COMP90054 Software Agents: Projection and Regression

Adrian Pearce

includes slides by Ray Reiter & Ryan Kelly

Outline

Projection and Regression on Situations

- Techniques for effective inductive reasoning over situations continued

Outline

Basic theories of actions

Recall that Σ denotes the four *foundational axioms* for situations.

- We now consider some metamathematical properties of these axioms when combined with *successor state* and *action precondition* axioms, and unique names axioms for actions.
- Such a collection of axioms will be called a *basic theory of actions*.
- First we must be more precise about what counts as successor state and action precondition axioms.

Uniform Formulae

Let σ be a term of sort *situation* .

Definition (Uniform Formulae)

A formula is *uniform in* σ iff it does not mention the predicates *Poss* or $\sqsubset$, it does not quantify over variables of sort *situation*, it does not mention equality on situations, and whenever it mentions a term of sort *situation* in the situation argument position of a fluent, then that term is σ.

- Formulas uniform in s = Markov property. The future (states) are determined by the present (state).

Action precondition and successor state axioms

- **Definition: Action Precondition Axiom**
  An action precondition axiom is a sentence of the form:

$$(\forall x_1, \cdots, x_n, s).Poss(A(x_1, \cdots, x_n), s) \equiv \Pi_A(x_1, \cdots, x_n, s),$$

  where A is an n-ary action function, and Π_A is a formula that is uniform in s and whose free variables are among $x_1, \cdots, x_n, s$.
- **Definition: Successor State Axiom** A successor state axiom for an $(n+1)$-ary fluent F is a sentence of the form:

$$(\forall a, s)(\forall x_1, \ldots, x_n).F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F, \qquad (1)$$

  where Φ_F is a formula uniform in s, all of whose free variables are among $a, s, x_1, \ldots, x_n$.

Basic Action Theories

$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ where

- Σ are the foundational axioms for situations.
- $\mathcal{D}_{ss}$ is a set of successor state axioms.
- $\mathcal{D}_{ap}$ is a set of action precondition axioms.
- $\mathcal{D}_{una}$ is the set of unique names axioms for actions.
- $\mathcal{D}_{S_0}$ is a set of first order sentences with the property that S_0 is the only term of sort *situation* mentioned by the fluents of a sentence of $\mathcal{D}_{S_0}$. Thus, no fluent of a formula of $\mathcal{D}_{S_0}$ mentions a variable of sort *situation* or the function symbol *do*. $\mathcal{D}_{S_0}$ will play the role of the initial situation of the world (i.e. the one we start off with, before any actions have been "executed").

Theorem (Relative Satisfiability)

$\mathcal{D}$ is satisfiable iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is.

Relative satisfiability assures us that provided the initial database together with the unique names axioms for actions are satisfiable, then unsatisfiability cannot be introduced by augmenting these with the foundational axioms, together with the action preconditions and successor state axioms. This as

Outline

Regression

The **Regressable Formulas**.

- The essence of a regressable formula is that each of its *situation* terms is rooted at S_0, and therefore, one can tell, by inspection of such a term, exactly how many actions it involves.

Examples of regressable formulae: can the actions in the sequence

$walk(A, B, Adrian), train(B, C, Adrian), walk(C, D, Adrian)$ be executed one after the other beginning in S_0?

$Poss(walk(A, B, Adrian), S_0) \wedge$
$Poss(train(B, C, Adrian), do(walk(A, B, Adrian), S_0)) \wedge$
$Poss(walk(C, D, Adrian), do([walk(A, B, Adrian), train(B, C, Adrian)], S_0)$

The following are not regressable:

$(\exists a) Poss(a, S_0), \ \ holding(x, do(walk(A, B, Adrian), s))$.

Another example of regressable formulae: The Gold Thief

The (potential) thief would like to know whether the following action sequence

$walk(A, B, Bruce), enter(bank(Bruce)), crackSafe(Bruce)$

can be executed one after the other beginning in S_0?

$Poss(walk(A, B, Bruce), S_0) \wedge$
$Poss(enter(bank(Bruce)), do(walk(A, B, Bruce), S_0)) \wedge$
$Poss(crackSafe(Bruce), do([walk(A, B, Bruce), enter(bank(Bruce)], S_0)).$

The following are not regressable:

$(\exists a) Poss(a, S_0),\ \ holding(Gold, do(pickup(Gold, Bruce), s)).$

Regression (notes)

- It is not necessary to be able to tell what those actions are, just how many they are. In addition, when a regressable formula mentions a *Poss* atom, we can tell, by inspection of that atom, exactly what is the action function symbol occurring in its first argument position, for example, that it is a *move* action.
- Assume a background axiomatisation that includes a set of successor state and action precondition axioms.

