Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Instructor: Richard Scherl
- Prerequisite: Some Course in Logic
- Course Web Page:

  `http://www.cis.njit.edu/~scherl/ESSLLI01/index.html`

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Introduction
  1. A.I.
  2. Knowledge Representation
  3. Reasoning
  4. KR Hypothesis
  5. Cognitive Robotics
  6. Action Logics

What is AI?

Using principled characterizations of interactions between agents and thier environments to guide explanation and design (Philip Agre 1995)

.... the study and construction of rational agents. (Stuart Russell and Peter Norvig 1995)

Acting rationally means acting in such a way as to achieve one's goals given one's beliefs.

An agent is just something that perceives and acts.

In this approach, AI is viewed as the study and construction of rational agents. (ibid.)

A Stronger Notion of Agency

A computer system that is either conceptualized or implemented using concepts that are more usually applied to humans. These concepts are mentalistic notions such as knowledge, belief, intention, obligation etc.

Knowledge

Intelligent entities seem to anticipate their environments and the consequences of their actions. They act as if they know, in some sense, what the results would be. We can account for this anticipatory behavior by assuming that intelligent entities themselves possess knowledge of their environments. (Genesereth and Nilsson p2.)

Propositions: Statements about the world that are either true or false, right or wrong.

Representation and Reasoning

Representation

Symbols stand for things in the world.

Reasoning

Manipulation of symbols encoding propositions to produce representations of new propositions.

Knowledge-Based Systems

KR Hypothesis: (Brian Smith)

Any mechanically embodied intelligent process will be comprised of structural ingredients that a) We as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.

Implicit and Explicit Knowledge

KB is a set of sentences

explicit statement of sentences believed

$\mathbf{KB} \models \alpha$

- Explicit Knowledge: KB
- Implicit Knowledge: $\{\alpha | \mathbf{KB} \models \alpha\}$

Knowledge-Based System

Start with a KB representing what is explicitly known.

What to influence behavior based on what is implicit in the KB. This requires reasoning.

We want to calculate for any α, for a given KB whether or not

$$\mathbf{KB} \models \alpha$$

Knowledge Representation and Reasoning

How do we write down in some language descriptions of the world that correspond correctly to a state of the world?

How do we do so in such a way that computers can manipulate these representations and come to new conclusions?

Should this language be a logical language?

Cognitive Robotics

Most current work in robotics emphasizes basic-level tasks like sensory processing, path planning, manipulator design and control, reactive agents, artificial insects etc. In contrast, research in cognitive robotics is concerned with the theory and the implementation of robots that reason, act and perceive in changing, incompletely known, unpredictable environments.

Such robots must have higher level cognitive functions that involve reasoning, for example, about goals, actions, when to perceive and what to look for, the cognitive states of other agents, time, collaborative task execution, etc.

Cognitive Robotics (cont)

> In short, Cognitive Robotics is concerned with integrating reasoning, perception and action within a uniform theoretical and implementation framework.

From the description of the 1998 AAAI Fall Symposium on Cognitive Robotics

Common Sense

To endow computers with common sense is one of the major long term goals of Artificial Intelligence research. Although we know how to build programs that excel at certain bounded or mechanical tasks which humans find difficult, such as playing chess, we have very little idea how to program computers to do well at commonsense taks which are easy for humans. One approach to this problem is to formalize commonsense reasoning using mathematical logic. (From the Call for Papers of Common Sense 2001)

Reasoning about Actions: Logics of Actions

- Situation Calculus
- "A" Language
- Event Calculus
- Temporal Logics
- Dynamic Logics
- Fluent Calculus

Toronto Approach to Cognitive Robotics

- Based on the Situation Calculus
- Agent Theory
- Agent Programming Language – GOLOG
- University of Toronto – Hector Levesque and Raymond Reiter

Knowledge Representation Topics

- Situation Calculus vs Event Calculus
- Frame Problem
- Time
- Knowledge

Schedule

Class I

1. Introduction: What is K.R.? Cognitive Robotics?
2. Background in Logic

Class II

3. Situation Calculus

Class IIII

4. Event Calculus

Schedule (cont)

Class IV

5. Adding Knowledge and Sensing to the Situation Calculus

Class V

6. The Golog Programming Language

7. Conclusions

In Course Reader

1. *Propositional, First-Order and Higher-Order Logics* by Stuart Shapiro.
2. Chapter 4 from *Logical Foundations of Artificial Intelligence* by Genesereth and Nilsson
3. Selections from *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems* by Raymond Reiter, MIT Press, 2001.
   (a) Chapter 3: Introduction to the Situation Calculus.
   (b) Chapter 4: Foundations of the Situation Calculus.
   (c) Chapter 6: Complex Actions, Procedures and Golog.
   (d) Chapter 11: Sensing and Knowledge.
4. *The Event Calculus in Classical Logic* by Rob Miller and Murray Shanahan.

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Logic Background
- Quick Overview [Assumed you have seen this before]
  1. Connectives, Truth Tables
  2. First-Order Logic: Syntax and Semantics
  3. Entailment, Derivability
  4. Resolution

Connectives

- **Conjunction** $A \wedge B$
- **Disjunction** $B \vee C$
- **Implication** $A \rightarrow B$

  premise, antecedent A

  conclusion, consequent B
- **Equivalence** $A \leftrightarrow B$
- **Negation** $\neg B$

Truth Tables

P	Q	$P \wedge Q$
true	true	true
false	true	false
true	false	false
false	false	false

P	Q	$P \vee Q$
true	true	true
false	true	true
true	false	true
false	false	false

Truth Tables

P	Q	$P \rightarrow Q$
true	true	true
false	true	true
true	false	false
false	false	true

P	Q	$P \leftrightarrow Q$
true	true	true
false	true	false
true	false	false
false	false	true

Truth Tables

P	$\neg P$
true	false
false	true

First-Order Logic

- First-Order Predicate Calculus
- Predicate Calculus
- FOL
- FOPC

  The world consists of **objects** that is thinks with individual identities and **properties** that distinguish them from other objects.

  relations between objects.

  functions

Semantics

An interpretation **I** is a mapping between the elements of the *language* and the elements of a conceptualization of the world.

First-Order Logic

- **Objects:** people, houses, colors, numbers
  $\mathcal{D} = \{a, b, c, d, e\}$
- **Relations:** brother of, bigger than, part of, has color, owned.
  $[\![brotherof]\!] = \{\langle a, b\rangle, \langle d, e\rangle\}$
- **Properties:** red, round, prime
  $[\![red]\!] = \{a, e\}$
- **Functions:** father of, best friend.
  $[\![bestfriend]\!] = \{\langle\langle e\rangle, a\rangle, \langle\langle c\rangle, e\rangle\}$

First-Order Logic

- **Term:** A Logical Expression that refers to an object.

  john
  fatherof(john)

- **Atomic Sentences:**

  $\text{brother}(\text{john}, \text{richard})$
  $\text{married}(\text{fatherof}(\text{richard}), \text{motherof}(\text{john}))$

- **Complex Sentences:**

  $\text{brother}(\text{richard}, \text{john}) \wedge$
  $\text{brother}(\text{john}, \text{richard})$

First-Order Logic

Quantifiers, Variables

- **Universal Quantification:** $\forall$

$$\forall x\ \mathrm{cat}(x) \rightarrow \mathrm{mammal}(x)$$
$$\forall x\ \mathrm{man}(x) \rightarrow \mathrm{mortal}(x)$$

- **Existential Quantification:** $\exists$

$$\exists x\ \mathrm{sister}(x, \mathrm{spot}) \wedge \mathrm{cat}(x)$$

- **terms:**

$$\mathrm{a}, \mathrm{fatherof}(\mathrm{a}), x$$

- **ground terms:**

$$\mathrm{a}, \mathrm{fatherof}(\mathrm{a})$$

First-Order Logic

Syntax

$$
\begin{aligned}
&Atomic\_Sentence \Rightarrow \\
&Predicate(term_1 \ldots) \\
&\qquad\qquad \mid term = term \\
&term \Rightarrow Function(term_1 \ldots) \\
&\qquad\qquad \mid constant \mid variable \\
&Sentence \Rightarrow Atomic_Sentence \\
&\qquad\qquad \mid Sentence\ Connective\ Sentence \\
&\qquad\qquad \mid Quantifier\ Variable\ Sentence \\
&\qquad\qquad \mid \neg Sentence \\
&\qquad\qquad \mid (Sentence)
\end{aligned}
$$

