Rational Action in Agent Programs with Prioritized Goals

Sebastian Sardiña
Dept. of Computer Science
University of Toronto
CANADA
ssardina@cs.toronto.edu

Steven Shapiro
School of Computer Science and Engineering
Univ. of New South Wales
AUSTRALIA
steven@cse.unsw.edu.au
Introduction
Introduction

1. Provide an account of rational action w.r.t. a set of prioritized goals

- *Rational*: The agent acts to the best of her abilities to bring about and maintain her goals
- Extend the single-goal approach in [Shapiro et al. 95]
Introduction

1. Provide an account of rational action w.r.t. a set of prioritized goals
   - *Rational*: The agent acts to the best of her abilities to bring about and maintain her goals
   - Extend the single-goal approach in [Shapiro et al. 95]

2. Integrate approach with INDILOG high-level agent programming language
   - New construct to execute programs the “best” way possible w.r.t. the goals
Situation Calculus

- A dialect of the predicate calculus with sorts for actions and situations

- **Actions**: openSafe, readComb, dial(13), ...

- **Situations**:
  - \( S_0 \) the initial situation
  - \( \text{do}(a, s) \) where \( a \) is an action term and \( s \) is a situation

- **Fluents**: predicates/functions whose value changes from situation to situation:

  \[
  \text{Locked}(S_0) \land \neg \text{Locked}(\text{do}(\text{dial}(2), S_0))
  \]

- use a distinguished predicate \( \text{Poss}(a, s) \) to assert that action \( a \) is possible to execute in situation \( s \)
Knowledge and Strategies
1. **Knowledge/Beliefs**: using possible worlds approach

- $K(s', s)$: situation $s'$ is accessible from situation $s$
- $\text{Know}(\alpha, s)$: agent knows/believes $\alpha$ in $s$
- Sensing actions affect knowledge. E.g., if $a = \text{senseLegible}$ either $\text{Know}(\text{PaperLegible}, do(a, s))$ or $\text{Know}(\neg\text{PaperLegible}, do(a, s))$ holds.
1. Knowledge/Beliefs: using possible worlds approach
   
   - $K(s', s)$: situation $s'$ is accessible from situation $s$
   - $\text{Know}(\alpha, s)$: agent knows/believes $\alpha$ in $s$
   - Sensing actions affect knowledge. E.g., if $a = \text{senseLegible}$ either $\text{Know}(\text{PaperLegible}, \text{do}(a, s))$ or $\text{Know}(\neg \text{PaperLegible}, \text{do}(a, s))$ holds.

2. Strategy: a mapping from situations to actions

   A strategy dictates one possible behavior at every situation

   $$
   \begin{align*}
   \sigma(S_0) &= \text{readComb} \\
   \sigma(\text{do(readComb}, S_0)) &= \text{dial}(1) \\
   \sigma'(S_0) &= \text{dial}(0) \\
   \sigma'(\text{do(dial}(0), S_0)) &= \text{open} \\
   \vdots \\
   \end{align*}
   $$
Knowledge and Strategies

1. Knowledge/Beliefs: using possible worlds approach

   - $K(s', s)$: situation $s'$ is accessible from situation $s$
   - $\text{Know}(\alpha, s)$: agent knows/believes $\alpha$ in $s$
   - Sensing actions affect knowledge. E.g., if $a = \text{senseLegible}$ either $\text{Know}($PaperLegible, do(a, s)$)$ or $\text{Know}(\neg\text{PaperLegible}, do(a, s))$ holds.

2. Strategy: a mapping from situations to actions

   A strategy dictates one possible behavior at every situation

   \[
   \begin{align*}
   \sigma(S_0) &= \text{readComb} \\
   \sigma(\text{do(readComb, } S_0)) &= \text{dial}(1) \\
   \sigma'(S_0) &= \text{dial}(0) \\
   \sigma'(\text{do(dial}(0), S_0)) &= \text{open}
   \end{align*}
   \]

3. Ability: $\text{Can}(\alpha, s)$: agent is capable of achieving $\alpha$ in $s$
Safe Example [Moore85] [Shapiro95]

An agent wants to open a safe by dialing its combination number (0 or 1), but she does not know its combination. The combination is written on a piece of paper. However, the paper may be illegible.

What is the most rational thing to do?
An agent wants to open a safe by dialing its combination number (0 or 1), but she *does not know* its combination. The combination is written on a piece of paper. However, the paper may be illegible.