The Regression Operator: Simple Version

W is a regressable formula of $\mathcal{L}_{sitcalc}$ that mentions no functional fluents.

1 If W is an atom, there are four possibilities:

1.1 W is a situation independent atom. Then $\mathcal{R}[W] = W$.

1.2 W is a *relational fluent* atom of the form $F(\vec{t}, S_0)$. Then $\mathcal{R}[W] = W$.

1.3 W is a regressable *Poss* atom, so it has the form $Poss(A(\vec{t}), \sigma)$ for terms $A(\vec{t})$ and σ of sort *action* and *situation* respectively. Here, A is an action function symbol of $\mathcal{L}_{sitcalc}$. Then there must be an action precondition axiom for A of the form

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

1.4 W is a relational fluent atom of the form $F(\vec{t}, do(\alpha, \sigma))$. Let F's successor state axiom in $\mathcal{D}_{ss}$ be $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$.

Then $\mathcal{R}[W] = \mathcal{R}[\Phi_F(\vec{t}, \alpha, \sigma)]$.

The Regression Operator: Simple Version (notes)

Assume that all quantifiers (if any) of $\Pi_A(\vec{x}, s)$ have had their quantified variables renamed to be distinct from the free variables (if any) of $Poss(A(\vec{t}), \sigma)$. Then

$$\mathcal{R}[W] = \mathcal{R}[\Pi_A(\vec{t}, \sigma)].$$

In other words, replace the atom $Poss(A(\vec{t}), \sigma)$ by a suitable instance of the right hand side of the equivalence in A's action precondition axiom, and regress that expression. The above renaming of quantified variables of $\Pi_A(\vec{x}, s)$ prevents any of these quantifiers from capturing variables in the instance $Poss(A(\vec{t}), \sigma)$.

The Regression Operator: Simple Version (notes)

Assume that all quantifiers (if any) of $\Phi_F(\vec{x}, a, s)$ have had their quantified variables renamed to be distinct from the free variables (if any) of $F(\vec{t}, do(\alpha, \sigma))$.

In other words, replace the atom $F(\vec{t}, do(\alpha, \sigma))$ by a suitable instance of the right hand side of the equivalence in F's successor state axiom, and regress this formula. The above renaming of quantified variables of $\Phi_F(\vec{x}, a, s)$ prevents any of these quantifiers from capturing variables in the instance $F(\vec{t}, do(\alpha, \sigma))$.

The Regression Operator: Simple Version

2 For non-atomic formulas, regression is defined inductively.

- 2.1 $\mathcal{R}[\neg W] = \neg\mathcal{R}[W]$,
- 2.2 $\mathcal{R}[W_1 \wedge W_2] = \mathcal{R}[W_1] \wedge \mathcal{R}[W_2]$,
- 2.3 $\mathcal{R}[(\exists v)W] = (\exists v)\mathcal{R}[W]$.

The Regression Operator: Simple Version (notes)

- Intuitively, the regression operator eliminates *Poss* atoms in favour of their definitions as given by action precondition axioms, and replaces fluent atoms about $do(\alpha, \sigma)$ by logically equivalent expressions about σ as given by successor state axioms. Moreover, it repeatedly does this until it cannot make such replacements any further.
- Each $\mathcal{R}$-step reduces the depth of nesting of the function symbol *do* in the fluents of W by substituting suitable instances of Φ_F for each occurrence of a fluent atom of W of the form $F(t_1, \ldots, t_n, do(\alpha, \sigma))$. Since no fluent atom of Φ_F mentions the function symbol *do*, the effect of this substitution is to replace each such F by a formula whose fluents mention only the situation term σ, and this reduces the depth of nesting by one.

The regression theorem

Theorem (The Regression Theorem)

Suppose W is a regressable sentence of $\mathcal{L}_{sitcalc}$ that mentions no functional fluents, and $\mathcal{D}$ is a basic theory of actions. Then,

$$\mathcal{D} \models W \text{ iff } \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W].$$

Consider a sequence $\alpha_1, \ldots, \alpha_n$ of ground action terms.