First-Order Logic

Syntax

$Connective \Rightarrow \ \wedge \mid \ \vee \ \mid \ \leftrightarrow$

$Quantifier \Rightarrow \ \forall \mid \exists$

$Constant \Rightarrow \ \text{a} \mid \text{john} \mid \ \ldots$

$Variable \Rightarrow \ x \mid s \mid \ \ldots$

$Predicate \Rightarrow \ \text{before} \mid \text{hascolor} \mid \ \ldots$

$Function \Rightarrow \ \text{mother} \mid \text{leftlegof} \mid \ \ldots$

Logic

- **Entailment**

  $$\mathbf{KB} \models \alpha$$

- **Derivability**

  $$\mathbf{KB} \vdash \alpha$$

- **Sound**
- **Complete**

First-Order Theorem Proving

Substitutions

EXAMPLE

$\{x/\mathrm{a}, y/\mathrm{f(b)}, z/w\}$

$\mathrm{p}(x, x, y, v)$

$\mathrm{p}(\mathrm{a}, \mathrm{a}, \mathrm{f(b)}, v)$

First-Order Theorem Proving

Substitutions

A set of expressions $\alpha_1 \ldots \alpha_n$ are unifiable if and only if there is a substitution σ that makes the expressions identical.

Given two substitutions τ, σ, the composition of τ and σ ($\sigma\tau$) is the substitution θ that gives the same result as applying σ and then τ to an arbitrary expression.

$$(\beta\sigma)\tau = \beta\theta$$

First-Order Theorem Proving

Unification

EXAMPLE

$\mathrm{p}(\mathrm{a}, y, z)$

and

$\mathrm{p}(x, \mathrm{b}, z)$

are unifiable with the substitution

$\{x/\mathrm{a}, y/\mathrm{b}, z/\mathrm{c}\}$

to yield

$\mathrm{p}(\mathrm{a}, \mathrm{b}, \mathrm{c})$

Most General Unifier

But the Most General Unifier (MGU) makes the least commitment.

$\{x/\mathrm{a}, y/\mathrm{b}\}$

$\mathrm{p}(\mathrm{a}, \mathrm{b}, z)$

$UNIFY(\alpha, \beta)$ returns the MGU of α and β.

Theorem Proving

Notation

$$\text{Clause} \stackrel{\text{def}}{=} \text{Set of Literals}$$

$$\text{Literal} \stackrel{\text{def}}{=} \begin{array}{l}\text{Atomic Sentence} \\ \text{or its negation}\end{array}$$

$\{P, \neg P, Q$

$\{P, \neg P\}$

$\{R\}$

$\{S\}$

1'st order logic:

$\{\neg P(x), Q(x, y)\}$

$\{Q(a, f(a))\}$

Conversion to clausal form eliminates quantifiers.

Theorem Proving

Resolution Rule of Inference

$$\frac{\alpha \vee \beta, \neg\beta \vee \delta}{\alpha \vee \delta}$$

- - - - - - - - - - - - - - - - -

Set Notation

Γ with $\alpha \in \Gamma$

Δ with $\neg\alpha \in \Delta$

$$(\Gamma - \alpha) \cup (\Delta - \neg\alpha)$$

Theorem Proving

The Procedure

$$\Delta \stackrel{?}{\models} \alpha$$

$$\Delta \cup \neg\alpha \stackrel{?}{\not\models}$$

1. Clausify $(\Delta \cup \neg\alpha)$
2. Repeatedly try to produce $\Box$ by performing resolution and adding the resolvent to the set of clauses.

Theorem Proving

Resolution Rule of Inference

$$\frac{\alpha \vee \beta_1, \neg\beta_2 \vee \delta}{(\alpha \vee \delta)\theta}$$

where θ = mgu of β_1 and β_2.

- - - - - - - - - - - - - - - - -

Set Notation

Γ with $\beta_1 \in \Gamma$

Δ with $\neg\beta_2 \in \Delta$

$$(\Gamma - \beta_1) \cup (\Delta - \neg\beta_2)\theta$$

Theorem Proving

The Procedure

1. CLAUSES $\leftarrow$ Clausify $(\Delta \cup \neg\alpha)$
2. Repeat
   - Pick two clauses in CLAUSES, c_1, c_j, such that c_i and c_j have a resolvent r_{ij} not already in CLAUSES.
   - If no such c_i, c_j exist then return $(\Delta \not\models \alpha)$.
   - If $r_{ij} = \Box$, then return $(\Delta \models \alpha)$.
   - Otherwize add r_{ij} to CLAUSES.

Example

Consider the following KB:

$\forall x\, man(x) \rightarrow mortal(x)$
$man(socrates)$

We wish to show:

$KB \models mortal(socrates)$

We do this by showing:

$KB \cup \neg mortal(socrates) \not\models$

The refutational proof is as follows:

1) $\{\neg man(x), mortal(x)\}$ given
2) $\{man(socrates)\}$ given
3) $\{\neg mortal(socrates)\}$ negation of hypothesis
4) $\{mortal(socrates)\}$ resolving 1,2
5) $\{\}$ resolving 3,4

Answer Literals

"Green's Trick"

$$Ans(v_1 \ldots v_n) \vee \neg\phi$$

where ϕ is the goal with the free variables $v_1 \ldots v_n$.

Instead of searching for $\Box$, search for $\{Ans(\vec{x})\}$.

Example

1) Art is the father of John
2) Bob is the father of Kim
3) Fathers are Parents

Query: Who is the parent of John?

$$\{\neg Parent(x, John), Ans(x)\}$$

Example (Continued)

1) $\{F(Art, John)\}$
2) $\{F(Bob, Kim)\}$
3) $\{\neg F(x, y), P(x, y)\}$
4) $\{\neg P(z, John), Ans(z)\}$
5) $\{P(Art, John)\}$ from 1,3
6) $\{Ans(Art)\}$ from 5,4

Readings

- *Propositional, First-Order and Higher-Order Logics:Basic Definitions, Rules of Inference, Examples* by Stuart C. Shapiro. From Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language edited by Lucja Iwanska and Stuart C. Shapiro. Available from

  `http://www.cse.buffalo.edu/faculty/shapiro`

  Provides a good overview of the logic background needed for this material.

- *Logical Foundations of Artificial Intelligence* by Michael R. Genesereth and Nils J. Nilsson. Chapters 1–3 provide a good introduction to logic. Chapter 4 covers resolution theorem proving including conversion to clausal form which was not covered in the lectures here.

Readings (cont)

See Also:

- *Artificial Intelligence: A Modern Approach* by Stuart J. Russell and Peter Norvig. Chapters 6, 7, 8, 9, and 10.
- *Computational Intelligence: A Logical Approach* by David Poole, Alan Mackworth, and Randy Goebel.

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Situation Calculus
  1. Fluents
  2. Effect Axioms
  3. Frame Problem
  4. Successor State Axioms
  5. Database Example

Reference

Chapter 3 from *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems* by Ray Reiter.

Review

We want to a uniform theoretical and implementation framework that integrates the reasoning, perception, and action of a robot.

We Adopt the K.R. Hypothesis – using 1'st order logic as our representation language.

a. Sentences in 1'st order logic will represent the knowledge of the agent about the world, actions, and effects.

b. These sentences play a causal role in engendering the behavior of the agent.

This is not only explicit knowledge, but also implicit knowledge.

$$\{\alpha | KB \models \alpha\}$$

Problem: In our discussion last time of 1'st order logic, can only represent knowledge at one point in time. How do we talk about change within logic?

Action Theories: Motivation

To perform non-trivial reasoning (like planning) an intelligent agent situated in a changing world needs:

- The knowledge of causal laws describing the actions or events that change the world.
- The ability to observe and record occurrences of these actions/events.
- A model of the state of the world.

Action theories are formalisms intended to represent and reason about actions and their effects.

The Situation Calculus (McCarthy)

- The world is conceived as being in some state s; this state can change only in consequence of some agent performing an action.
- Often the constant S_0 is used to denote the initial situation.
- $do(\alpha, s)$ denotes the successor state to s resulting from performing the action α.
- Actions may be parameterized:
  $\textsc{put}(x, y)$ might stand for the action of putting object x on object y;
  $do(\textsc{put}(\textsc{block}_1, \textsc{block}_2), s)$ denotes that state resulting from placing $\textsc{block}_1$ on $\textsc{block}_2$ when the world is in state s.

