15-780: Graduate AI

Lecture 3. FOL proofs

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman

Admin

HW1

- *Out today*
- *Due Tue, Feb. 1 (two weeks)*
  - *hand in hardcopy at beginning of class*
- *Covers propositional and FOL*
- *Don't leave it to the last minute!*

Collaboration policy

- *OK to discuss general strategies*
- *What you hand in must be your own work*
  - *written with no access to notes from joint meetings, websites, etc.*
- *You must acknowledge all significant discussions, relevant websites, etc., on your HW*

Late policy

- *5 late days to split across all HWs*
  - *these account for conference travel, holidays, illness, or any other reasons*
- *After late days, out of 70th %ile for next 24 hrs, 40th %ile for next 24, no credit thereafter (but still must turn in)*
- *Day = 24 hrs or part thereof, HWs due at 10:30AM*

Office hours

- *My office hours this week (usually 12–1 Thu) are canceled*
- *Email if you need to discuss something with me*

Review

NP

- *Decision problems*
- *Reductions: A reduces to B means B at least as hard as A*
  - *Ex: k-coloring to SAT, SAT to CNF-SAT*
  - *Sometimes a practical tool*
- *NP = reduces to SAT*
- *NP-complete = both directions to SAT*
- $P \stackrel{?}{=} NP$

Propositional logic

- *Proof trees, proof by contradiction*
- *Inference rules (e.g., resolution)*
- *Soundness, completeness*
- *First nontrivial SAT algorithm*
- *Horn clauses, MAXSAT, nonmonotonic logic*

FOL

- *Models*
  - *objects, function tables, predicate tables*
- *Compositional semantics*
  - *object constants, functions, predicates*
  - *terms, atoms, literals, sentences*
  - *quantifiers, variables, free/bound, variable assignments*

Proofs in FOL

- *Skolemization, CNF*
- *Universal instantiation*
- *Substitution lists, unification*
- *MGU (unique up to renaming, exist efficient algorithms to find it)*

Proofs in FOL

Quiz

- *Can we unify*

  $knows(John, x) \quad knows(x, Mary)$

- *What about*

  $knows(John, x) \quad knows(y, Mary)$

Quiz

- *Can we unify*

  $knows(John, x) \quad knows(x, Mary)$

  No!

- *What about*

  $knows(John, x) \quad knows(y, Mary)$

  $x \rightarrow Mary, y \rightarrow John$

Standardize apart

- *But knows(x, Mary) is logically equivalent to knows(y, Mary)!*
- *Moral: standardize apart before unifying*

First-order resolution

- *Given clauses* $(\alpha \vee c)$, $(\neg d \vee \beta)$, *and a substitution list L unifying c and d*
- *Conclude* $(\alpha \vee \beta) : L$
- *In fact, only ever need L to be MGU of c, d*

Example

$rains \wedge outside(x) \Rightarrow wet(x)$

$wet(x) \Rightarrow rusty(x) \vee rustproof(x)$

$robot(x) \Rightarrow \neg rustproof(x)$

$rains$

$guidebot(Robby)$

$guidebot(x) \Rightarrow robot(x) \wedge outside(x)$

$rains \wedge outside(x) \Rightarrow wet(x)$ $\rightsquigarrow$ $\neg rains \vee \neg outside(x) \vee wet(x)$ 6

$wet(x) \Rightarrow rusty(x) \vee rustproof(x)$ $\neg wet(x) \vee rusty(x) \vee rustproof(x)$ 10

$robot(x) \Rightarrow \neg rustproof(x)$ $\neg robot(x) \vee \neg rustproof(x)$ 4

$rains$ $rains$ 7

$guidebot(Robby)$ $guidebot(R)$ 1

$guidebot(x) \Rightarrow robot(x) \wedge outside(x)$ $\neg guidebot(x) \vee robot(x)$ 2