*What is the most rational thing to do?*

1. Check whether the paper is readable or not;
Safe Example [Moore85] [Shapiro95]

An agent wants to open a safe by dialing its combination number (0 or 1), but she does not know its combination. The combination is written on a piece of paper. However, the paper may be illegible.

What is the most rational thing to do?

1. Check whether the paper is readable or not;
2. If it is readable, then read the combination number and dial it: the safe will be unlocked!
Safe Example [Moore85] [Shapiro95]

An agent wants to open a safe by dialing its combination number (0 or 1), but she does not know its combination. The combination is written on a piece of paper. However, the paper may be illegible.

What is the most rational thing to do?

1. Check whether the paper is readable or not;

2. If it is readable, then read the combination number and dial it: the safe will be unlocked!

3. Otherwise, just guess a number randomly and dial it! (we may be lucky and open the safe!)
**Safe Example** [Moore85] [Shapiro95]

An agent wants to open a safe by dialing its combination number (0 or 1), but she *does not know* its combination. The combination is written on a piece of paper. However, the paper may be illegible.

*What is the most rational thing to do?*

1. Check whether the paper is readable or not;

2. If it is readable, then read the combination number and dial it: the safe will be unlocked!

3. Otherwise, just guess a number randomly and dial it! (we may be lucky and open the safe!)

What if the safe explodes when the wrong number is dialed?
Safe Example [Moore85] [Shapiro95]

An agent wants to open a safe by dialing its combination number (0 or 1), but she does not know its combination. The combination is written on a piece of paper. However, the paper may be illegible.

What is the most rational thing to do?

1. Check whether the paper is readable or not;

2. If it is readable, then read the combination number and dial it: the safe will be unlocked!

3. Otherwise, just guess a number randomly and dial it! (we may be lucky and open the safe!)

What if the safe explodes when the wrong number is dialed?

⇒⇒ What is needed: two goals with different priorities!
Representing Multiple Goals with Priorities

- \textit{PATH} = \textit{Strategy} + \textit{Situation} \\
  (\sigma, s): \text{ path obtained from executing strategy } \sigma \text{ from situation } s

- We represent \( k \) goals at different levels of priorities by characterizing the “happy” paths:

  \[
  \forall \sigma. H(\sigma, s, 1) \equiv \phi_1(\sigma, s) \quad \leftarrow \text{ MOST IMPORTANT} \\
  \vdots \\
  \forall \sigma. H(\sigma, s, k) \equiv \phi_k(\sigma, s) \quad \leftarrow \text{ LESS IMPORTANT} \\
  \forall n. n > k \supset \forall \sigma. H(\sigma, s, n) \equiv \text{TRUE}
  \]

- For our safe example: \textbf{Always}(-\textit{Exploded}) > \textbf{Eventually}(-\textit{Open})
Representing Multiple Goals with Priorities

- **PATH = Strategy + Situation**
  
  \((\sigma, s)\): path obtained from executing strategy \(\sigma\) from situation \(s\)

- We represent \(k\) goals at different levels of priorities by characterizing the “happy” paths:

  \[
  \forall \sigma. H(\sigma, s, 1) \equiv \phi_1(\sigma, s) \quad \text{← MOST IMPORTANT}
  \]
  
  \[
  \vdots
  \]
  
  \[
  \forall \sigma. H(\sigma, s, k) \equiv \phi_k(\sigma, s) \quad \text{← LESS IMPORTANT}
  \]
  
  \[
  \forall n. n > k \supset \forall \sigma. H(\sigma, s, n) \equiv TRUE
  \]

- For our safe example: **Always**(\(\neg Exploded\)) > **Eventually**(\(Open\))

  \[
  \forall \sigma. H(\sigma, S_0, 1) \equiv \text{Always}(\neg Exploded, \sigma, S_0)
  \]
  
  \[
  \forall \sigma. H(\sigma, S_0, 2) \equiv \text{Eventually}(\text{Open}, \sigma, S_0)
  \]
  
  \[
  \forall n. n > 2 \supset \forall \sigma. H(\sigma, S_0, n) \equiv TRUE
  \]
Comparing Strategies

Dominance at one level (w.r.t. the agent’s knowledge):

\[ \sigma_1 \succeq_s \sigma_2 \overset{\text{def}}{=} \forall s'. K(s', s) \land H(\sigma_2, n, s') \supset H(\sigma_1, n, s') \]