Axiomatization of Initial Situation

$$\neg \textsc{Holding}(\textsc{robot}, \textsc{obj}_1, S_0)$$
$$\textsc{location}(\textsc{robot}, S_0) = 0$$
$$\textsc{location}(\textsc{obj}_1, S_0) = 1$$
$$\textsc{location}(\textsc{obj}_2, S_0) = 2$$
$$\textsc{Color}(\textsc{obj}_1, \textsc{red}, S_0)$$

Can also say:

$$\textsc{Holding}(\textsc{robot}, \textsc{obj}_1, S_0) \vee$$
$$\textsc{Holding}(\textsc{robot}, \textsc{obj}_2, S_0)$$

The Situation Calculus (Continued)

- *Fluents* = those relations whose truth values may vary from state to state.
  - Denoted by predicate symbols taking a state term as one of their arguments.
  - In a world in which it is possible to paint objects, we might have a fluent $\textsc{Color}(x, c, s)$, meaning that the color of object x is c when the world is in state s.
  - Functional Fluents $\textsc{location}(x, s)$.

Preconditions of Actions

- Actions have *preconditions*: necessary conditions which a world state must satisfy if the action can be performed in this state.
  - If it is possible for a robot r to pick up an object x in the world state s then the robot is not holding any object, it is next to x, and x is not heavy:

$$Poss(\textsc{pickup}(r, x), s) \rightarrow [(\forall z)\neg \textsc{Holding}(r, z, s)] \wedge \neg \textsc{Heavy}(x) \wedge \textsc{Nexto}(r, x, s).$$

  - Whenever it is possible for a robot to repair an object, then the object must be broken, and there must be glue available:

$$Poss(\textsc{Repair}(r, x), s) \rightarrow \textsc{Hasglue}(r, s) \wedge \textsc{Broken}(x, s).$$

Effects of Actions

- World dynamics are specified by *effect axioms* which specify the effect of a given action on the truth value of a given fluent.
  - The effect on the fluent BROKEN of a robot dropping an object:

$$Poss(\text{DROP}(r, x), s) \wedge \text{FRAGILE}(x) \rightarrow \text{BROKEN}(x, do(\text{DROP}(r, x), s)).$$

  - A robot repairing an object causes it not to be broken:

$$Poss(\text{REPAIR}(r, x), s) \rightarrow \neg\text{BROKEN}(x, do(\text{REPAIR}(r, x), s)).$$

Effects of Actions (cont)

- $KB \models \textsc{Broken}(\textsc{obj}_1, do(\textsc{drop}(r, \textsc{obj}_1), S_0))$
- $KB \not\models \textsc{Color}(\textsc{obj}_1, \textsc{red}, do(\textsc{repair}(r, \textsc{obj}_1), S_0))$

The Qualification Problem

- What needs to be true for a particular action to be possible.
- Need to axiomatize when a *pickup* is possible.

But many other factors however unlikely could prevent the pickup action from being possible.

$\neg$GLUEDTOFLOOR(x, s)
$\neg$ARMSTIED(r, s)
$\neg$HITBYTRUCK(r, s)
$\vdots$

As long as

$$\forall z \neg \text{HOLDING}(r, z, s) \wedge \neg \text{HEAVY}(\text{OBJ}_1) \wedge \text{NEXTTO}(r, \text{OBJ}_1, s)$$

the robot can execute the pickup actions, assuming we don't know that any of the other minor qualifications are true.

Nonmonotonic Reasoning

$\textsc{Bird}(\textsc{tweety}) \vdash \textsc{Flies}(\textsc{tweety})$

but

$\textsc{Bird}(\textsc{tweety}) \wedge \textsc{Penguin}(\textsc{tweety}) \vdash$
$\neg\textsc{Flies}(\textsc{tweety})$

$\textsc{Bird}(x) \wedge \neg\textsc{Penguin}(x) \wedge$
$\neg\textsc{Ostrich}(x) \ldots \rightarrow \textsc{Flies}(x)$

Qualification Problem (cont)

Here we will ignore the minor qualifications/exceptions.

- One approach: Assume that for each action $A(\vec{x})$, we have an axiom of the form

$$Poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s),$$

  where $\Pi_A(\vec{x}, s)$ is a first order formula with free variables $\vec{x}, s$ which does not mention *do*. We shall call these *action precondition axioms*.

- *Example*:

$$Poss(\textsc{pickup}(r, x), s) \leftrightarrow$$
$$[(\forall z)\neg\textsc{Holding}(r, z, s)] \wedge \neg\textsc{Heavy}(x) \wedge$$
$$\textsc{Nexto}(r, x, s).$$

The Frame Problem (McCarthy and Hayes)

- Axioms other than effect axioms are required for formalizing dynamic worlds. These are called *frame axioms*, and they specify the action *invariants* of the domain, i.e., those fluents unaffected by the performance of an action.
  - A positive frame axiom – dropping things does not affect an object's color:

$$Poss(\textsc{drop}(r, x), s) \wedge \textsc{color}(y, c, s) \rightarrow color(y, c, do(\textsc{drop}(r, x), s)).$$

  - A negative frame axiom – not breaking things:

$$Poss(\textsc{drop}(r, x), s) \wedge \neg\textsc{Broken}(y, s) \wedge [y \neq x \vee \neg\textsc{Fragile}(y)] \rightarrow \neg\textsc{Broken}(y, do(\textsc{drop}(r, x), s)).$$

The Frame Problem (Continued)

- Problem: Vast number of frame axioms; only relatively few actions will affect the truth value of a given fluent. All other actions leave the fluent invariant.
- $\sim 2 \times \mathcal{A} \times \mathcal{F}$ frame axioms.
- The *frame problem*:
  - Axiomatizer must think of all these quadratically many frame axioms.
  - System must reason efficiently in the presence of so many axioms.

the Frame Problem (continued)

A General Solution demands nonmonotonic reasoning. Generally *circumscription* is used. The solution that we discuss here works under certain simplifying assumptions:

- No Ramifications – state constraints.

$$\forall s \mathrm{P}(s) \leftrightarrow \mathrm{Q}(s)$$

- Only deterministic actions.

A Simple Solution to the FP (Reiter, Hass, Schubert)

- **Example**: Suppose there are two positive effect axioms for the fluent *broken*:

$$Poss(\textsc{drop}(r, x), s) \wedge y = x \wedge \textsc{Fragile}(y) \rightarrow \textsc{Broken}(y, do(\textsc{drop}(r, x), s)),$$

$$Poss(\textsc{explode}(b), s) \wedge \textsc{Nexto}(b, y, s) \rightarrow \textsc{Broken}(y, do(\textsc{Explode}(b), s)).$$

- And one negative effect axiom:

$$Poss(\textsc{repair}(r, x), s) \wedge y = x \rightarrow \neg\textsc{Broken}(y, do(\textsc{Repair}(r, x), s)).$$

- Now appeal to the following completeness assumption:
  The above axioms characterize all the conditions under which action a leads to y being broken.

Successor State Axioms

- For the above example, the successor state axiom for BROKEN is:

$$Poss(a, s) \rightarrow [\textsc{Broken}(y, do(a, s)) \leftrightarrow$$
$$(\exists r)\{a = \textsc{drop}(r, y) \wedge \textsc{Fragile}(y)\} \vee$$
$$(\exists b)\{a = \textsc{explode}(b) \wedge \textsc{Nexto}(b, y, s)\} \vee$$
$$\textsc{Broken}(y, s) \wedge \neg(\exists r)a = \textsc{repair}(r, y)].$$

The Solution

- Reiter's solution yields the following axioms:
  1. Successor state axioms: for each fluent R,
     $$Poss(a, s) \rightarrow [R(do(a, s)) \leftrightarrow \gamma_R^+(a, s) \vee R(s) \wedge \neg\gamma_R^-(a, s)].$$
  2. For each action A, a single action precondition axiom of the form:
     $$Poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s).$$
  3. Unique names axioms for actions.
- Ignoring the unique names axioms (whose effects can be compiled), this axiomatization requires $\mathcal{F} + \mathcal{A}$ axioms in total, compared with the $2 \times \mathcal{A} \times \mathcal{F}$ explicit frame axioms that would otherwise be required.

