

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) knows(x, Mary)

- *What about*

knows(John, x) knows(y, Mary)

Quiz

- *Can we unify*

knows(John, x) knows(x, Mary)

No!

- *What about*

knows(John, x) 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

$\text{rains} \wedge \text{outside}(x) \Rightarrow \text{wet}(x)$

$\text{wet}(x) \Rightarrow \text{rusty}(x) \vee \text{rustproof}(x)$

$\text{robot}(x) \Rightarrow \neg \text{rustproof}(x)$

rains

$\text{guideboat}(\text{Robby})$

$\text{guideboat}(x) \Rightarrow \text{robot}(x) \wedge \text{outside}(x)$

$\text{rains} \wedge \text{outside}(x) \Rightarrow \text{wet}(x) \rightarrow \neg \text{rains} \vee \neg \text{outside}(x) \vee \text{wet}(x)$ ⁶
 $\text{wet}(x) \Rightarrow \text{rusty}(x) \vee \text{rustproof}(x) \rightarrow \neg \text{wet}(x) \vee \text{rusty}(x) \vee \text{rustproof}(x)$ ¹⁰
 $\text{robot}(x) \Rightarrow \neg \text{rustproof}(x) \rightarrow \neg \text{robot}(x) \vee \neg \text{rustproof}(x)$ ⁴

rains

guidebot(Robby)

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

$\neg (\exists x. \text{rusty}(x)) \rightarrow \neg \text{rusty}(x)$ ¹⁴

1,2 \models robot(R) ⁵ (x \rightarrow R)

1,3 \models outside(R) ⁹ (x \rightarrow 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

rains ⁷

guidebot(R) ¹

\neg guidebot(x) ² \vee robot(x)

\neg guidebot(x) \vee outside(x) ³

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***

$$\begin{aligned} \forall P. \forall x, y. (x=y) &\Rightarrow P(x) \Leftrightarrow P(y) \\ \forall P. \forall x, y, z. (x=y) &\Rightarrow P(x, z) \Leftrightarrow P(y, z) \\ &\Rightarrow P(z, x) \Leftrightarrow P(z, y) \end{aligned}$$

Restricted semantics

- *Only check one finite, propositional KB*
 - *NP-complete much better than RE*
- *Unique names: objects with different names are different (John \neq 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., $\text{loves}(x,y) \Rightarrow \text{happy}(x)$ 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)*

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 $\exists x. \text{killed}(x, JR)$, hope the proof is constructive*
 - *may not work even if constr. proof exists*

Answer literals

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

Example

1 $\text{kills}(\text{Jack}, \text{Cat}) \vee \text{kills}(\text{Curiosity}, \text{Cat})$

2 $\neg \text{kills}(\text{Jack}, \text{Cat})$

3 $\neg \text{kills}(x, \text{Cat})$

4 $3 \models \text{kills}(J, C) \quad (x \rightarrow \text{Curiosity})$

4,2 $\models F$

Example

¹ $\text{kills}(\text{Jack}, \text{Cat}) \vee \text{kills}(\text{Curiosity}, \text{Cat})$

² $\neg \text{kills}(\text{Jack}, \text{Cat})$

³ $\neg \text{kills}(x, \text{Cat})$

^{1,3} $(x \rightarrow J) \models \text{kills}(\text{Curiosity}, \text{Cat})$ ⁴

^{3,4} $(x \rightarrow Cu) \models F$

Example

¹ $\text{kills}(\text{Jack}, \text{Cat}) \vee \text{kills}(\text{Curiosity}, \text{Cat})$

² $\neg \text{kills}(\text{Jack}, \text{Cat})$

³ $\neg \text{kills}(x, \text{Cat}) \vee \text{Answer}(x)$

$1,3 \models \text{kills}(J, C) \vee \text{answer}(C)$ ⁴

$2,4 \models \text{answer}(C)$



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
 - $likes(x, kittens) \vee \neg likes(y, x) \rightarrow A \vee \neg B$
- KB ground and KB_0 unsatisfiable $\Rightarrow KB$ unsatisfiable

Propositionalizing

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

Example

$\text{kills}(\text{Jack}, \text{Cat}) \vee \text{kills}(\text{Curiosity}, \text{Cat})$

