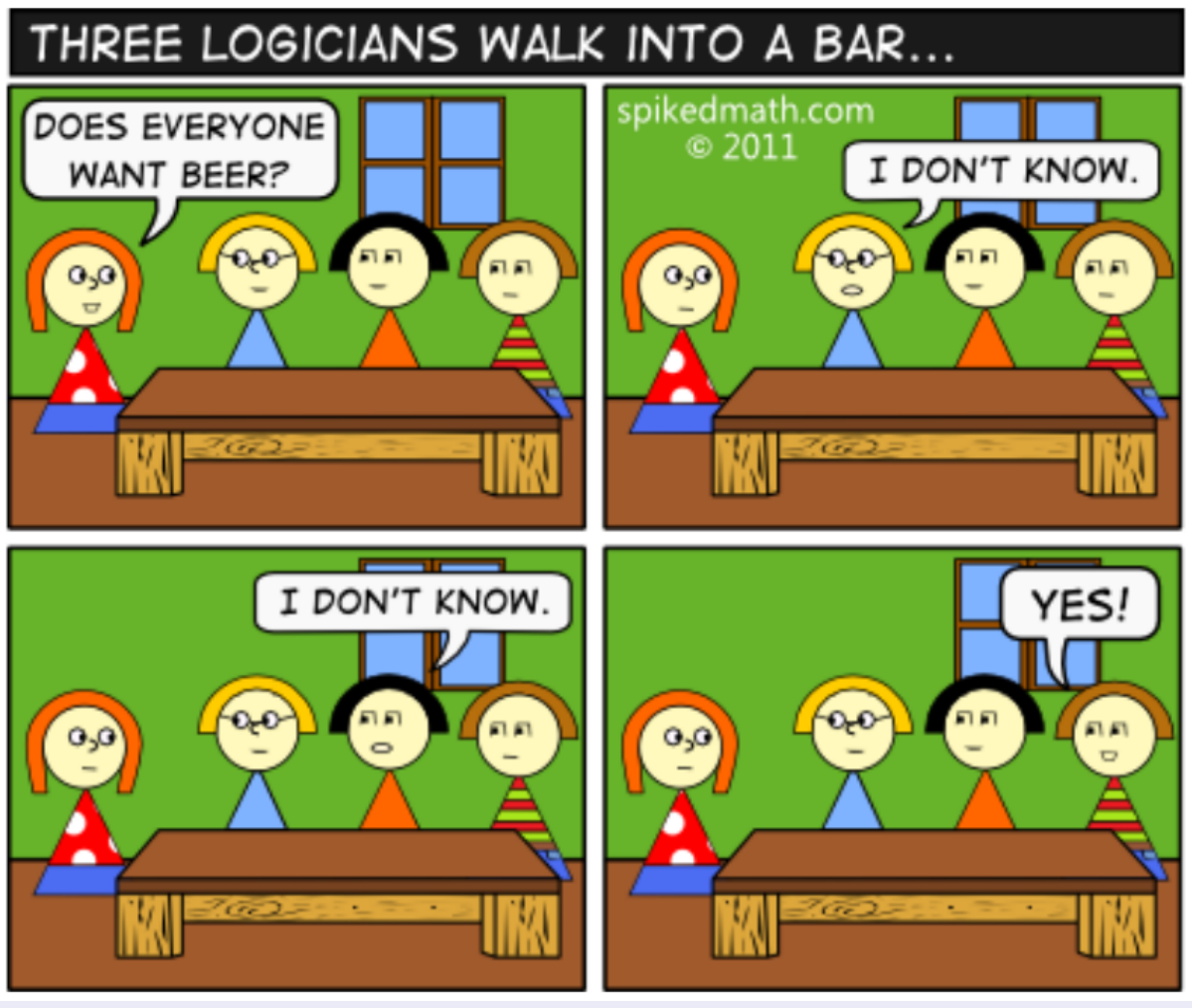
Consider the previous example.

