TDT4136 Introduction to Artificial Intelligence

Lecture 6: First Order Logic

Chapter 8 in the textbook

Keith L. Downing

The Norwegian University of Science and Technology
Trondheim, Norway
keithd@idi.ntnu.no

September 25, 2024

Outline

- Recap - Propositional logic
- FOL (Predicate) logic
  - Syntax and semantics
- Next week: Inference in FOL (chapter 9)

- Logic is the scientific study of validity - allows us to test validity related to
  - answering questions
  - making decisions about plans, verifying designs, solving problems in general

- Logic is the scientific study of validity - allows us to test validity related to
  - answering questions
  - making decisions about plans, verifying designs, solving problems in general
- Logic provides a knowledge representation language and an inference mechanism

- Logic is the scientific study of validity - allows us to test validity related to
    - answering questions
    - making decisions about plans, verifying designs, solving problems in general
- Logic provides a knowledge representation language and an inference mechanism
- There are various logical languages with different expressive power

Problem solving in Logic

- Problem description:
  - If it is autumn, then it is lamb-meat season in Norway.
  - If it is lamb-meat season in Norway, then "får i kål" is delicious.
  - "Får i kål" is not delicious.
- Question: Is it autumn?
- How to translate this to Propositional Logic?

Propositional Logic - ”Får i kål” example

- Propositions:

Propositional Logic - "Får i kål" example

- Propositions:
  P: It is autumn
  Q: It is lamb-meat season
  R: "Får i kål" is delicious
- Premises?

Propositional Logic - "Får i kål" example

- Propositions:
  P: It is autumn
  Q: It is lamb-meat season
  R: "Får i kål" is delicious

- Premises?
  $P \implies Q$
  $Q \implies R$
  $\neg R$

Propositional Logic - "Får i kål" example

- Propositions:
  P: It is autumn
  Q: It is lamb-meat season
  R: "Får i kål" is delicious

- Premises?
  $P \implies Q$
  $Q \implies R$
  $\neg R$

- How to answer the Question "Is it autumn", i.e., whether the proposition **P** is true or false?

Is it Autumn?

Two main methods for testing/checking if **P** is true - i.e., if KB entails P:

- by Model Checking with Truth Table

- by Theorem Proving

Model checking with a Truth table - "Får i kål" example

P	Q	R	P→Q	Q→R	¬R	P	¬P
T	T	T	T	T	F	T	F
T	T	F	T	F	T	T	F
T	F	T	F	T	F	T	F
T	F	F	F	T	T	T	F
F	T	T	T	T	F	F	T
F	T	F	T	F	T	F	T
F	F	T	T	T	F	F	T
F	F	F	T	T	T	F	T

Theorem Proving. - "Får i kål" example

Proof: a sequence of sentences, where each is a premise (i.e., given) or is derived from earlier sentences in the proof by an inference rule.

At the end of the sequence of the sentences there will be the target/goal/query if it can be proven true.

Theorem Proving. - "Får i kål" example

Proof: a sequence of sentences, where each is a premise (i.e., given) or is derived from earlier sentences in the proof by an inference rule.

At the end of the sequence of the sentences there will be the target/goal/query if it can be proven true.

To answer P? (Is it autumn?) we'll use the *resolution rule* twice:

Theorem Proving. - "Får i kål" example

Proof: a sequence of sentences, where each is a premise (i.e., given) or is derived from earlier sentences in the proof by an inference rule.

At the end of the sequence of the sentences there will be the target/goal/query if it can be proven true.

To answer P? (Is it autumn?) we'll use the *resolution rule* twice:

Resolution rule:

$$\frac{\begin{array}{l} A \lor B \\ \neg A \end{array}}{B}$$

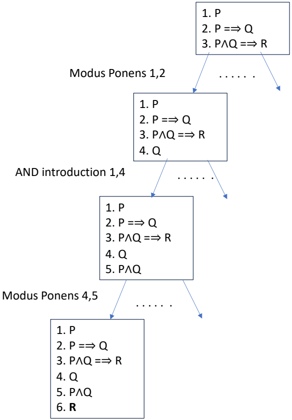
Theorem Proving - "Får i kål" example

- The KB has these sentences (premises), and we ASK the logic system if it is autumn?
  1. P $\implies$ Q (If it is autumn, then it is lamb-meat season)
  2. Q $\implies$ R. (If it is lamb-meat season then "får i kål" is delicious)
  3. $\neg R$ ("Får i kål" is not delicious)
- Proof:
  4. **Resolution rule** on 2 and 3: $\neg$ Q
  5. **Resolution rule** on 1 and 4: $\neg$ P - Goal sentence.

Proof as Search

- At any step in the proof process, there may be several inferences that can be applied to the KB.
- We can therefore imagine the proof like a search process.
- Problem definition
    - What is a state?
    - what is an action?
    - what is goal?

Proof as search- example



State: KB

Initial State: $\mathbf{P}, \mathbf{P} \implies \mathbf{Q}, \mathbf{P} \wedge \mathbf{Q} \implies \mathbf{R}.$

Actions: Inference rules

Goal: KB including the goal sentence (i.e., **R**)

More efficient Inference methods

- Searching on the state space (shown as above) is exponential in nr. of propositions
- More efficient methods, using specific forms of sentences

  Conjuctive Normal Form -CNF

  Horn Clasuses

CNF and Resolution Refutation

- CNF: Conjunction of Clauses (i.e.,disjunctive literals)
- Used in Resolution Refutation
  - To prove P, given KB, show that KB and $\neg$ P is unsatisfiable/false/empty clause.
  - Repeatedly use only one inference rule: Resolution rule.

Horn Clauses and Forward/backward Chaning)

- Horn clause: A clause with at most one positive literal
- Used in Forward/Backward chaining
- To prove P, given KB, repeatedly apply Modus Ponens in KB.

Propositional logic

- Less expressive than First order/Predicate logic
- PL assumes that the world contains *facts* that are true or false
- Propositional constants refer to atomic propositions:
  - R: It is raining
  - S: It is snowing
  - W: It is wet
- Compound sentences capture *relationships* among propositions:
  $R \vee S \implies W$
- Ontological commitment of PL: does not include objects, properties of objects, and relationships between objects.

