CSC 481: Expressing Knowledge

Rodrigo Canaan
Assistant Professor
Computer Science Department
Cal Poly, San Luis Obispo
rcanaan@calpoly.edu

Goal

Review the knowledge engineering process by going over the First-Order Logic statements that make up the “Soap Opera” world

- At the end of this class, you should be able to represent simple english sentences in FOL

Vocabulary

Types of objects

Types of objects (Named individuals, or constants)

- People
- Animals
- Corporations
- Locations
- Physical objects (knives, earrings…)

Vocabulary

Properties

Properties (1-ary predicates)

“Types”	“Attributes”
• Person • Man • Place • Company	• Beautiful • HappilyMarried • ClosedFor Repairs • Bloody

What’s the conceptual difference between “Types” and “Attributes”?

Vocabulary

Relationships

Relationships (n-ary predicates)

- MarriedTo
- DaughterOf
- LivesAt
- HasCEO
- HadAnAffairWith
- LoveTriangle (3-ary)
- OccursInTimeInterval (3-ary: event, start time, end time)

Basic Facts

Basic Facts

- Man(john)
- Rich(john)
- Company(faultyInsuranceCompany)
- ¬HappilyMarried(jim)
- bestFriendOf(jim)=john
- fic=faultyInsuranceCompany
- ceoOf(fic) = john

- **Married(Alice, Bob)**

Complex Facts

Complex Facts

- $\forall y[Rich(y) \wedge Man(y) \rightarrow Loves(y, jane)]$
- $\forall y[Woman(y) \wedge y \neq jane \rightarrow Loves(y, john)]$
- $\forall x, y[Loves(x, y) \rightarrow \neg Blackmails(x, y)]$
- $Loves(jane, john) \vee Loves(jane, jim)$
- $\exists x[Adult(x) \wedge Blackmails(x, john)]$
- $\forall x[Lawyer(x) \rightarrow x = jane \vee x = jack \vee x = jim \vee \ldots]$
- $jane \neq john\ jane \neq john$

How would you write “every CEO is rich”?

Terminological Facts

Terminological Facts

- $\forall x[Man(x) \rightarrow \neg Woman(x)]$ (disjointness)
- $\forall x[Surgeon(x) \rightarrow Doctor(x)]$ (subtypes)
- $\forall x[Adult(x) \rightarrow Man(x) \vee Woman(x)]$ (exhaustiveness)
- $\forall x[MarriedTo(x, y) \rightarrow MarriedTo(y, x)]$ (symmetry)[1]
- $\forall x[ChildOf(x, y) \rightarrow ParentOf(y, x)]$ (inverses)
- $\forall x, y[MarriedTo(x, y) \rightarrow Person(x) \wedge Person(y)]$ (type restriction)
- $\forall x[RichMan(x) \equiv Rich(x) \wedge Man(x)]$ (full definition)

How would you write a sentence indicating that a predicate (e.g. TallerThan) is transitive?

Order of quantifiers

- Careful when mixing existential and universal quantifiers!
  - $\forall x \exists y : loves(x, y)$ “everybody loves someone” (possibly themselves)
  - $\exists y \forall x : loves(x, y)$ “someone is loved by everyone” (including themselves)
  - $\forall x \exists y : loves(x, y) \wedge x \neq y$ “everybody loves someone **else**”
  - $\exists y \forall x : x \neq y \rightarrow loves(x, y)$ “someone is loved by everyone (possibly excluding themselves)”
- If you’re using only one type of quantifier, order doesn’t matter

"Is there a company whose CEO loves Jane?"

$$KB \models \exists x[Company(x) \wedge Loves(ceoOf(x), jane)]?$$

Basic Facts

- Man(john)
- Rich(john)
- Company(faultyInsuranceCompany)
- ¬HappilyMarried(jim)
- bestFriendOf(jim)=john
- fic=faultyInsuranceCompany
- ceoOf(fic) = john

Complex Facts

- $\forall y[Rich(y) \wedge Man(y) \rightarrow Loves(y, jane)]$
- $\forall y[Woman(y) \wedge y \neq jane \rightarrow Loves(y, john)]$
- $\forall x, y[Loves(x, y) \rightarrow \neg Blackmails(x, y)]$
- $Loves(jane, john) \vee Loves(jane, jim)$
- $\exists x[Adult(x) \wedge Blackmails(x, john)]$
- $\forall x[Lawyer(x) \rightarrow x = jane \vee x = jack \vee x = jim \vee \ldots]$
- $jane \neq john\, jane \neq john$

Terminological Facts

- $\forall x[Man(x) \rightarrow \neg Woman(x)]$ (disjointness)
- $\forall x[Surgeon(x) \rightarrow Doctor(x)]$ (subtypes)
- $\forall x[Adult(x) \rightarrow Man(x) \vee Woman(x)]$ (exhaustiveness)
- $\forall x[MarriedTo(x, y) \rightarrow MarriedTo(y, x)]$ (symmetry)[1]
- $\forall x[ChildOf(x, y) \rightarrow ParentOf(y, x)]$ (inverses)
- $\forall x, y[MarriedTo(x, y) \rightarrow Person(x) \wedge Person(y)]$ (type restriction)
- $\forall x[RichMan(x) \equiv Rich(x) \wedge Man(x)]$ (full definition)