Next, we combine all levels:

\[
\text{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \\
(\forall n. \sigma_2 \succeq_n \sigma_1 \supset \sigma_1 \succeq_n \sigma_2) \lor \\
\exists m. \sigma_1 \succeq_m \sigma_2 \land \forall i. i < m \supset (\sigma_2 \succeq^i_i \sigma_1 \supset \sigma_1 \succeq^i_i \sigma_2)
\]
Comparing Strategies

Dominance at one level (w.r.t. the agent’s knowledge):

\[
\sigma_1 \succeq^s_n \sigma_2 \overset{\text{def}}{=} \forall s'. K(s', s) \land H(\sigma_2, n, s') \supset H(\sigma_1, n, s')
\]

Next, we combine all levels:

\[
\text{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \\
(\forall n. \sigma_2 \succeq^s_n \sigma_1 \supset \sigma_1 \succeq^s_n \sigma_2) \lor \\
\exists m. \sigma_1 \succeq^s_m \sigma_2 \land \forall i. i < m \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2)
\]
Comparing Strategies

Dominance at one level (w.r.t. the agent’s knowledge):

\[ \sigma_1 \succeq^s_n \sigma_2 \overset{\text{def}}{=} \forall s'. K(s', s) \land H(\sigma_2, n, s') \supset H(\sigma_1, n, s') \]

Next, we combine all levels:

\[ \text{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \left( \forall n. \sigma_2 \succeq^s_n \sigma_1 \supset \sigma_1 \succeq^s_n \sigma_2 \right) \lor \right. \]
\[ \exists m. \sigma_1 \succeq^s_m \sigma_2 \land \forall i. i < m \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2) \]
Comparing Strategies

Dominance at one level (w.r.t. the agent’s knowledge):

$$\sigma_1 \succeq^s_n \sigma_2 \overset{\text{def}}{=} \forall s'. K(s', s) \land H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\text{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=}$$

$$\left(\forall n. \sigma_2 \succeq^s_n \sigma_1 \supset \sigma_1 \succeq^s_n \sigma_2\right) \lor$$

$$\exists m. \sigma_1 \succeq^s_m \sigma_2 \land \forall i. i < m \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2)$$

\[\sigma_1 \succeq^s_1 \sigma_2 \quad \sigma_1 \succeq^s_2 \sigma_2 \quad \cdots \quad \neg(\sigma_1 \succeq^s_k \sigma_2)\]

MOST IMPORTANT

\[\neg(\sigma_2 \succeq^s_k \sigma_1)\]

LESS IMPORTANT
Comparing Strategies

Dominance at one level (w.r.t. the agent’s knowledge):

\[ \sigma_1 \preceq^s_n \sigma_2 \overset{\text{def}}{=} \forall s'. K(s', s) \land H(\sigma_2, n, s') \supset H(\sigma_1, n, s') \]

Next, we combine all levels:

\[ \text{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \]

\[ \left( \forall n. \sigma_2 \preceq^s_n \sigma_1 \supset \sigma_1 \preceq^s_n \sigma_2 \right) \lor \]

\[ \exists m. \sigma_1 \preceq^s_m \sigma_2 \land \forall i. i < m \supset (\sigma_2 \preceq^s_i \sigma_1 \supset \sigma_1 \preceq^s_i \sigma_2) \]

\[
\begin{array}{cccccccc}
\sigma_1 & \preceq^s_1 & \sigma_2 & \cdots & \neg(\sigma_1 \preceq^s_k \sigma_2) & \cdots & \sigma_1 & \preceq^s_n \sigma_2 \\
\text{MOST IMPORTANT} & & & & \neg(\sigma_2 \preceq^s_k \sigma_1) & \text{LESS IMPORTANT}
\end{array}
\]
Defining Rational Strategies

A strategy can be followed by the agent iff every step is known to the agent and physically possible:

$$\text{CanFollow}(\sigma, s) \overset{\text{def}}{=} \forall s'. \text{OnPath}(\sigma, s, s') \supset \exists a \text{ Know}(\sigma = a, s') \land \text{Poss}(\sigma(s'), s')$$
Defining Rational Strategies

A strategy can be followed by the agent iff every step is known to the agent and physically possible:

\[
\text{CanFollow}(\sigma, s) \overset{\text{def}}{=} \forall s'. \text{OnPath}(\sigma, s, s') \supset \exists a \text{ Know}(\sigma = a, s') \land \text{Poss}(\sigma(s'), s')
\]

A strategy is rational iff the agent knows she can follow it and that it is as good as any other strategy she can follow:

\[
\text{RationalPrio}(\sigma, s) \overset{\text{def}}{=} \text{Know}(\text{CanFollow}(\sigma), s) \land \\
(\forall \sigma'. \text{Know}(\text{CanFollow}(\sigma'), s) \supset \text{AsGood}(\sigma, \sigma', s))
\]
Properties and Results

- Goal priorities have a lexicographic order;
- Our rationality account is conservative. There may be no rational strategy (e.g., infinite chain of strategies).
Properties and Results

- Goal priorities have a lexicographic order;
- Our rationality account is conservative. There may be no rational strategy (e.g., infinite chain of strategies).

