Handling Uncertainty System in the Situation Calculus with Macro-actions

Yilan Gu
Department of Computer Science
University of Toronto
November 28, 2002
Outline

- Review – the situation calculus
- Introducing macro-actions
- Developing the knowledge base for the macro-actions
- Reuse of the macro-actions
- Conclusion and future work
The basic elements of the situation calculus

- **Actions**: \( kick(x) \): kick object \( x \)
- **Situations**: \( s_0, do(a,s) \)
- **Objects**: \( mary, boxA \)

The basic action theory

- Action precondition axioms
- Successor state axioms
- Initial database

Complex actions

\[ \alpha;\beta, \ p?, \ \alpha |\beta, \ (\pi x)\alpha(x), \ if\text{-}then\text{-}else, \ while \ loop \]

\[ \text{proc name body endproc} \]

Golog program – a bunch of procedures followed by complex actions
The regression operator

- Regressable formulas
  example: $\text{Poss}(\text{pickup}(x),S_0) \land \text{ontable}(x,\text{do}(\text{putTable}(x),S_0))$

- The regression operator $\mathcal{R}$
  examples: Suppose $\text{Poss}(\text{pickup}(t),s) \equiv \text{handfree}(s)$, $\text{ontable}(t,\text{do}(a,s)) \equiv a = \text{putTable}(t) \lor \text{ontable}(t) \land \neg a = \text{picup}(t)$
  then $\mathcal{R}[\text{Poss}(\text{pickup}(x),S_0) \land \text{ontable}(x,\text{do}(\text{putTable}(x),S_0))]$
      $= \mathcal{R}[\text{Poss}(\text{pickup}(x),S_0)] \land \mathcal{R}[\text{ontable}(x,\text{do}(\text{putTable}(x),S_0))]
      = \mathcal{R}[\text{handfree}(S_0)] \land \mathcal{R}[\text{true}]
      = \text{handfree}(S_0)$

- The regression theorem
  For any regressable formula $W$, $\mathcal{D} | W$ iff $\mathcal{D}_{S_0} \cup D_{una} = \mathcal{R}[W]$. 
Review (cont.)

- Uncertainty system
  - Stochastic actions
    Examples: \( \text{choice}(\text{pickup}(x), a) \equiv a = \text{pickupS}(x) \lor a = \text{pickupF}(x) \)
  - Probability
    \[ \text{prob}(a, \beta, s) = p \equiv \text{choice}(\beta, a) \land \text{Poss}(a, s) \land p = \text{prob}_0(a, \beta, s) \lor \]
    \[ \neg [\text{choice}(\beta, a) \land \text{Poss}(a, s) ] \land p = 0 \]
    example: if \( \text{Poss}(\text{pickupS}(x), S_0), \neg \text{Poss}(\text{pickupS}(x), S') \),
    \( \text{prob}_0(\text{pickupS}(x), \text{pickup}(x), s) = 0.9 \), then
    \( \text{prob}(\text{pickupS}(x), \text{pickup}(x), S_0) = 0.9 \),
    \( \text{prob}(\text{pickupS}(x), \text{pickup}(x), S') = 0. \)
  - Modified Golog interpreter – stGolog
Example of Robot climbing Stairs

(0) ready

(1) liftUpperLeg(h)

(2) forwLowLeg

(3) stepDown(main)

(4) moveBarycenter(main)

(5) straightMain

(6) forwSupLeg

(7) stepDown(supporting)

(8) moveBarycenter(supporting)

(ready again)
**Nature’s choices**

\[
\text{choice}(\text{liftUpperLeg}(h), a) \equiv a = \text{liftTill}(h) \lor a = \text{malfunc}(h).
\]

\[
\text{choice}(\text{forwLowLeg}, a) \equiv a = \text{forwLowLegS} \lor a = \text{forwLowLegF}.
\]

\[
\text{choice}(\text{stepDown}(l), a) \equiv a = \text{stepDownS}(l) \lor a = \text{stepDownF}(l).
\]

\[
\text{choice}(\text{moveBarycenter}(l), a) \\equiv a = \text{moveBarycenterS}(l) \lor a = \text{moveBarycenterF}(l).
\]

\[
\text{choice}(\text{straightLeg}, a) \equiv a = \text{straightLeg}.
\]

\[
\text{choice}(\text{forwSupLeg}, a) \equiv a = \text{forwSupLegS} \lor a = \text{forwSupLegF}.
\]