Planning

Find a sequence of actions that if performed in a world with an aximatized intitial situation, will lead to a situation in which some goal statement will be true.

$do(\text{PUTDOWN}, do(\text{MOVE}, do(\text{PICKUP}, S_0)))$
[PICKUP, MOVE, PUTDOWN]
$KB \models \exists s G(s)$
$KB \models G(s)\{s/do(\ldots)\}$

Can Use Answer Extraction and Resolution.

$$KB \cup \{\neg \text{G}(s) \vee \text{ANS}(s)\}$$

Shakey Project (SRI), Abandoned resolution theorem proving for planning and developed STRIPS planning.

Projection Problem

Does a particular sentence hold in the situation resulting from the execution of a particular sequence of actions?

One could use resolution, but note that the successor state axioms contain equivalences which when converted to clause form would create a very large space.

Can we avoid using a general purpose reasoning mechanism with the successor state axioms.

Regression

- Regression operators transform something of the form

$$G(do(A_1, do(\ldots, S_0)\ldots)$$

to

$$R(S_0)$$

- **Theorem**

$$\mathcal{F} \models G(do(A_1, do(\ldots, S_0)\ldots)$$
$$iff$$
$$\mathcal{F} - \mathcal{F}_{ss} \models R(S_0)$$

An Education Database

- **Relations**
  1. ENROLLED($st, course, s$): st is enrolled in $course$ when the database is in state s.
  2. GRADE($st, course, grade, s$): The grade of st in $course$ is $grade$ when the database is in state s.
  3. PREREQU($pre, course$): pre is a prerequisite course for $course$.

Education Database (Continued)

- **Initial Database State**
  These will be arbitrary first order sentences,
  the only restriction being that fluents
  mention only the initial state S_0.

$$\textsc{Enrolled}(\textsc{sue}, \textsc{c}100, \mathrm{S}_0) \vee \textsc{Enrolled}(\textsc{sue}, \textsc{c}200, \mathrm{S}_0),$$

$$(\exists c)\textsc{Enrolled}(\textsc{bill}, c, \mathrm{S}_0),$$

$$(\forall p).\textsc{Prerequ}(p, \textsc{p}300) \leftrightarrow p = \textsc{p}100 \vee p = \textsc{m}100,$$

$$(\forall p)\neg\textsc{Prerequ}(p, \textsc{c}100),$$

$$(\forall c).\textsc{Enrolled}(\textsc{bill}, c, \mathrm{S}_0) \leftrightarrow$$
$$c = \textsc{m}100 \vee c = \textsc{c}100 \vee c = \textsc{p}200,$$

$$\textsc{Enrolled}(\textsc{mary}, \textsc{c}100, \mathrm{S}_0),$$

$$\neg\textsc{Enrolled}(\textsc{john}, \textsc{m}200, \mathrm{S}_0), \ldots$$

$$\textsc{Grade}(\textsc{sue}, \textsc{p}300, 75, \mathrm{S}_0),$$

$$\textsc{Grade}(\textsc{bill}, \textsc{m}200, 70, \mathrm{S}_0), \ldots$$

Example: Continued

- **Database Transactions**
  - Denote by function symbols.
  - Treat exactly like actions in situation calculus planning.
- For the example, there are three transactions:
  1. REGISTER(st, c),
  2. CHANGE(st, c, g),
  3. DROP(st, c).

Example: Continued

- Registration Prerequisites:

$$Poss(\text{REGISTER}(st, c), s) \leftrightarrow \{(\forall p).\text{PREREQU}(p, c) \rightarrow (\exists g).\text{GRADE}(st, p, g, s) \wedge g \geq 50\}.$$

- It is possible to change a student's grade iff he has a grade which is different than the new grade:

$$Poss(\text{CHANGE}(st, c, g), s) \leftrightarrow (\exists g').\text{GRADE}(st, c, g', s) \wedge g' \neq g.$$

- A student may drop a course iff the student is currently enrolled in that course:

$$Poss(\text{DROP}(st, c), s) \leftrightarrow \text{ENROLLED}(st, c, s).$$

- **Update Specifications**

$$Poss(a, s) \rightarrow [\text{ENROLLED}(st, c, do(a, s)) \leftrightarrow a = \text{REGISTER}(st, c) \vee \text{ENROLLED}(st, c, s) \wedge a \neq \text{DROP}(st, c)],$$

Example: Continued

$$Poss(a, s) \rightarrow [\textsc{Grade}(st, c, g, do(a, s)) \leftrightarrow$$
$$a = \textsc{change}(st, c, g) \vee$$
$$\textsc{Grade}(st, c, g, s) \wedge (\forall g')a \neq \textsc{change}(st, c, g')]$$

Projection Problem

$$KB \models \exists c\, \textsc{Enrolled}(\textsc{john}, c, \\ do(\textsc{register}(\textsc{mary}, \textsc{c}100), \\ do(\textsc{drop}(\textsc{john}, \textsc{c}100), S_0))))$$

Is John erolled in any courses after the sequence of actions
$[\textsc{drop}(\textsc{john}, c100), \textsc{register}(\textsc{mary}, \textsc{c}100)]$
has been executed?

References

J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969.

R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

R. Reiter. On specifying database updates. *Journal of Logic Programming.* 25:25-91, 1995

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Event Calculus
  - Ontology and Predicates of the Event Calculus
  - Simple Axioms
  - Frame Problem
  - Full Axioms
  - Examples

Sources

- R. A. Kowalski and M.J. Sergot, A Logic-Based Calculus of Events, in *New Generation Computing*, vol 4 (1986), pp. 67-95.
- M.P. Shanahan, The Event Calculus Explained, in *Artificial Intelligence Today*, ed. by M.J. Woolridge and M. Veloso, Springer Lecture Notes in Artificial Intelligence no. 1600, Springer(1999), pages 409-430. Available from `http://casbah.ee.ic.ac.uk/ mpsha/pubs.html`
- *The Event-Calculus in Classical Logic – Alternative Axiomatizations* by Rob Miller and Murray Shanahan.
- M.P. Shanahan, *Solving the Frame Problem*, MIT Press, 1997.

Event Calculus vs. Situation Calculus

Narrative: a course of actual events about which we may have incomplete information.

The Event Calculus is a narrative based formalism.

It does not have states, situations, *do* function.

It has an explicit representation of time, independent of action occurrences.

The Situation Calculus is not.

- All events are essentially hypothetical.
- Does not naturally represent incompletely specified sequences.

Narratives

Before breakfast one morning, Mary checks her briefcase to make sure her lecture notes are inside, which indeed they are. She eats her breakfast, and then carries her briefcase to college. Mary knows that if she carries her briefcase from A to B then its contents will also be carried from A to B. So she concludes that, at the end of this sequence of events, her lecture notes are at college along with her briefcase. However, shortly after she sits down at her desk, her husband telephones to apologise for accidentally removing her notes from the briefcase before she left for work. With her new knowledge of her husband's actions, Mary can no longer conclude that her notes are at college. (Shanahan '97)

The Ontology and Predicates of the Event Calculus

- Initiates(α, β, τ) Fluent β starts to hold after action α at time τ.
- Terminates(α, β, τ) Fluent β ceases to hold after action α at time τ.
- InitiallyP(β) Fluent β holds from time 0.
- $\tau_1 < \tau_2$ Time point τ_1 is before time point τ_2.
- Happens(α, τ) Action α occurs at time τ.
- HoldsAt(β, τ) Fluent β holds at time τ.
- Clipped(τ_1, β, τ_2) Fluent β is terminated between times τ_1 and τ_2.

Representational Decisions

Fluents are reified terms. They can be quantified over.

The types of things over which we can quantify are actions (events), time points, and fluents. They are in the domain of the models of 1'st order logic.