Limitations of propositional logic

- Assume we want to express *Every student likes vacation*:

  John likes vacation $\wedge$
  Mary likes vacation $\wedge$
  Ann likes vacation $\wedge$
  ....

Limitations of propositional logic

- Assume we want to express *Every student likes vacation*:

  John likes vacation $\wedge$
  Mary likes vacation $\wedge$
  Ann likes vacation $\wedge$
  ....
- Problem: KB grows large
- Possible solution: ?

Limitations of propositional logic

- Assume we want to express *Every student likes vacation*:

  John likes vacation $\wedge$
  Mary likes vacation $\wedge$
  Ann likes vacation $\wedge$
  ....

- Problem: KB grows large
- Possible solution: ?
  All students like vacation.

Limitations of propositional logic

- Assume the KB has:
  - Stig is older than Sissel
  - Sissel is older than Paul
  - Stig is older than Sissel $\wedge$ Sissel is older than Paul $\implies$ Stig is older than Paul

Limitations of propositional logic

- Assume the KB has:
  - Stig is older than Sissel
  - Sissel is older than Paul
  - Stig is older than Sissel $\wedge$ Sissel is older than Paul $\implies$ Stig is older than Paul

  - We can derive *Stig is older than Paul*

Example on Limitations of propositional logic

- Assume we add *Hanne is older than Sissel* into the KB
- The current KB now:
  - Stig is older than Sissel
  - Sissel is older than Paul
  - Stig is older than Sissel $\wedge$ Sissel is older than Paul $\implies$ Stig is older than Paul
  - *Hanne is older than Sissel*

Example on Limitations of propositional logic

- Assume we add *Hanne is older than Sissel* into the KB
- The current KB now:
  - Stig is older than Sissel
  - Sissel is older than Paul
  - Stig is older than Sissel $\wedge$ Sissel is older than Paul $\implies$ Stig is older than Paul
  - *Hanne is older than Sissel*
- What else do we need to have in the KB in order to derive *Hanne is older than Paul*?

Example on Limitations of propositional logic

- Assume we add *Hanne is older than Sissel* into the KB
- The current KB now:
  - Stig is older than Sissel
  - Sissel is older than Paul
  - Stig is older than Sissel $\wedge$ Sissel is older than Paul $\implies$ Stig is older than Paul
  - *Hanne is older than Sissel*
- What else do we need to have in the KB in order to derive *Hanne is older than Paul*?
- We need:

  Hanne is older than Sissel $\wedge$ *Sissel is older than Paul* $\implies$ *Hanne is older than Paul*

Limitations of propositional logic -Example 2

- KB grows large
- Possible solution: ?
  PersA is older than PersB $\wedge$ PersB is older than PersC $\implies$ PersA is older than PersC

Limitations of Propositional Logic -example 3: Wumpus

- Consider the statement "If there is breeze in a square,there must be pit in an adjacent square"
- In propositional logic we need 16 sentences (one for each square) to represent this statement (for 4x4 grid):
  - B1,1 $\implies$ P1,2 $\vee$ P2,1
  - B1,2 $\implies$ P1,2 $\vee$ P1,3 $\vee$ P2,2
  - ...
  - ...
- **We want to be able to say this in one single sentence.**

How to say it in one sentence

- Our statement above refers to 2 types objects (*pit* and *square*). The square has the property to be *breezy*. The relationship between a square and pit is *adjacency*, i.e., neighbourhood.

- In FOL, this statement is represented by means of the following formula - instead of 16 sentences in propositional logic,:

  $\forall$ square, adjacent(square,pit) $\implies$ breezy(square)

First Order Logic (FOL)

- More expressive than propositional logic. While PL assumes that the world contains facts, FOL assumes the world contains
  - Objects: trees, people, numbers, movies,Trump, maps, colours, hypotheses, Wumpus....

First Order Logic (FOL)

- More expressive than propositional logic. While PL assumes that the world contains facts, FOL assumes the world contains
  - Objects: trees, people, numbers, movies,Trump, maps, colours, hypotheses, Wumpus....
  - Relations: square, smelly, brother, older than, owns, has colour, adjacent to....

First Order Logic (FOL)

- More expressive than propositional logic. While PL assumes that the world contains facts, FOL assumes the world contains
    - Objects: trees, people, numbers, movies,Trump, maps, colours, hypotheses, Wumpus....
    - Relations: square, smelly, brother, older than, owns, has colour, adjacent to....
    - Functions: brother of, colour-of, adjacent to,....

Syntax of FOL - elements

- Constants represents objects. NTNU, KingHarald, 5, ...
- Predicates represents relations. Brother, $>$, $=$, ...
- Functions represents functions: Sqrt, LeftLegOf
- Variables: x, y, a, b, ...
- Connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Quantifiers: $\forall, \exists$

Functions versus Relations

- Functions are a way of referring to individuals indirectly, e.g.,

  BrotherOf(Janne) and Edvard would refer to the same individual if Janne's brother is the person named Edvard.

Functions versus Relations

- Functions are a way of referring to individuals indirectly, e.g.,
  BrotherOf(Janne) and Edvard would refer to the same individual if Janne's brother is the person named Edvard.

- Relations hold among objects - unary, binary, n-ary relations
  Brother(Janne , Edvard) is true if Edvard is Janne's brother

Syntax

- Atomic sentence: predicate(term$_1$,...,term$_n$)
  Term: constant, variable, or function(term$_1$,...,term$_m$)
- Complex sentences: Composed of atomic sentences using connectives

  $\neg S$, $S_1 \vee S_2$, $S_1 \wedge S_2$, $S_1 \implies S_2$, $S_1 \Leftrightarrow S_2$

  Examples:
  - Brother(JonSnow, AryaStark) $\implies$ Sister(AryaStark, JonSnow)
  - Dreaming(Garcia) $\vee$ Today(Monday)

Quantifiers

- Quantifications express properties of collections of objects.
- Universal Quantifier, $\forall$: "For all"
- $\forall \langle \textit{variable} \rangle \ \langle \textit{sentence} \rangle$

Quantifiers

- Quantifications express properties of collections of objects.
- Universal Quantifier, $\forall$: "For all"
- $\forall \langle \textit{variable} \rangle \ \ \langle \textit{sentence} \rangle$
- We can state $\forall$ x P(x)
  - English translation: "for all values of x, P(x) is true"
  - Example: P(x) : $x+1 \geq x$
  - English translation: "for all values of x, $x+1 \geq x$ is true"

