



**OPPA European Social Fund
Prague & EU: We invest in your future.**

Introduction, Semantic Networks and the Others

...

Petr Křemen
petr.kremen@fel.cvut.cz

FEL ČVUT

Our plan

Course Information

Crisp Knowledge Representation

Semantic Networks

Frames

Thesauri

Topic Maps

Conceptual Graphs

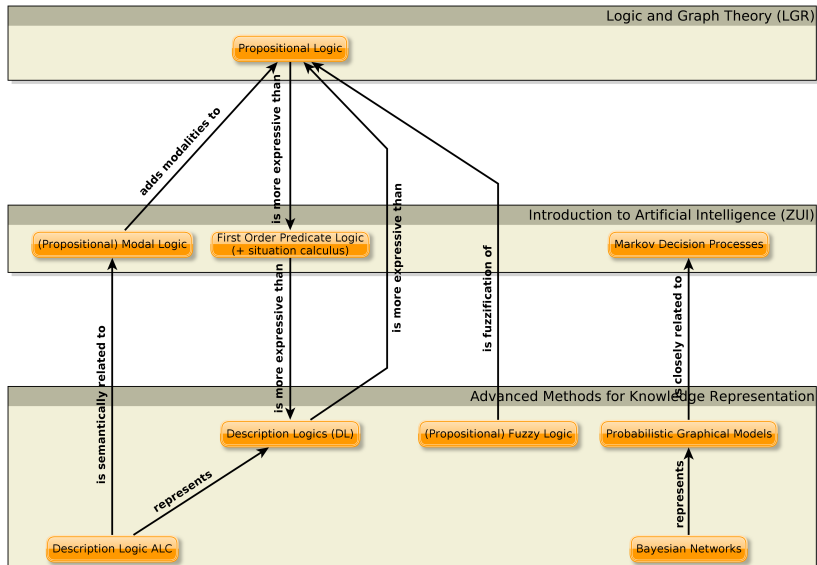
Course Information

- web page [currently in czech]:
<http://cw.felk.cvut.cz/doku.php/courses/a4m33rzn/start>
- three basic topics: description logics, probabilistic models, fuzzy logic
- Please go through the course web page carefully !!!

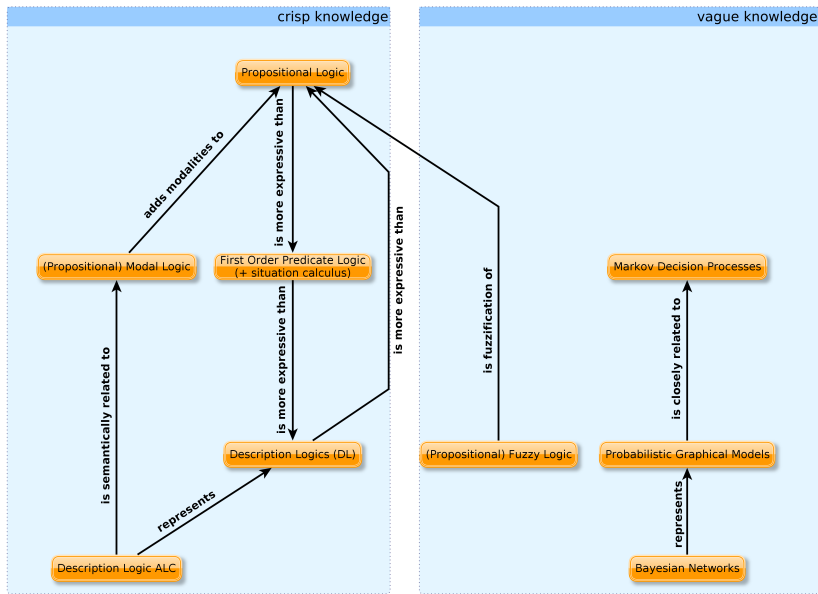
- web page [currently in czech]:
<http://cw.felk.cvut.cz/doku.php/courses/a4m33rzn/start>
- three basic topics: description logics, probabilistic models, fuzzy logic
- Please go through the course web page carefully !!!

- web page [currently in czech]:
<http://cw.felk.cvut.cz/doku.php/courses/a4m33rzn/start>
- three basic topics: description logics, probabilistic models, fuzzy logic
- **Please go through the course web page carefully !!!**

Course Roadmap



Course Roadmap



Crisp Knowledge Representation

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: "Which bachelor course should I enroll in order to get at least 6 credits ?"
 - Teacher : "How many hours per week am I going to teach this term ?"
 - Dean : "Which courses are popular among students ?"
- Knowledge tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - "Which courses are common?"
 - "The next semester, which courses can be selected with 6 credits?"
 - "Which are popular?"
 - "Every head of a department is a natural employee."

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : “How many hours per week am I going to teach this term ?”
 - Dean : “Which courses are popular among students ?”
- Knowledge tries to capture relationships in the domain, so that they can be used for answering various types of queries.

“Which courses are popular?”

“How many hours per week am I going to teach this term?”

“Which courses should I enroll in?”

“Which courses are popular among students?”

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : **“How many hours per week am I going to teach this term ?”**
 - Dean : **“Which courses are popular among students ?”**
- **Knowledge** tries to capture relationships in the domain, so that they can be used for answering various types of queries.

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : **“How many hours per week am I going to teach this term ?”**
 - Dean : **“Which courses are popular among students ?”**
- Knowledge tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - “Bachelor courses are courses.”

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : **“How many hours per week am I going to teach this term ?”**
 - Dean : **“Which courses are popular among students ?”**
- **Knowledge** tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - “Bachelor courses are courses.”
 - “In most cases a course can be opened only if 2 or more students are enrolled.”
 - “Every head of a department is a school employee.”

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : **“How many hours per week am I going to teach this term ?”**
 - Dean : **“Which courses are popular among students ?”**
- **Knowledge** tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - **“Bachelor courses are courses.”**
 - “In most cases a course can be opened only if 2 or more students are enrolled.”
 - “Every head of a department is a school employee.”

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **"Which bachelor course should I enroll in order to get at least 6 credits ?"**
 - Teacher : **"How many hours per week am I going to teach this term ?"**
 - Dean : **"Which courses are popular among students ?"**
- **Knowledge** tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - **"Bachelor courses are courses."**
 - **"In most cases a course can be opened only if 2 or more students are enrolled."**
 - "Every head of a department is a school employee."

- Let's have the domain of a university. Each stakeholder needs different type of information:
 - Student: **“Which bachelor course should I enroll in order to get at least 6 credits ?”**
 - Teacher : **“How many hours per week am I going to teach this term ?”**
 - Dean : **“Which courses are popular among students ?”**
- **Knowledge** tries to capture relationships in the domain, so that they can be used for answering various types of queries.
 - **“Bachelor courses are courses.”**
 - **“In most cases a course can be opened only if 2 or more students are enrolled.”**
 - **“Every head of a department is a school employee.”**

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively x procedurally ? – this course will deal with declarative knowledge. napf.
($\forall P$)(BachelorCourse(P) \Rightarrow Course(P))
 - subject domain learner (crisp) vs. with uncertainty – this course will cover both, starting with crisp knowledge. napf.
($\forall K$)(Course(K) \Rightarrow (Course(K) \wedge Professor(K)) \vee (Student(K)) \wedge (FDE, FGE) \wedge Course(K)) \wedge (FDE, FGE) \wedge Student(K))
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic search, associated with)
 - intelligent systems – systems of knowledge engineering
 - machine learning – knowledge bases
 - ... all AI in the knowledge base

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (documents, personal information, ...)
 - knowledge engineering – expert systems
 - machine learning – learning from data

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)
 - multiagent systems – content of messages sent between agents
 - machine learning – language bias
 - ... all AI branches

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)
 - multiagent systems – content of messages sent between agents
 - machine learning – language bias
 - ... all AI branches