Theorem 1. *If there is a single goal (i.e., level 1 goal), our account is equivalent to [Shapiro et al. 95]*
Properties and Results

• Goal priorities have a lexicographic order;
• Our rationality account is conservative. There may be no rational strategy (e.g., infinite chain of strategies).

Theorem 1. If there is a single goal (i.e., level 1 goal), our account is equivalent to [Shapiro et al. 95]

Theorem 2. Suppose strategy $\sigma$ is rational but the agent believes that $\sigma$ does not guarantee some achievable goal $\phi_k$. Then, the agent believes that strategy $\sigma$ has a “better chance” of achieving some more important goal than any strategy that guarantees $\phi_k$. 

**IndiGolog Overview**

**IndiGolog** is an agent programming language based on the Sit. Calc.

Programmer provides:

1. An *action theory* to specify the domain dynamics, and

2. A *non-deterministic program* to specify agent behavior

A full range of programming constructs including concurrency is supported.

To do planning in **IndiGolog**, programmer provides non-deterministic program/plan skeleton, which is put in a "**search block**" (Σδ)

Interpreter must do *lookahead* and find a plan that ensures successful execution of the search block.
The Safe Problem in IndiGolog

What would be an agent program for opening the safe?

```prolog
proc open_safe
  (πa.a)*;
  (holding(paper))?;
  senseLegible*; /* sensing action */
  readComb*; /* sensing action */
  (πc.dial(c) | (true)?)
  (open | call(locksmith));
end;
```

High-level programs have many advantages, but they lack a declarative notion of goal/objective:
The Safe Problem in IndiGoLog

What would be an agent program for opening the safe?

```plaintext
proc open_safe
  (πa.a)*;
  (holding(paper))?;
  senseLegible*;  /* sensing action */
  readComb*;  /* sensing action */
  (πc.dial(c) | (true)?);
  (open | call(locksmith));
end;
```

High-level programs have many advantages, but they lack a declarative notion of goal/objective:

What is the goal of the above program?
How to execute the above program differently depending on the current goals?
A Rational Search Construct: $\Delta_{rat}(\delta : \phi_1 > ... > \phi_n)$

Produces a simple and ready-to-execute plan whose execution will respect both the given nondeterministic program and the set of declarative objectives.

$$\Delta_{rat}(\delta : \phi_1 > ... > \phi_n) \implies \delta_s$$

where $\delta_s$:

- is in the same agent language, i.e., IndiGOLOG;
- is simple, i.e., deterministic and with no search;
- is ready-to-be-executed, i.e., is sufficiently detailed and possible;
- respects the original program, i.e., any execution of $\delta_s$ is an execution of $\delta$;
- respects goals, i.e., it is rational w.r.t. the goals $\phi_1 > ... > \phi_n$. 
Defining Rational Programs

**Induced**($\delta, \sigma, s$): if any situation reached with the strategy $\sigma$ from $s$ can be reached with program $\delta$ from $s$.

**Equivalent**($\sigma_1, \sigma_2, s$): if $\sigma_1$ and $\sigma_2$ dictate the same actions from situation $s$.

**UInduced**($\delta, \sigma, s$): if $\sigma$ is induced by $\delta$ and any other induced strategy $\sigma'$ is equivalent to it in $s$.

Recast the strategy comparison in the language:
Defining Rational Programs

**Induced**($\delta, \sigma, s$): if any situation reached with the strategy $\sigma$ from $s$ can be reached with program $\delta$ from $s$.

**Equivalent**($\sigma_1, \sigma_2, s$): if $\sigma_1$ and $\sigma_2$ dictate the same actions from situation $s$.

**ULInduced**($\delta, \sigma, s$): if $\sigma$ is induced by $\delta$ and any other induced strategy $\sigma'$ is equivalent to it in $s$.

Recast the strategy comparison in the language:

$$\text{AsGood}(\sigma_1, \sigma_2, s) \implies \text{AsGood}^L(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$$
Defining Rational Programs (cont.)