Universal quantification

- Everyone at NTNU is smart:
  $\forall x \;\; At(x, NTNU) \implies Smart(x)$
- $\forall x \;\; P$ is true in a model m iff P is true with x being each possible object in the model

Universal quantification

- Everyone at NTNU is smart:
  $\forall x \;\; At(x, NTNU) \implies Smart(x)$
- $\forall x \;\; P$ is true in a model m iff P is true with x being each possible object in the model
- Equivalent to the conjunction of instantiations of P

$$\begin{array}{ll} & (At(KingJohn, NTNU) \implies Smart(KingJohn)) \\ \wedge & (At(Richard, NTNU) \implies Smart(Richard)) \\ \wedge & (At(NTNU, NTNU) \implies Smart(NTNU)) \\ \wedge & \ldots \end{array}$$

Existential quantification

- $\exists$: "There exist a/some"
- $\exists \langle variables \rangle \; \langle sentence \rangle$
- Someone at NTNU is smart:
  $\exists x \;\; At(x, NTNU) \land Smart(x)$

Existential quantification

- $\exists$: "There exist a/some"
- $\exists \langle variables \rangle \ \langle sentence \rangle$
- Someone at NTNU is smart:
  $\exists x \ At(x, NTNU) \wedge Smart(x)$
- $\exists x \ P$ is true in a model m iff P is true with x being **some** possible object in the model
- Equivalent to the disjunction of instantiations of P

$$
\begin{array}{ll}
 & (At(KingJohn, NTNU) \wedge Smart(KingJohn)) \\
\vee & (At(Richard, NTNU) \wedge Smart(Richard)) \\
\vee & (At(NTNU, NTNU) \wedge Smart(NTNU)) \\
\vee & \ldots
\end{array}
$$

Exercise: A Few First-Order Logic Translations

Convert these sentences to First Order Logic

- I am happy only if my spouse is happy.
- Claudia is happy if every one of her friends clicks *like* for all of her posts.
- Claudia is happy only if at least one of her friends *shares* all of her posts.
- Wherever you go, Big Brother is watching you.
- Joe never works past 5 pm.

Answers

These work. Others are possible as well.

- $Happy(me) \Rightarrow Happy(spouse(me))$ ::: In most countries, spouse is a function: single output for any input

  ...alternatively....

- $\exists Y : spouse(me, Y) \wedge [Happy(me) \rightarrow Happy(Y)]$
- $$[\forall F, P : (post(Claudia, P) \wedge friend(Claudia, F)) \Rightarrow likes(F, P)] \\ \Rightarrow Happy(Claudia)$$
- $$Happy(Claudia) \Rightarrow \\ [\exists F \forall P : (friend(Claudia, F) \wedge post(Claudia, P)) \Rightarrow shares(F, P)]$$
- $\forall X : [location(X) \wedge at(You, X)] \Rightarrow watching(BigBro, You, X)$
- $\forall T : [time(T) \wedge works(Joe, T)] \Rightarrow \sim later(T, 5pm)$

Example for comparison of Propositional Logic and FOL

Primitives in Propositional Logic. Ski-race example.[1]

Objects

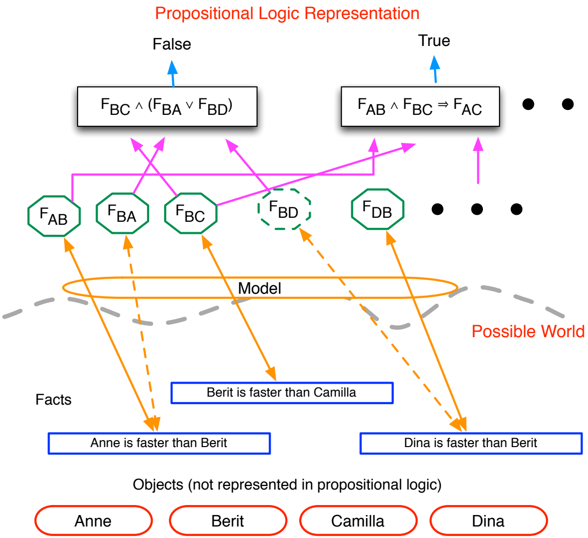
- Anne, Berit , Camilla, Dina
- These are not actually represented in propositional logic.
- Only True-or-False facts **about** them are represented.
- Objects alone do not have a truth value, whereas all primitives in propositional logic do.

Propositional Symbols

- F_{AB}:(Anne is faster than Berit) ; F_{BA} : (Berit is faster than Anne), .. F_{AC}
- These have truth values and are atomic.
- Their logical combinations into sentences (representing specific facts or general rules) also have truth values.

[1]Thanks to Keith Downing for this example

Example- Ski-race in Propositional Logic



Example- Ski Race in FOL

Objects

- Anne, Berit, Camilla, Dina
- These are now represented in the logic, even though they still have no truth value.

Functions

- best10K(person) $\rightarrow$ time. Mapping from athlete to their best 10K time.
- rank(person) $\rightarrow$ integer. Mapping from athlete to their seeding in the competition.
- start(person) $\rightarrow$ integer. Mapping from athlete to start order in the race, where slowest start first.
- These have no truth value and map one primitive object (person) to another (number).

Example- Ski Race in FOL

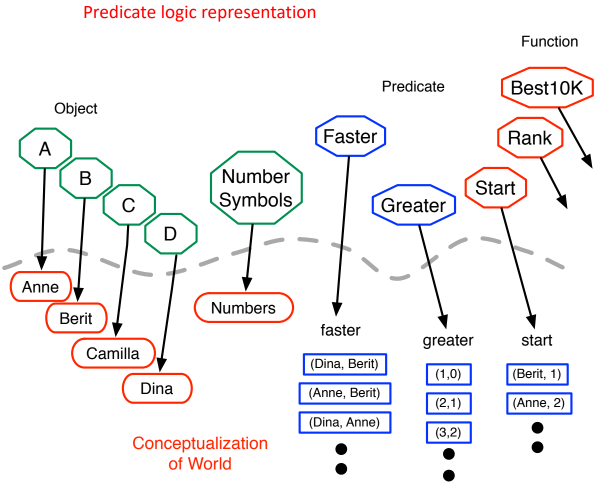
Relations

- greater(X,Y) $\rightarrow$ {True, False }. Is number X greater than number Y?
- faster(X,Y) $\rightarrow$ {True, False }. Is athlete X faster than athlete Y?
- These always have a truth value.
- These are often viewed as explicit lists of tuples, one list for each TRUE relation. So in one possible world, faster is represented by:

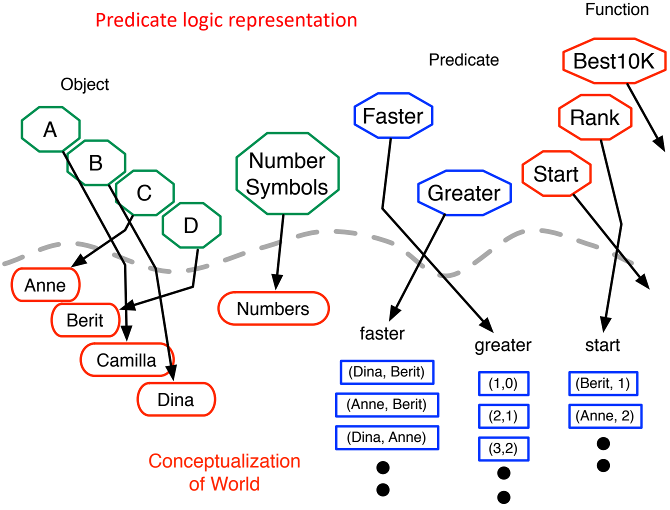
  { (anne, berit), (anne, camilla), (dina, anne), (dina, camilla), (camilla, berit), (dina, berit) }

Interpretations in First-Order Logic

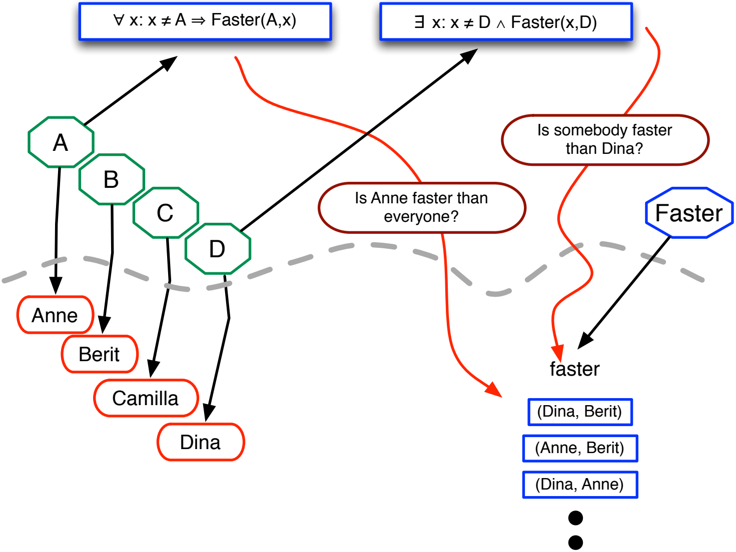
Interpretation = Mapping from constant, function and predicate symbols of the representation to the conceptualization.



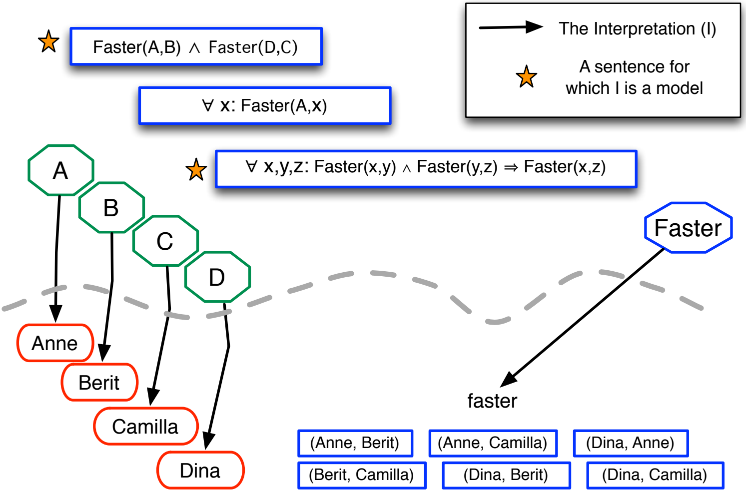
Another Legal Interpretation



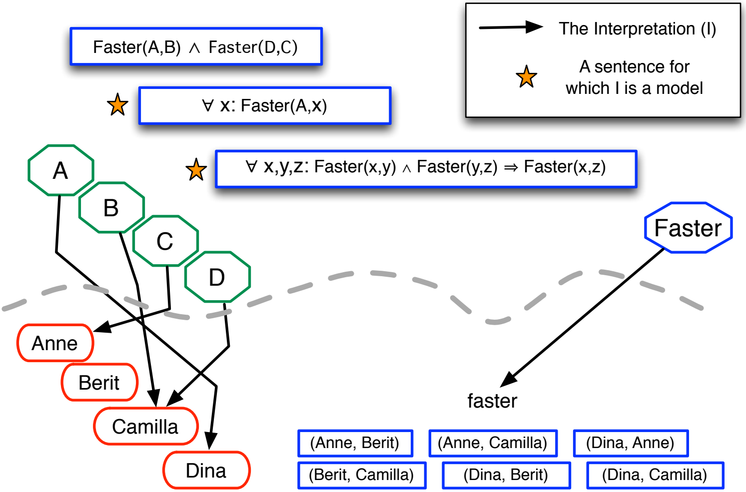
Evaluating Sentences with Quantified Variables



Example - Ski Models, Interpretation 1



Example - Ski Models for Interpretation 2



Common mistakes with Quantifiers

- Typically, $\implies$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Let us represent this sentence: "Everybody at NTNU is smart"

Common mistakes with Quantifiers

- Typically, $\implies$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Let us represent this sentence: "Everybody at NTNU is smart"

$$\forall x \;\; At(x, NTNU) \wedge Smart(x)$$

– Correct?

Common mistakes with Quantifiers

- Typically, $\implies$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Let us represent this sentence: "Everybody at NTNU is smart"

$$\forall x \;\; At(x, NTNU) \wedge Smart(x)$$

– Correct?