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)
 - multiagent systems – content of messages sent between agents
 - machine learning – language bias
 - ... all AI branches

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)
 - multiagent systems – content of messages sent between agents
 - machine learning – language bias
 - ... all AI branches

Motivation (2)

- So, two questions remain ...
 - How to formally represent knowledge ?
 - declaratively \times procedurally ? – this course will deal with **declarative knowledge**. např.
 $(\forall P)(BachelorCourse(P) \Rightarrow Course(P))$
 - without uncertainty (crisp) \times with uncertainty – this course will cover both, starting **without uncertainty**. např.
 $(\forall K)(Course(K) \Rightarrow (CourseWithException(K) \vee ((\exists X_1, X_2)IsEnrolledTo(X_1, K) \wedge IsEnrolledTo(X_2, K) \wedge X_1 \neq X_2)))$
 - How to make use of the knowledge representation ?
 - knowledge management – search engines (databases, semantic servers, semantics web)
 - multiagent systems – content of messages sent between agents
 - machine learning – language bias
 - ... all AI branches

Declarative Knowledge Representation without Uncertainty

- **sémantic networks, frames,**
- thesauri, topic maps
- relational databases (relational calculus)
- rule-based systems, Prolog (first-order predicate logics)
- sémantic web, RDF(S), OWL, OWL 2 (description logics)

Declarative Knowledge Representation without Uncertainty

- sémantic networks, frames,
- thesauri, topic maps
- relational databases (relational calculus)
- rule-based systems, Prolog (first-order predicate logics)
- sémantic web, RDF(S), OWL, OWL 2 (description logics)

Declarative Knowledge Representation without Uncertainty

- sémantic networks, frames,
- thesauri, topic maps
- relational databases (relational calculus)
- rule-based systems, Prolog (first-order predicate logics)
- sémantic web, RDF(S), OWL, OWL 2 (description logics)

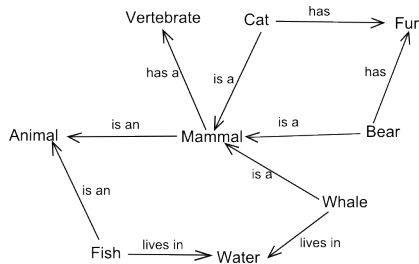
Declarative Knowledge Representation without Uncertainty

- sémantic networks, frames,
- thesauri, topic maps
- relational databases (relational calculus)
- rule-based systems, Prolog (first-order predicate logics)
- sémantic web, RDF(S), OWL, OWL 2 (description logics)

- sémantic networks, frames,
- thesauri, topic maps
- relational databases (relational calculus)
- rule-based systems, Prolog (first-order predicate logics)
- sémantic web, RDF(S), OWL, OWL 2 (description logics)

Semantic Networks

Semantic Networks



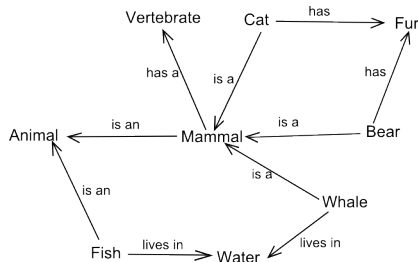
(©wikipedia.org)

Nodes = entities (individuals, classes),

Edges = binary relations

- The only possible inference is *inheritance* by means of **is a** relationship.

Semantic Networks



(©wikipedia.org)

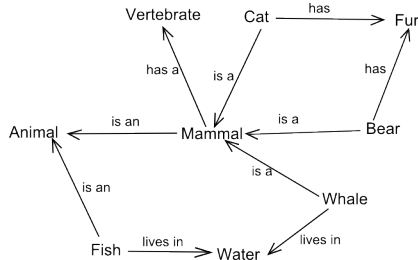
Nodes = entities (individuals, classes),

Edges = binary relations

- The only possible inference is *inheritance* by means of **is a** relationship.

Each Cat has a Vertebrate, since each Cat is a Mammal.

Semantic Networks



(©wikipedia.org)

Nodes = entities (individuals, classes),

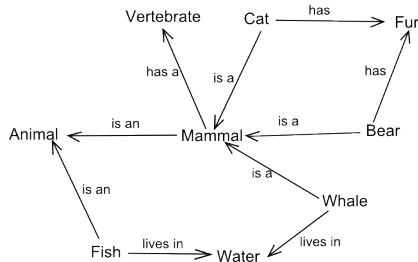
Edges = binary relations

- The only possible inference is *inheritance* by means of **is a** relationship.

Example

Each Cat **has a** Vertebrate, since each Cat **is a** Mammal.

Semantic Networks



(©wikipedia.org)

Nodes = entities (individuals, classes),

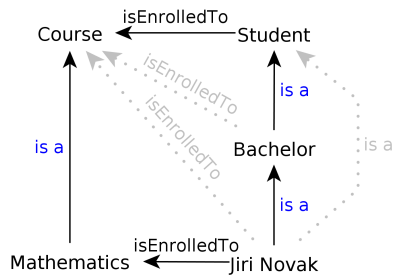
Edges = binary relations

- The only possible inference is *inheritance* by means of **is a** relationship.

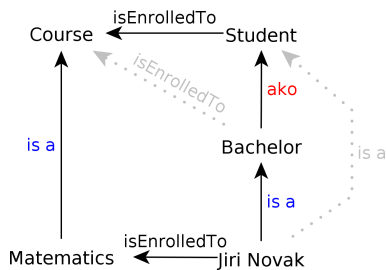
Example

Each Cat **has a** Vertebrate, since each Cat **is a** Mammal.

Semantic Networks (2)



However, this does not allow distinguishing individuals (instances) and groups (classes).



To solve this, a new relationship type “is a kind of” **ako** can be introduced and used for inheritance, while **is a** relationships would be restricted to expressing individual-group relationships.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$

$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$

$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$\text{relation}(X, Y) \wedge \text{ako}(Z, X) \Rightarrow \text{relation}(Z, Y).$$

$$\text{isa}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{isa}(X, Z).$$

$$\text{ako}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{ako}(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$

$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$

$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$\text{relation}(X, Y) \wedge \text{ako}(Z, X) \Rightarrow \text{relation}(Z, Y).$$

$$\text{isa}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{isa}(X, Z).$$

$$\text{ako}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{ako}(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$

$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$

$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$\text{relation}(X, Y) \wedge \text{ako}(Z, X) \Rightarrow \text{relation}(Z, Y).$$

$$\text{isa}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{isa}(X, Z).$$

$$\text{ako}(X, Y) \wedge \text{ako}(Y, Z) \Rightarrow \text{ako}(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”

Semantic Networks (3)

- ☺ are simple – from the point of logics they are not much more than a binary structure + **ako** and **isa** relationships with the following semantics:

$$relation(X, Y) \wedge ako(Z, X) \Rightarrow relation(Z, Y).$$

$$isa(X, Y) \wedge ako(Y, Z) \Rightarrow isa(X, Z).$$

$$ako(X, Y) \wedge ako(Y, Z) \Rightarrow ako(X, Z).$$

- ☹ no way to express non-monotonous knowledge (like FOL).
- ☹ no easy way to express n-ary relationships (**reification** needed).
- ☹ no way to express binary relationships characteristics – transitivity, functionality, reflexivity, etc., or their hierarchies “to be a father means to be a parent”, aj.,
- ☹ no way to express more complex constructs, like cardinality restrictions: “Each person has at most two legs.”
- Wordnet, Semantic Wiki, aj.

Wordnet (<http://wordnet.princeton.edu>) and MultiWordnet (<http://multiwordnet.itc.it>) are lexical databases. They are represented as semantic networks extended with a bit more semantics, e.g. :

hyponyms, hypernyms correspond to the **ako** relationship.

meronyms, holonyms denote “part-of” relationships between terms.

synonyms, antonyms synonyms are grouped into *synsets* – i.e. sets of terms that build up a single semantic context/meaning (e.g.