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

$$[\neg]\mathbf{K}_{Lois}[\neg]\mathbf{K}_{Lois}[\neg]I,$$
$$[\neg]\mathbf{K}_{Lois}[\neg]\mathbf{K}_{Lois}[\neg]R,$$
$$[\neg]\mathbf{K}_{Sup.}[\neg]\mathbf{K}_{Sup.}[\neg]I,$$
$$[\neg]\mathbf{K}_{Sup.}[\neg]\mathbf{K}_{Sup.}[\neg]R,$$

Exercise

- Why does the third logician answers “Yes”?
- Formalize and solve the problem by means of modal logic



(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

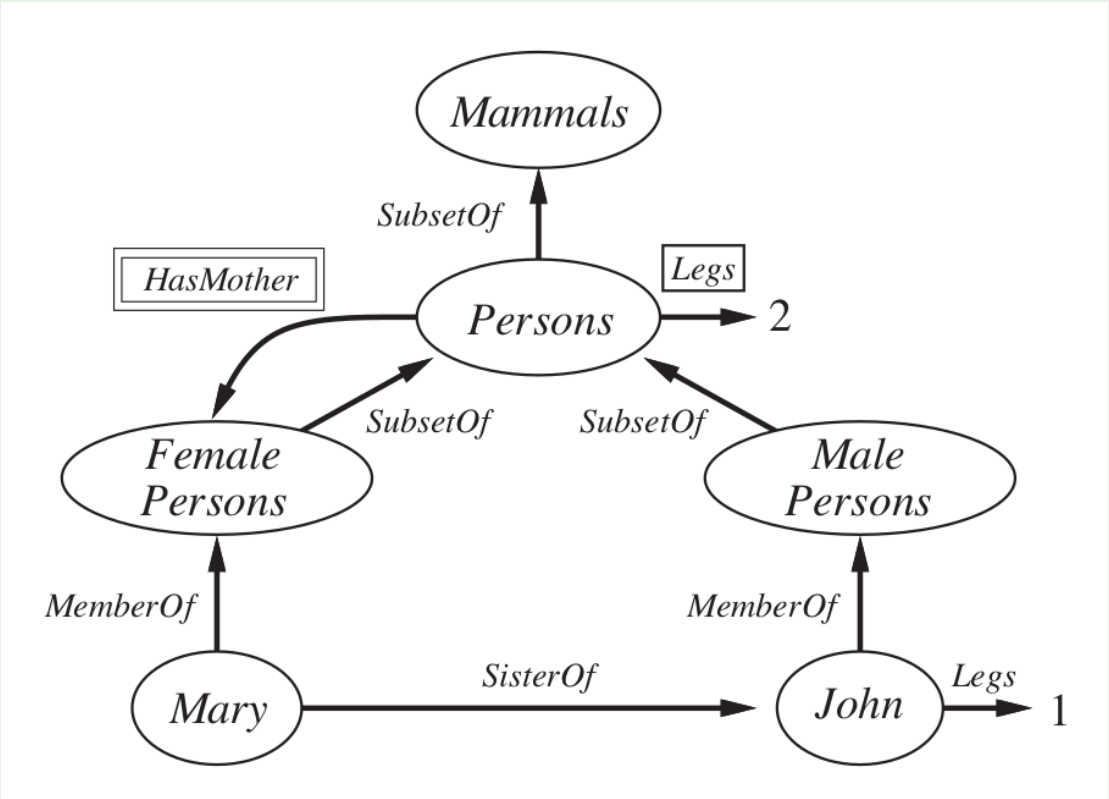
- Allow for representing individual objects, categories of objects, and relations among objects
- A Semantic Network is a graph where:
  - nodes, with a label, correspond to concepts
  - arcs, labelled and directed, correspond to binary relations between concepts (aka roles)
- Two kinds of nodes:
  - 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 $\forall x.(x \in Persons \rightarrow [\forall y.(HasMother(x, y) \rightarrow y \in FemalePersons)])$
  - similar for "Legs"

