CS 7180: Behavioral Modeling and Decision-making in AI

First-order Logic & Situation Calculus

Guest Lecturer: Prof. Thomas Wahl

September 14, 2012

Outline

- Propositional logic (review of the review)
- First-order logic (review)
- Actions and events in FOL
  - Situation calculus
  - Frame problem

Warning!

"Logic, like whiskey, loses its beneficial effect when taken in too large quantities."

—Lord Dunsany

Review: clauses and inference

- **Literal** is an atomic sentence or negated atom (P, $\neg P$)
- **Clause** is a **disjunction** of literals (i.e., $P \vee \neg Q \vee R$)
- **Knowledge base** is a set of true propositional sentences, i.e., conjunction of sentences ($\wedge$ is implicit)
- KB is in **conjunctive Normal Form (CNF)** if written as aconjunction of disjunctions, i.e., conjunction of clauses
- With KB in CNF, resolution is **sound and complete** inference procedure in a single rule!!
- **Theorem**: any set of sentences can be transformed into CNF

Horn clauses

- **Horn clause**—clause with at most one positive literal

  $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n$
- **Definite clause**—Horn clause with exactly one positive literal

  $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n \vee R$
- **Goal clause**—Horn clause with no positive literals

  $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n$
- Closed under resolution (i.e., resolution of Horn clauses will return Horn clause)
- Special properties of KBs with Horn clauses
  1. Definite clauses can be written as implication rules <body> ⇒ <head>

     $P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow R$
  2. Two inference methods that work for Horn clauses
     - **Forward chaining** (data driven)
     - **Backward chaining** (goal driven)
  3. Entailment can be decided in linear time w.r.t. size of KB

Forward chaining

- Determines if query q is entailed by KB of definite clauses
  - Starts with known facts and derives new knowledge

- **Horn clauses:**
  C1. $\neg P_1 \vee \neg P_2 \vee P_4$
  C2. $\neg P_4 \vee P_5$

Rules:
$P_1 \wedge P_2 \Rightarrow P_4$
$P_4 \Rightarrow P_5$

- **Facts:** P_1 , P_2

- Step 1: Facts P_1 and P_2 resolve with C1 to get P_4 (Add P_4 to KB)
- Step 2: Resolve P_4 with C2 to get P_5
  - This is called **rule chaining**
- Derive conclusions from incoming **facts**

Backward chaining

- Goal-driven reasoning triggered by a **query**
- Basis of backward chaining

$P \land R \Rightarrow Q$ is an assertion in our KB

Q is a query we want to prove (or disprove)

Set up P and R as sub-queries, if true then Q is proved

- What if we cannot find Q or a rule that proves Q?
  - Answer False—**negation by failure** (not the same as a real proof of $\neg Q$)
- Note: $P \Rightarrow \neg Q$ is not a Horn clause
  - Normalizes to $P \lor Q$, which has two positive literals

Backward chaining example

- Works backward to determine if the query q is true

- **Horn clauses:**
  C1. $\neg P_1 \vee \neg P_2 \vee P_4$
  C2. $\neg P_4 \vee P_5$

Rules:
$P_1 \wedge P_2 \Rightarrow P_4$
$P_4 \Rightarrow P_5$

- **Facts:** P_1 , P_2

Goal: P_5

- Subgoal: prove P_4
  - Sub-sub goal: prove P_2
  - Sub-sub goal: prove P_1
- Very efficient—only touches relevant facts/rules

Pros and cons of propositional logic

- Propositional logic is **declarative**
- PL allows **partial/disjunctive/negated** information
  - Horn clauses are a nice intermediate form
- Propositional logic is **compositional**
  - Meaning of $L_{1,1} \wedge up$ is derived from the meaning of $L_{1,1}$ and of *up*
- Meaning in PL is **context-independent**
  - Unlike natural language, where meaning depends on context

- Propositional logic has very **limited expressive power**
  - Unlike natural language…
  - Can become impractical even for simple worlds