Axioms of the Simple Event Calculus

SC1

$$\forall f, t\, \mathrm{HoldsAt}(\mathrm{f}, \mathrm{t}) \leftarrow$$
$$\mathrm{InitiallyP}(\mathrm{f}) \wedge \neg \mathrm{Clipped}(0, \mathrm{f}, \mathrm{t})$$

SC2

$$\forall f, a, t_1, t, 2\, \mathrm{HoldsAt}(\mathrm{f}, \mathrm{t}_2) \leftarrow$$
$$\mathrm{Happens}(\mathrm{a}, \mathrm{t1}) \wedge \mathrm{Initiates}(\mathrm{a}, \mathrm{f}, \mathrm{t}_1) \wedge$$
$$t_1 < t_2 \wedge \neg \mathrm{Clipped}(\mathrm{t}_1, \mathrm{f}, \mathrm{t}_2)$$

SC3

$$\forall f, t_1, t_2\, \mathrm{Clipped}(\mathrm{t}_1, \mathrm{f}, \mathrm{t}_2) \leftrightarrow$$
$$\exists a, t[\mathrm{Happens}(\mathrm{a}, \mathrm{t}) \wedge$$
$$t_1 < t < t_2 \wedge \mathrm{Terminates}(\mathrm{a}, \mathrm{f}, \mathrm{t})$$

Yale Shooting Problem

- Actions
  - Load
  - Sneeze
  - Shoot
- Fluents
  - Loaded
  - Alive
  - Dead
- Effects
  - Load makes Loaded hold.
  - Shoot makes Dead hold and Alive not hold as long as Loaded holds.
  - Sneeze has no effects.

Yale Shooting Problem (cont)

Scenario

1. Load
2. Sneeze
3. Shoot

Yale Shooting Problem (cont)

Y1.1 $\text{Initiates}(\text{Load}, \text{Loaded}, \text{t})$

Y1.2 $\text{Initiates}(\text{Shoot}, \text{Dead}, \text{t}) \leftarrow$
$\text{HoldsAt}(\text{Loaded}, \text{t})$

Y1.3 $\text{Terminates}(\text{Shoot}, \text{Alive}, \text{t}) \leftarrow$
$\text{HoldsAt}(\text{Loaded}, \text{t})$

Yale Shooting Problem

Y2.1 InitiallyP(Alive, t)

Y2.2 Happens(Load, T1)

Y2.3 Happens(Sneeze, T2)

Y2.4 Happens(Shoot, T3)

Y2.5 $T1 < T2$

Y2.6 $T2 < T3$

Y2.7 $T3 < T4$

Yale Shooting Problem (cont)

We want:

$$\Sigma \wedge \Delta \wedge SC \models \text{HoldsAt}(\text{Dead}, \text{T4})$$

Frame Problem

Unique Name Axioms

Y3.1 UNA[Load, Sneeze, Shoot]

Y3.2 UNA[Loaded, Alive, Dead]

Predicate Completion

Y4.1 [replaces Y1.1]

$$\text{Initiates}(a, f, t) \leftrightarrow [a = \text{Load} \wedge f = \text{Loaded}] \vee$$
$$[a = \text{Shoot} \wedge f = \text{Dead} \wedge \text{HoldsAt}(\text{Loaded}, t)]$$

Y4.2 [replaces Y1.2] $\text{Terminates}(a, f, t) \leftrightarrow a = \text{Shoot} \wedge f = \text{Dead} \wedge \text{HoldsAt}(\text{Loaded}, t)$

Yale Shooting Problem(Cont !)

Y5.1 (Y2.1) InitiallyP(Alive, t)

Y5.2 (replaces Y2.2, Y2.3, Y2.4)
Happens(a, t) $\leftrightarrow$ [a = Load $\wedge$ t = T1] $\vee$ [a = Sneeze $\wedge$ t = T2] $\vee$ [a = Shoot $\wedge$ t = T3]

Y5.3 (Y2.5) $T1 < T2$

Y5.4 (Y2.6) $T2 < T4$

Y5.5 (Y2.7) $T3 < T4$

$$\Sigma \wedge \Delta \wedge SC \models \text{HoldsAt(Dead, T4)}$$

The Full Event Calculus

New Predicates

- Releases(α, β, τ) Fluent β is not subject to inertia after action α at time τ.
- InitiallyN(β) Fluent β does not holds from time 0.
- Happens(α, τ_1, τ_2) Action α starts at time τ_1 and ends at time τ_2.
- Declipped(τ_1, β, τ_2) Fluent β is initiated between times τ_1 and τ_2.

New Axioms

EC1 $\text{HoldsAt}(f, t) \leftarrow$
$\text{InitiallyP}(f) \wedge \neg\text{Clipped}(0, f, t)$

EC2 $\text{HoldsAt}(f, t3) \leftarrow \text{Happens}(a, t1, t2) \wedge$
$\text{Initiates}(a, f, t1) \wedge t2 < t3 \wedge \neg\text{Clipped}(t1, f, t3)$

EC3 $\text{Clipped}(t1, f, t4) \leftrightarrow$
$\exists a, t2, t3[\text{Happens}(a, t2, t3) \wedge t1 < t3 \wedge t2 <$
$t4 \wedge [\text{Terminates}(a, f, t2) \vee \text{Releases}(a, f, t2)]]$

EC4 $\neg\text{HoldsAt}(f, t) \leftarrow$
$\text{InitiallyN}(f) \wedge \neg\text{Declipped}(0, f, t)$

EC5 $\neg\text{HoldsAt}(f, t3) \leftarrow$
$\text{Happens}(a, t1, t2) \wedge \text{Terminates}(a, f, t1) \wedge t2 <$
$t3 \wedge \neg\text{Declipped}(t1, f, t3)$

EC6 $\text{Declipped}(t1, f, t4) \leftrightarrow$
$\exists a, t2, t3[\text{Happens}(a, t2, t3) \wedge t1 < t3 \wedge t2 <$
$t4 \wedge [\text{Initiates}(a, f, t2) \vee \text{Releases}(a, f, t2)$

EC7 $\text{Happens}(a, t1, t2) \rightarrow t1 \leq t2$

Frame Problem Solutions

Both the Situation Calculus and the Event Calculus use predicate completion to incorporate a default completeness assumption.

Completion can be shown correct with respect to *circumscription* which is used to formalize the common sense law of inertia.

Normally, given any action (event type) and any fluent, the action doesn't affect the fluent. Circumscription minimizes the changes from state (time) to state (time).

Reasoning

$$KB \models HoldsAt(\ldots)$$

The basic task is to determine whether the above formula holds for various arguments to $HoldsAt$.

The KB includes terminates and initiates formulas along with temporal ordering formulas, formulas constructed out of $Happens$ amd $Initially_P$.

Planning

$\Delta \wedge \Sigma \wedge SC \models HoldsAt(G, t)$

where Δ contains statements involving *Happens*, *Ordering*, and *Initially*.

The Planning task is to add statements to Δ such that the above holds.

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Knowledge-Producing Actions
- Representing Knowledge
- The Frame Problem with Knowledge
- Reasoning

Sensing Actions

- Generally, agents do not have complete knowledge of the world.
  - formalism must distinguish between what is true in the world and what the agent knows
- Agents must reason about:
  - actions that produce knowledge
    - perception
    - reading
    - communicative acts
  - the knowledge prerequisites of actions

Knowledge and Action

McCarthy and Hayes 1969, McCarthy 1963, Moore 1980, Moore, 1985

- If John is at the same place as SF_1 and he *Knows* the combination of the safe, he can open the safe by dialing the combination.
- If John is at the same place as both SF_1 and the piece of paper PPR_1 and he *knows* that the combination of SF_1 is the number written on PPR_1, he can open SF_1 by *reading* the piece of paper and dialing the combination.
- If C_1 is the combination of SF_1, and if John tries to open SF_1 by dialing C_1, he will then *know* that C_1 is the combination of SF_1.

Goal

- Account of Knowledge of Agent
- Knowledge-Producing Actions (Sensing)
- Ordinary Actions
- Solution to Frame Problem
- Reasoning Methods

Omelet Problem

The problem is to make a 3 egg omelet from a set of eggs some of which may be bad. None of the eggs in the omelet should be bad. We have two bowls; we can only see if an egg is bad if it is in a bowl. We can throw out the whole bowl.

We can can assume a limited number of eggs (say 5), and simplify it even further by adding the statement that there are at least 3 good eggs. (Savage, Poole)

Omelet Problem (cont)

- Actions
  - Break an egg into a bowl.
  - Pour the contents of one bowl into another.
  - Throw out the contents of one bowl.
  - Inspect a bowl to see if there are any bad eggs in it.
- Goal
  - Have three eggs in a bowl that are not bad.