**Precondition axioms: (example)**

\[
\text{Poss}(\text{straightLeg}, s) \equiv \\
\text{straightMain}(s) \land \text{footOnGround}(\text{main}, s) \land \text{barycenter}(\text{main}, s).
\]
Fluents

- Relational Fluents:
  - \textit{straightMain}(s), \textit{Barycenter}(l,s), \textit{footOnGround}(l,s), \textit{overNewStair}(l,s)

- Functional Fluents: \textit{mainToCurr}(s)

Successor State Axioms: (example)

\[ over\text{NewStair}(l,do(a,s)) \equiv \]
\[ a = \text{forwSupLegS} \land l = \text{supporting} \lor \]
\[ a = \text{forwLowLegS} \land l = \text{main} \lor \]
\[ over\text{NewStair}(l,s) \land \neg a = \text{stepDownS}(l). \]
Probabilities: (examples)

\[
\begin{align*}
prob_0(liftTill(h), liftUpperLeg(h), s) &= h/(h+100). \\
prob_0(malfunc(h), liftUpperLeg(h), s) &= h/(h+100).
\end{align*}
\]

Initial Database:

\[
\begin{align*}
\text{straightMain}(S_0), & \quad \text{mainToCurr}(0, S_0), \quad \neg \text{overNewStair}(l, S_0), \\
\text{barycenter}(\text{supporting}, S_0), & \quad \text{legalStair}(h) \equiv \text{number}(h) \land 0<h<20. \\
\text{footOnGround}(l, S_0) & \equiv l = \text{main} \lor l = \text{supporting}.
\end{align*}
\]

Procedure of climbing a stair of height \( h \)

\[
\begin{align*}
\text{proc } \text{climbing}(h) \\
\text{legalStair}(h)?; \text{liftUpperLeg}(h); \text{forwLowLeg}; \text{stepDown}(\text{main}); \\
\text{moveBarycenter}(\text{main}); \text{straightLeg}; \text{forwSupLeg}; \\
\text{stepDown}(\text{supporting}); \text{moveBarycenter}(\text{supporting})
\end{align*}
\]
endproc
Motivation

- Create intelligent autonomous agents to help human on particular topics
- Agents often meet same local situations and asked to achieve same tasks and perform same strategies
- Human beings often act “without thinking”
- Purpose: save time on re-computation
- Idea:
  - Consider certain kind of complex actions as a whole
  - Compute and keep intermediate knowledge
  - Reuse the existing intermediate knowledge
Introducing macro-actions

- Discard uncertain logic actions $\alpha | \beta$ and $(\pi x)\alpha(x)$
- Not consider $p?$, if-then-else or while loop as a part of macro-actions
- Small sequence of actions is appropriate
  - Uniform syntax
  - Easy to find its deterministic choices and information of corresponding probabilities
  - Feasible to give extended precondition axioms and successor state axiom
Introducing macro-actions (Cont.)

- Sequence of stochastic actions \( \alpha = \alpha_1; \alpha_2; \ldots; \alpha_n \) and a sequence of deterministic actions \( A = A_1; A_2; \ldots; A_m \)

- Nature’s choice of \( \alpha \)
  
  \[
  \text{choiceMac}(\alpha, a) \equiv \frac{a \in \{A_1; A_2; \ldots; A_m \mid m \in \mathbb{N} \land 1 \leq m \leq n \land (\land_{i=1}^{m} \text{choice}(\alpha_i, A_i))\}}{}
  \]

- Extended precondition axiom of \( A \)
  
  \[
  \text{Poss}(A, s) \equiv \land_{i=2}^{m} \text{Poss}(A_i, \text{do}([A_1; A_2; \ldots; A_{i-1}], s)) \land \text{Poss}(A, s) \\
  \equiv \land_{i=1}^{m} \pi_{A_i}(t_i, s)
  \]