First-order logic

- Propositional logic limits world models to atomic facts
  E.g. $P_{1,2} \Rightarrow B_{2,2}$
- First-order logic (like natural language) can manipulate world models that include
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother of, bigger than, outside, part of, has color, occurs after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father, nationality, one more than, plus…

Basic syntax of FOL

- **Constant symbols** — ***KingJohn, 2, NEU, …***
- **Predicate symbols** — ***IsHappy, Likes, >, …***
- **Function symbols** — ***Sqrt, Nationality, …***
- **Variables** — ***x, y, a, b, …***
- Connectives — $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Equality — $=$
- Quantifiers — $\forall, \exists$

- Constant, predicate, and function symbols called a **logical language**—provided by user

Atomic sentences in FOL

- Term = $function(term_1,...,term_n)$ or constant or variable
- A term with no variables is a **ground term**
- Examples
  - Brother(KingJohn, RichardTheLionheart)
  - >(AgeOf(Richard), AgeOf(John))
  - Brother(AgeOf(Richard), AgeOf(John))

- Atomic sentence = $predicate(term_1,...,term_n)$ or $term_1 = term_2$
- An **atomic sentence** has value true or false

Complex and quantified sentences

- Complex sentences are made from atomic sentences using connectives
  - Connectives have same semantics as propositional logic
  - $\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$
- Examples
  - *Sibling*(*John*,*Richard*) ⇔ *Sibling*(*Richard*,*John*)
  - $>(1,2) \wedge \leq(1,2)$
  - $>(1,2) \wedge \neg >(1,2)$

- A **quantified sentence** adds quantifiers ∀ and ∃
- A **well-formed formula (wff)** is contains no "free" variables—all variables are "bound" by quantifiers

  $(\forall x)P(x,y)$ has *x* bound as a universally quantified variable, but *y* is free

Semantics of FOL

- **Domain *M***: the set of all objects in the world (of interest)
- **Interpretation *I***:
  - Assigns each constant to object in M
  - Defines each function of n arguments as a mapping $M^n \rightarrow M$
  - Defines each predicate as a mapping $M^n \rightarrow \{True, False\}$
- Every **ground predicate** will have a truth value
- In general there is an infinite number of interpretations because $|M|$ is infinite
- **Model**: interpretation of a set of sentences such that every sentence is *True*
- A sentence is
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there is no interpretation where the sentence is true

Semantics of quantifiers

- **Universal quantification**
  - $\forall x\, P(x)$ means that P holds for **all** values of x
    E.g., $\forall x\ dolphin(x) \Rightarrow mammal(x)$
- **Existential quantification**
  - $\exists\, x\, P(x)$ means that P holds for **some** value of x
    E.g., $\exists\, x\ mammal(x) \wedge lays\text{-}eggs(x)$
  - Permits one to make a statement about some object without naming it

Universal quantification examples

- $\forall$ *<variables> <sentence>*

- Everyone at NU is smart:
  $\forall x\ [\ At(x, NEU) \Rightarrow Smart(x)\ ]$

- $\forall x\ P$ is true in a model *m* iff *P* is true with *x* being each possible object in the model

- Roughly speaking, equivalent to the **conjunction** of all **instantiations** of *P*

$$
\begin{aligned}
& At(KingJohn, NEU) \Rightarrow Smart(KingJohn) \\
\wedge\ & At(Richard, NEU) \Rightarrow Smart(Richard) \\
\wedge\ & At(NEU, NEU) \Rightarrow Smart(NEU) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$

- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x\, At(x, \text{NEU}) \wedge Smart(x)$$

means "Everyone is at NEU and everyone is smart"

Existential quantification examples

- $\exists$<*variables*> <*sentence*>

- Someone at NEU is smart:
  $\exists x\ [At(x,NEU) \wedge Smart(x)]$

- $\exists x\ P$ is true in a model m iff P is true with x being some possible object in the model