The Axiomatization

- *Actions.*

  $\textsc{break\_into}(bowl)$, $\textsc{fetch}(container)$,
  $\textsc{pour}(bowl1, bowl2)$, $\textsc{throw\_out}(bowl)$

- *Fluents*

  $\textsc{In}(egg, bowl, s)$, $\textsc{Bad}(egg, s)$,
  $\textsc{Broken}(egg, s)$, $\textsc{Holding}(egg, s)$,
  $\textsc{number\_eggs}(bowl, s)$

- *Non-Fluents*

  $\textsc{Egg}(x)$, $\textsc{small\_bowl}$, $\textsc{large\_bowl}$, $\textsc{basket}$

- *Effect Axioms (Causal Laws)*

  $\textsc{Holding}(e, s) \rightarrow$
  $\textsc{In}(e, b, do(\textsc{break\_into}(b), s))$
  $\neg\textsc{In}(e, b_1, do(\textsc{pour}(b_1, b_2), s))$

The Axiomatization (Continued)

- For each action A, an *action precondition axiom* is needed.

$$\textsc{Poss}(\textsc{break\_into}(bowl), s) \leftrightarrow \exists egg \; \neg\textsc{Broken}(egg, s) \wedge \textsc{Holding}(egg, s)$$

$$\textsc{Poss}(\textsc{pour}(b_1, b_2), s) \leftrightarrow \neg \exists e \; \textsc{Holding}(e, s) \wedge \textsc{number\_eggs}(b_1, s) \geq 0$$

$$\textsc{Poss}(\textsc{fetch}(e, conn), s) \leftrightarrow \textsc{In}(e, con, s) \wedge \neg\exists e \textsc{Holding}(e, s)$$

- An axiomatization of S_0, the initial state.

$$\neg\textsc{Broken}(\textsc{egg1}, S_0) \qquad \textsc{Egg}(\textsc{egg1})$$

Solving the Frame Problem (Continued)

- For all fluents F, a *successor state axiom* is needed.

$$\textsc{Poss}(a, s) \rightarrow [\textsc{Broken}(e, do(a, s)) \leftrightarrow$$
$$(\textsc{holding}(e, s) \wedge \exists b\, a = \textsc{break\_into}(b)) \vee$$
$$\textsc{Broken}(e, s)]$$

$$\textsc{Poss}(a, s) \rightarrow [\textsc{In}(e, b_1, do(a, s)) \leftrightarrow$$
$$(\textsc{holding}(e, s) \wedge a = \textsc{break\_into}(b_1)) \vee$$
$$(\exists\, b_2\ a = \textsc{pour}(b2, b) \wedge \textsc{In}(e, b_2, s) \vee$$
$$\textsc{In}(e, b_1, s) \wedge \neg((a = \textsc{throw\_out}(b) \vee$$
$$\exists\, b2\, a = \textsc{pour}(b, b2))]$$

An Epistemic Fluent

- We (following Moore) adapt the standard possible-world model of knowledge to the situation calculus.
- We introduce a binary relation $K(s', s)$, read as "s' is accessible from s".
- So something is known in s if it is true in every s' accessible from s, and conversely something is not known if it is false in some accessible situation.
- We can now introduce the notation $\mathbf{Knows}(P, s)$ as an abbreviation for a formula that uses K.

$$\mathbf{Knows}(\textsc{Broken}(y), s) \stackrel{\text{def}}{=}$$
$$\forall s'\ K(s', s)\ \rightarrow \textsc{Broken}(y, s').$$

Solving the Frame Problem

- For non-knowledge-producing actions (e.g. $\textsc{drop}(x)$), the specification (based on Moore) is as follows:

$$\forall s, s'',\ K(s'', do(\textsc{drop}(x), s)) \leftrightarrow$$
$$\exists s' \left(K(s', s) \wedge \textsc{Poss}(\textsc{drop}(x), s') \wedge \right.$$
$$\left. (s'' = do(\textsc{drop}(x), s'))\right)$$

- The only change in knowledge that occurs in moving from s to $do(\textsc{drop}(x), s)$ is the knowledge that the action $\textsc{drop}(x)$ has been performed.

Solving the Frame Problem

$$\forall s, s'',\ K(s'', do(\text{DROP}(x), s)) \leftrightarrow$$
$$\exists s' \left(K(s', s) \wedge \text{POSS}(\text{DROP}(x), s') \wedge \right.$$
$$\left. (s'' = do(\text{DROP}(x), s'))\right)$$

Solving the Frame Problem

- Now consider the simple case of a knowledge-producing action (e.g. $\textsc{inspect}(bowl)$) that determines whether or not there is a bad egg in the bowl.

$$\textsc{Poss}(\textsc{inspect}(b), s) \leftrightarrow$$
$$\exists e\, \textsc{Egg}(e) \wedge \textsc{In}(e, b, s) \wedge$$
$$\textsc{Broken}(e)$$

$$\textsc{Poss}(\textsc{inspect}(b), s) \rightarrow$$
$$[K(s'', do(\textsc{inspect}(b), s)) \leftrightarrow \exists s'(K(s', s) \wedge$$
$$(s'' = do(\textsc{inspect}(b), s')) \wedge \textsc{Poss}(\textsc{inspect}(b))$$
$$\wedge (\textsc{Bad}(b, s) \leftrightarrow \textsc{Bad}(b, s'))))]$$

- The idea here (based on Moore) is that in moving from s to $do(\textsc{inspect}(b), s)$, the agent not only knows that the action $\textsc{inspect}(b)$ has been performed (as before), but also the truth value of the predicate $\textsc{Bad}(b)$.

Solving the Frame Problem

$$\forall s, s'',\ K(s'', do(\textsc{inspect}(bowl), s)) \leftrightarrow$$
$$[\exists s' (K(s', s) \wedge \textsc{Poss}(\textsc{inspect}(bowl))$$
$$\wedge (s'' = do(\textsc{inspect}(bowl), s')))$$
$$\wedge (\textsc{Bad}(bowl, s) \leftrightarrow \textsc{Bad}(bad, s'))]$$

The Solution

- In general, there may be many knowledge-producing actions. For each sensing action a, we enter an axiom of the following form, where SF stands for *sensed fluent*:

$$\mathrm{SF}(a, s) \leftrightarrow \varphi_i(s)$$

  For the INSPECT example, the formula would be:

$$\mathrm{SF}(\textsc{inspect}, s) \leftrightarrow \textsc{Bad}(e, s)$$

$$\mathrm{SF}(\textsc{drop}, s) \leftrightarrow T$$

- **Successor State Axiom for K**

$$\textsc{Poss}(a, s) \rightarrow [\mathrm{K}(s'', do(a, s)) \leftrightarrow$$
$$\exists s' \, (\textsc{Poss}(a, s') \wedge \mathrm{K}(s', s) \wedge (s'' = \textsc{do}(a, s'))) \wedge$$
$$\mathrm{SF}(a, s') \leftrightarrow \mathrm{SF}(a, s)]$$

Properties of the Solution

- After performing an action, agents always know that the action has been performed.
- Agents know the effects of actions.
- The solution to the frame problem for ordinary actions is maintained.
- Knowledge only changes as appropriate.
- Memory.

Reasoning

- [FETCH, BREAK_INTO, INSPECT]
- If starting in the initial situation, first an egg is fetched from the basket, then it is broken into the small bowl, and then the bowl is inspected will it be known whether or not the egg in the bowl is bad.

$$\mathbf{Knows}(\text{BAD}(\text{SMALL\_BOWL}),\ do(\text{INSPECT}(\text{SMALLBOWL}),\ do(\text{BREAK\_INTO}(\text{SMALLBOWL}),\ do(\text{FETCH}(\text{BASKET}), S_0))))$$

$\vee$

$$\mathbf{Knows}(\neg\text{BAD}(\text{SMALL\_BOWL}),\ do(\text{INSPECT}(\text{SMALLBOWL}),\ do(\text{BREAK\_INTO}(\text{SMALLBOWL}),\ do(\text{FETCH}(\text{BASKET}), S_0))))$$

What about Reading?

A READ_τ action makes known the denotation of a term τ.