$S_1 = \{\text{man, adult male}\}$, $S_2 = \{\text{man, human being}\}$)

Semantic Networks – Wordnet, MultiWordnet

Wordnet (<http://wordnet.princeton.edu>) and MultiWordnet (<http://multiwordnet.itc.it>) are lexical databases. They are represented as semantic networks extended with a bit more semantics, e.g. :

hyponyms, hypernyms correspond to the **ako** relationship.

meronyms, holonyms denote “part-of” relationships between terms.

synonyms, antonyms synonyms are grouped into *synsets* – i.e. sets of terms that build up a single semantic context/meaning (e.g.

$S_1 = \{\text{man, adult male}\}$, $S_2 = \{\text{man, human being}\}$)

Wordnet (<http://wordnet.princeton.edu>) and MultiWordnet (<http://multiwordnet.itc.it>) are lexical databases. They are represented as semantic networks extended with a bit more semantics, e.g. :

hyponyms, hypernyms correspond to the **ako** relationship.

meronyms, holonyms denote “part-of” relationships between terms.

synonyms, antonyms synonyms are grouped into *synsets* – i.e. sets of terms that build up a single semantic context/meaning (e.g.

$S_1 = \{\text{man, adult male}\}$, $S_2 = \{\text{man, human being}\}$)

Frames

frame: Škoda Favorit

slots:

is a: car

has engine: four-stroke engine

has transmission system: manual

has carb: *value:* Jikov

default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.
- ⊙ Facets allow non-monotonic reasoning.
- ⊙ *Daemons* are triggers for actions performed on facets (read, write, delete). Can be used e.g. for consistency checking.

frame: Škoda Favorit

slots:

is a: car

has engine: four-stroke engine

has transmission system: manual

has carb: *value:* Jikov

default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.
- ☺ Facets allow non-monotonic reasoning.
- ☺ *Daemons* are triggers for actions performed on facets (read, write, delete). Can be used e.g. for consistency checking.

frame: Škoda Favorit

slots:

is a: car

has engine: four-stroke engine

has transmission system: manual

has carb: *value:* Jikov

default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.

- ☺ Facets allow non-monotonic reasoning.
- ☺ *Daemons* are triggers for actions performed on facets (read, write, delete). Can be used e.g. for consistency checking.

frame: Škoda Favorit

slots:

is a: car

has engine: four-stroke engine

has transmission system: manual

has carb: *value:* Jikov

default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.

😊 Facets allow non-monotonic reasoning.

😊 *Daemons* are triggers for actions performed on facets (read, write, delete). Can be used e.g. for consistency checking.

frame: Škoda Favorit

slots:

is a: car

has engine: four-stroke engine

has transmission system: manual

has carb: *value:* Jikov

default: Pierburg

- more structured than semantic networks
- forms that contain **slots** (binary relationships).

([MvL93])

- Every slot has several **facets** (slot use restrictions), e.g. cardinality, defaults, etc.
- 😊 Facets allow non-monotonic reasoning.
- 😊 *Daemons* are triggers for actions performed on facets (read, write, delete). Can be used e.g. for consistency checking.

Example

Typically, Škoda Favorit **has carb** of type Pierburg, but this particular Škoda Favorit **has carb** of type Jikov.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - <http://www.ai.sri.com/okbc>
- Protégé - <http://protege.stanford.edu/overview/protege-frames.html>
- Apollo - <http://apollo.open.ac.uk>
- Apollo CH - <http://labe.felk.cvut.cz/falc/Apollo>

Example

Typically, Škoda Favorit **has carb** of type Pierburg, but this particular Škoda Favorit **has carb** of type Jikov.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - <http://www.ai.sri.com/okbc>
- Protégé - <http://protege.stanford.edu/overview/protege-frames.html>
- Apollo - <http://apollo.open.ac.uk>
- Apollo CH - <http://labe.felk.cvut.cz/falc/Apollo>

Example

Typically, Škoda Favorit **has carb** of type Pierburg, but this particular Škoda Favorit **has carb** of type Jikov.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - <http://www.ai.sri.com/okbc>
- Protégé - <http://protege.stanford.edu/overview/protege-frames.html>
- Apollo - <http://apollo.open.ac.uk>
- Apollo CH - <http://labe.felk.cvut.cz/falc/Apollo>

Example

Typically, Škoda Favorit **has carb** of type Pierburg, but this particular Škoda Favorit **has carb** of type Jikov.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - <http://www.ai.sri.com/okbc>
- Protégé - <http://protege.stanford.edu/overview/protege-frames.html>
- Apollo - <http://apollo.open.ac.uk>
- Apollo CH - <http://labe.felk.cvut.cz/falc/Apollo>

Example

Typically, Škoda Favorit **has carb** of type Pierburg, but this particular Škoda Favorit **has carb** of type Jikov.

- frames can be grouped into *scenarios* that represent typical situations, e.g. going into a restaurant. [MvL93]
- OKBC - <http://www.ai.sri.com/okbc>
- Protégé - <http://protege.stanford.edu/overview/protege-frames.html>
- Apollo - <http://apollo.open.ac.uk>
- Apollo CH - <http://labe.felk.cvut.cz/falc/Apollo>

Frames (3) - Apollo CH

The screenshot shows the Apollo 0.28.0 interface. On the left is a hierarchical tree of classes. The 'general-medical-knowledge' class is expanded, showing sub-classes like 'drug', 'my-class', 'blood-pressure', 'disease-disorder', 'cerebrovascular-disease', 'heart-disease', 'renal-disease', 'vascular-disease', 'medical-condition-disorder', 'ace-inhibitors', 'advanced-hypertensive-retinopathy', 'alpha-blockers', 'angiotensin-2-antagonists', 'beta-blockers', 'calcium-antagonists', and 'cerebrovascular-disease'. The 'disease-disorder' class is selected. The main area displays details for 'disease-disorder', including its super-classes, slots, and sub-classes.

Super-classes
disorder

Slots using focus

disease-disorder

Slot	Type	Value	Cardinality	Documen.
description	string		R	

Documentation | **Slots** | **Relational**

Sub-classes and instances
cerebrovascular-disease
heart-disease
renal-disease
vascular-disease

Classes used by focus
string

Current : general-medical-knowledge

Frames (4) - Protégé

The screenshot displays the Protégé 3.2.1 interface. The main window is titled "CLASS EDITOR" and shows the configuration for the "STANDARD-SLOT" class, which is an instance of "STANDARD-CLASS".

CLASS EDITOR Details:

- Name:** STANDARD-SLOT
- Role:** Concrete
- Template Slots Table:**

Name	Cardinality	Type	Other Facets
ASSOCIATED-FACET	single	Instance of FACET	inverse-slot=ASSOCIATED-SLOT
DIRECT-DOMAIN	multiple	Instance of CLASS	inverse-slot=DIRECT-TEMPLATE-SLOTS
DIRECT-SLOTS	multiple	Instance of SLOT	inverse-slot=DIRECT-SUPERLOTS
DIRECT-SUPERLOTS	multiple	Instance of SLOT	inverse-slot=DIRECT-INSTANCES
DIRECT-TYPE	multiple	Class with superclass SLOT	
DOCUMENTATION	multiple	String	
NAME	single	String	
SLOT-CONSTRAINTS	multiple	Instance of CONSTRAINT	
SLOT-DEFAULTS	multiple	Any	
SLOT-INVERSE	single	Instance of SLOT	inverse-slot=SLOT-INVERSE default=1
SLOT-MAXIMUM-CARDINALITY	single	Integer	
SLOT-MINIMUM-CARDINALITY	single	Integer	
SLOT-NUMERIC-MAXIMUM	single	Float	
SLOT-NUMERIC-MINIMUM	single	Float	
SLOT-VALUE-TYPE	multiple	Any	default=String
SLOT-VALUES	multiple	Any	