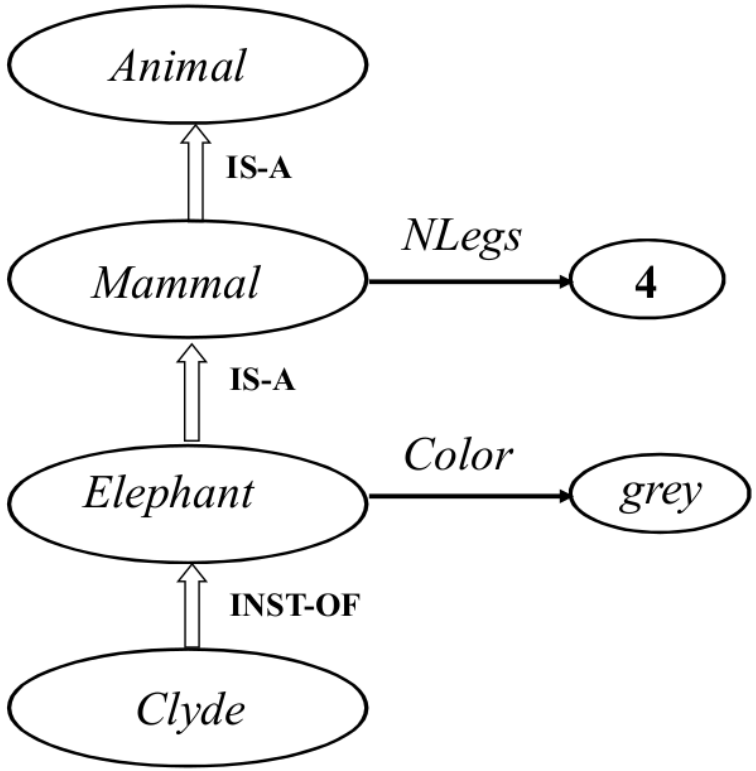


Inheritance in Semantic Networks

- 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



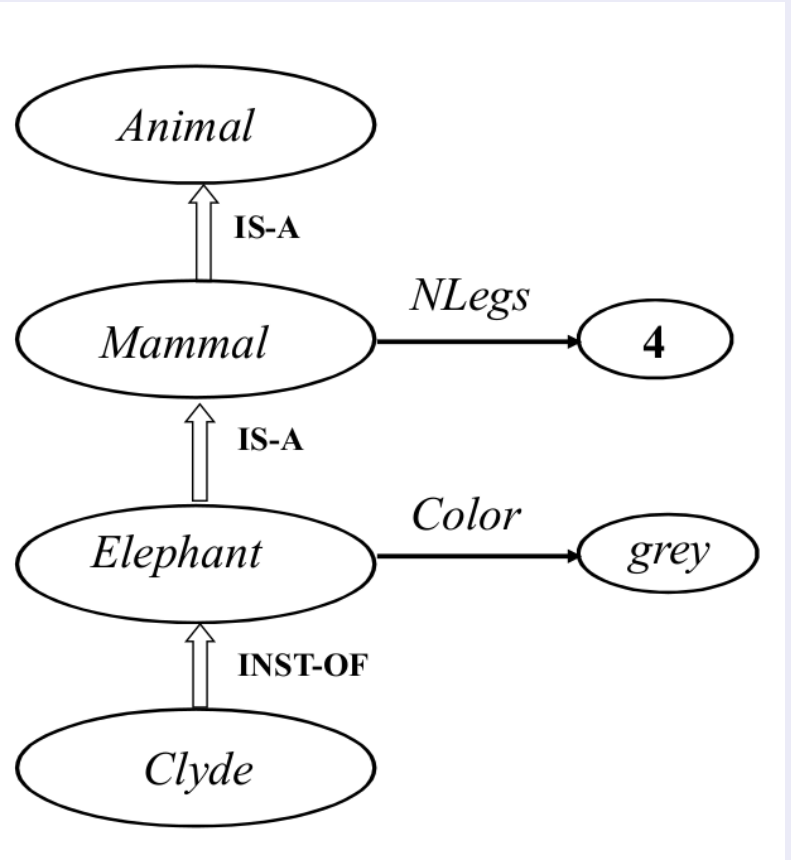
(Courtesy of Maria Simi, UniPI)