$\tau(s) = \tau(s')$

$$\mathbf{Kref}(\tau, s) \stackrel{\text{def}}{=} \exists x \mathbf{Knows}(\tau = x).$$

Regression

- Regression operators transform something of the form

$$G(do(A_1, do(\ldots, S_0)\ldots)$$

to

$$R(S_0)$$

- **Theorem**

$$\mathcal{F} \models G(do(A_1, do(\ldots, S_0)\ldots)$$
$$iff$$
$$\mathcal{F} - \mathcal{F}_{ss} \models R(S_0)$$

- All reasoning about actions is done by regression.
- $R(S_0)$ can be expressed in modal logic.

References

Richard Scherl and Hector Levesque. Knowledge Action and the Frame Problem, 2001, submitted.

Richard Scherl and Hector Levesque. The Frame Problem and Knowledge Producing Actions. pp. 689-695, in Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93). Washington, DC. July 1993. (Menlo Park: AAAI Press/ The MIT Press)

Chapter 11 from Reiter *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.*

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Complex Actions in the Situation Calculus
  1. DO
  2. GoLog
  3. Hypertext Example
  4. Adding Sensing

Complex Actions

- if carInDriveway then drive
  else walk endIf

- while ($\exists$ block) ontable(block)
  do
  removeAblock
  end While

- proc removeAblock
  (π x) [pickup(x); putaway(x)]
  endProc

GOLOG

- $\mathbf{Do}(a, s, s') \stackrel{\text{def}}{=} \text{Poss}(a, s) \wedge s' = do(a, s)$
- **Do** is a macro $\mathbf{Do}(\delta, s, s')$

$$\mathbf{Do}([\text{A}; \text{B}], s, s') \stackrel{\text{def}}{=}$$
$$\exists s^* \, \mathbf{Do}(\text{A}, s, s*) \wedge \mathbf{Do}(\text{B}, s^*, s')$$

- Need notation for complex actions
  - $\delta_1; \delta_2$ — sequences
  - $\delta_1 | \delta_2$ — nondeterministic choice of actions
  - **if** ϕ **then** δ_1 **else** δ_2 — conditionals
  - **while** ϕ **do** δ — while loops
  - $(\Pi x)\delta$ — nondeterministic choice of parameters
  - recursion, procedures
  - tests

Definitions of Complex Actions

1. Tests:

$$Do(p?, s, s') \stackrel{\Delta}{=} p[s] \wedge s = s'.$$

Here, p is a pseudo-fluent expression (not a situation calculus formula) which stands for a formula in the language of the situation calculus, but with all state arguments suppressed. $p[s]$ denotes the situation calculus formula obtained from p by restoring state variable s as the suppressed state argument. *Example:* If p denotes

$$(\forall x).\textsc{Ontable}(x) \wedge \neg \textsc{On}(x, \textsc{a}),$$

then $p[s]$ denotes

$$(\forall x).\textsc{Ontable}(x, s) \wedge \neg \textsc{On}(x, \textsc{a}, s).$$

2. Sequence:

$$Do([a_1; a_2], s, s') \stackrel{\Delta}{=} (\exists s'').Do(a_1, s, s'') \wedge Do(a_2, s'', s').$$

Definitions of Complex Actions

3. Nondeterministic choice of two actions:

   $Do([a_1 \mid a_2], s, s') \stackrel{\Delta}{=} Do(a_1, s, s') \vee Do(a_2, s, s')$.

4. Nondeterministic choice of action parameters:

   $Do((\pi\ x)\delta(x), s, s') \stackrel{\Delta}{=} (\exists x) Do(\delta(x), s, s')$.

5. Conditional actions:

   if p **then** a_1 **else** $a_2 \stackrel{\Delta}{=} [p?; a_1] \mid [\neg p?; a_2]$.

Definitions of Complex Actions

6. While loops:

$$\textbf{while}\ p\ \delta \stackrel{\Delta}{=} [p?;\delta]^*;\neg p?.$$

7. Nondeterministic iteration: Execute a 0 or more times.

$$\begin{aligned} Do(\delta^*, s, s') \stackrel{\Delta}{=} \\ & (\forall P).[(\forall s_1)P(s_1, s_1)] \wedge \\ & [(\forall s_1, s_2, s_3).P(s_1, s_2) \wedge Do(a, s_2, s_3) \rightarrow \\ & \quad P(s_1, s_3)] \\ & \quad \rightarrow P(s, s'). \end{aligned}$$

Examples

- "Walk to car, get in, start it, drive to Helsinki."

  walkToCar; enterCar; startCar; driveTo(Helsinki)

- "Move one of the blocks on the table onto the floor."

  Πb Block(b) $\wedge$ OnTable(b)?;
  pickup(b); putOnFloor(b)

- "Move everything off the table onto the floor."

  while$(\exists x)$OnTable(x)
  do
  (Πx)OnTable(x)?;
  pickup(x); putOnFloor(x)
  endWhile

The Computation Task

$$Axioms \models (\exists s)\mathbf{Do}(\delta, s_0, s)$$

The theorem proving task is to find the possible values for s; possible execution histories.

Example

$$\mathbf{Do}(\delta, s_0, s)$$

$$\delta = (\Pi x)\ \text{A}(x); \phi?; [\text{B}(x)|(\text{C}(x); \psi?)]$$

Procedures

1. For any predicate P of arity $n+2$, taking a pair of situation arguments:

   $$Do(P(t_1,\ldots,t_n),s_1,s_2) \stackrel{\Delta}{=} P(t_1,\ldots,t_n,s_1,s_2).$$

2. (Recursive) procedure P with formal parameters $x_1,\ldots,x_n$ and actual parameters $t_1,\ldots,t_n$ and body α:

   $$Do([\mathbf{proc}\ P(x_1,\ldots,x_n)\ \alpha\ \mathbf{end}](t_1,\ldots,t_n),s_1,s_2) \stackrel{\Delta}{=}$$
   $$(\forall P)\{(\forall x_1,\ldots,x_n,s_1',s_2')[Do(P(x_1,\ldots,x_n),s_1',s_2') \leftrightarrow$$
   $$Do(\alpha,s_1',s_2')]$$
   $$\rightarrow P(t_1,\ldots,t_n,s_1,s_2)\}.$$

Example

Procedure Clean: Put all the blocks into the box

proc CLEAN$(\forall x)(\forall x)$
$[\text{BLOCK}(x) \rightarrow \text{IN}(x, \text{BOX})]? \mid$
$(\Pi x)[(\forall y)\neg\text{ON}(y, x)?;$
$\text{PUT}(x, \text{BOX})]; \text{CLEAN}$
ENDPROC

An Example: Specification of a Hypertext System

1. NODE(x, y, z, s) x is a node with content expression y and semantic type z in situation s.
2. LINK(u, v, w, x, y, z, s) u is a link from source node v to sink node w with semantic type x, operation type y, and either display mode or procedure identifier z in situation s.
3. BUTTON(x, y, z, s) x is a button representing link y with context expression z in situation s.
4. WINDOW(s) denotes that which is displayed in the main window in situation s (a functional fluent).
5. CURRENT_NODE(x, s) x is the current node in situation s.
6. POP_UP_WINDOW(s) denotes the display in the pop_up_window in situation s (a functional fluent).