Class Hierarchy (Left Panel):

- THING
 - SYSTEM-CLASS
 - META-CLASS
 - CLASS
 - STANDARD-CLASS
 - SLOT
 - STANDARD-SLOT
- CONSTRAINT
- ANNOTATION
- RELATION
- Author
 - New_Service
 - Columnist
 - Editor
 - Reporter
- Content
 - Advertisement
 - Personals_Ad
 - Standard_Ad
 - Article
 - Library
 - Newspaper

Superclasses (Bottom Panel):

- SLOT

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:
 - thesauri
 - expert shells

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:
 - thesauri
 - topic maps

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:
 - **thesauri**
 - topic maps

Frames and Semantics Networks – Summary

- 😊 very simple structures for knowledge representation,
- 😊 nonmonotonic reasoning,
- 😞 ad-hoc reasoning procedures, that complicates (and broadens ambiguity during) translation to First Order Predicate Logic (FOPL),
- 😞 problems – querying, debugging.
 - ... but semantic networks are basis for other technologies:
 - **thesauri**
 - **topic maps**

Thesauri

thesaurus is a taxonomy (hierarchy of terms) enriched with new types of relationships, e.g.:

BT/NT (broader/narrower term) = term hierarchy.

Example

beef → NT → meat

SN (scope note) explains meaning of a given term.

Example

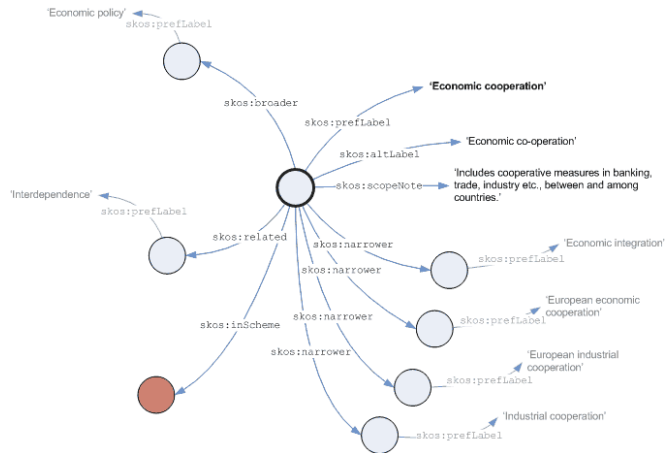
school → SN → institution for education

RT (related term) describes general term relationships (excluding BT/NT, USE, ...).

Příklad

topic maps → RT → knowledge management.

Thesauri – Example



prefix skos: <<http://www.w3.org/2004/02/skos/core#>>

<http://metadaten-twr.org/2011/01/19/>

skos-simple-knowledge-organisation-system, cit. 16.9.2012

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- ☺ simple, easy-to-use by non-experts in knowledge engineering
- ☹ problems in formal semantics:

☹ relationships can be used in several meanings:

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- ☹ problems in formal semantics:

BT relationships can be used in several meanings:

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

• subsumption, e.g. fruit BT apple,

• instance of, e.g. man BT David,

• part-whole, e.g. auto BT wheel.

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,
instance of , e.g. man BT David,
part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,

instance of , e.g. man BT David,

part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,

instance of , e.g. man BT David,

part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,

instance of , e.g. man BT David,

part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,

instance of , e.g. man BT David,

part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

Thesauri – Summary

- two ISO standards: single-language thesauri (ISO 2788:1986) or multiple-language thesauri (ISO 5964:1985).
- 😊 simple, easy-to-use by non-experts in knowledge engineering
- 😞 problems in formal semantics:

Example

BT relationships can be used in several meanings:

subsumption , e.g. fruit BT apple,

instance of , e.g. man BT David,

part of , e.g. auto BT wheel.

...

- semantic search, disambiguation, NLP

Topic Maps

- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurrences** and mutual **associations**.
- topics

Topic Maps – Topics

- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurences** and mutual **associations**.
- topics
 - ★ represent concepts – classes, instances, properties, etc.

- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurrences** and mutual **associations**.
- topics
 - represent concepts – classes, instances, properties, etc.
 - topics can have several **topic types**. The relationship “has type” build up a hierarchy of topics (analogy to *isa* relationships in semantics networks, or property *rdf:type* in RDF(S)).
 - each topics can have one or more **names** (e.g. nick, formal name, login name, etc.), each of which in different **variants** (e.g. visualization vs. sorting).

- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurrences** and mutual **associations**.
- topics
 - represent concepts – classes, instances, properties, etc.
 - topics can have several **topic types**. The relationship “has type” build up a hierarchy of topics (analogy to *isa* relationships in semantics networks, or property *rdf:type* in RDF(S)).
 - each topics can have one or more **names** (e.g. nick, formal name, login name, etc.), each of which in different **variants** (e.g. visualization vs. sorting).

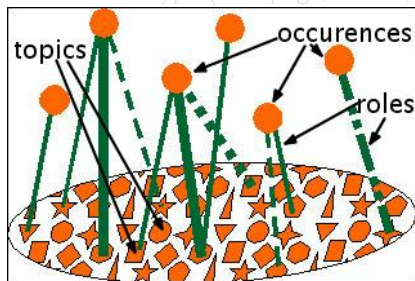
- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurrences** and mutual **associations**.
- topics
 - represent concepts – classes, instances, properties, etc.
 - topics can have several **topic types**. The relationship “has type” build up a hierarchy of topics (analogy to *isa* relationships in semantics networks, or property *rdf:type* in RDF(S)).
 - each topics can have one or more **names** (e.g. nick, formal name, login name, etc.), each of which in different **variants** (e.g. visualization vs. sorting).

- ISO standard – ISO/IEC 13250:2003
- three types of objects : **topics**, their **occurrences** and mutual **associations**.
- topics
 - represent concepts – classes, instances, properties, etc.
 - topics can have several **topic types**. The relationship “has type” build up a hierarchy of topics (analogy to *isa* relationships in semantics networks, or property *rdf:type* in RDF(S)).
 - each topics can have one or more **names** (e.g. nick, formal name, login name, etc.), each of which in different **variants** (e.g. visualization vs. sorting).

Topic Maps – Occurrences

- occurrences

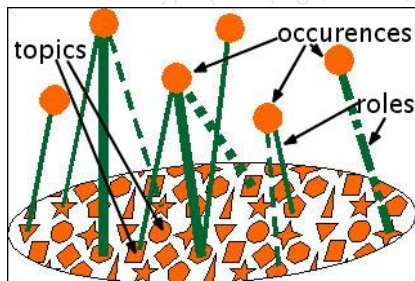
- represent “links” from topics to real documents/information resources.
- a topic is connected with an occurrence by means of a **role**, that determines the occurrence type (web page, article, book, etc.)



(<http://www.ontopia.net/topicmaps/materials/tao.html>)

Topic Maps – Occurrences

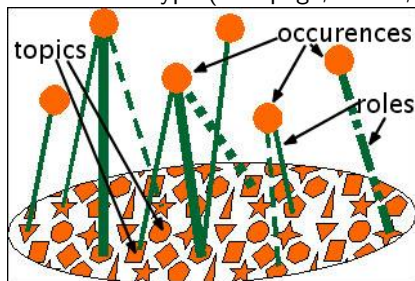
- occurrences
 - represent “links” from topics to real documents/information resources.
 - a topic is connected with an occurrence by means of a **role**, that determines the occurrence type (web page, article, book, etc.)



(<http://www.ontopia.net/topicmaps/materials/tao.html>)

Topic Maps – Occurrences

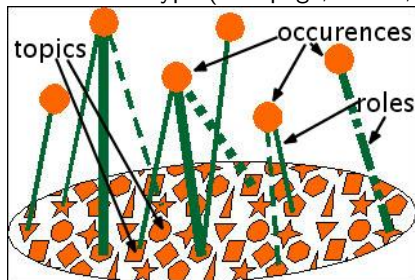
- occurrences
 - represent “links” from topics to real documents/information resources.
 - a topic is connected with an occurrence by means of a **role**, that determines the occurrence type (web page, article, book, etc.)