- Extended successor state axiom of \( a = a_1; a_2; \ldots; a_m \) (\( a_i \) variables of sort deterministic actions)
  
  \[
  \begin{align*}
  m=1, \ F(x, \text{do}(a, s)) & \equiv \phi_F(x, a, s) \\
  m>1, \ F(x, \text{do}(a, s)) & \equiv \phi_F(x, a_m, \text{do}(a_1; a_2; \ldots; a_{m-1}, s)) \\
  & \equiv \psi_F(x, a_1, a_2, \ldots, a_m, s)
  \end{align*}
  \]
Extended probabilities

\[ \text{probMac}(A, \alpha, s) = p \equiv \text{choiceMac}(\alpha, A) \wedge \text{Poss}(A, s) \]

\[ \wedge p = \text{prob}_0(A_1, \alpha_1, s) \ast \ldots \ast \text{prob}_0(A_m, \alpha_m, \text{do}([A_1, \ldots, A_{m-1}], s)) \lor \]

\[ \neg(\text{choiceMac}(\alpha, A) \wedge \text{Poss}(A, s)) \land p=0 . \]

Properties

1. \(( \forall a, s). 0 \leq \text{probMac}(a, \alpha, s) \leq 1.\)
2. \(( \forall a, s). \neg \text{choiceMac}(\alpha, a) \supset \text{probMac}(a, \alpha, s)=0.\)
3. \(( \forall s). \text{Poss}(A, s) \equiv \text{probMac}(A, \alpha, s)>0.\)
4. \(\forall (A= A_1, \ldots, A_n \land \text{choiceMac}(\alpha, A) \land \text{Poss}(A, s)) \supset \)

\[ \sum_{A \in \maxPoss(\alpha, s)} \text{probMac}(A, \alpha, s) = 1, \text{ where} \]

\[ \maxPoss(\alpha, s) \equiv \{ A= A_1, \ldots, A_m \mid \text{choiceMac}(\alpha, A) \land \text{Poss}(A, s) \}

\[ \wedge (m = n \lor m < n \land ((\forall a). \text{Choice}(\alpha_{m+1}, a) \supset \neg \text{Poss}(A; a, s))) \}. \]
Specifying macro-actions

- **Reason:**
  - Should be distinguish from ordinary sequential actions
  - Need to let the autonomous agent know

- **Syntax:**

  ```
  macro name \( \alpha_1 ; \alpha_2 ; \ldots ; \alpha_n \) endmacro
  ```

- **Extended definition:**

  ```
  choiceMac(name, a) \equiv choiceMac(\alpha_1 ; \alpha_2 ; \ldots ; \alpha_n, a),
  probMac(A, name, s)=p \equiv probMac(A, \alpha_1 ; \alpha_2 ; \ldots ; \alpha_n, s)=p,
  seqLength(name) = seqLength(\alpha_1 ; \alpha_2 ; \ldots ; \alpha_n) = n,
  maxPoss(name, s)=l \equiv maxPoss(\alpha_1 ; \alpha_2 ; \ldots ; \alpha_n, s)=l.
  ```
Example of macro-actions

Example 1

macro stepMain(h)
    liftUpperLeg(h); forwLowLeg; stepDown(main);
    moveBarycenter(main); straightLeg
endmacro

macro stepSupp
    forwSupLeg; stepDown(supporting); moveBarycenter(supporting)
endmacro

proc climbing(h)
    legalStair(h)?; stepMain(h); stepSupp
endproc
Example of macro-actions (Cont.)

