# Formalizing Commonsense Reasoning

Lecture 7

February 27, 2018

# Take a Deep Breath . . .

- So what have we learned so far?
- A lot: We have learned that there is a powerful logic (namely FOPL) for which there are sound and complete, albeit semi-decidable, inference procedures.
- Two things remain, however:
  1. How powerful is FOPL? Is it appropriate for the kind of commonsense reasoning that we need?
  2. Assuming it is powerful enough, how easy is it to formalize commonsense reasoning?
- In the coming lectures, we look at these two points in depth.
- In this lecture, we venture into a brief investigation of the second point.
- Take a deep breath . . .

# Translation

- We have seen that reasoning could be reduced to the formal manipulation of symbol structures.
- Commonsense reasoning is the formal manipulation of symbol structures representing commonsense knowledge.
- Commonsense knowledge is naturally expressible in natural language statements.
- Thus, before attempting to formalize commonsense reasoning, we need to look at issues in the translation from natural language to symbol structures, namely FOPL.
- Without loss of generality, we will concentrate on the case of English.

# Translation Basics

- A typical translation scheme:
  - Proper nouns $\leftrightarrow$ Constants.
  - Verbs (verb to-be is special, though) $\leftrightarrow$ Predicate symbols (of arity 1, 2 or 3).
  - Nouns $\leftrightarrow$ Unary predicate symbols.
  - Adjectives $\leftrightarrow$ Unary predicate symbols.
  - Prepositions $\leftrightarrow$ Predicate symbols of arity 2 or more.
- **Possible paper topics:**
  - **Study linguistic theories of one of the above categories and propose a more elaborate representation scheme.**
  - **What about adverbs? Determiners?**
- For auxiliary verbs, tense, and aspect, wait for the coming lectures.

# Examples

- – Fido is a dog.
  – $\textsc{Dog}(\mathsf{Fido})$
- – Fido is a black dog.
  – $\textsc{Dog}(\mathsf{Fido}) \wedge \textsc{Black}(\mathsf{Fido})$
- – Fido likes Lacy.
  – $\textsc{Likes}(\mathsf{Fido}, \mathsf{Lacy})$

# Beware!

- –
  - $\textsc{Dog}(\mathsf{Fido})$
- –
  - $\textsc{Dog}(\mathsf{Fido}) \wedge \textsc{Black}(\mathsf{Fido})$
- –
  - $\textsc{Likes}(\mathsf{Fido}, \mathsf{Lacy})$
- The pretend-it's-English semantics is often convenient, but it is often dangerous.
  - It gives the false impression of completeness.
- Informal intensional semantics is useful here.
  - Extensional semantics is useful when we need to prove something about the system, but it is often not convenient.

# Life is Not That Simple

- – Fido is a smart dog.
  – ??? $\textsc{Dog}(\mathsf{Fido}) \wedge \textsc{Smart}(\mathsf{Fido})$
- – Scooby-Doo is a cartoon dog.
  – ??? $\textsc{Dog}(\mathsf{SD}) \wedge \textsc{Cartoon}(\mathsf{SD})$
- – Tweety is a water bird.
  – ??? $\textsc{Bird}(\mathsf{Tweety}) \wedge \textsc{Water}(\mathsf{Tweety})$

# Quantification

- - Some lion is brave.
  - There is a brave lion.
  - $\exists x\ (\textsc{Lion}(x) \wedge \textsc{Brave}(x))$
- - All lions are brave.
  - Every lion is brave.
  - Each lion is brave.
  - $\forall x\ (\textsc{Lion}(x) \Rightarrow \textsc{Brave}(x))$
- What is wrong with
  - $\exists x\ (\textsc{Lion}(x) \Rightarrow \textsc{Brave}(x))$ and
  - $\forall x\ (\textsc{Lion}(x) \wedge \textsc{Brave}(x))$

  respectively?

# Scope and Ambiguity

- - Every man loves a woman.
  - $\forall x(\textsc{Man}(x) \Rightarrow \exists y(\textsc{Woman}(y) \ \wedge \ \textsc{Loves}(x, y)))$
  - $\exists y(\textsc{Woman}(y) \wedge \forall x(\textsc{Man}(x) \ \Rightarrow \ \textsc{Loves}(x, y)))$
- How about
  - All men love a woman.
- - There is a barber in town who shaves every man that does not shave himself.
  - **Do it.**

# Uniqueness

- – There is exactly one brave lion.
  – ?

# Uniqueness

- – There is exactly one brave lion.
  – $\exists x[\textsc{Lion}(x) \wedge \textsc{Brave}(x) \wedge \forall y[(\textsc{Lion}(y) \wedge \textsc{Brave}(y)) \Rightarrow y = x]\,]$
- How about two lions? At least? At most?

# Tricky Cases

- Generics
  - A lion is brave.
- Donkey sentences
  - Every farmer who owns a donkey beats it.

# Generalized Light-Switch World: I

- Consider attempting to formalize a generalized light-switch world.
- We start with a rather simple world:
  - Parallel branches each consisting of a single switch and a single light source connected in series.
- We need to setup a KB such that a "user" may enter a description of any particular situation and query the KB about which lights are on.