- **Problem**: Determine whether this is executable. Is it the case that:

$$\mathcal{D} \models executable(do([\alpha_1, \ldots, \alpha_n], S_0))$$

Executable Action Sequences

It is straightforward to show that:

$$\Sigma \models (\forall a_1, \ldots, a_n).executable(do([a_1, \ldots, a_n], S_0)) \equiv \bigwedge_{i=1}^{n} Poss(\alpha_i, do([\alpha_1, \ldots, \alpha_{i-1}], S_0)).$$

Theorem

Suppose that $\alpha_1, \ldots, \alpha_n$ is a sequence of ground action terms of $\mathcal{L}_{sitcalc}$. Then

$$\mathcal{D} \models executable(do([\alpha_1, \ldots, \alpha_n], S_0))$$

iff

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \bigwedge_{i=1}^{n} \mathcal{R}[Poss(\alpha_i, do([\alpha_1, \ldots, \alpha_{i-1}], S_0))].$$

Executable Action Sequences (notes)

This provides a systematic, regression-based method for determining whether a ground situation $do([\alpha_1, \ldots, \alpha_n], S_0)$ is executable. Moreover, it reduces this test to a theorem-proving task *in the initial database* $\mathcal{D}_{S_0}$, together with unique names axioms for actions.

Example: Executibility Testing (notes)

Another database example:

- Compute the legality test for the transaction sequence $register(Bill, C100), drop(Bill, C100), drop(Bill, C100)$ which intuitively should fail because the first *drop* leaves *Bill* unenrolled in $C100$, so that the precondition for the second *drop* will be false.

Executibility Testing (Continued)

Legality test for the transaction sequence
$register(Bill, C100), drop(Bill, C100), drop(Bill, C100)$.

First compute

$$\mathcal{R}[Poss(register(Bill, C100), S_0)] \wedge$$
$$\mathcal{R}[Poss(drop(Bill, C100), do(register(Bill, C100), S_0))] \wedge$$
$$\mathcal{R}[Poss(drop(Bill, C100), do(drop(Bill, C100), do(register(Bill, C100), S_0)))],$$

which is

$$\mathcal{R}[(\forall p).prerequ(p, C100) \supset (\exists g).grade(Bill, p, g, S_0) \wedge g \geq 50] \wedge$$
$$\mathcal{R}[enrolled(Bill, C100, do(register(Bill, C100), S_0))] \wedge$$
$$\mathcal{R}[enrolled(Bill, C100, do(drop(Bill, C100), do(register(Bill, C100), S_0)))].$$

This yields $\{(\forall p).prerequ(p, C100) \supset$
$(\exists g).grade(Bill, p, g, S_0) \wedge g \geq 50\} \wedge true \wedge false$

transaction sequence is illegal!

Another Example: Legality Testing

- $change(Bill, C100, 60), register(Sue, C200), drop(Bill, C100)$.
- First compute

$$\mathcal{R}[(\exists g')grade(Bill, C100, g', S_0) \wedge g' \neq 60] \wedge$$
$$\mathcal{R}[(\forall p)prerequ(p, C200) \supset$$
$$(\exists g)grade(Sue, p, g, do(change(Bill, C100, 60), S_0)) \wedge g \geq 50] \wedge$$
$$\mathcal{R}[enrolled(Bill, C100, do(register(Sue, C200),$$
$$do(change(Bill, C100, 60), S_0)))].$$

- This simplifies to

$$\{(\exists g').grade(Bill, C100, g', S_0) \wedge g' \neq 60\} \wedge$$
$$\{(\forall p).prerequ(p, C200) \supset$$
$$Bill = Sue \wedge p = C100 \vee (\exists g).grade(Sue, p, g, S_0) \wedge g \geq 50\} \wedge$$
$$\{Sue = Bill \wedge C200 = C100 \vee enrolled(Bill, C100, S_0)\}.$$

- So the transaction sequence is legal iff this sentence is entailed by the initial database together with unique names axioms for actions.