(<http://www.ontopia.net/topicmaps/materials/tao.html>)

Topic Maps – Occurrences

- occurrences
 - represent “links” from topics to real documents/information resources.
 - a topic is connected with an occurrence by means of a **role**, that determines the occurrence type (web page, article, book, etc.)



(<http://www.ontopia.net/topicmaps/materials/tao.html>)

- associations

- represent relationships between topics – analogy of n-ary relationships,
- an *association type* (which is a topic) is assigned to an association (topic type is a special association type),
- topics have so called **association roles** when connected to associations,
- each association role is assigned **association role type**, which is a topic, in turn.

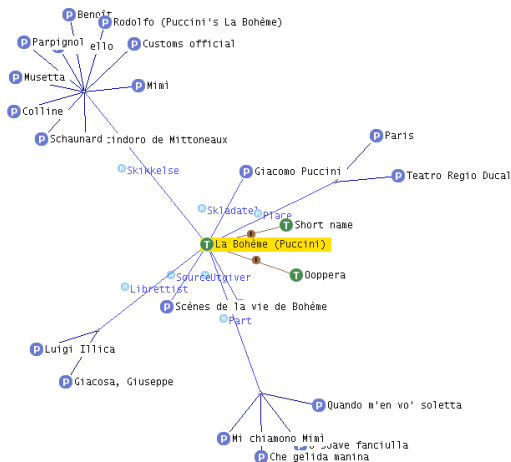
- associations
 - represent relationships between topics – analogy of n-ary relationships,
 - an *association type* (which is a topic) is assigned to an association (topic type is a special association type),
 - topics have so called **association roles** when connected to associations,
 - each association role is assigned **association role type**, which is a topic, in turn.

- associations
 - represent relationships between topics – analogy of n-ary relationships,
 - an *association type* (which is a topic) is assigned to an association (topic type is a special association type),
 - topics have so called **association roles** when connected to associations,
 - each association role is assigned **association role type**, which is a topic, in turn.

- associations
 - represent relationships between topics – analogy of n-ary relationships,
 - an *association type* (which is a topic) is assigned to an association (topic type is a special association type),
 - topics have so called **association roles** when connected to associations,
 - each association role is assigned **association role type**, which is a topic, in turn.

- associations
 - represent relationships between topics – analogy of n-ary relationships,
 - an *association type* (which is a topic) is assigned to an association (topic type is a special association type),
 - topics have so called **association roles** when connected to associations,
 - each association role is assigned **association role type**, which is a topic, in turn.

Topic Maps – Example

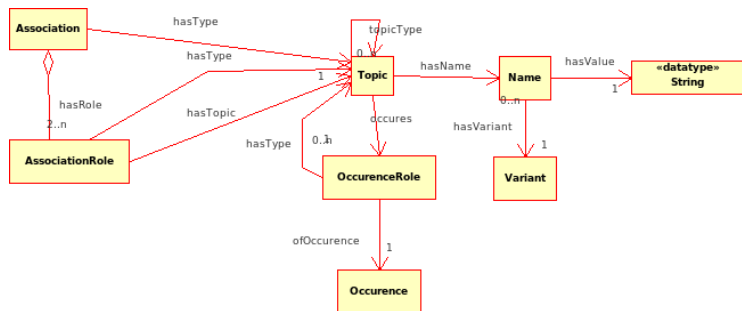


T ... topics

P ... partially expanded topics (except topic types)

R ... associations

Topic Maps – Model



- additionally, topic maps can be grouped into **contexts** (scopes, themes).
- querying using
 - TMQL
 - [http://www.gerstner.com/gerstner/STC/](#)

- additionally, topic maps can be grouped into **contexts** (scopes, themes).
- querying using
 - TMQL
 - tolog (syntactically similar to SQL)

- additionally, topic maps can be grouped into **contexts** (scopes, themes).
- querying using
 - TMQL
 - tolog (syntactically similar to SQL)

- additionally, topic maps can be grouped into **contexts** (scopes, themes).
- querying using
 - TMQL
 - tolog (syntactically similar to SQL)

- selected tools:

- Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
- TM4L
- TM4J

- links:

- <http://www.topicmapsonline.org/>
- <http://www.topicmapsonline.org/ontology/ontology.html>
- <http://www.topicmapsonline.org/ontology/ontology.html>

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:
 - <http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html>
 - <http://www.ontopia.net/topicmaps/>

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:
 - <http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html>
 - <http://www.kosek.cz/xml/tmtut/>

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:
 - <http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html>
 - <http://www.kosek.cz/xml/tmtut/>

- selected tools:
 - Ontopia (Ontopoly, Omnigator, Vizigator) – main stakeholder in Topic Maps
 - TM4L
 - TM4J
- links:
 - <http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html>
 - <http://www.kosek.cz/xml/tmtut/>

TM4L Viewer

The screenshot displays the TM4L Viewer interface with the following components:

- Topic Maps:** M file://home/krmen/downloads/zyopera.xml
- Topic Maps Indexes:** Subject Topics list including 'O voto', 'I work by Paul de Choudens', 'A Masked Ball', 'A pasant', 'Abbess', 'Abbi', 'Abbé de Chazeuil', 'Abdalo', 'Abhijanasakuntala', 'Abigail', 'Abreschviller', 'Acciano', 'Adam Mickiewicz', 'Adami', 'Adhemar de Monthell', 'Admèro', 'Adrianna', 'Adriana Lecocqueur', 'Adriana Lecocqueur (character)', 'Adrienne Lecocqueur', 'Aegean Island', 'Alra', 'Agnese', 'Ah! Fer l'ultima votal', 'Ah, il suo nome', 'Ah-Joe', 'Ah, Vergine Maria', 'Aida', 'Aida (character)', 'Alcindoro de Mittonneau', 'Aida di Ratenow', 'Aldobrandino dei Rangoni', 'Aldwort', 'Alaxis', 'Alfano', 'Alfo', 'Alfons Gernont', 'Alice', 'Alice Ford', 'Algi', 'Alms Collector', 'Alsace', 'Abernathivan', 'Alchiana', 'Alvaro', 'Alyse Badoero', 'Alzira', 'Alzira (character)', 'Alzire, ou Les Américains', 'Amalia'.
- Tree View:** Hierarchical structure of the topic map for 'La Bohème (Puccini)', including categories like 'Oppera', 'Short name', 'Web page', 'Skikkelse', 'Place', 'Librettist', 'Illustration', 'Synopsis', 'Utgiver', 'Ricordi', 'Premiere date', and 'Part'.
- Graph/Text View:** A network graph showing relationships between nodes such as 'Paris', 'Teatro Regio Duca', 'Giacosa, Giuseppe', 'Luigi Illica', 'Scènes de la vie de Bohème', 'Giacosa, Giuseppe', 'Luigi Illica', 'Part', 'Ricordi', 'La Bohème (Puccini)', 'Short name', 'Skikkelse', 'Parignot', 'Customs official', 'Narcello', 'Rodolfo (Puccini's La Bohème)', 'Mimi', 'Alcindoro de Mittonneau', 'Musetta', 'Schauvard', and 'senat'.
- Log:** A message at the bottom: 'Cannot activate TopicMapReference, because its in state StoreState (opening)'.

omnigator

zsyOpera | Custom | Filter | Export | Merge | Statistics | Query | Edit | No schema | Vlogate

La Bohème (Puccini)

Type(s): Opera

Untyped Names (1)

- La Bohème (Puccini)
 - Bohème (Puccini) - Scope: Sort

Short name (1)