- Roughly speaking, equivalent to the **disjunction** of all **instantiations** of P

$$
\begin{aligned}
& At(KingJohn,NEU) \wedge Smart(KingJohn) \\
\vee\ & At(Richard,NEU) \wedge Smart(Richard) \\
\vee\ & At(NEU,NEU) \wedge Smart(NEU) \\
\vee\ & \ldots
\end{aligned}
$$

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

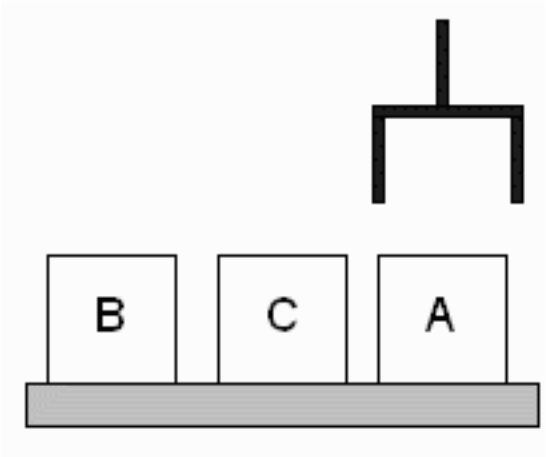
$$\exists x\ At(x, NEU) \Rightarrow Smart(x)$$

is true if there is no one at NEU!

Blocks world in FOL

- Constants: Block, Table, A, B, C, Cube, Brick, Sphere, Pyramid, Red, Blue, Large, Small
- Functions: Size
- Relations: Cleartop, Color, Loc, On, Held, Above…

- Atomic sentences:
  Loc(*y*,*z*), *On*(*x*,*y*); *Cleartop*(*x*); *Held*(*x*); *Isa*(*x*, *z*), *Color*(*x*, *c*)



Blocks world sentences

- Axioms for general world knowledge:
  $\forall x$ [$Isa(x, Block) \Rightarrow$
  ($\exists y$ $Isa(y, Block) \wedge On(x, y)$) $\vee$ $On(x, Table)$]

- To prevent something silly (we need "common sense"):
  $\forall x,y$ [$On(x, y) \Rightarrow \neg = (x, y)$]

- Axioms for specific state of the world:
  $Isa(A, Block)$
  $Isa(B, Block)$
  $Isa(C, Block)$
  $On(A, Table)$
  $On\ (B, Table)$
  $On\ (C, Table)$

Substitution of variables

- A **substitution** is a set of variable-term pairs: {x/term, y/term, ...}
  - Often referred to using the symbol θ [theta]
  - No variable can occur more than once.

- For any term or formula *A*:
  - *Subst*(θ, *A*) also written *A*θ is result of replacing each variable with the corresponding term
    (Remember a term is a constant, variable symbol, or function)
    $S = Smarter(x,y)$
    $\theta = \{x/Hillary, y/Bill\}$
    $S\theta = S\,\{x/Hillary, y/Bill\} = Smarter(Hillary, Bill)$

- Given set of sentences *KB*, return some/all θ s.t. $KB \models S\theta$

Unification

- Two formulas A and B **unify** if there is a substitution θ such that $A\theta = B\theta$.
  - ***θ* is not unique!**
- Example—unify $A = Knows(John,x)$ and $B = Knows(y,z)$
  $\theta_1 = \{y/John, x/z\}$ or $\theta_2 = \{y/John, x/Sue, z/Sue\}$
- The first unifier is **more general** than the second
- There is a single **most general unifier** (MGU) that is unique up to renaming variables
  MGU = $\{y/John, x/z\}$ or $\{y/John, z/x\}$

Most general unifier

- If θ_1 is a unifier for formulas A and B, it is a **MOST GENERAL UNIFIER** (MGU) iff:

  There is no other unifier θ_2 for A and B s.t. $A\theta_2$ subsumes $A\theta_1$
- A formula F **subsumes** a formula G if there is a non-trivial substitution Π s.t. $F\Pi = G$

- **Example**

  $A = Knows(John,x)$ and $B = Knows(y,z)$

  $\theta_1 = \{y/John, x/z\}$ or $\theta_2 = \{y/John, x/Sue, z/Sue\}$

  $A\theta_1 = Knows(John, z)$ $A\theta_2 = Knows(John, Sue)$

  $A\theta_1$ subsumes $A\theta_2$ so $\theta_2 = \{y/John, x/Sue, z/Sue\}$ is not an MGU

  NOTE: What is Π?