A strategy is rational w.r.t. a program and a set of goals:

\[ \text{RationalPrio}(\sigma, \delta, g, s) \overset{\text{def}}{=} \]
\[ \text{Know}(\text{CanFollow}(\sigma), s) \land \]
\[ [\forall \sigma'. \text{Know}(\text{Induced}(\delta, \sigma') \land \text{CanFollow}(\sigma'), s) \supset A_{\text{Good}}^L(\sigma, \sigma', g, s)] \]
Defining Rational Programs (cont.)

A strategy is rational w.r.t. a program and a set of goals:

\[
\text{RationalPrio}(\sigma, \delta, g, s) \overset{\text{def}}{=} \\
\text{Know(CanFollow}(\sigma), s) \land \\
[\forall \sigma'. \text{Know( Induced}(\delta, \sigma') \land \\
\text{CanFollow}(\sigma'), s) \supset \text{AsGood}^L(\sigma, \sigma', g, s)]
\]

A plan \( \delta_r \) is a rational implementation of \( \delta \) if \( \delta_r \) induces a unique rational strategy:

\[
\text{DRationalSol}(\delta, \delta_r, g, s) \overset{\text{def}}{=} \\
\exists \sigma. \text{Know}[ \text{UnInduced}(\delta_r, \sigma) \land \text{Induced}(\delta, \sigma), s] \land \\
\text{RationalPrio}(\sigma, \delta, g, s)
\]

Note: The most important and implicit “goal” is to execute the original program!
Rational Program for the Safe Problem

\texttt{proc open\_safe}
\begin{align*}
& (\pi a.a)^*; \\
& (\text{holding(paper)})?; \\
& \text{senseLegible}^*; \\
& \text{readComb}^*; \\
& (\pi c. \text{dial}(c) \mid (\text{true})?; \\
& (\text{open} \mid \text{call(locksmith)})); \\
\end{align*}
\texttt{end;}

\Delta_{rat}(open\_safe : \text{Always}(\neg Exploded) > \text{Eventually}(\text{Open}))
Rational Program for the Safe Problem

\textbf{proc} \texttt{open\_safe} \\
\hspace{5mm} (\pi a.a)^*; \\
\hspace{5mm} (\text{holding(paper)})?; \\
\hspace{5mm} \text{senseLegible}^*; \\
\hspace{5mm} \text{readComb}^*; \\
\hspace{5mm} (\pi c.\text{dial}(c) \mid (\text{true})?); \\
\hspace{5mm} (\text{open} \mid \text{call(locksmith)}); \\
\textbf{end}; \\

\[ \Delta_{rat}(\text{open\_safe} : \text{Always}(\neg\text{Exploded}) > \text{Eventually}(\text{Open})) \]

\[ \delta_0 \overset{\text{def}}{=} \text{pickup(paper)}; \text{dial}(1) \; ; \text{open} \quad \text{If combination num is known to be 1} \]
Rational Program for the Safe Problem

\[\Delta_{rat}(open_{-}safe : Always(\neg Exploded) > Eventually(Open))\]

\textbf{proc} open\_safe
(\pi a.a)*;
(holding(paper))?;
senseLegible*;
readComb*;
(\pi c.dial(c) | (true)?);
(open | call(locksmith));
end;

\(\delta_0 \overset{\text{def}}{=} \) pickup(paper); dial(1) ; open \hspace{1cm} \text{If combination num is known to be 1}

\(\delta_1 \overset{\text{def}}{=} \) pickup(paper); senseLegible;
\hspace{1cm} \text{If safe may explode and comb. is unkown}
\hspace{1cm} \textbf{if} PaperLegible \textbf{then} \{readComb; dial(Comb); open\}
\hspace{1cm} \textbf{else} call(locksmith)
Rational Program for the Safe Problem

\begin{equation}
\Delta_{rat}(open_{safe} : \text{Always}(\neg \text{Exploded}) > \text{Eventually}(\text{Open}))
\end{equation}