- La Bohème - Scope: Puccini, Giacomo
 - Bohème - Scope: Puccini, Giacomo: Sort

Associations (21)

- Based on
 - Scènes de la vie de Bohème
- Composed by
 - Puccini, Giacomo
- Contexts
 - Cha galdà marina
 - Is chamazzo Mini
 - 0 saavi faveulla
 - Quando m'en vor soletta
- Dramatis personae
 - Alcindoro de Mittenieux
 - Bienet
 - Colline
 - Systema official
 - Marcello
 - Mimi
 - Musetta
 - Parginal
 - Rodolfo (Puccini's La Bohème)
 - Schaunard
- First performed at
 - Teatro Regio Ducal
- Libretto by
 - Giacosa, Giuseppe
 - Illica, Luigi
- Published by
 - Ricordi
- Takes place in
 - Paris

Subject Identifiers (1)

- http://en.wikipedia.org/wiki/La_Bohème

Internal Occurrences (2)

- Première date
 - 1899-02-01
- Video recording
 - 190 D&S

External Occurrences (11)

- Article
 - http://en.wikipedia.org/wiki/La_Bohème - Scope: Web; Wikipedia
- Illustration
 - <http://localhost:8080/operamap/occurrences/operapuccini/la-boheme-poster.jpg> - Scope: Local
- Libretto
 - <http://opera.starford.edu/operamapuccini/LaBohemeLibretto.html> - Scope: Opera Glass; Web
- Poster
 - http://www-4s.com/operapuccini/images/pictures/boheme_score.htm - Scope: OperaResource; Web
 - http://www-4s.com/operapuccini/images/pictures/boheme_poster.htm - Scope: OperaResource; Web
- Sound clip
 - <http://localhost:8080/operamap/occurrences/la-boheme.wav> - Scope: Local
 - <http://www.artsplus.net/spotmap/sample/operasounds/la-boheme.wav> - Scope: Web
- Synopsis
 - http://localhost:8080/operamap/occurrences/Bo_syn.php3.htm - Scope: Arizona Opera; Local
 - <http://www.arizonaopera.com/learn/synopsis/boheme.htm> - Scope: Arizona Opera; Web
 - <http://www.metopera.org/synopsis/boheme.html> - Scope: Opera News; Web
- Web page
 - <http://opera.starford.edu/operamapuccini/LaBohemeMain.html> - Scope: Opera Glass; Web

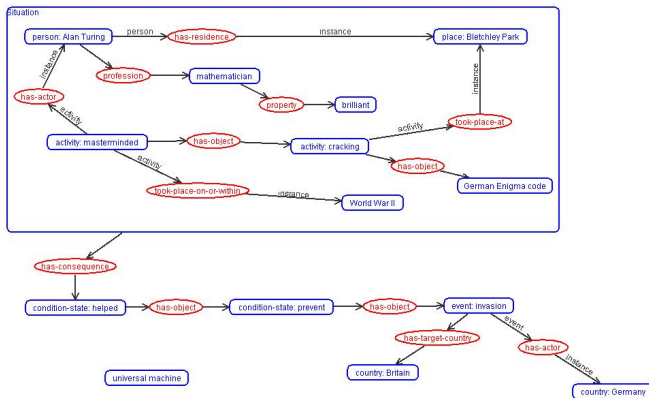
Scoped Names (1)

- Rodolfo (Puccini's La Bohème) (Rodolfo (Puccini's La Bohème))

Scoped Association Types (2)

Conceptual Graphs

Example



Conceptual Graphs

conceptual graph is a bipartite graph with two types of nodes (1) **concepts** a (2) **relations**.

concept has the form **concept type : referent**.

dog : Lucky

Dog Lucky

$\exists x \text{Dog}(x) \wedge \text{Name}(x, \text{Lucky})$

dog

Some dog

$\exists x \text{Dog}(x)$

dog : \forall

All dogs

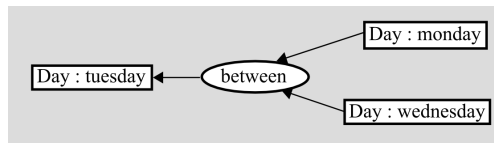
$\forall x \text{Dog}(x) \rightarrow \dots$

dog : {*}

Set of dogs

$\{ \}$ no FOPL

relation = predicate of arbitrary arity > 0 .



Conceptual Graphs

conceptual graph is a bipartite graph with two types of nodes (1) **concepts** a (2) **relations**.

concept has the form **concept type : referent**.

Example (Quantifier type)

dog : Lucky

Dog Lucky

$\exists x \text{Dog}(x) \wedge \text{Name}(x, \text{Lucky}) \dots$

dog

Some dog

$\exists x \text{Dog}(x) \dots$

dog : \forall

All dogs

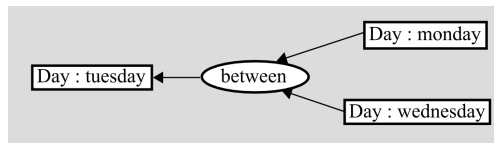
$\forall x \text{Dog}(x) \rightarrow \dots$

dog : {*}

Set of dogs

⊕ no FOPL

relation = predicate of arbitrary arity > 0 .



Conceptual Graphs

conceptual graph is a bipartite graph with two types of nodes (1) **concepts** a (2) **relations**.

concept has the form **concept type : referent**.

Example (Quantifier type)

dog : Lucky

Dog Lucky

$\exists x \text{Dog}(x) \wedge \text{Name}(x, \text{Lucky}) \dots$

dog

Some dog

$\exists x \text{Dog}(x) \dots$

dog : \forall

All dogs

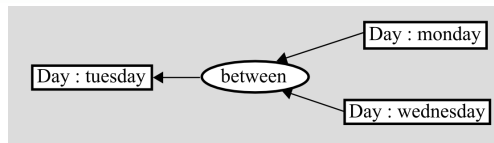
$\forall x \text{Dog}(x) \rightarrow \dots$

dog : {*}

Set of dogs

☹ no FOPL

relation = predicate of arbitrary arity > 0 .



Conceptual Graphs

conceptual graph is a bipartite graph with two types of nodes (1) **concepts** a (2) **relations**.

concept has the form **concept type : referent**.

Example (Quantifier type)

dog : Lucky

Dog Lucky

$\exists x \text{Dog}(x) \wedge \text{Name}(x, \text{Lucky}) \dots$

dog

Some dog

$\exists x \text{Dog}(x) \dots$

dog : \forall

All dogs

$\forall x \text{Dog}(x) \rightarrow \dots$

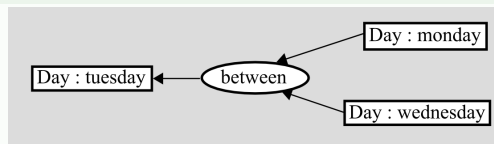
dog : {*}

Set of dogs

☹ no FOPL

relation = predicate of arbitrary arity > 0 .

Example (Ternary relation)



Conceptual Graphs

conceptual graph is a bipartite graph with two types of nodes (1) **concepts** a (2) **relations**.

concept has the form **concept type : referent**.

Example (Quantifier type)

dog : Lucky

Dog Lucky

$\exists x \text{Dog}(x) \wedge \text{Name}(x, \text{Lucky}) \dots$

dog

Some dog

$\exists x \text{Dog}(x) \dots$

dog : \forall

All dogs

$\forall x \text{Dog}(x) \rightarrow \dots$

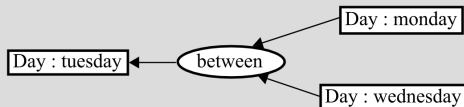
dog : {*}

Set of dogs

☹ no FOPL

relation = predicate of arbitrary arity > 0 .