$\neg guidebot(x) \vee outside(x)$ 3

$\neg(\exists x.\ rusty(x)) \rightsquigarrow \neg rusty(x)$ 14

$1,2 \models robot(R)$ 5 $(x \to R)$

$1,3 \models outside(R)$ 9 $(x \to R)$

$4,5 \models \neg rustproof(R)$ 13

$6,7 \models \neg outside(x) \vee wet(x)$ 8

$8,9 \models wet(R)$ 11

$10,11 \models rusty(R) \vee rustproof(R)$ 12

$12,13 \models rusty(R)$ 15

$14,15 \models F$

First-order factoring

- *When removing redundant literals, we have the option of unifying them first*
- *Given clause ($a \vee b \vee \theta$), substitution L*
- *If a : L and b : L are syntactically identical*
- *Then we can conclude ($a \vee \theta$) : L*
- *Again L = MGU is enough*

Completeness

Jacques Herbrand
1908–1931

- *First-order resolution (w/ FO factoring) is sound and complete for FOL w/o equality (famous theorem due to Herbrand and Robinson)*
- *Unlike propositional case, may be infinitely many possible conclusions*
- *So, FO entailment is **semidecidable** (entailed statements are **recursively enumerable**)*

Algorithm for FOL

- *Put KB $\wedge$ $\neg$S in CNF*
- *Pick an application of resolution or factoring (using MGU) by some fair rule*
  - *standardize apart premises*
- *Add consequence to KB*
- *Repeat*

Variations

Equality

- ***Paramodulation** is sound and complete for FOL+equality (see RN)*
- *Or, resolution + factoring +* ***axiom schema***

$$\forall P.\ \forall x,y.\ (x=y) \Rightarrow P(x) \Leftrightarrow P(y)$$

$$\forall P.\ \forall x,y,z.\ (x=y) \Rightarrow P(x,z) \Leftrightarrow P(y,z)$$

$$\Rightarrow P(z,x) \Leftrightarrow P(z,y)$$

Restricted semantics

- *Only check one finite, propositional KB*
  - *NP-complete much better than RE*
- ***Unique names**: objects with different names are different (John ≠ Mary)*
- ***Domain closure**: objects without names given in KB don't exist*
- ***Known functions**: only have to infer predicates*

Uncertainty

- *Same trick as before: many independent random choices by Nature, logical rules for their consequences*
- *Two new difficulties*
  - *ensuring satisfiability (not new, harder)*
  - *describing set of random choices*

Markov logic

- *Assume unique names, domain closure, known fns: only have to infer propositions*
- *Each FO statement now has a known set of ground instances*
  - *e.g., loves(x,y) ⇒ happy(x) has n^2 instances if there are n people*
- *One random choice per rule instance: enforce w/p p (KBs that violate the rule are (1–p) times less likely)*

Richardson & Domingos

Independent Choice Logic

David Poole

- *Generalizes Bayes nets, Markov logic, Prolog programs—incomparable to FOL*
- *Use only **acyclic** KBs (always feasible), **minimal model** (cf. nonmonotonicity)*
- *Assume all **syntactically distinct** ground terms are distinct (so we know what objects are in our model—perhaps infinitely many)*
- *Label some predicates as **choices**: values selected independently for each grounding*

Inference under uncertainty

- *Wide open topic: lots of recent work!*
- *We'll cover only the special case of propositional inference under uncertainty*
- *The extension to FO is left as an exercise for the listener*

Second order logic

- *SOL adds quantification over predicates*
- *E.g., principle of mathematical induction:*
  - $\forall P.\ P(0) \wedge (\forall x.\ P(x) \Rightarrow P(S(x)))$
    $\Rightarrow \forall x.\ P(x)$
- *There is no sound and complete inference procedure for SOL (Gödel's famous incompleteness theorem)*

Others

- *Temporal and modal logics ("P(x) will be true at some time in the future," "John believes P(x)")*
- *Nonmonotonic FOL*
- *First-class functions (lambda operator, application)*
- …

Who? What? Where?