Unification examples

Unify**(*α*,*β*) = *θ* if *αθ* = *βθ***

- Find the unifier θ

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, Barak)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, Barak)$	

Unification examples

Unify*(*α*,*β*) = *θ* if *αθ* = *βθ

- Find the unifier θ

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	**$\{x/Jane\}$**
$Knows(John,x)$	$Knows(y,Barak)$	
$Knows(John,x)$	$Knows(y,Mother(y))$	
$Knows(John,x)$	$Knows(x,Barak)$	

Unification examples

$\mathbf{\mathit{Unify}(\alpha,\beta) = \theta}$ **if** $\mathbf{\alpha\theta = \beta\theta}$

- Find the unifier θ

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	$\{x/Jane\}$
$Knows(John,x)$	$Knows(y,Barak)$	$\{x/Barak,y/John\}$
$Knows(John,x)$	$Knows(y,Mother(y))$	
$Knows(John,x)$	$Knows(x,Barak)$	

Unification examples

$Unify(\alpha,\beta) = \theta$ if $\alpha\theta = \beta\theta$

- Find the unifier θ

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	$\{x/Jane\}$
$Knows(John,x)$	$Knows(y,Barak)$	$\{x/Barak,y/John\}$
$Knows(John,x)$	$Knows(y,Mother(y))$	$\{x/Mother(John),y/John\}$
$Knows(John,x)$	$Knows(x,Barak)$	

Unification examples

$Unify(\alpha,\beta) = \theta$ if $\alpha\theta = \beta\theta$

- Find the unifier θ

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	**$\{x/Jane\}$**
$Knows(John,x)$	$Knows(y,Barak)$	**$\{x/Barak,y/John\}$**
$Knows(John,x)$	$Knows(y,Mother(y))$	**$\{x/Mother(John),y/John\}$**
$Knows(John,x)$	$Knows(x,Barak)$	**{fail}**

Unification examples

$Unify(\alpha,\beta) = \theta$ if $\alpha\theta = \beta\theta$

- Find the unifier θ

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	**{x/*Jane*}**
$Knows(John,x)$	$Knows(y,Barak)$	**{x/*Barak*,y/*John*}**
$Knows(John,x)$	$Knows(y,Mother(y))$	**{x/*Mother*(*John*),y/*John*}**
$Knows(John,x)$	$Knows(x,Barak)$	**{fail}**

- Last one fails because we need to **standardize apart** the variables
  - Rename x in $Knows(x,Barak)$ to a
  - Unify with $\theta = \{x/Barak, a/John\}$

Inference in FOL

- **Rules of inference** same as in propositional logic in ground cases
- **Generalize (lift)** rules to FOL case with variables and quantifiers

- Universal instantiation (UI)—every **instantiation** of a universally quantified sentence is entailed by it

$$\frac{\forall v\ \alpha}{Subst(\{v/g\}, \alpha)}$$

  for any variable v and ground term g

- Existential instantiation (EI)—for any sentence α, variable v, and constant symbol k that does **NOT** appear elsewhere in the knowledge base

$$\frac{\exists v\ \alpha}{\mathrm{Subst}(\{v/k\}, \alpha)}$$

- Apply EI followed by UI to get fully ground version of *KB*

Quantifier inference examples

- **Universal instantiation**
  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

- **Existential instantiation**
  $\exists x\ Crown(x) \wedge OnHead(x,John)$ yields:

  $Crown(C_1) \wedge OnHead(C_1,John)$

  where C_1 is a **brand-new** constant not occurring in any sentence
  - Also known as skolemization—constant is a **skolem constant**