Inheritance in Semantic Networks

- Inheritance conveniently implemented as link traversal

Q. How many legs has Clyde?

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



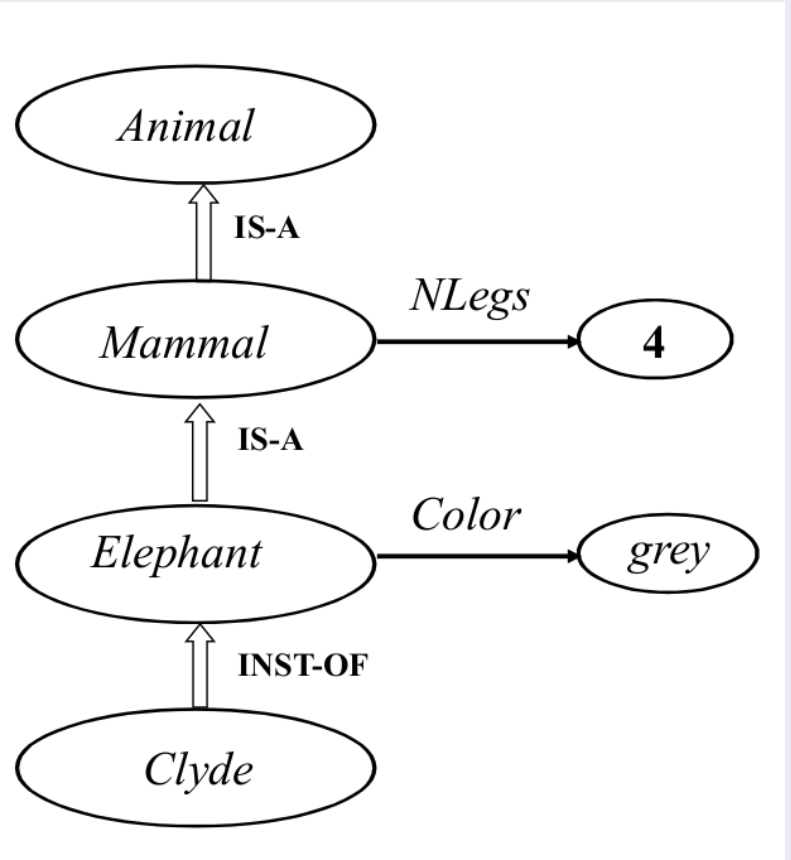
(Courtesy of Maria Simi, UniPI)

Inheritance in Semantic Networks

- Inheritance conveniently implemented as link traversal

Q. How many legs has Clyde?

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



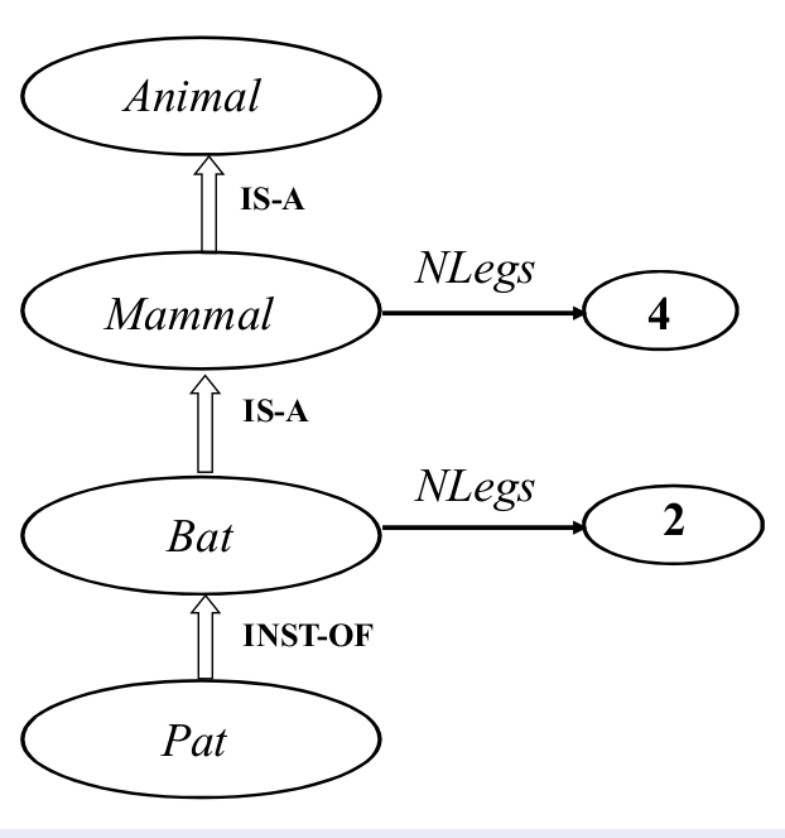
(Courtesy of Maria Simi, UniPI)