- No, it means "Everyone is at NTNU and everyone is smart"

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\implies$ as the main connective with $\exists$:
- Let us represent this sentence:"There is a smart person at NTNU"

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\implies$ as the main connective with $\exists$:
- Let us represent this sentence:"There is a smart person at NTNU"

$$\exists x \;\; At(x, NTNU) \implies Smart(x)$$

– Correct?

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\implies$ as the main connective with $\exists$:
- Let us represent this sentence:"There is a smart person at NTNU"

$$\exists x \;\; At(x, NTNU) \implies Smart(x)$$

– Correct?

- No, because it becomes true if there is anyone who is not at NTNU!

Connections between $\forall$ and $\exists$

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation.
- Negation rules/laws:
  - $\neg \exists \equiv \forall \neg$

Connections between ∀ and ∃

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation.
- Negation rules/laws:
    - $\neg\,\exists \equiv \forall\,\neg$
    - $\neg\forall \equiv \exists\neg$

Connections between ∀ and ∃

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation.
- Negation rules/laws:
  - $\neg \exists \equiv \forall \neg$
  - $\neg\forall \equiv \exists\neg$
  - $\neg\forall\neg \equiv \exists$

Connections between $\forall$ and $\exists$

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation.
- Negation rules/laws:
  - $\neg \exists \equiv \forall \neg$
  - $\neg\forall \equiv \exists\neg$
  - $\neg\forall\neg \equiv \exists$
  - $\neg\exists\neg \equiv \forall$

Connections between $\forall$ and $\exists$

- All statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation.
- Negation rules/laws:
  - $\neg\exists \equiv \forall\neg$
  - $\neg\forall \equiv \exists\neg$
  - $\neg\forall\neg \equiv \exists$
  - $\neg\exists\neg \equiv \forall$

Connections between ∀ and ∃

- Remember De Morgan's Rules
  - $\mathrm{P} \wedge \mathrm{Q} \equiv (\neg(\neg P \vee \neg \mathrm{Q}))$
  - $\mathrm{P} \vee \mathrm{Q} \equiv (\neg(\neg P \wedge \neg \mathrm{Q}))$
  - $\neg(P \wedge \mathrm{Q}) \equiv \neg P \vee \neg \mathrm{Q}$
  - $\neg(P \vee \mathrm{Q}) \equiv \neg P \wedge \neg \mathrm{Q}$

- Generalized De Morgan's rules
  - $\forall \mathrm{x}\ \mathrm{P(x)} \equiv \neg\exists x(\neg P(x))$
  - $\exists \mathrm{x}\ \mathrm{P(x)} \equiv \neg\forall x(\neg P(x))$
  - $\neg\forall \mathrm{x}\ \mathrm{P(x)} \equiv \exists x(\neg P(x))$
  - $\neg\exists \mathrm{x}\ \mathrm{P(x)} \equiv \forall x(\neg P(x))$

Multiple Quantifiers

More complex sentences can be formulated by multiple variables and by nesting quantifiers

- "For all x, there exists a y such that P(x,y)"
  - $\forall x \, \exists y P(x, y)$
  - Example: $\forall x \, \exists y (x + y = 0)$

Multiple Quantifiers

More complex sentences can be formulated by multiple variables and by nesting quantifiers

- "For all x, there exists a y such that P(x,y)"
  - $\forall x \, \exists y P(x, y)$
  - Example: $\forall x \, \exists y (x + y = 0)$
- "There exists an x such that for all y P(x,y) is true"
  - $\exists x \, \forall y P(x, y)$
  - Example: $\exists x \, \forall y (x * y = 0)$

Order of Multiple Quantifiers

- Reversing the order of the same type of consecutive quantifiers does not change the truth value of a sentence

  $\forall x \, \forall y P(x, y)$ is the same as $\forall y \, \forall x P(x, y)$

  - $\forall x, \forall y \; Parent(x, y) \implies Child(y, x)$ can be written as $\forall y \, \forall x Parent(x, y) \implies Child(y, x)$

Order of Multiple Quantifiers

Order of consecutive quantifiers of opposite type cannot be changed.

- $\forall x \, \exists y \; Loves(x, y)$
    - Everyone in the world loves someone.
    - With paranteheses: $\forall x \, (\exists y \; Loves(x, y)\,)$
    - y is inside the scope of x

Order of Multiple Quantifiers

Order of consecutive quantifiers of opposite type cannot be changed.

- $\forall x\, \exists y\, Loves(x, y)$
  - Everyone in the world loves someone.
  - With paranteheses: $\forall x\, (\exists y\, Loves(x, y)\,)$
  - y is inside the scope of x
- $\exists x\, \forall y\, Loves(x, y)$
  - There is a person who loves everyone n the world.
  - The same person x loves everybody.
  - y is inside the scope of x

Order of Multiple Quantifiers

Order of consecutive quantifiers of opposite type cannot be changed.

- $\forall x \, \exists y \; Loves(x, y)$
  - Everyone in the world loves someone.
  - With paranteheses: $\forall x \, (\exists y \; Loves(x, y)\,)$
  - y is inside the scope of x
- $\exists x \, \forall y \; Loves(x, y)$
  - There is a person who loves everyone n the world.
  - The same person x loves everybody.
  - y is inside the scope of x
- $\exists y \, \forall x Loves(x, y)$
  - There is someone whom everybody likes
  - Everybody likes the same y.
  - x is inside the scope of y.

Order of Multiple Quantifiers

Order of consecutive quantifiers of opposite type cannot be changed.

- $\forall x \, \exists y \; Loves(x, y)$
  - Everyone in the world loves someone.
  - With paranteheses: $\forall x \, (\exists y \; Loves(x, y) \;)$
  - y is inside the scope of x
- $\exists x \, \forall y \; Loves(x, y)$
  - There is a person who loves everyone n the world.
  - The same person x loves everybody.
  - y is inside the scope of x
- $\exists y \, \forall x Loves(x, y)$
  - There is someone whom everybody likes
  - Everybody likes the same y.
  - x is inside the scope of y.
- What about "There is a puppy that likes every woman."

Order of Multiple Quantifiers

Order of consecutive quantifiers of opposite type cannot be changed.