Wh-questions

- *We've shown how to answer a question like "is Socrates mortal?"*
- *What if we have a question whose answer is not just yes/no, like "who killed JR?" or "where is my robot?"*
- *Simplest approach: prove* $\exists x.\ killed(x, JR)$, *hope the proof is constructive*
  - *may not work even if constr. proof exists*

Answer literals

- *Instead of $\neg P(x)$, add $(\neg P(x) \vee answer(x))$*
  - *answer is a **new** predicate*
- *If there's a proof of $P(foo)$, can eliminate $\neg P(x)$ by resolution and unification, leaving $answer(x)$ **with x bound** to foo*

Example

1 $Kills(Jack, Cat) \vee Kills(Curiosity, Cat)$

2 $\neg Kills(Jack, Cat)$

3 $\neg Kills(x, Cat)$

$1,3 \vdash Kills(J, C)$ 4 $(x \to Curiosity)$

$4,2 \vdash F$

Example

1 Kills(Jack, Cat) ∨ Kills(Curiosity, Cat)

2 ¬Kills(Jack, Cat)

3 ¬Kills(x, Cat)

1,3 $(x \to J)$ ⊨ Kills(Curiosity, Cat) 4

3,4 $(x \to Cu)$ ⊨ F

Example

1 Kills(Jack, Cat) ∨ Kills(Curiosity, Cat)

2 ¬Kills(Jack, Cat)

3 ¬Kills(x, Cat) ∨ Answer(x)

1,3 ⊨ kills(J, C) ∨ answer(Cu) 4

2,4 ⊨ answer(Cu)

Instance Generation

Bounds on KB value

- *If we find a model M of KB, then KB is satisfiable*
- *If L is a substitution list, and if (KB: L) is unsatisfiable, then KB is unsatisfiable*
  - *e.g., $mortal(x) \rightarrow mortal(uncle(x))$*

Bounds on KB value

- *KB_0 = KB w/ each **syntactically distinct** atom replaced by a different 0-arg proposition*
  - $likes(x, kittens) \lor \neg likes(y, x) \rightarrow A \lor \neg B$
- *KB ground and KB_0 unsatisfiable $\Rightarrow$ KB unsatisfiable*

Propositionalizing

- *Let L be a **ground** substitution list*
- *Consider KB' = $(KB: L)_0$*
  - *KB' unsatisfiable ⇒ KB unsatisfiable*
  - *KB' is propositional*
- *Try to show contradiction by handing KB' to a SAT solver: if KB' unsatisfiable, done*
- *Which L?* $x \to foo \quad y \to foo \quad z \to foo, \ldots$

Example

$Kills(Jack, Cat) \vee Kills(Curiosity, Cat)$

$\neg Kills(Jack, Cat)$

$\neg Kills(x, Cat) \rightsquigarrow \neg Kills(foo, Cat)$

$A \vee B$

$\neg A$

$\neg C$

M'

$A = F$

$B = T$

$C = F$

M

$Kills(J, C) = F$

$Kills(Cu, C) = T$

$Kills(x, C) = F \quad x \neq (J, Cu)$

Lifting

- *Suppose KB' satisfiable by model M'*
- *Try to* ***lift*** *M' to a model M of KB*
  - *assign each atom in M the value of its corresponding proposition in M'*
  - *break ties by* ***specificity*** *where possible*
  - *break any further ties arbitrarily*

Example

kills(Jack, Cat) ∨ kills(Curiosity, Cat)

¬kills(Jack, Cat)

¬kills(x, Cat)

¬kills(Jack, Cat)
kills(Curiosity, Cat)
¬kills(Foo, Cat)

M'

Discordant pairs

- *Atoms kills(x, Cat), kills(Curiosity, Cat)*
  - *each **tight** for its clause in M'*
  - *assigned opposite values in M'*
  - *unify: MGU is x $\rightarrow$ Curiosity*
- *Such pairs of atoms are **discordant***
- *They suggest useful ways to instantiate*

Example

$Kills(Jack, Cat) \vee Kills(Curiosity, Cat)$

$\neg Kills(Jack, Cat)$

$\neg Kills(x, Cat)$

$\neg Kills(Curiosity, Cat)$

$A \vee B$

$\neg A$

$\neg B$

InstGen

- *Propositionalize KB→KB', run SAT solver*
- *If KB' unsatisfiable, done*
- *Else, get model M', lift to M*
- *If M satisfies KB, done*
- *Else, pick a discordant pair according to a fair rule; use to instantiate clauses of KB*
- *Repeat*