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

$\Longrightarrow$ ability to represent default values for categories



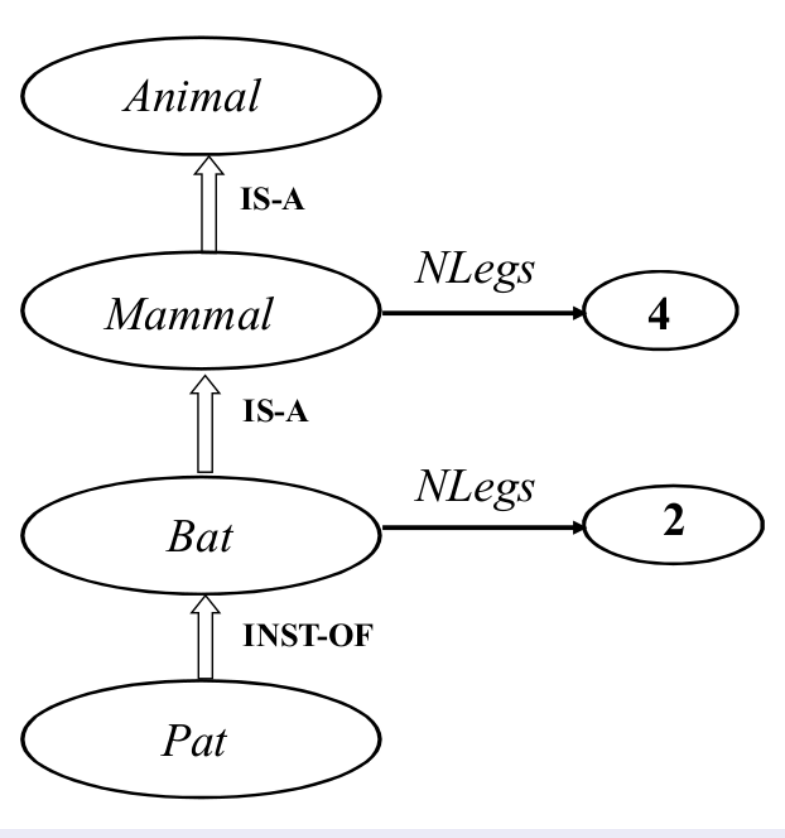
(Courtesy of Maria Simi, UniPI)

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



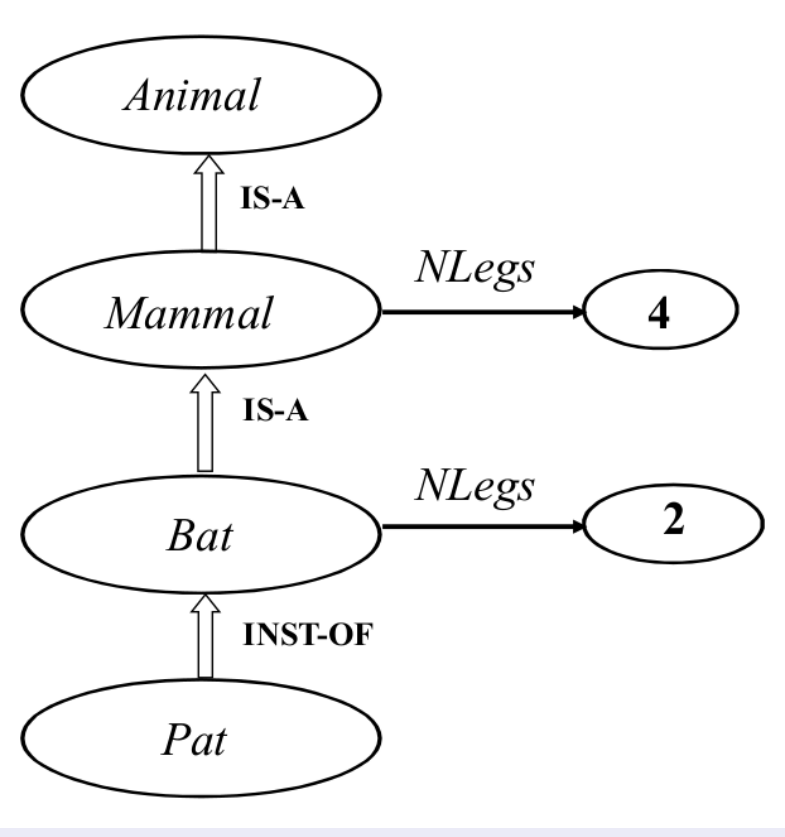
(Courtesy of Maria Simi, UniPI)

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?