- $\forall x \, \exists y \; Loves(x, y)$
  - Everyone in the world loves someone.
  - With paranteheses: $\forall x \, (\exists y \; Loves(x, y) \;)$
  - y is inside the scope of x
- $\exists x \, \forall y \; Loves(x, y)$
  - There is a person who loves everyone n the world.
  - The same person x loves everybody.
  - y is inside the scope of x
- $\exists y \, \forall x Loves(x, y)$
  - There is someone whom everybody likes
  - Everybody likes the same y.
  - x is inside the scope of y.
- What about "There is a puppy that likes every woman."
  $\exists p Puppy(p) \wedge (\forall w Woman(w) \implies Loves(p, w))$

Negating multiple Quantifiers

- Recall negation rules for single quantifiers:

  $\neg\forall x\, P(x) = \exists x \neg P(x)$
  $\neg\exists\, x P(x) = \forall x \neg P(x)$

- You change the quantifier(s), and negate what it's quantifying:

Negating multiple Quantifiers

- Recall negation rules for single quantifiers:

  $\neg \forall x\, P(x) = \exists x \neg P(x)$
  $\neg \exists\, x P(x) = \forall x \neg P(x)$

- You change the quantifier(s), and negate what it's quantifying:

  $\neg(\forall x\, \exists y\, P(x, y)) \equiv \exists x\, \neg\, \exists y P(x, y) \equiv \exists x\, \forall y\, \neg P(x, y)$

Higher-Order Logic

- FOL is called first-order because it allows quantifiers to range over objects but not properties, relations, or functions applied to those objects.
- Second-order logic allows quantifiers to range over predicates and functions as well:

  - $\forall x \forall y[(x = y) \leftrightarrow (\forall P\, P(x) \leftrightarrow P(y))]$
  Means that two objects are equal if and only if they have exactly the same properties.

Higher-Order Logic

- FOL is called first-order because it allows quantifiers to range over objects but not properties, relations, or functions applied to those objects.
- Second-order logic allows quantifiers to range over predicates and functions as well:

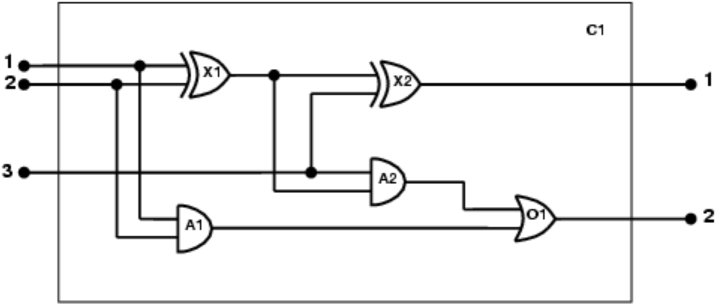
- $\forall x \, \forall y [(x = y) \leftrightarrow (\forall P \, P(x) \leftrightarrow P(y))]$
  Means that two objects are equal if and only if they have exactly the same properties.
- $\forall F \, \forall G [(F = G) \leftrightarrow (\forall x \, F(x) = G(x))]$
  Means that two functions are equal if and only if they have the same value for all possible arguments.

Knowledge engineering in FOL

1. Identify the problem/task you want to solve
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



Knowledge engineering in FOL

1. Identify the task

– Does the circuit actually add properly? (circuit verification)

2. Assemble the relevant knowledge

– Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)

– Irrelevant attributes: size, shape, colour, cost of gates

Knowledge engineering in FOL

Decide on a vocabulary. Gate, Circuit, Terminal, Out, Connected,. Type....

Alternatives:

Type(X1) $=$ XOR
Type(X1, XOR)
XOR(X1)

Knowledge engineering in FOL

Encode general knowledge of the domain.

- $\forall$ t1, t2 Connected (t1, t2) $\implies$ Signal(t1)=Signal(t2)
- $\forall$ t Signal(t)=1 $\vee$ Signal(t) = 0
- $1 \neq 0$
- $\forall$t1,t2 Connected(t1, t2) $\implies$ Connected(t2, t1)
- $\forall$g Type(g)=OR $\implies$ Signal(Out(1,g))=1 $\iff$ $\exists$ n Signal(In(n,g))=1
- $\forall$g Type(g)=AND $\implies$ Signal(Out(1,g))=0 $\iff$ $\exists$n Signal(In(n,g))=0
- $\forall$g Type(g) = XOR $\implies$ Signal(Out(1,g)) = 1 $\iff$ Signal(In(1,g)) $\neq$ Signal(In(2,g))
- $\forall$ g Type(g) = NOT $\implies$ Signal(Out(1,g)) $\neq$ Signal(In(1,g))

Knowledge engineering in FOL

Encode the specific problem instance

- Type(X1) = XOR Type(X2) = XOR
- Type(A1) = AND Type(A2) = AND
- Type(O1) = OR

- Connected(Out(1,X1),In(1,X2))
- Connected(Out(1,X1),In(2,A2))
- Connected(Out(1,A2),In(1,O1))
- Connected(Out(1,A1),In(2,O1))
- Connected(Out(1,X2),Out(1,C1))
- Connected(Out(1,O1),Out(2,C1))

- Connected(In(1,C1),In(1,X1))
- Connected(In(1,C1),In(1,A1))
- Connected(In(2,C1),In(2,X1))
- Connected(In(2,C1),In(2,A1))
- Connected(In(3,C1),In(2,X2))
- Connected(In(3,C1),In(1,A2))

Knowledge engineering in FOL

Pose queries to the inference procedure.

What are the possible sets of values of all the terminals for the adder circuit?

$$\begin{aligned}\exists\ i1,i2,i3,o1,o2\ & \mathrm{Signal}(\mathrm{In}(1,C1)) = i1 \land \mathrm{Signal}(\mathrm{In}(2,C1)) = i2 \land \\ & \mathrm{Signal}(\mathrm{In}(3,C1)) = i3 \land \mathrm{Signal}(\mathrm{Out}(1,C1)) = o1 \land \\ & \mathrm{Signal}(\mathrm{Out}(2,C1)) = o2\end{aligned}$$

Debug the KB

- Perturb the KB to see what kinds of erroneous. Behaviors emerge.

Knowledge Engineering for Euclidean Geometry

Can we give a First-Order Logic Theorem Prover enough Knowledge to do Euclidean Geometry?