"Is there a company whose CEO loves Jane?"

$$KB \models \exists x[Company(x) \wedge Loves(ceoOf(x), jane)]?$$

Basic Facts

- Man(john)
- Rich(john)
- Company(faultyInsuranceCompany)
- ¬HappilyMarried(jim)
- bestFriendOf(jim)=john
- fic=faultyInsuranceCompany
- ceoOf(fic) = john

Yes

Complex Facts

- $\forall y[Rich(y) \wedge Man(y) \rightarrow Loves(y, jane)]$
- $\forall y[Woman(y) \wedge y \neq jane \rightarrow Loves(y, john)]$
- $\forall x, y[Loves(x, y) \rightarrow \neg Blackmails(x, y)]$
- $Loves(jane, john) \vee Loves(jane, jim)$
- $\exists x[Adult(x) \wedge Blackmails(x, john)]$
- $\forall x[Lawyer(x) \rightarrow x = jane \vee x = jack \vee x = jim \vee \ldots]$
- $jane \neq john\ jane \neq john$

Terminological Facts

- $\forall x[Man(x) \rightarrow \neg Woman(x)]$ (disjointness)
- $\forall x[Surgeon(x) \rightarrow Doctor(x)]$ (subtypes)
- $\forall x[Adult(x) \rightarrow Man(x) \vee Woman(x)]$ (exhaustiveness)
- $\forall x[MarriedTo(x, y) \rightarrow MarriedTo(y, x)]$ (symmetry)[1]
- $\forall x[ChildOf(x, y) \rightarrow ParentOf(y, x)]$ (inverses)
- $\forall x, y[MarriedTo(x, y) \rightarrow Person(x) \wedge Person(y)]$ (type restriction)
- $\forall x[RichMan(x) \equiv Rich(x) \wedge Man(x)]$ (full definition)

"If no man is blackmailing John, is he being blackmailed by someone he loves?"

$$KB \models \forall x[Man(x) \rightarrow \neg Blackmails(x, john)] \rightarrow$$
$$\exists y[Loves(john, y) \wedge Blackmails(y, john)]?$$

Basic Facts

- Man(john)
- Rich(john)
- Company(faultyInsuranceCompany)
- ¬HappilyMarried(jim)
- bestFriendOf(jim)=john
- fic=faultyInsuranceCompany
- ceoOf(fic) = john

Complex Facts

- $\forall y[Rich(y) \wedge Man(y) \rightarrow Loves(y, jane)]$
- $\forall y[Woman(y) \wedge y \neq jane \rightarrow Loves(y, john)]$
- $\forall x, y[Loves(x, y) \rightarrow \neg Blackmails(x, y)]$
- $Loves(jane, john) \vee Loves(jane, jim)$
- $\exists x[Adult(x) \wedge Blackmails(x, john)]$
- $\forall x[Lawyer(x) \rightarrow x = jane \vee x = jack \vee x = jim \vee \ldots]$
- $jane \neq john\ jane \neq john$

Terminological Facts

- $\forall x[Man(x) \rightarrow \neg Woman(x)]$ (disjointness)
- $\forall x[Surgeon(x) \rightarrow Doctor(x)]$ (subtypes)
- $\forall x[Adult(x) \rightarrow Man(x) \vee Woman(x)]$ (exhaustiveness)
- $\forall x[MarriedTo(x, y) \rightarrow MarriedTo(y, x)]$ (symmetry)[1]
- $\forall x[ChildOf(x, y) \rightarrow ParentOf(y, x)]$ (inverses)
- $\forall x, y[MarriedTo(x, y) \rightarrow Person(x) \wedge Person(y)]$ (type restriction)
- $\forall x[RichMan(x) \equiv Rich(x) \wedge Man(x)]$ (full definition)

Entailments

"If no man is blackmailing John, is he being blackmailed by someone he loves?"

$$KB \models \forall x[Man(x) \rightarrow \neg Blackmails(x, john)] \rightarrow$$
$$\exists y[Loves(john, y) \wedge Blackmails(y, john)]?$$

Yes!

What if we don't assume that no man is blackmailing John?

Then no!

Reification

Turning a predicate or statement into an addressable object

- Expressing statements at an arbitrary level of granularity
- Handling incompatible statements,
- Handling intentions, beliefs, etc held by different actors
- Representing values that can be expressed in many different units

Shortcomings of FOL

It is hard to represent:

- Statistical and probabilistic facts (“Jane is 50% likely to go to the party)
- “Fuzzy” degrees of truth (“John is somewhat tall”, “Bob is very tall”)
- Default and prototypical facts (“Most CEOs are rich”)
- Temporal events
- Intentional facts (Beliefs, desires, beliefs about beliefs, etc.)
  - Some of these are somewhat addressable with reification

Next chapter - Resolution

- So far, we have done inference “by hand”
- Reasoning with FOL is usually done by asking questions of the form:

$$KB \models \alpha?$$

- This is the equivalent of asking if α is satisfied in every interpretation where KB is satisfied.
- We want to mechanize the process of answering this question with the rule of Resolution