Implementing universal instantiation

All humans are mortal
Jack is human

Jack is mortal

- How do we represent this in FOL?
  R1. $\forall \mathrm{x}\ Human(x) \Rightarrow Mortal(x)$
  F1. $Human(Jack)$
- **Modus ponens** for FOL
  - Let R1 be $p \Rightarrow q$, Let F1 be be p'
  - If p and p' **unify**, conclude q'—find the substitution θ

Applying unification to reasoning

- **Modus ponens** says:

  Given $p \Rightarrow q$ and p

  Conclude q
- In FOL

  Given $p \Rightarrow q$, and p' (where p and p' unify by θ)

  Conclude q' s.t. $q' = q\theta$
- Suppose our KB includes:

  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

  $King(John)$
- Modus ponens won't quite work since we have $p_1 \wedge p_2 \Rightarrow q$
  - But we can see using human reasoning that it should!

Generalized Modus Ponens (GMP)

$$\frac{(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q),\ p_1',\ p_2',\ \ldots,\ p_n'}{q\theta}$$ **where $p_i'\theta = p_i\theta$ for all i**

- **Example**

  p_1' is $King(John)$ — p_1 is $King(x)$

  p_2' is $Greedy(y)$ — p_2 is $Greedy(x)$

  θ is {x/John,y/John} — q is $Evil(x)$

  $q\theta$ is $Evil(John)$

- GMP uses **definite clauses** (**exactly** one positive literal)
- All variables assumed universally quantified
- Follows from the resolution rule for FOL…

Resolution in FOL

- First-order resolution rule:

$$\frac{l_1 \vee \ldots \vee l_k,\ m_1 \vee \ldots \vee m_k}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\ \theta}$$