The Knowledge Engineering Process

1. Identify questions to be asked of the Knowledge Base.
2. Knowledge Acquisition - gather the proper kwg *
3. Ontology Design - determine constants, predicates and functions
4. Encode general kwg about the domain – e.g. axioms that define the main concepts.
5. Encode an instance in the problem domain.**
6. Pose queries and get answers.
7. Debug

* Kwg is sometimes used as an abbreviation for Knowledge

** Steps 4 and 5 will often reveal faults in the ontology and encoded domain kwg, thus forcing a return to step 3. Many such returns were necessary during the making of these slides !

Step 1: Identify Relevant Knowledge-Base Queries

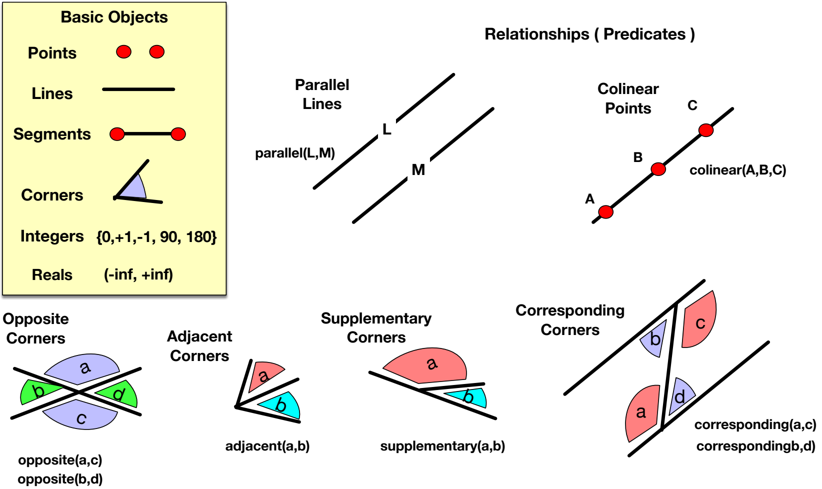
- Are angles ABC and ABD adjacent? Are they supplementary?
- Are angles ABC and XYZ congruent (same degrees, radians)?
- Do angles ABC and DBE refer to the same location (a.k.a. corner)?
- Are points P and Q on the same side of line L?
- Are lines L and M parallel? Perpendicular?

  ... and eventually....
- Are triangles ABC and ADE congruent? Similar?
- Are the ratios of the lengths of two pairs of segments equal?

Step 2: Identify Relevant Knowledge

- Two lines can intersect at at most one point.
- Intersecting lines form two pairs of opposite angles.
- Opposite angles are congruent.
- Parallel lines form corresponding angles.
- Corresponding angles are congruent.
- Many identical angles can be formed when two intersecting lines contain many points.
- Adjacent angles sum to a larger angle with the same corner point.
- Supplementary and Complementary angles are important.
- For any point (P) and line (L), P is either on L or on exactly one of L's two sides.
- The side of line on which a point lies is important for determining angles and their adjacencies and equalities.
- Angles refer to unique locations (*corners*), but it's a many-to-one mapping. This could be tricky to represent!

Step 3: Design the Ontology

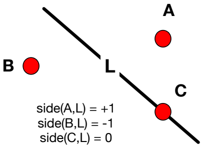


*** These are predicates, since they map to a truth value: {True, False}**

Step 3: Design the Ontology (continued)

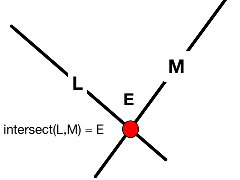
Functions

side(point,line) = {0,+1,-1}

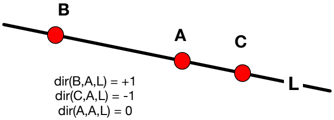


*** These are functions, since every set of inputs maps to a single object.**

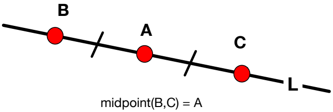
intersect(line,line) = point



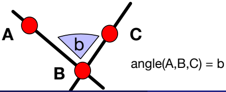
dir(point,point,line) = {0,+1,-1}



midpoint(point,point) = point



angle(point,point,point) = corner



measure(corner) = (-180, 180]

measure(b) = 85º

Step 4: Encode General Domain Knowledge

- Defining *on*

$$\forall L, P : [line(L) \land pt(P) \land side(P, L) = 0] \Leftrightarrow on(P, L)$$

- Defining *colinear*

$$\forall A, B, C, L : pt(A) \land pt(B) \land pt(C) \land line(L) \Rightarrow$$
$$\{(on(A, L) \land on(B, L) \land on(C, L)) \Leftrightarrow colinear(A, B, C)\}$$

- Defining corresponding corners (Note: angle(A,B,C) $\rightarrow$ corner)

$$\forall L, M, N, A, B, C, D : line(L) \land line(M) \land line(N) \land$$
$$pt(A) \land pt(B) \land pt(C) \land pt(D) \Rightarrow$$
$$\{[parallel(L, M) \land intersect(L, N) = B \land intersect(M, N) =$$
$$C \land on(A, L) \land on(D, M) \land side(A, N) \neq side(D, N)]$$
$$\Leftrightarrow corresponding(angle(A, B, C), angle(B, C, D))\}$$

- Equivalence of corresponding corners

$$\forall F, G : corner(F) \land corner(G) \Rightarrow \{corresponding(F, G) \Leftrightarrow (measure(F) = measure(G))\}$$

- Two lines can intersect at at most one point.

$$\forall L, M : [line(L) \land line(M) \land intersect(L, M, A) \land intersect(L, M, B)] \Rightarrow (A = B)$$

- Symmetry of angle function

$$\forall A, B, C : pt(A) \land pt(B) \land pt(C) \Rightarrow angle(A, B, C) = angle(C, B, A)$$

Step 4: Encode General Domain Knowledge (cont.)