$\neg \text{kills}(\text{Jack}, \text{Cat})$

$\neg \text{kills}(x, \text{Cat}) \implies \neg \text{kills}(\text{foo}, \text{Cat})$

$A \vee B$

$\neg A$

$\neg B$

M'

$A = F$

$B = T$

$C = F$

M

$\text{kills}(J, C) = F$

$\text{kills}(Cu, C) = T$

$\text{kills}(x, C) = F$

$x \neq (J, Cu)$

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

$\text{kills}(\text{Jack}, \text{Cat}) \vee \text{kills}(\text{Curiosity}, \text{Cat})$

$\neg \text{kills}(\text{Jack}, \text{Cat})$

$\neg \text{kills}(x, \text{Cat})$

$\neg \text{kills}(\text{Jack}, \text{Cat})$
 $\text{kills}(\text{Curiosity}, \text{Cat})$
 $\neg \text{kills}(\text{Foo}, \text{Cat})$

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 $x \rightarrow \text{Curiosity}$*
- *Such pairs of atoms are **discordant***
- *They suggest useful ways to instantiate*

Example

$\text{Kills}(\text{Jack}, \text{Cat}) \vee \text{Kills}(\text{Curiosity}, \text{Cat})$

$\neg \text{Kills}(\text{Jack}, \text{Cat})$

$\neg \text{Kills}(x, \text{Cat})$

$\neg \text{Kills}(\text{Curiosity}, \text{Cat})$

$A \vee B$

$\neg A$

$\neg B$

InstGen

- *Propositionalize $KB \rightarrow KB'$, 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 =

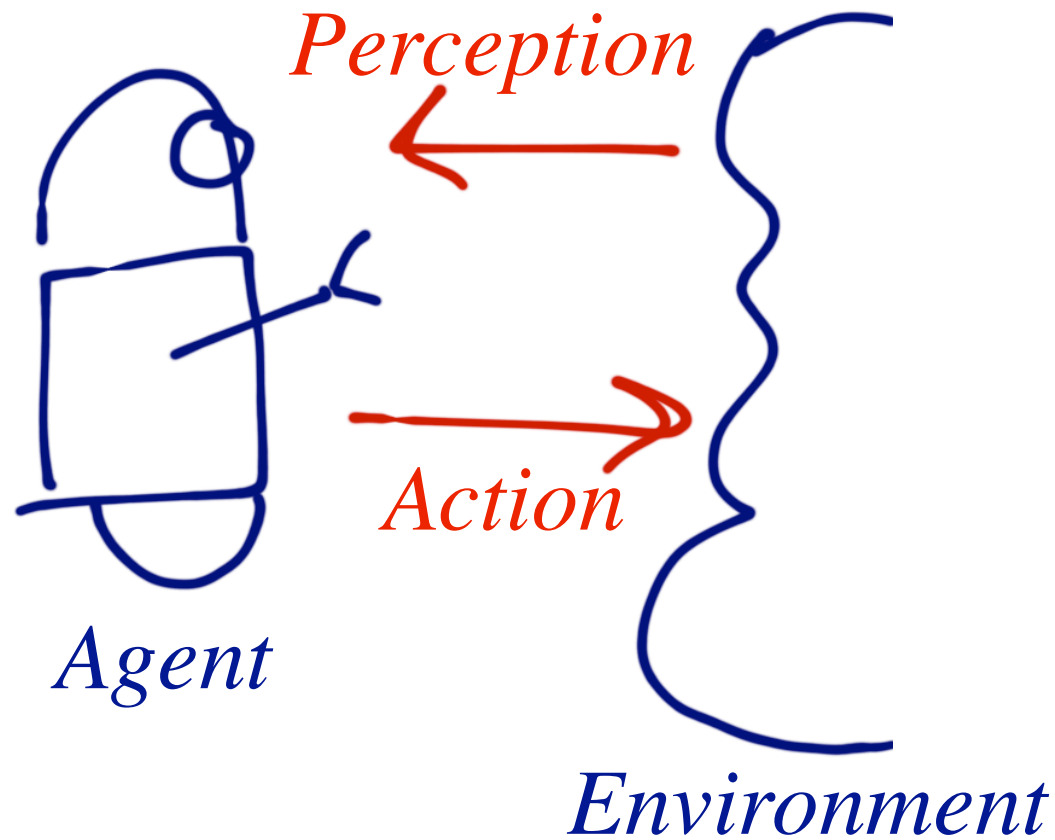
- *We've already argued soundness*
- *Completeness theorem: if KB is unsatisfiable but KB' is satisfiable, must exist a discordant pair wrt M' which generates a new instantiation of a clause from KB —and, \exists a finite sequence of such instantiations will find an unsatisfiable propositional formula*