Hyperdocument

NODE(1,['Pieridae:','This is a large family',' of more than ..... 'peculiar to this family', 'of butterflies. Species:' button(1), button(2), button(3)],DESCRIPTION,S_0)
NODE(2,['Orange Albatross', ... 'Region: Indo-Australian'], DESCRIPTION,S_0)
NODE(3,['Bali is a small island,'..... 'in the arts ',button(4), button(5), 'and religion', button(6)'.'],DESCRIPTION,S_0)
⋮
LINK(1,1,2,MEMBERS_OF,DISPLAY, FULL_WINDOW,S_0)
LINK(2,3,4,MORE_INFORMATION,DISPLAY, FULL_WINDOW,S_0)
⋮
BUTTON(1,1,'Appias nero',S_0)
BUTTON(2,2,'Mylothris chloris',S_0)
CURRENT_NODE(3,S_0)

Actions

1. TRAVERSE_SINK(x, y) moves the current node from x to y.
2. TRAVERSE_SOURCE(x, y) moves the current node from y to x.
3. MAKE_DISPLAY_TEXT(x) displays in a window the text encoded in content expression x.
4. DELETE_NODE(x)
5. CREATE_LINK(u, v, w, x, y, z)
6. DELETE_LINK(u)

Successor State Axioms

$$\begin{aligned}
&\textsc{Poss}(a, s) \rightarrow [\textsc{Current_node}(x, do(a, s)) \leftrightarrow \\
&\quad a = \textsc{traverse_sink}(y, x) \vee \\
&\quad\quad a = \textsc{traverse_source}(x, y) \vee \\
&\quad (\neg(a = \textsc{traverse_sink}(y, x) \vee \\
&\quad\quad a = \textsc{traverse_source}(x, y)) \\
&\quad\quad \wedge \textsc{Current_node}(x, s))]
\end{aligned}$$

$$\begin{aligned}
&\textsc{Poss}(\textsc{traverse_sink}(y, x), s) \leftrightarrow \\
&\quad \textsc{current_node}(y, s) \ \wedge \ \exists u \ \textsc{link}(u, y, x, _, _, _, s)
\end{aligned}$$

$$\begin{aligned}
&\textsc{Poss}(\textsc{traverse_source}(x, y), s) \leftrightarrow \\
&\quad \textsc{current_node}(y, s) \ \wedge \ \exists u \ \textsc{link}(u, x, y, _, _, _, s)
\end{aligned}$$

Run-time Layer

Actions performed by user:

- RIGHTCLICK(x)
- MIDDLECLICK(x)
- LEFTCLICK(x)

Additionally, we need the following action:

- DEACTIVATE(x)

Also needed:

- ACTIVEBUTTONRIGHT(x)
- ACTIVEBUTTONMIDDLE(x)
- ACTIVEBUTTONLEFT(x)

Some GOLOG

proc CONTROL

OPEN_SESSION;

request;

[**while** $\neg$ ACTIVE(0) **do**

FIND_BUTTON;

request];

CLOSE_SESSION

end.

Some GOLOG (Continued)

proc PROCESS_COMMAND(n)

if ACTIVEBUTTONMIDDLE(n)

then $(\pi\, u)$[CURRENT_NODE(u)?;

DISPLAY_NODE_ATTR(u)

else [**if** ACTIVEBUTTONLEFT(n)

then $(\pi\, y)$BUTTON(n, y, z)?;

DISPLAY_LINK_ATTR(y)

else$(\pi\, y)$BUTTON(n, y, z)?;

TRAVERSE(y)]]

end.

proc TRAVERSE(x) $(\pi\ y, z)$[LINK($x, y, z, \_, \_, \_$)?;

if CURRENTNODE(z)

then TRAVERSE_SOURCE(y, z)

else TRAVERSE_SINK(y, z)]

end.

A GOLOG Program for the Omelet Problem

While $\neg(\text{NUMBER\_EGGS}(\text{LARGE\_BOWL}) = 3)$

(Πe) ACHIEVE(HOLDING(e));

BREAK_INTO(SMALL_BOWL);

INSPECT(SMALL_BOWL);

if BAD(SMALL_BOWL)

then THROW_OUT(SMALL_BOWL);

else POUR(SMALL_BOWL, LARGE_BOWL);

Other Issues

Integrating sensing with Golog. See Indigolog

Congolog (Giacomo, Lespérance, Levesque)

`http://www.cs.toronto.edu/cogrobo`
Prolog interpreters available.

References

Richard Scherl, Michael Bieber, and Fabio Vitali. A Situation Calculus model of Hypertext. In the "Logic Modeling" minitrack of the Thirty-First Hawaii International Conference on System Sciences (HICSS-31), January 1998.

Hector Levesque, Ray Reiter, Yves Lespérance and Fangzhen Lin, and Richard Scherl. GOLOG: A logic programming language for dynamic domains, with In the *Journal of Logic Programming*, vol 31(1-3), May 1997.

Ray Reiter, chapter 6 of *Knowledge in Action*

Cognitive Robotics: Introduction to Knowledge Representation and Reasoning

- Conclusions
  1. Further Issues
  2. Applications
  3. References
  4. Pointers

Theory of Actions: Further Issues

- Exogenous Actions
- Probabilistic action occurrences and effects
- Ability
- Time
- Concurrency
- Hypothetical and Counterfactual Reasoning
- Agent beliefs, desires, intentions
- Real time, resource bounded behavior
- Belief revision
- Execution monitoring and failure recovery

Concurrency in Situation Calculus

walk and chew gum

- $\textsc{Walking}(x, y, s)$ (fluent)
- $\textsc{startWalk}(x, y)$ (action)
- $\textsc{endWalk}(x, y)$ (action)

$[\{\textsc{startWalk}(\textsc{a}, \textsc{b}), \textsc{startChewGum}\}$
$\{\textsc{endChewGum}, \textsc{startSing}\},$
$\{\textsc{endWalk}(\textsc{a}, \textsc{b})\}$

Concurrency (cont)

$$Poss(a, s) \rightarrow [\textsc{Walking}(x, y, do(a, s)) \leftrightarrow$$
$$(\exists r)\{a = \textsc{startWalk}(x, y); \vee$$
$$\textsc{Walking}(x, y, s) \wedge a \neq \textsc{walk}(x, y)].$$

$$Poss(\textsc{startWalk}(x, y), s) \leftrightarrow$$
$$[\neg(\exists u, v)\textsc{Walking}(u, v, s) \wedge \textsc{location}(x) = x]$$

$$Poss(\textsc{endWalk}(x, y), s) \leftrightarrow$$
$$\textsc{Walking}(x, y, s)$$

Commonsense Reasoning)

(From the Call for Papers of Common Sense 2001)

- change, action, and causality
- ontologies, including space, time, shape, and matter, ontologies of networks and structures
- axiomatizations of benchmark commonsense problems.
- non-monotonic reasoning
- formal models of probabilistic reasoning
- belief change, update, revision
- cognitive robotics
- mental attitudes including knowledge, belief, intention, and planning.
- applications of formal representations to applications, such as natural language processing.

Applications

- Robots
  - Legolog: Inexpensive Experiments in Cognitive Robotics (Levesque and Pagnucco)

    `http://www.cs.toronto.edu/~morri/Legolog`
  - Museum Tour Guide [Bonn – CMU]

    `http://www-i5.informatik.rwth-aachen.de/kbs`
- Hypertext —Scherl, Bieber, and Vitali
- Animated Characters – John Funge *A.I. for Games and Animation*, A.K. Peters, 1999
- Software Agents —Ruman and Levesque, Banking Agent
- Using the internet
  - Sheila A. McIlraith, Tran Cao Son and honglei Zeng,"Semantic Web Service," *IEEE Intelligent Systems*, March/April 2001
  - McIlraith and Son, "Adapting Golog for the Semantic Web," Commonsense 2001.

Knowledge Representation and Reasoning

- Davis, Ernie. Representations of Commonsense Knowledge. Morgan Kaufmann. 1986.
- Genesereth, Michael and Nilsson, Nils. Logical Foundations of Artificial Intelligence. Morgan Kaufmann. 1987
- Brachman, Ronald and Levesque, Hector. (eds) Readings in Knowledge Representation. Morgan Kaufmann. 1985.
- Poole, David and Mackworth, Alan and Goebel, Randy. Computational Intelligence: A Logical Approach. Oxford University Press. 1998.

Cognitive Robotics

- Raymond Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, 2001, to appear.
- Murray Shanahan, Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press. 1997.
- Erik Sanderwall. Features and Fluents: The Representation of Knowledge about Dynamical Systems. Clarendon Press: Oxford. 1994.

Pointers

- `http://www.cs.toronto.edu/cogrobo` University of Toronto Cognitive Robotics Group
- `http://casbah.ee.ic.ac.uk/ mpsha` Murray Shanahan's web page
- http://www.wv.inf.tu-dresden.de/Research/Robotics/ Dresden Cognitive Robotics Group
- `http://www-i5.informatik.rwth-aachen.de/kbsg/` Aachen Knowledge Based Systems Group

Pointers (cont)

- http://www.ida.liu.se/
- http://www.public.asu.edu/ cbaral/
  Chitta Baral
- http://www.kr.org/kr/kr.html
  Principles of Knowledge Representation and Reasoning, Incorporated
- http://www-formal.Stanford.edu/leora/cs/
  Common Sense Problems
- http://www.ida.liu.se/ext/etai/lmw
  Logic Modeling Workshop