Example 2

```plaintext
macro climbStair(h)
    liftUpperLeg(h); forwLowLeg; stepDown(main);
    moveBarycenter(main); straightLeg; forwSupLeg;
    stepDown(supporting); moveBarycenter(supporting)
endmacro

proc climbing(h)
    legalStair(h)?; climbStair(h)
endproc
```

```
The knowledge base

- What intermediate information do we want to keep?

Static part:
- Definition of macro-actions
- Maximal length of current existing macro-actions
- The extended successor state axioms
- The Extended precondition axioms of nature’s choices of macro-actions that are not equivalent to false
- The Extended probabilities of nature’s choices of macro-actions

Dynamic part:
- Facts: \( \text{maxPossBase}(\text{list}, \alpha, S) \equiv \text{list} = \text{maxPoss}(\alpha, S) \) for particular situation instance \( S \).
Extended regression operator

- Reason: reuse the existing information in the knowledge base
- s-regressable formula
  - Situation terms of from $do([\alpha_1, \ldots, \alpha_n], s)$
  - Regressable formula is same as $S_0$-regressable
- The extended regression operator $R^*$ for s-regressable formula $W$
  - Boarder than original regression operator
  - Save computational steps
  - Example
    $$R^*[F(x, do(a_1;a_2;a_3, s))], \text{ no } R[F(x, do([a_1, a_2, a_3], s))];$$
    $$R^*[F(x, do(a_1;a_2;a_3, S_0))] = R[F(x, do([a_1, a_2, a_3], S_0))], \text{ but has different computational steps}$$
The regression theorems for $\mathcal{R}^*$

- $\mathcal{L}_{sc}$, and $\mathcal{L}_{sc}'$ (allow notation $do(a_1; a_2; \ldots; a_n, s)$)

- Given $s$-regressable sentence $W$ in $\mathcal{L}_{sc}'$ and a basic action theory $\mathcal{D}$, $\mathcal{R}^*[W]$ is a sentence uniform in $s$ and
  \[ \mathcal{D} \models W \equiv \mathcal{R}^*[W] \]

- Given regressable sentence $W$ in $\mathcal{L}_{sc}$ and a basic action theory $\mathcal{D}$, then $\mathcal{D} \models \mathcal{R}[W] \equiv \mathcal{R}^*[W]$
  \[ \mathcal{D} \models W \text{ iff } \mathcal{D}_{S0} \cup \mathcal{D}_{una} \models \mathcal{R}^*[W] \]

- Given $S_0$-regressable sentence $W_1$ in $\mathcal{L}_{sc}'$, $W_2$ be the corresponding sentence by changing $do(a_1; a_2; \ldots; a_n, s)$ to be $do([a_1, a_2, \ldots, a_n], s)$, and a basic action theory $\mathcal{D}$, then
  \[ \mathcal{D} \models W_2 \text{ iff } \mathcal{D}_{S0} \cup \mathcal{D}_{una} \models \mathcal{R}^*[W_1] \]
Developing knowledge base for macro-action
Reuse of macro-actions

- After developing the knowledge base of macro-actions given by the user, we want to use the remembered information.

- A simple application – macGolog
  - Modified stGolog
  - Program now allows to include macro-actions
  - Adding the test checking that if an action is a macro-action
  - Checking if there exists fact $maxPossBase$ for macro-action at current situations (i.e., the dynamic part of the knowledge base), if exists, choose the maximal nature’s choices from it, otherwise, compute the fact, insert it to base and use the computed information.
Implementation of macGolog

- Implemented in Polog
- Has same functions as stGolog
- Benefit: same computing steps in the searching trees, therefore save computational time
- Example:
  The robot climbing stairs, comparing running the procedures of climbing a stair without or with macro-actions, we have following different searching trees
Searching trees (Fig. 1)
The searching tree for climbing a stair without macro-actions
Searching Trees (Fig. 2)

- The searching tree of climbing a stair for using two macro-actions
- The searching tree of climbing a stair for using one macro-action
Experiments of climbing stairs

- Pre-assumption
  - Test on 3 different definitions of climbing stair procedure
  - No macro-actions (using stGolog)
  - Define 2 short macro-actions: $\text{stepMain}(h)$, $\text{stepSupp}$
  - Define 1 long macro-action: $\text{climbStair}(h)$
  - 5 tests – simulating different environments
    - Test 1 – all stairs are of the same height
    - Test 2 – at most 10 different heights
    - Test 3 – at most 50 different heights
    - Test 4 – at most 800 different heights
    - Test 5 – all stairs are of different height
Comparing Results

-- 2 macro-actions

-- 1 macro-action

-- no macro-action
Discussion: benefits and limits

- Benefits:
  - Saving computational time
  - Agents have “memories” in some sense and keep the “experiences”

- Limits, especially under current implementation:
  - User’s duty to choose suitable macro-actions
  - The agents can not be aware of similar local situations themselves
**Conclusions and future work**

- **What we did:**
  - Introduce concept of macro-action
  - Define extended regression operation
  - Develop knowledge base for macro-actions
  - macGolog – primary implementation of reuse macro-actions

- **What we can do in the future:**
  - Improve implementation of reuse macro-actions
  - Possible to formalize the way of resetting the local situation, make agent forgettable
  - Try to make agents aware of similar local situations
  - Consider if we can adopt macro-action into the problem of solving MDP by using the situation calculus