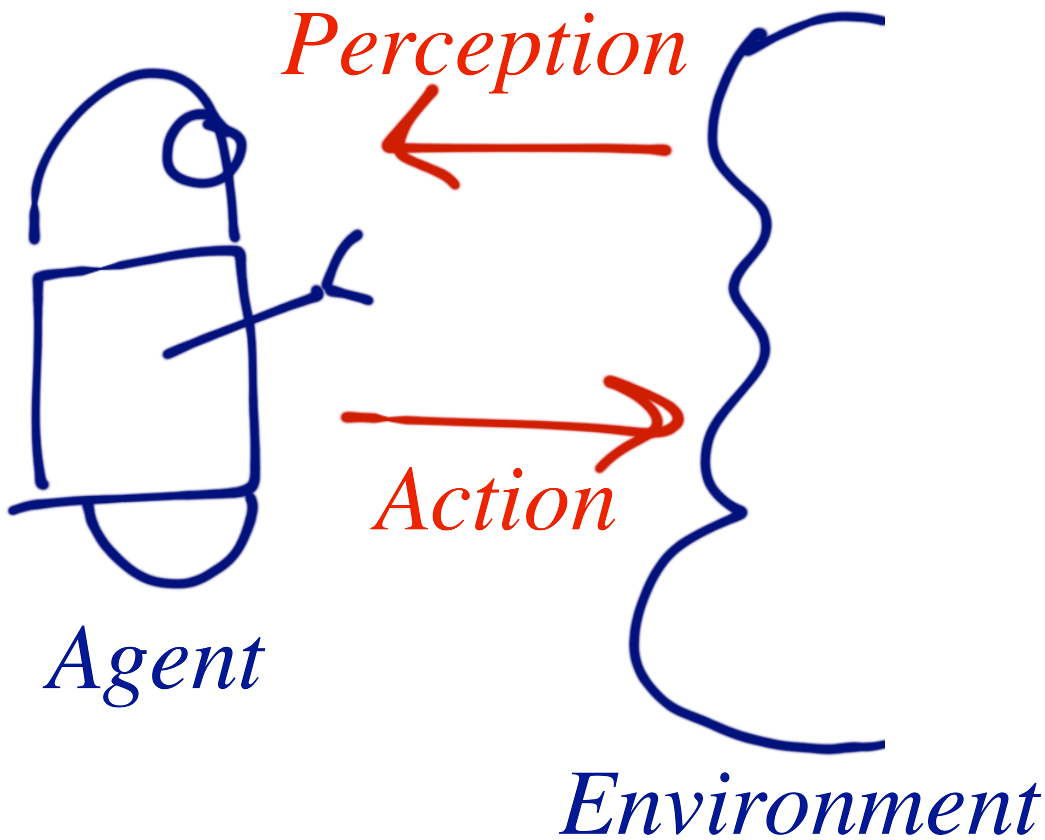
Soundness and completeness

for FOL w/o =

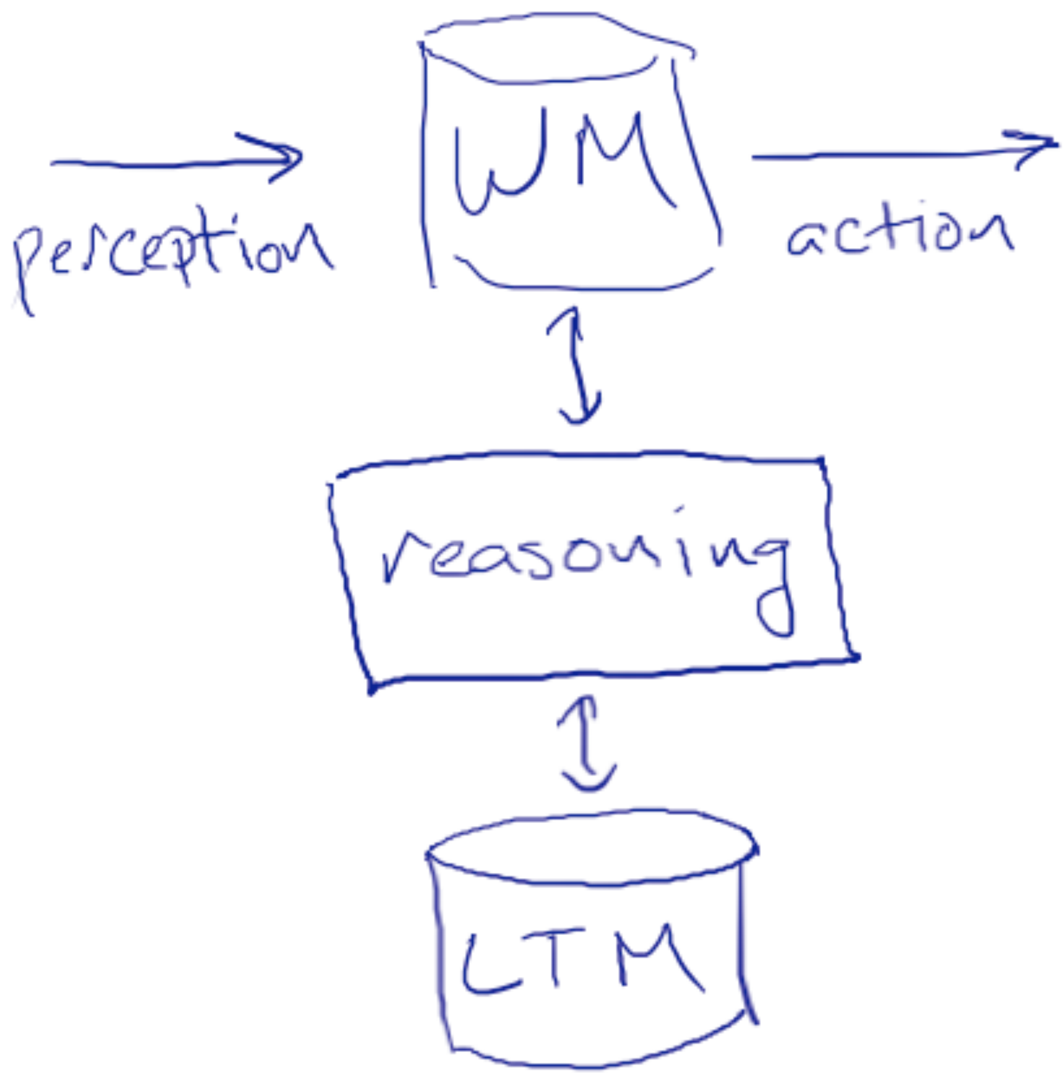
- *We've already argued soundness*
- *Completeness theorem: if KB is unsatisfiable but KB' is satisfiable, must exist a discordant pair wrt M' which generates a new instantiation of a clause from KB—and, $\exists$ a finite sequence of such instantiations will find an unsatisfiable propositional formula*