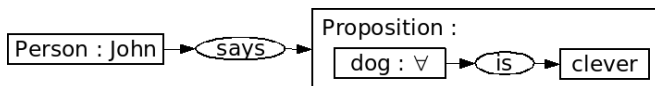
Example (Ternary relation)



Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

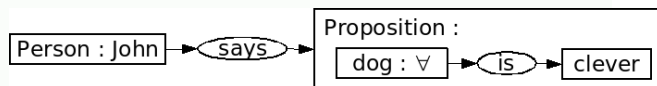


Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

Example (Context)

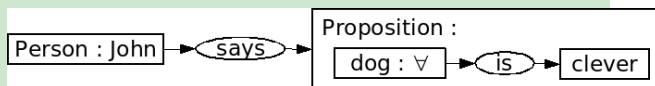


Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

Example (Context)



"John says, that all dogs are clever."

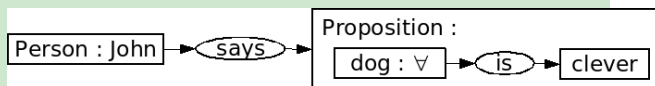
© no FOL

Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

Example (Context)



"John says, that all dogs are clever."

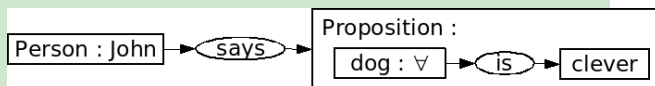
⊖ no FOL

Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

Example (Context)



"John says, that all dogs are clever."

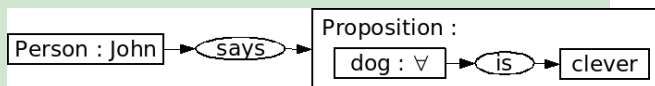
⊖ no FOL

Conceptual Graphs (2)

referent consists of **quantifier** (existential, or defined (universal, collective, etc.)), **designator** (instance identifier, e.g. name) and possibly **descriptor** (conceptual graph describing the concept).

context is a concept with empty descriptor

Example (Context)



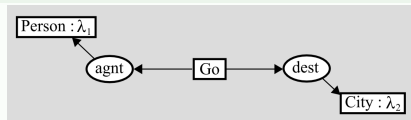
"John says, that all dogs are clever."

☹ no FOL

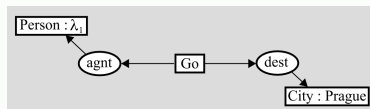
Conceptual Graphs (3)

lambda expressions correspond to “macros” – they allow defining relations by means of a “pattern” of the conceptual graph. Placeholders are denoted by λ_i symbols.

Example (lambda expressions)



def. binary relation “Go”.

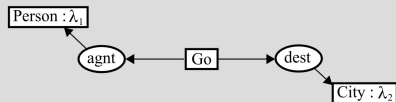


def. unary relation “Go to Prague”.

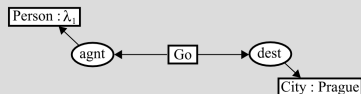
Conceptual Graphs (3)

lambda expressions correspond to “macros” – they allow defining relations by means of a “pattern” of the conceptual graph. Placeholders are denoted by λ_i symbols.

Example (lambda expressions)



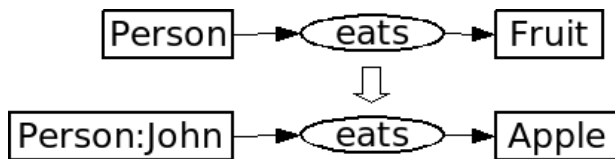
def. binary relation “Go”.



def. unary relation “Go to Prague”.

Conceptual Graphs – Inference

- inference makes use of several forward chaining rules¹ (graph generalization, specialization, equivalent changes).
- querying is performed using **projection** that looks for a conceptual graph pattern in another conceptual graph making use of the conceptual type hierarchy and conceptual relations.

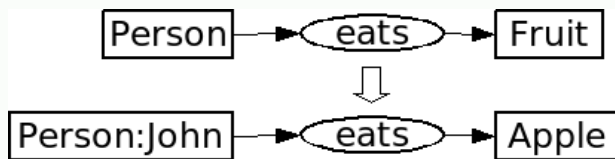


¹<http://www.jfsowa.com/cg/cgstandw.htm>

Conceptual Graphs – Inference

- inference makes use of several forward chaining rules¹ (graph generalization, specialization, equivalent changes).
- querying is performed using **projection** that looks for a conceptual graph pattern in another conceptual graph making use of the conceptual type hierarchy and conceptual relations.

Example – Projection

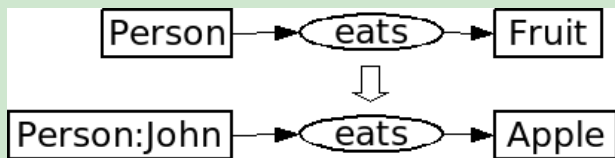


¹<http://www.jfsowa.com/cg/cgstandw.htm>

Conceptual Graphs – Inference

- inference makes use of several forward chaining rules¹ (graph generalization, specialization, equivalent changes).
- querying is performed using **projection** that looks for a conceptual graph pattern in another conceptual graph making use of the conceptual type hierarchy and conceptual relations.

Example – Projection



¹<http://www.jfsowa.com/cg/cgstandw.htm>

CharGer – CG editor

(<http://sourceforge.net/projects/charger>)

Notio – Java library + API for CG manipulation

(<http://backtrack.uwaterloo.ca/CG/projects/notio>)

Prolog+CG – inference engine for CG in Prolog

(<http://prologpluscg.sourceforge.net>)

Amine – newer version of Prolog+CG

(<http://amine-platform.sourceforge.net>)

DNA – annotation tool that visualizes the knowledge base
using CG

(<http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNAThome.html>)

Conceptual Graphs – Tools

CharGer – CG editor

(<http://sourceforge.net/projects/charger>)

Notio – Java library + API for CG manipulation

(<http://backtrack.uwaterloo.ca/CG/projects/notio>)

Prolog+CG – inference engine for CG in Prolog

(<http://prologpluscg.sourceforge.net>)

Amine – newer version of Prolog+CG

(<http://amine-platform.sourceforge.net>)

DNA – annotation tool that visualizes the knowledge base
using CG

(<http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNAThome.html>)

Conceptual Graphs – Tools

CharGer – CG editor

(<http://sourceforge.net/projects/charger>)

Notio – Java library + API for CG manipulation

(<http://backtrack.uwaterloo.ca/CG/projects/notio>)

Prolog+CG – inference engine for CG in Prolog

(<http://prologpluscg.sourceforge.net>)

Amine – newer version of Prolog+CG

(<http://amine-platform.sourceforge.net>)

DNA – annotation tool that visualizes the knowledge base using CG

(<http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNATHome.html>)

Conceptual Graphs – Tools

CharGer – CG editor

(<http://sourceforge.net/projects/charger>)

Notio – Java library + API for CG manipulation

(<http://backtrack.uwaterloo.ca/CG/projects/notio>)

Prolog+CG – inference engine for CG in Prolog

(<http://prologpluscg.sourceforge.net>)

Amine – newer version of Prolog+CG

(<http://amine-platform.sourceforge.net>)

DNA – annotation tool that visualizes the knowledge base using CG

(<http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNATHome.html>)

CharGer – CG editor

(<http://sourceforge.net/projects/charger>)

Notio – Java library + API for CG manipulation

(<http://backtrack.uwaterloo.ca/CG/projects/notio>)

Prolog+CG – inference engine for CG in Prolog

(<http://prologpluscg.sourceforge.net>)

Amine – newer version of Prolog+CG

(<http://amine-platform.sourceforge.net>)

DNA – annotation tool that visualizes the knowledge base using CG

(<http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNATHome.html>)

The screenshot displays the Amine4 software interface, titled "CG Operations Interface - Untitled". It features a menu bar (File, Edit, Format, Parameters, Ontology, CG Operations, Animation, Help) and a toolbar. The interface is divided into several sections:

- Input CG1:** A window showing a simple ontology with a box labeled "Sex = male" connected to a box labeled "Human" via a relationship labeled "sexOf".
- Input CG2:** A window showing a conceptual graph with nodes "Human", "Push", and "Object". Edges include "ageOf" from "Human" to "Push", "objPush" from "Object" to "Push", and "on" from "Push" to "Object".
- Output CG:** A window showing a combined conceptual graph that merges the elements from the two input windows.
- Left Panel:** A menu of operations such as "Equal", "Generalize", "Subsume", "SubsumeDiff", "Compare", "IsCanonic", and "Contract", each with a right-pointing arrow.
- Bottom Status Bar:** Shows the current ontology path: "Current LexiconOntology: /home/loisemen/programs/Amine4/samples/ontology/MaxOntology/Contract.xml".