# Language

- Constants:
  - $\mathsf{S}i$ ($i \in \mathbb{N}$), which we will use to name switches.
  - $\mathsf{L}i$ ($i \in \mathbb{N}$), which we will use to name light sources.
- Predicate symbols:
  1. $\textsc{Lt}$, where $[\textsc{Lt}(x)] =$ $[x]$ is a light source.
  2. $\textsc{Sw}$, where $[\textsc{Sw}(x)] =$ $[x]$ is a switch.
  3. $\textsc{On}$, where $[\textsc{On}(x)] =$ light source $[x]$ is on.
  4. $\textsc{Down}$, where $[\textsc{Down}(x)] =$ switch $[x]$ is down.
  5. $\textsc{Con}$, where $[\textsc{Con}(x, y)] =$ light source $[x]$ is connected to the power supply through switch $[y]$.
  6. $=$, where $[x = y] =$ $[x]$ is identical to $[y]$.
- Note: we probably need a **sorted logic** here.

# Axioms

1. $\forall x[\textsc{Lt}(x) \Rightarrow (\textsc{On}(x) \Leftrightarrow \exists y[\textsc{Sw}(y) \ \wedge \ \textsc{Con}(x,y) \ \wedge \ \textsc{Down}(y)]) \ ]$
2. $\forall x, y[(\textsc{Lt}(x) \ \wedge \ \textsc{Sw}(y) \ \wedge \ \textsc{Con}(x,y)) \Rightarrow$
   $\forall z[(\textsc{Lt}(z) \ \wedge \ \textsc{Con}(z,y)) \Rightarrow z = x]]$
3. $\forall x, y[(\textsc{Lt}(x) \ \wedge \ \textsc{Sw}(y) \ \wedge \ \textsc{Con}(x,y)) \Rightarrow$
   $\forall z[(\textsc{Sw}(z) \ \wedge \ \textsc{Con}(x,z)) \Rightarrow z = y]]$

# Example

- Situation I:
  - $\textsc{Lt}(\mathsf{L1})$, $\textsc{Lt}(\mathsf{L2})$, $\textsc{Lt}(\mathsf{L3})$
  - $\textsc{Sw}(\mathsf{S1})$, $\textsc{Sw}(\mathsf{S2})$, $\textsc{Sw}(\mathsf{S3})$
  - $\textsc{Con}(\mathsf{L1}, \mathsf{S1})$
  - $\textsc{Con}(\mathsf{L2}, \mathsf{S2})$
  - $\textsc{Con}(\mathsf{L3}, \mathsf{S3})$
  - $\textsc{Down}(\mathsf{S1})$
  - $\textsc{Down}(\mathsf{S2})$
  - $\neg\textsc{Down}(\mathsf{S3})$
- Can we now infer which lights are on and which are not?
  - Yes, but we need to enhance our inference engine with a rule for dealing with "=". This rule is referred to as **paramodulation**.

# Generalized Switch-World: II

- Consider the following generalization:

  A single light source may be connected to multiple switches.

- Remove Axiom 3.

# Example

- Situation II:
  - $\textsc{Lt}(\mathsf{L1})$, $\textsc{Lt}(\mathsf{L2})$, $\textsc{Lt}(\mathsf{L3})$
  - $\textsc{Sw}(\mathsf{S1})$, $\textsc{Sw}(\mathsf{S2})$, $\textsc{Sw}(\mathsf{S3})$
  - $\textsc{Con}(\mathsf{L1}, \mathsf{S1})$
  - $\textsc{Con}(\mathsf{L2}, \mathsf{S2})$
  - $\textsc{Con}(\mathsf{L3}, \mathsf{S3})$
  - $\textsc{Down}(\mathsf{S1})$
  - $\textsc{Down}(\mathsf{S2})$
  - $\neg\textsc{Down}(\mathsf{S3})$
- Can we now infer which lights are on and which are not?

# Domain Closure

- We can infer $\textsc{On}(\mathsf{L1})$ and $\textsc{On}(\mathsf{L2})$.
- We cannot infer $\neg\,\textsc{On}(\mathsf{L3})$, though.
- Why?
  - There could be other switches in addition to $[\mathsf{S1}]$, $[\mathsf{S2}]$, and $[\mathsf{S3}]$.
- Add the **domain closure axiom**:

$$\forall x[\textsc{Sw}(x) \Rightarrow (x = \mathsf{S1} \;\vee\; x = \mathsf{S2} \;\vee\; x = \mathsf{S3})]$$

- Can we now derive $\neg\,\textsc{On}(\mathsf{L3})$?

# Unique Names Axioms

- No, we'll get stuck trying to prove that [$\mathsf{L3}$] is neither [$\mathsf{L1}$] nor [$\mathsf{L2}$]!
- We need to add the **unique names axioms**:

  $\mathsf{L1} \neq \mathsf{L2}$

  $\mathsf{L1} \neq \mathsf{L3}$

  $\mathsf{L2} \neq \mathsf{L3}$
- Do we need to add axioms of the form

  $\mathsf{L}i \neq \mathsf{S}j$?
- How many such axioms would we need, in general?
- Note that, now, we can derive $\neg\, \textsc{On}(\mathsf{L3})$.

# Generalized Switch-World: III

- Consider the following generalization:

  Multiple light sources may be connected to multiple switches.

- Remove Axiom 2.

# Sad News

- We can infer $\textsc{On}(\mathsf{L1})$ and $\textsc{On}(\mathsf{L2})$.
- We cannot infer $\neg\, \textsc{On}(\mathsf{L3})$, though.
- Why?
  - Nothing prevents [L3] from being connected to [S1], [S2], or [S3].
- Here, we have to explicitly state, not only which connections do exist, but also which connections do not!
- How do we deal with this mess? Wait for "non-monotonic logic".