where $Unify(l_i, \neg m_j) = \theta$

- The two clauses are assumed to be **standardized apart**

$$\frac{\neg Rich(x) \vee Unhappy(x) \quad Rich(Ken)}{Unhappy(Ken)}$$

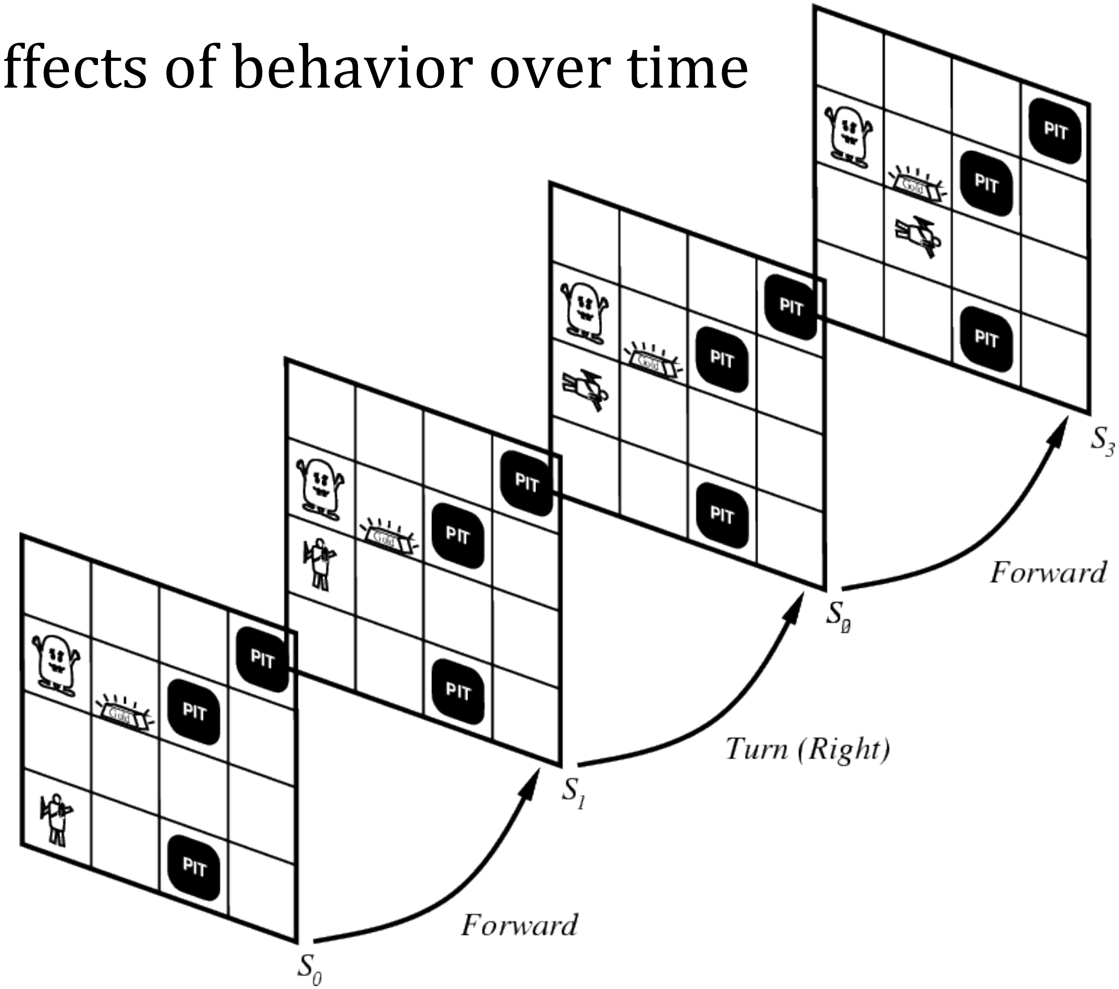
with $\theta = \{x/Ken\}$

Logic of events—the situation calculus

- We can model dynamic behavior with logic and resolution
  - FOL representation of the events over time
  - Resolution to find a solution
- **Key idea:** represent a snapshot of the world, called a "situation" explicitly
- **Fluents**—statements that are true or false in any given situation, e.g., "I am at home"
- Actions map situations to situations
  - When an agent performs action A in situation S_1, the result is a new situation S_2

Situations in the Wumpus world

- Each action leads to a new situation
- Models effects of behavior over time



Describing actions and effects

- **Situation**—snapshot of the world at an interval of time where nothing changes
- Every true or false statement is made w.r.t. a particular situation
  - Add **situation variables** to every predicate
  - *At*(*Agent*,1,1) becomes *At*(*Agent*,1,1,*s0*)—At(*Agent*,1,1) is true in situation (i.e., state) *s0*
- Result function, ***result(a,s)***, mapping situation *s* into a new situation as a result of performing action *a*
  - E.g., *result*(*forward, s*) is a function that returns the successor state (situation) to *s* after we move forward

Blocks world example

- A move action:
  $Move(x, loc)$
- Use of the Result function:
  $Result(Move(x, loc), state)$ → the state resulting from doing the *Move* action
- An axiom about moving:
  $$\forall x\ \forall loc\ \forall s\ [\ At(x, loc, Result(Move(x, loc), s))\ ]$$
  If you move some object to a location, then in the resulting state that object is at that location

- $At(B1, Table, S0)$
- $At(B1, Top(B2), Result(Move(B1, Top(B2)), S0))$
  using the axiom

Monkeys and bananas problem

- The **monkey-and-bananas problem** is faced by a **monkey** standing under some **bananas** which are hanging out of reach from the ceiling. There is a box in the corner of the room that will enable the **monkey** to reach the **bananas** if he climbs on it.