The Projection Problem

Definition (The Projection Problem)

Given an action sequence $\alpha_1, \ldots, \alpha_n$ of *ground* action terms, and a query $Q(s)$ whose only free variable is the situation variable s, what is the answer to Q in that situation resulting from performing this action sequence, beginning with the initial world situation S_0? Formally, the problem of determining whether

$$\mathcal{D} \models Q(do([\alpha_1, \ldots, \alpha_n], S_0)).$$

- Note that $Q(do([\alpha_1, \ldots, \alpha_n], S_0))$ will normally be regressable formulae.
- So, by the Regression Theorem, regress $Q(do([\alpha_1, \ldots, \alpha_n], S_0))$, and verify it in the initial situation with unique names axioms for actions.

Projection problem example: Database Query Evaluation

- Consider again the transaction sequence

  $\mathbf{T} = change(Bill, C100, 60), register(Sue, C200), drop(Bill, C100).$

- Suppose the query is

$$(\exists st).enrolled(st, C200, do(\mathbf{T}, S_0)) \wedge$$
$$\neg enrolled(st, C100, do(\mathbf{T}, S_0)) \wedge$$
$$(\exists g).grade(st, C200, g, do(\mathbf{T}, S_0)) \wedge g \geq 50.$$

- Regress this query, after some simplification, assuming that $\mathcal{D}_{S_0} \models C100 \neq C200$, we obtain

$$(\exists st).[st = Sue \vee enrolled(st, C200, S_0)] \wedge$$
$$[st = Bill \vee \neg enrolled(st, C100, S_0)] \wedge$$
$$[(\exists g).grade(st, C200, g, S_0) \wedge g \geq 50].$$

- The answer to the query is obtained by evaluating this last formula in $\mathcal{D}_{S_0}$, together w. unique names axioms for actions.

Projection problem example: Planning

For example, a projection query for the sequence of actions

$walk(A, B, Adrian), train(B, C, Adrian), walk(C, D, Adrian)$

might be: Will Adrian get home from the Jazz Festival tonight?

$$\mathcal{D} \models getHome(Adrian, do([walk(VillaCelimontana, B, Adrian), train(B, C, Adrian), walk(C, D, Adrian)], S_0)).$$

Regression (re-cap)

Regression operates, intuitively, by unwinding actions one at a time:

$$\mathcal{R}(Holding(agt, obj, do(c, s))) \Rightarrow$$
$$pickup(agt, obj) \in c$$
$$\vee Holding(agt, obj, s) \wedge \neg (drop(agt, obj) \in c)$$

By repeatedly applying it, we get a query about S_0:

$$\mathcal{D} \models \phi[do(c_1, do(c_2, \ldots, S_0))] \text{ iff } \mathcal{D} \models \mathcal{R}^*(\phi)[S_0]$$

If you don't know the current situation, you cannot reason using regression.

Effective reasoning in the situation calculus

- Effective reasoning in the situation calculus is generally based on *syntactic* manipulation of a query into a form that is more amenable to automated reasoning.
- In the general case, answering a query about a basic action theory $\mathcal{D}$ is a theorem-proving task in second-order logic (denoted SOL) due to the induction axiom included in the foundational axioms:

$$\mathcal{D} \models_{SOL} \psi$$

- If a query only performs *existential* quantification over situation terms, it can be answered without the induction axiom (denoted I) and thus using only first-order logic (FOL):

$$\mathcal{D} \models_{SOL} \exists s : \psi(s) \quad \text{iff} \quad \mathcal{D} - \{I\} \models_{FOL} \exists s : \psi(s)$$

Effective reasoning in the situation calculus

- Simpler still, queries uniform in the initial situation, can be answered using only first-order logic and a limited set of axioms:

$$\mathcal{D} \models_{SOL} \phi[S_0] \quad \text{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models_{FOL} \phi[S_0]$$

References

- Ray Reiter. Knowledge in Action: Logical Foundations for Specifying and implementing Dynamical Systems, The MIT Press, 2001
- Ray Reiter. The frame problem in situation the 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 Professional, Inc., 1991.
- Fiora Pirri and Ray Reiter. Some contributions to the metatheory of the situation calculus. Journal of the ACM, 46(3):325-361, 1999.
- Ray Reiter. Proving Properties of States in the Situation Calculus. Artificial Intelligence, 64:337-351, 1993.

References (continued)

- Ryan F. Kelly and Adrian R. Pearce. Property Persistence in the Situation Calculus. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), pages 1948-1953, 2007
- Ryan Kelly. "Asynchronous Multi-Agent Reasoning in the Situation Calculus", PhD Thesis, The University of Melbourne, 2009

Summary