- editing/viewing ontologies
- editing/viewing conceptual graphs
- CG operations – e.g.: JOIN
- CG+Prolog inference
- multiagent systems

The screenshot displays the Amine4 software interface, titled "CG Operations Interface - Untitled". It features a menu bar (File, Edit, Font, Parameters, Ontology, CG Operations, Animation, Help) and a toolbar. The interface is divided into several panes:

- Input CG1:** Shows a simple graph with a node "Sex = male" connected to "Human" via a "sexOf" relationship.
- Input CG2:** Shows a more complex graph with nodes "Human", "Push", and "Object". Relationships include "ageOf" from "Human" to "Push", "push" from "Push" to "Object", and "on" from "Push" to "Object".
- Output CG:** Shows a combined graph that merges the elements of the two input graphs.
- Left Panel:** A menu of operations such as "Equal", "Generalize", "Subsume", "Subsumed", "Compare", "IsCanonic", and "Contract".
- Bottom Status Bar:** Indicates the current ontology path: "Current LexiconOntology: /home/loisemen/programs/Amine4/samples/ontology/MaxOntology/Contract.xml".

- editing/viewing ontologies
- editing/viewing conceptual graphs
- CG operations – e.g.: JOIN
- CG+Prolog inference
- multiagent systems

The screenshot displays the Amine4 CG Operations Interface, which is used for editing and viewing ontologies and conceptual graphs. The interface is divided into several panes:

- Input CG1:** Shows a simple graph with a node 'Sex = male' connected to 'Human' via a 'sexOf' relationship.
- Input CG2:** Shows a more complex graph with nodes 'Human', 'Push', and 'Object'. 'Human' is connected to 'Push' via 'ageOf', and 'Push' is connected to 'Object' via 'on'.
- Output CG:** Shows the result of a 'JOIN' operation, combining elements from the input graphs. It includes nodes 'Sex = male', 'Human', 'Push', and 'Object', with relationships 'sexOf', 'ageOf', and 'on'.

On the left side, there is a menu with various operations such as 'Generalize', 'Subsume', 'SubsumeDiff', 'Analog', 'Compare', 'Coverably', 'IsCanonic', 'Expand', and 'Contract'. The status bar at the bottom indicates the current ontology path: 'Current LexiconOntology: /home/loewen/programs/Amine4/samples/ontology/MaxOntology/Context.xml'.

- editing/viewing ontologies
- editing/viewing conceptual graphs
- CG operations – e.g.: JOIN
 - CG+Prolog inference
 - multiagent systems

The screenshot displays the Amine4 CG Operations Interface, which is used for editing and viewing ontologies and conceptual graphs. It features three main editing windows:

- Input CG1:** Shows a simple graph with a node 'Sex = male' connected to 'Human super' via a 'sexOf' relation.
- Input CG2:** Shows a graph with 'Human' connected to 'age1', which is connected to 'push', which is connected to 'Object' via 'age1' and 'push' relations.
- Output CG:** Shows a more complex graph with 'Sex = male', 'Human', 'age1', 'push', and 'Object' nodes and their relationships.

The interface also includes a sidebar with various operations such as 'Equal', 'Generalize', 'Subsume', 'SubsumeDiff', 'Compare', 'IsCanonic', and 'Contract'. The current ontology is identified as 'MaxOntology/Context.xml'.

- editing/viewing ontologies
- editing/viewing conceptual graphs
- CG operations – e.g.: JOIN
- CG+Prolog inference
- multiagent systems

The screenshot displays the Amine4 CG Operations Interface, which is used for editing and viewing ontologies and conceptual graphs. It features three main editing windows:

- Input CG1:** Shows a simple graph with a node 'Sex = male' connected to 'Human super' via a 'sexOf' relation.
- Input CG2:** Shows a graph with 'Human' connected to 'age1', which is connected to 'push', which is connected to 'Object' via 'age1' and 'push' relations.
- Output CG:** Shows a more complex graph with 'Sex = male', 'Human', 'age1', 'push', and 'Object' nodes and their relationships.

The interface also includes a sidebar with various operations such as 'Equal', 'Generalize', 'Subsume', 'SubsumeDiff', 'Analog', 'Compare', 'Coverably', 'IsCanonic', 'Expand', and 'Contract'. The current ontology is identified as 'MaxOntology/Context.xml'.

- editing/viewing ontologies
- editing/viewing conceptual graphs
- CG operations – e.g.: JOIN
- CG+Prolog inference
- multiagent systems

Conceptual Graphs – Summary

- CG's (J.F. Sowa 80's) are representatives of formal (machine readable) and at the same time well readable, intuitive languages,
- are based on Pierce existential graphs [Sow00], [Dau01],
- are more expressive than FOPL – undecidability,
- to keep things decidable, so called *simple graphs* (J.F. Sowa 80's), were defined. They restrict the form of referents and prohibit contexts.

Conceptual Graphs – Summary

- CG's (J.F. Sowa 80's) are representatives of formal (machine readable) and at the same time well readable, intuitive languages,
- are based on Pierce existential graphs [Sow00], [Dau01],
- are more expressive than FOPL – undecidability,
- to keep things decidable, so called *simple graphs* (J.F. Sowa 80's), were defined. They restrict the form of referents and prohibit contexts.

Conceptual Graphs – Summary

- CG's (J.F. Sowa 80's) are representatives of formal (machine readable) and at the same time well readable, intuitive languages,
- are based on Pierce existential graphs [Sow00], [Dau01],
- are more expressive than FOPL – undecidability,
- to keep things decidable, so called *simple graphs* (J.F. Sowa 80's), were defined. They restrict the form of referents and prohibit contexts.

Conceptual Graphs – Summary

- CG's (J.F. Sowa 80's) are representatives of formal (machine readable) and at the same time well readable, intuitive languages,
- are based on Pierce existential graphs [Sow00], [Dau01],
- are more expressive than FOPL – undecidability,
- to keep things decidable, so called *simple graphs* (J.F. Sowa 80's), were defined. They restrict the form of referents and prohibit contexts.

Summary – what else ?

- we only quickly flew through the most important milestones in the crisp knowledge representation during last decades,
- ☹ most of these approaches have poorly defined semantics, which is a necessary condition for automated processing of large datasets,
- now, let's spend several weeks on formally precise logic-based knowledge representation languages with acceptable computational properties.

Summary – what else ?

- we only quickly flew through the most important milestones in the crisp knowledge representation during last decades,
- ☹ most of these approaches have poorly defined semantics, which is a necessary condition for automated processing of large datasets,
- now, let's spend several weeks on formally precise logic-based knowledge representation languages with acceptable computational properties.

Summary – what else ?

- we only quickly flew through the most important milestones in the crisp knowledge representation during last decades,
- ☹ most of these approaches have poorly defined semantics, which is a necessary condition for automated processing of large datasets,
- now, let's spend several weeks on formally precise logic-based knowledge representation languages with acceptable computational properties.



**OPPA European Social Fund
Prague & EU: We invest in your future.**