Agent Architectures

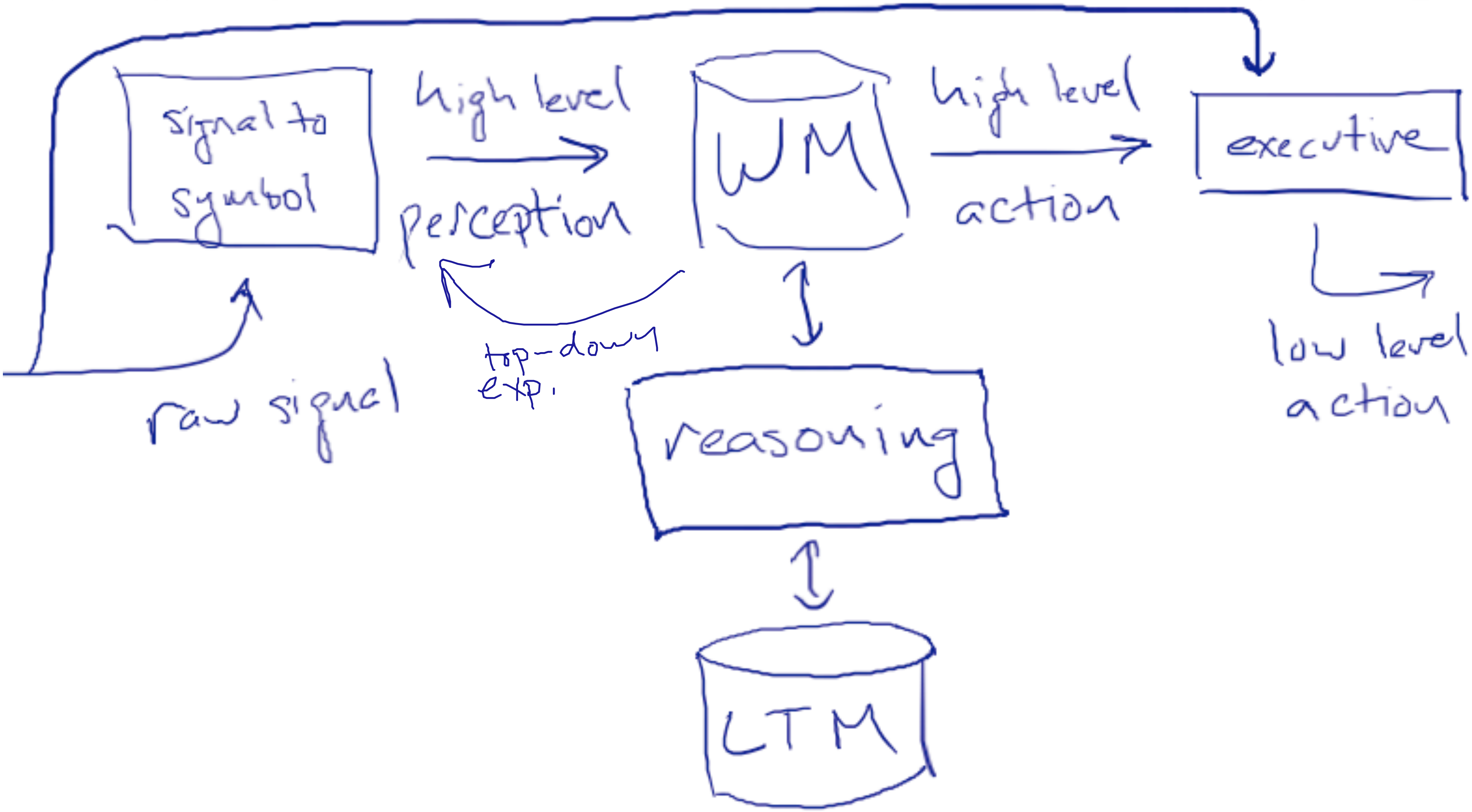
Situated agent



Inside the agent



Inside the agent



Knowledge Representation

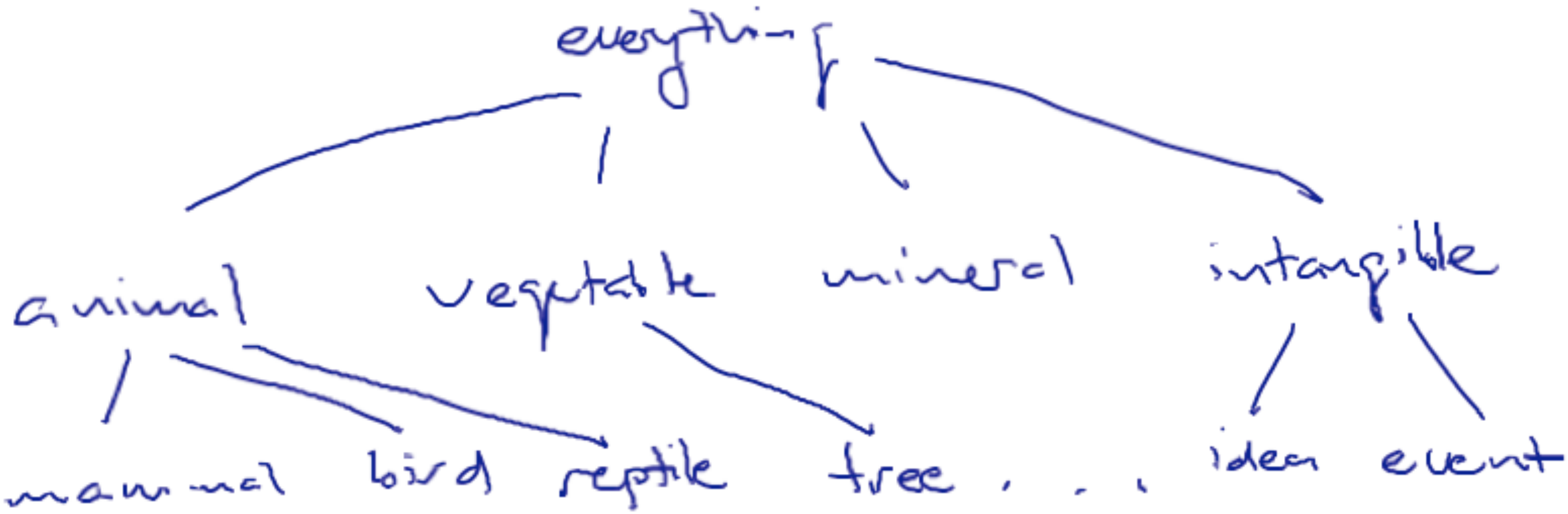
Knowledge Representation

- *is the process of*
  - *Identifing relevant objects, functions, and predicates*
  - *Encoding general background knowledge about domain (reusable)*
  - *Encoding specific problem instance*
- *Sometimes called knowledge engineering*

Common themes

- *RN identifies many common idioms and problems for knowledge representation*
- *Hierarchies, fluents, knowledge, belief, …*
- *We'll look at a couple*

Taxonomies



- *isa(Mammal, Animal)*
- *disjoint(Animal, Vegetable)*
- *partition({Animal, Vegetable, Mineral, Intangible}, Everything)*

Inheritance

- *Transitive:* $isa(x, y) \wedge isa(y, z) \Rightarrow isa(x, z)$
- *Attach properties anywhere in hierarchy*
  - $isa(Pigeon, Bird)$
  - $isa(x, Bird) \Rightarrow flies(x)$
  - $isa(x, Pigeon) \Rightarrow gray(x)$
- *So,* $isa(Tweety, Pigeon)$ *tells us Tweety is gray and flies*

Physical composition

- *partOf(Wean4625, WeanHall)*
- *partOf(water37, water3)*
- *Note distinction between* ***mass*** *and* ***count*** *nouns: any partOf a mass noun also isa that mass noun*

Fluents

- *Fluent = property that changes over time*
  - *at(Robot, Wean4623, 11AM)*
- *Actions change fluents*
- *Fluents chain together to form possible worlds*
- $at(x, p, t) \wedge adj(p, q) \Rightarrow poss(go(x, p, q), t) \wedge at(x, q, result(go(x, p, q), t))$

Frame problem

- *Suppose we execute an unrelated action (e.g., talk(Professor, FOL))*
- *Robot shouldn't move:*
  - *if at(Robot, Wean4623, t), want at(Robot, Wean4623, result(talk(Professor, FOL)))*
- *But we can't prove it without adding appropriate rules to KB!*

Frame problem

- *The **frame problem** is that it's a pain to list all of the things that don't change when we execute an action*
- *Naive solution: **frame axioms***
  - *for each fluent, list actions that can't change fluent*
  - *KB size: O(AF) for A actions, F fluents*

Frame problem

- *Better solution:* ***successor-state*** *axioms*
- *For each fluent, list actions that* ***can*** *change it (typically fewer): if go(x, p, q) is possible,*