- Use situation calculus to represent this problem
- Once we have a representation, we can solve it using resolution—i.e., determine

Representation of Monkey/banana problem

- **Fluents**

  At(*x*, *loc*, *s*)

  On(*x*, *y*, *s*)

  Reachable(*x*, *Bananas*, *s*)

  Has(*x*, *y*, *s*)

- **Other predicates**

  Moveable(*x*), *Climbable*(*x*)

  Can-move(*x*)

- **Actions**

  Climb-on(*x*, *y*) *Move*(*x*, *loc*)

  Reach(*x*, *y*) *Push*(*x*, *y*, *loc*)

Constants

BANANAS

MONKEY

BOX

S0

CORNER

UNDER-BANANAS

Monkey/bananas axioms

1. $\forall x1, s1$ [$Reachable(x1, BANANAS, s1) \Rightarrow$
   $Has(x1, BANANAS, Result(Reach(x1, BANANAS), s1))$]
   If a person (or monkey!) can reach the bananas ,then the result of reaching them is to have them

2. $\forall s2$ [$At(BOX, UNDER\text{-}BANANAS, s2) \wedge On(MONKEY, BOX, s2) \Rightarrow$
   $Reachable(MONKEY, BANANAS, s2)$
   If a box is under the bananas and the monkey is on the box then the monkey can reach the bananas

3. $\forall x3, loc3, s3$ [$Can\text{-}move(x3) \Rightarrow$
   $At(x3, loc, Result(Move(x3, loc3), s3))$]
   The result of moving to a location is to be at that location

Monkey/bananas axioms (cont.)

4. $\forall x4, y4, s4\ [\exists loc4\ [At(x4, loc4, s4) \land At(y4, loc4, s4)] \land$
   $Climbable(y4) \Rightarrow On(x4, y4, Result(Climb\text{-}on(x4, y4), s4))]$
   The result of climbing on an object is to be on the object

5. $\forall x5, y5, loc5, s5\ [\exists loc\ [At(x5, loc0, s) \land At(y5, loc0, s5)\ ] \land$
   $Moveable(y5) \Rightarrow At(y5, loc5, Result(Push(x5, y5, loc5), s5))$
   The result of x pushing y to a location is y is at that location

6. $<same> \Rightarrow At(x6, loc6, Result(Push(x6, y6, loc6), s6))\ ]$
   The result of x pushing y to a location is x is at that location

Monkey/bananas problem (cont.)

- **Initial state** (S0)
  F1. *Moveable*(*BOX*)
  F2. *Climbable*(*BOX*)
  F3. *Can-move*(*MONKEY*)
  F4. *At*(*BOX, CORNER, S0*)
  F5. *At*(*MONKEY, UNDER-BANANAS, S0*)
- **Goal**
  Has(*MONKEY, BANANAS, s*)

- **How to solve?**
  - Apply resolution to prove
    Has(*MONKEY, BANANAS*, ***Result*(*Reach*(. . .), *Result*(. .) . .), *S0***)
  - Will give the actions from s0 to goal in reverse order
- For the proof to work, we need two additional **frame axioms**…

Frame axioms for monkey/bananas problem

- Frame axioms keep track of what **does not** change from one situation to the next

7. $\forall x, y, loc, s\ [\ At(x, loc, s) \Rightarrow$
   $At(x, loc, Result(Move(y, loc), s))\ ]$
   The location of an object does not change as a result of someone moving to the same location

8. $\forall x, y, loc, s\ [\ At(x, loc, s) \Rightarrow$
   $At(x, loc, Result(Climb\text{-}on(y, x), s))\ ]$
   The location of an object does not change as a result of someone climbing on it

Limitation of situation calculus

- **Frame problem** (it's back again!!)
- I go from home to the store—new situation *S'*
- In *S'*
  - The store still sells chips
  - My age is still the same
  - Los Angeles is still the largest city in California…

- How can we efficiently represent everything that doesn't change?
  Successor state axioms…

Successor state axioms

- Normally, things stay true from one state to the next—unless an action **changes** them
  $At(p, loc, Result(a, s))$ iff $a = Go(p, x)$
  or $[At(p, loc, s)$ and $a \mathrel{!}= Go(p, y)]$

- We need one or more of these for every fluent
- Now we can use theorem proving (or possibly backward chaining) to infer an agent's course of action
  - Still not very practical
- Need for **effective heuristics** for situation calculus