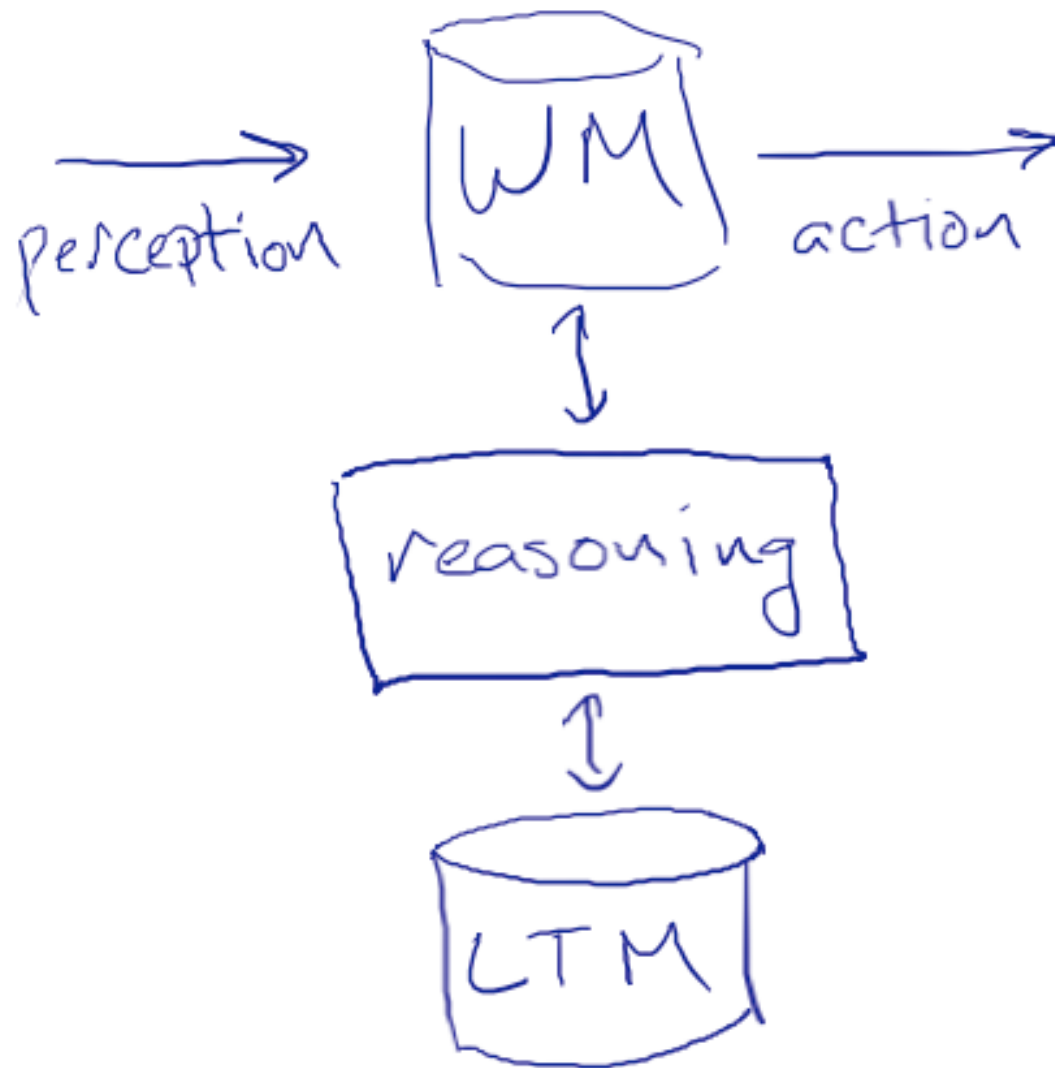
Agent

Architectures

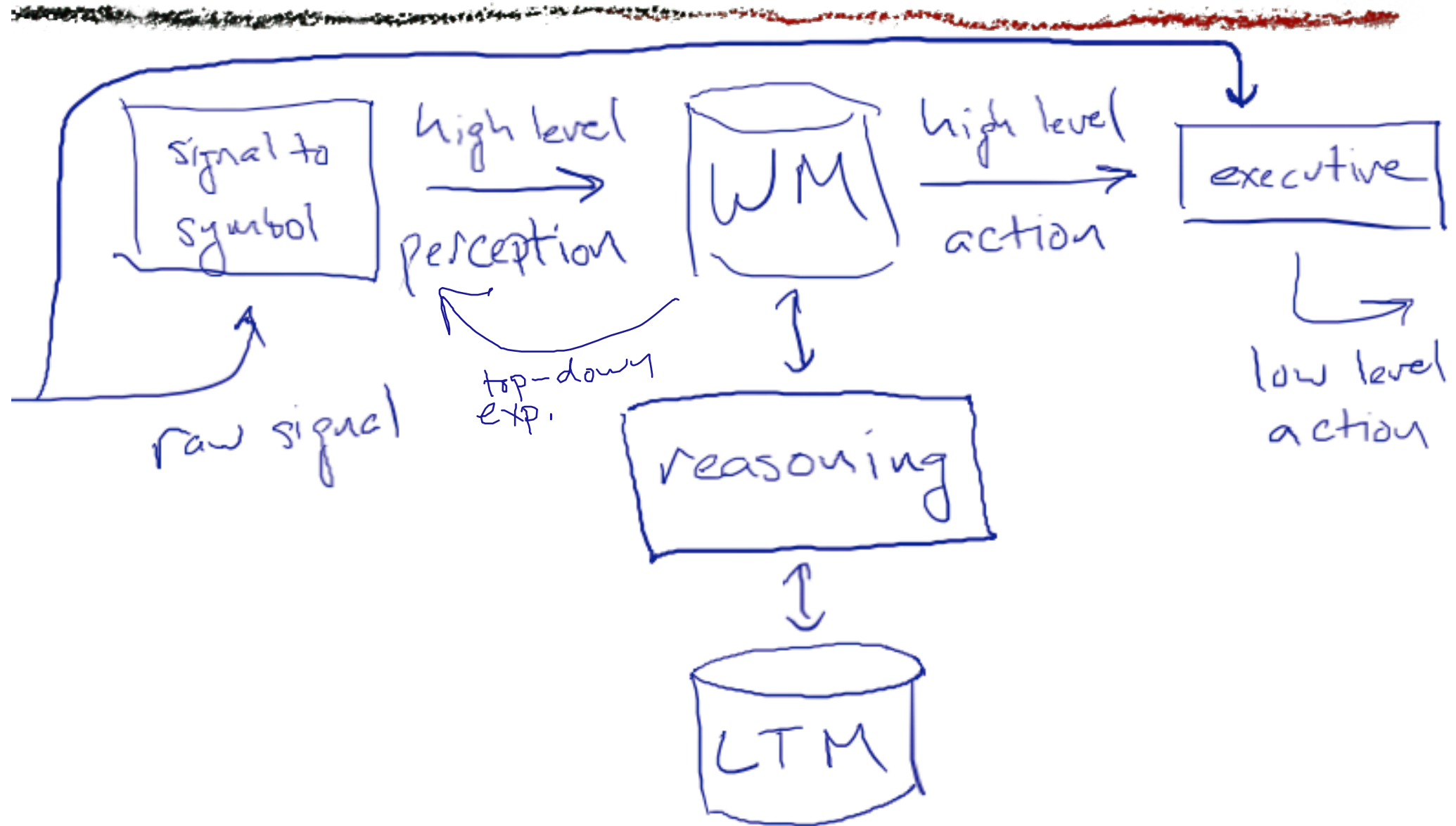
Situated agent



Inside the agent



Inside the agent





Knowledge Representation

Knowledge Representation



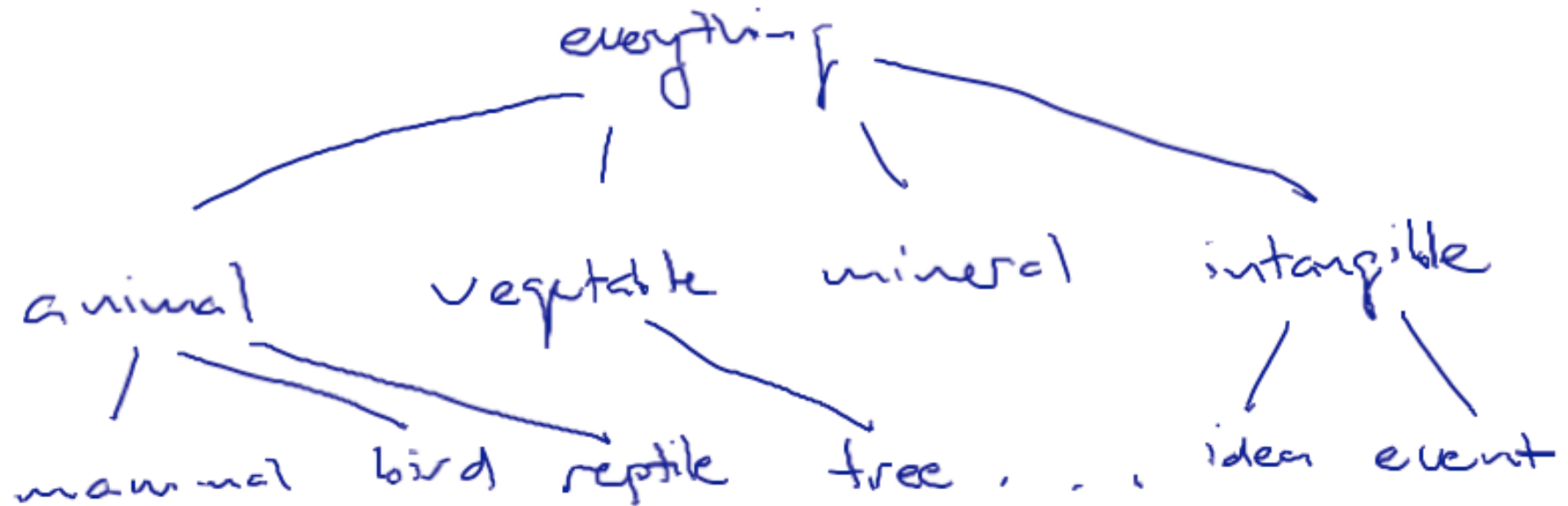
- *is the process of*
 - *Identifying relevant objects, functions, and predicates*
 - *Encoding general background knowledge about domain (reusable)*
 - *Encoding specific problem instance*
- *Sometimes called knowledge engineering*

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., $\text{talk}(\text{Professor}, \text{FOL})$)*
- *Robot shouldn't move:*
 - *if $\text{at}(\text{Robot}, \text{Wean4623}, t)$, want $\text{at}(\text{Robot}, \text{Wean4623}, \text{result}(\text{talk}(\text{Professor}, \text{FOL})))$*