- Defining separation

$$\forall L, A, B : line(L) \wedge pt(A) \wedge pt(B) \Rightarrow$$
$$\{(side(A, L) \neq side(B, L) \wedge side(A, L) \neq 0 \wedge side(B, L) \neq 0) \Leftrightarrow$$
$$separates(L, A, B)\}$$

- Defining opposite corners

$$\forall L, M, E : line(L) \wedge line(M) \wedge pt(E) \wedge intersect(L, M) = E \Rightarrow$$
$$\{(on(A, L) \wedge on(C, L) \wedge separates(M, A, C) \wedge on(B, M) \wedge on(D, M) \wedge$$
$$separates(L, B, D)) \Leftrightarrow$$
$$(opposite(angle(A, E, B), angle(D, E, C)) \wedge$$
$$opposite(angle(B, E, C), angle(A, E, D)))\}$$

- Equivalence of opposite corners

$$\forall F, G : corner(F) \wedge corner(G) \Rightarrow \{opposite(F, G) \Leftrightarrow (measure(F) = measure(G))\}$$

- Many angles can map to the same corner

$$\forall L, M, A, B, C, D : line(L) \wedge line(M) \wedge pt(A) \wedge pt(B) \wedge pt(C) \wedge pt(D) \Rightarrow$$
$$\{(intersect(L, M) =$$
$$D \wedge on(A, L) \wedge on(B, M) \wedge on(C, M) \wedge side(B, L) = side(C, L)) \Rightarrow$$
$$angle(A, D, B) = angle(A, D, C)\}$$

Step 4: Encode General Domain Knowledge (cont.)

Defining adjacent angles/corners*

$$\forall L, M, N \in lines, A, B, C, D \in points :$$
$$[D = intersect(L, M) \wedge D = intersect(M, N) \wedge on(A, M) \wedge on(B, L) \wedge on(C, N) \wedge side(C, M) \neq side(B, M)]$$
$$\Rightarrow adjacent(angle(B, D, A), angle(C, D, A))$$

Summing adjacent angles

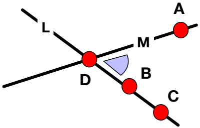
$$\forall A, B, C, D \in points : adjacent(angle(B, D, A), angle(C, D, A)) \rightarrow measure(angle(B, D, A)) + measure(angle(B, D, A)) = measure(angle(B, D, C))$$

Supplementary Adjacent Angles

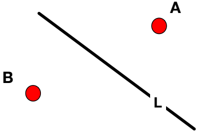
$$\forall A, B, C, D \in points : adjacent(angle(B, D, A), angle(C, D, A)) \wedge colinear(B, D, C) \rightarrow measure(angle(B, D, C)) = 180^\circ$$

*A perfect definition of adjacency is a bit more complicated.

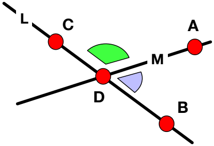
General Knowledge in Diagrams



angle(A,D,B) = angle(A,D,C)



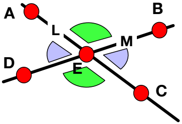
separates(L,A,B)



supplementary(angle(B,D,A), angle(C,D,A))

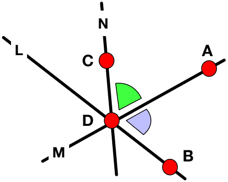
$\longleftrightarrow$

measure(angle(B,D,A)) + measure(angle(C,D,A) = 180°



opposite(angle(A,E,B), angle(D,E,C))

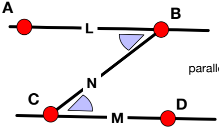
opposite(angle(A,E,D), angle(B,E,C))



adjacent(angle(B,D,A), angle(C,D,A))

$\longrightarrow$

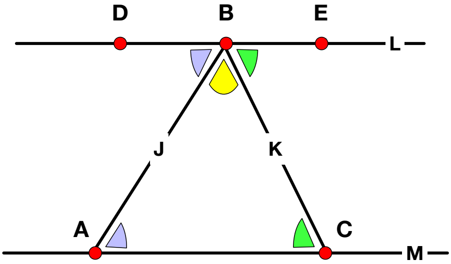
measure(angle(B,D,A)) + measure(angle(C,D,A) =
measure(angle(B,D,C))



parallel(L,M)

corresponding(angle(A,B,C), angle(D,C,B))

Step 5: Encode the Current Problem



Premises

line(J) line(K)
line(L) line(M)

parallel(L,M)

pt(A) pt(B) pt(C)
pt(D) pt(E)

B = intersect(J,L)
B = intersect(K,L)
A = intersect(J,M)
C = intersect(K,M)

side(D,J) = +1
side(D,K) = -1
side(D,L) = 0
side(D,M) = +1
side(A,J) = 0
side(A,M) = 0
side(A,K) = -1
side(A,L) = -1
:
:

Prove: angle(B,A,C) + angle(A,C,B) + angle(A,B,C) = 180º

Step 6: Pose Queries and Get Answers

Query

- m(angle(B,A,C))+ m(angle(A,C,B)) + m(angle(A,B,C)) = ??
  where m() = measure()

Answer (Hopefully)

- 180°
- The chain of deductions from the premises and the domain knowledge to 180°

Chain of Reasoning

*m() = measure()

- $m^*(\angle ABD) + m(\angle ABE) = 180^\circ$ ::: Adjacent, supplementary angles
- $m(\angle ABE) = m(\angle ABC) + m(\angle CBE)$ ::: Adjacent angles
- $m(\angle ABD) + m(\angle ABC) + m(\angle CBE) = 180^\circ$ ::: Simple substitution
- $corresponding(\angle ABD, \angle BAC)$ ::: Def. of corresponding angles
- $corresponding(\angle CBE, \angle ACB)$ ::: Def. of corresponding angles
- $m(\angle ABD) = m(\angle BAC)$ ::: Equality of corresponding angles
- $m(\angle CBE) = m(\angle ACB)$ ::: Equality of corresponding angles
- $m(\angle BAC) + m(\angle ACB) + m(\angle ABC) = 180^\circ$ ::: Simple substitution

This is a general proof that the 3 angles of a triangle always sum to 180°.

Step 7: Debug

- The ontology and general knowledge base require frequent updates (e.g. Tore Amble's BusTuc)
- Early on, many queries force these updates.
- With each change, try to anticipate future queries and thus make the KB as general and flexible as possible.
- *Representation without reasoning is an idle exercise*...Ken Forbus (well-known AI researcher)

  $\Rightarrow$ Building a truly general-purpose representation is a HARD job !!

Summary

Let us summarize together!