  $at(x, q, result(a, t)) \Leftrightarrow$
  $a = go(x, p, q) \vee (at(x, q, t) \wedge a \neq go(x, q, z))$

- *Size O(AE+F) if each action has E effects*

Debugging KB

- *Sadly always necessary…*
  - *Severe bug: logical contradictions*
  - *Less severe: undesired conclusions*
  - *Least severe: missing conclusions*
- *First 2: trace back chain of reasoning until reason for failure is revealed*
- *Last: trace desired proof, find what's missing*

Examples

A simple data structure

- $(ABB) \equiv \text{cons}(A, \text{cons}(B, \text{cons}(B, \text{nil})))$
- $\text{input}(x) \Leftrightarrow r(x, \text{nil})$
- $r(\text{cons}(x, y), z) \Leftrightarrow r(y, \text{cons}(x, z))$
- $r(\text{nil}, x) \Leftrightarrow \text{output}(x)$

Caveat

- $\mathrm{input}(x) \Leftrightarrow r(x, \mathrm{nil})$
- $r(\mathrm{cons}(x, y), z) \Leftrightarrow r(y, \mathrm{cons}(x, z))$
- $r(\mathrm{nil}, x) \Leftrightarrow \mathrm{output}(x)$

A context-free grammar

- S := NP VP
- NP := D Adjs N
- VP := Advs V PPs | Advs V DO PPs | Advs V IO DO PPs
- PP := Prep NP
- DO := NP
- IO := NP
- Adjs := Adj Adjs | {}
- Advs := Adv Advs | {}
- PPs := PP PPs | {}
- D := a | an | the | {}
- Adj := errant | atonal | squishy | piquant | desultory
- Adv := quickly | excruciatingly
- V := throws | explains | slithers
- Prep := to | with | underneath
- N := aardvark | avocado | accordion | professor | pandemonium

A context-free grammar

- S := NP
- NP := D
- VP := A … DO PPs
- PP := Pr
- DO := N
- IO := NI
- Adjs :=
- Advs :=
- PPs := P
- D := a |
- Adj := e
- Adv := c
- V := thr
- Prep :=
- N := aardvark | avocado | accordion | professor | pandemonium

the errant professor explains the desultory avocado to the squishy aardvark

a piquant accordion quickly excruciatingly slithers underneath the atonal pandemonium

Shift-reduce parser

input(x) ⇒ parse(x, nil)

parse(cons(x, y), z) ⇒ parse(y, cons(x, z))

parse(x, (VP NP . y)) ⇒ parse(x, (S . y))

parse(x, (N Adjs D . y)) ⇒ parse(x, (NP . y))

parse(x, y) ⇒ parse(x, (Adjs . y))

parse(x, (aardvark . y)) ⇒ parse(x, (N . y))

…

parse(nil, (S)) ⇒ parsed

An example parse

- input((the professor slithers))

More careful

$\text{input}(x) \wedge \text{input}(y) \Rightarrow (x = y)$

$NP \neq VP \wedge NP \neq S \wedge NP \neq \text{the} \wedge \text{avocado} \neq \text{aardvark} \wedge \text{avocado} \neq \text{the} \wedge \ldots$

$\text{terminal}(x) \Leftrightarrow x = \text{avocado} \vee x = \text{the} \vee \ldots$

$\text{input}(x) \Leftrightarrow \text{parse}(x, \text{nil})$

$\text{parse}(\text{nil}, (S)) \Leftrightarrow \text{parsed}$

More careful (cont'd)

terminal(x) $\Rightarrow$

[parse(cons(x, y), z) $\Leftrightarrow$ parse(y, cons(x, z))]

[parse(x, (aardvark . y)) $\vee$ parse(x, (avocado . y))

$\vee$ …] $\Leftrightarrow$ parse(x, (N . y))

[parse(x, y) $\vee$ parse(x, (Adjs Adj . y)]

$\Leftrightarrow$ parse(x, (Adjs . y))

…

Extensions

- *Probabilistic CFG*
- *Context-sensitive features (e.g., coreference: John and Mary like to sail. His yacht is red, and hers is blue.)*