- *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

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

A context-free grammar

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

A context-free grammar

- S := NP VP
- NP := D N
- VP := A VP
- PP := Pr N
- DO := N
- IO := N
- Adjs := A
- Advs := A
- PPs := P N
- D := a | the
- Adj := e | o | u | i | a
- Adv := c | q | s | u | n | d | e | r
- V := thr | slithers
- Prep := u | n | d | e | r
- 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

DO PPs

Shift-reduce parser

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

$\text{parse}(\text{cons}(x, y), z) \Rightarrow \text{parse}(y, \text{cons}(x, z))$

$\text{parse}(x, (\text{VP NP} \cdot y)) \Rightarrow \text{parse}(x, (\text{S} \cdot y))$

$\text{parse}(x, (\text{N Adjs D} \cdot y)) \Rightarrow \text{parse}(x, (\text{NP} \cdot y))$

$\text{parse}(x, y) \Rightarrow \text{parse}(x, (\text{Adjs} \cdot y))$

$\text{parse}(x, (\text{aardvark} \cdot y)) \Rightarrow \text{parse}(x, (\text{N} \cdot y))$

...

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

An example parse



- `input((the professor slithers))`

More careful

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

$\text{NP} \neq \text{VP} \wedge \text{NP} \neq \text{S} \wedge \text{NP} \neq \text{the} \wedge \text{avocado} \neq$
 $\text{armadillo} \wedge \text{avocado} \neq \text{the} \wedge \dots$

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

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

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

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.)*