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

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

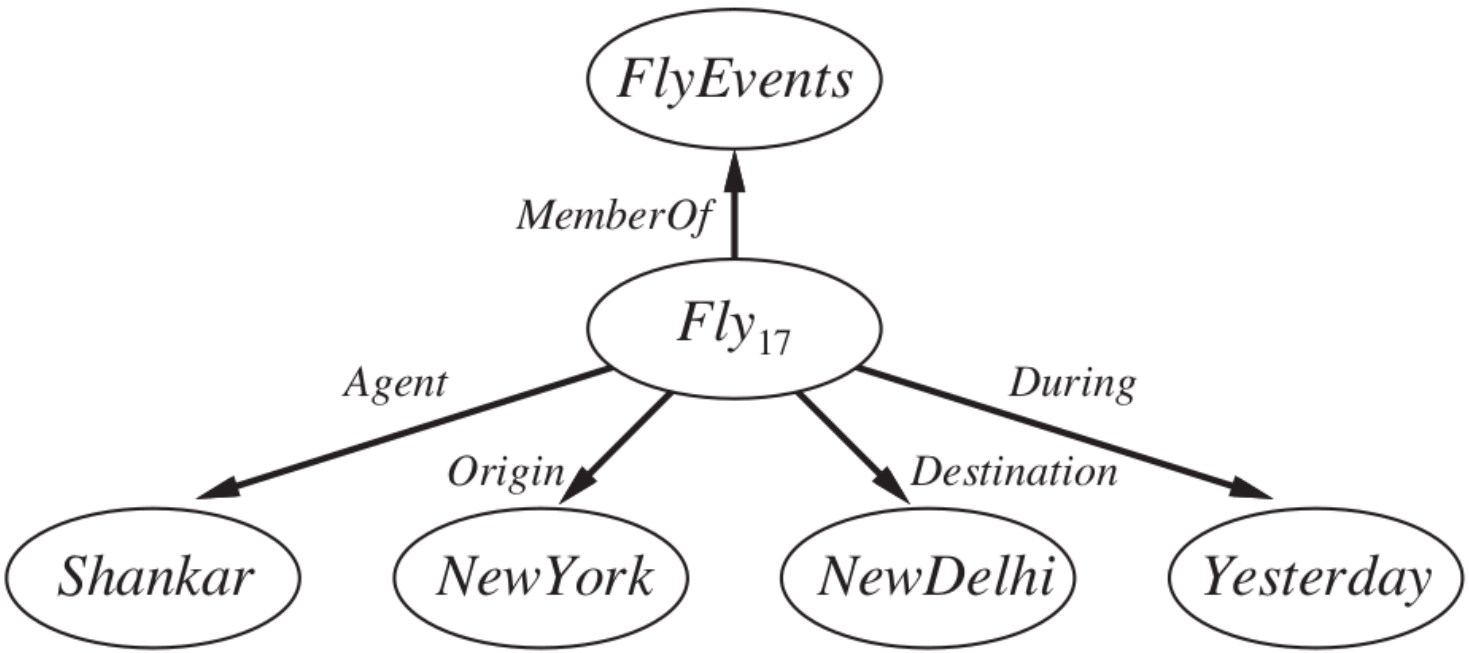
Encoding N-Ary Relations

- Semantic networks allow only binary relations

Q. How to represent n-ary relations?

$\Longrightarrow$ 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

Description Logics

- Designed to describe definitions and properties about categories
- Principal inference tasks:
  - Subsumption: check if one category is the subset 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 the subset 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 the subset 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
  - 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 of at least three female children: $\mathit{Woman} \sqcap \exists \mathit{hasChildren}.\mathit{Person} \sqcap \geq 3\ \mathit{hasChild}.\mathit{Female}$
  - ex: articles that have authors and whose authors are all journalists: $\mathit{Article} \sqcap \mathit{hasAuthor}.\top \sqcap \forall \mathit{hasAuthor}.\mathit{Journalist}$
- Roles corresponding to binary relations
  - ex: hasAuthor, hasChild
  - can be combined with operators for constructing complex roles
  - $\mathit{hasChildren} \equiv \mathit{hasSon} \sqcup \mathit{hasDaughter}$
- Individuals (used in assertions only)
  - ex Mary, John

Concepts, Roles, Individuals

- Concepts, corresponding to unary relations
  - 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 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: $Article \sqcap hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- 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
  - 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 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: $Article \sqcap hasAuthor.\top \sqcap \forall hasAuthor.Journalist$
- 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-Box: Example (Logic $\mathcal{ALCN}$)

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

(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