\textbf{proc} open\_safe
\begin{align*}
(\pi a.a)^*; \\
(\text{holding(paper)})?; \\
\text{senseLegible}^*; \\
\text{readComb}^*; \\
(\pi c.\text{dial}(c) \mid (\text{true})?); \\
(\text{open} \mid \text{call(locksmith)});
\end{align*}
\textbf{end;}

\begin{align*}
\delta_0 & \text{ def } \text{pickup(paper)}; \text{dial}(1); \text{open} & \text{If combination num is known to be 1} \\
\delta_1 & \text{ def } \text{pickup(paper)}; \text{senseLegible}; & \text{If safe may explode and comb. is unknown} \\
& \textbf{if} \text{PaperLegible} \textbf{then} \{\text{readComb}; \text{dial(Comb)}; \text{open}\} \\
& \textbf{else} \text{call(locksmith)} \\
\delta_2 & \text{ def } \text{pickup(paper)}; \text{senseLegible}; & \text{If safe never explodes and comb. is unknown} \\
& \textbf{if} \text{PaperLegible} \textbf{then} \{\text{readComb}; \text{dial(Comb)}; \text{open}\} \\
& \textbf{else} \{\text{dial}(1); \text{open}\}
\end{align*}
Conclusions

What we have done here:

- Account of prioritized goals and rational action in the situation calculus;
- A construct for an agent programming language that mixes programs with declarative goals.

Future issues:

1. Develop an implementation (maybe using DTGolog [Boutilier et al. 2000]);
2. Introduce soft goals (at each level? use numbers?);
3. Other criteria of rational action (from decision-theory, e.g., minimax regret);
4. Drop strategies and use programs only.
THE END
Useful Definitions I

A situation $s'$ is in the situation sequence defined by $\sigma$ and $s$:

$$\text{OnPath}(\sigma, s, s') \overset{\text{def}}{=} s \leq s' \land \forall a, s^*(s < \text{do}(a, s^*) \leq s' \supset \sigma(s^*) = a)$$

Formula $\alpha$ is eventually true in the path defined by $\sigma$ and $s$:

$$\text{Eventually}(\alpha, \sigma, s) \overset{\text{def}}{=} \exists s^*. \text{OnPath}(\sigma, s, s^*) \land \alpha(\sigma, s^*)$$

Formula $\alpha$ is always true in the path defined by $\sigma$ and $s$:

$$\text{Always}(\alpha, \sigma, s) \overset{\text{def}}{=} \forall s^*. \text{OnPath}(\sigma, s, s^*) \supset \alpha(\sigma, s^*)$$
Useful Definitions II

A situation $s'$ is in the situation sequence defined by $\sigma$ and $s$:

$$\text{OnPath}(\sigma, s, s') \overset{\text{def}}{=} s \leq s' \land \forall a, s^*(s < do(a, s^*) \leq s' \supset \sigma(s^*) = a)$$

A non-deterministic program induces strategies:

$$\text{Induced}(\delta, \sigma, s) \overset{\text{def}}{=} \forall s^*. \text{OnPath}(\sigma, s, s^*) \supset \exists \delta'. \text{Trans}^*(\delta; \delta_{\text{noOp}}, s, \delta', s^*)$$

Two strategies may be equivalent at a particular situation:

$$\text{Equivalent}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \forall s^*. \text{OnPath}(\sigma_1, s, s^*) \supset \sigma_1(s^*) = \sigma_2(s^*)$$

A program may uniquely induce a strategy:

$$\text{UInduced}(\delta, \sigma, s) \overset{\text{def}}{=} \text{Induced}(\delta, \sigma, s) \land (\forall \sigma'. \text{Induced}(\delta, \sigma', s) \supset \text{Equivalent}(\sigma, \sigma', s))$$
**IndiGolog Semantics**

Based on transition systems

\[ \text{Trans}(\delta, s, \delta', s') \]
means that can make transition \((\delta, s) \rightarrow (\delta', s')\) by executing a single primitive action or test.

\[ \text{Final}(\delta, s) \]
means that computation can be terminated in \((\delta, s)\)

\[ \text{Do}(\delta, s, s') \]
means that there is an execution of \(\delta\) starting in \(s\) and terminating in \(s'\)

\[ \text{Do}(\delta, s, s') \overset{\text{def}}{=} \exists \delta', s'. \text{Trans}^*(\delta, s, \delta', s') \land \text{Final}(\delta', s') \]
where \(\text{Trans}^*\) is the reflexive transitive closure of \(\text{Trans}\).
Theorems

\[ P(\sigma_1, \sigma_2, n, s) \equiv \sigma_1 \succeq_n \sigma_2 \land (\forall n'. n' < n \land P(\sigma_2, \sigma_1, n', s) \supset P(\sigma_1, \sigma_2, n', s)) \]  

(1)

**Theorem 1.** Let \( \mathcal{H} \) be the axioms \( \{I(1), \forall \sigma, s. H(\sigma, s) \equiv H(\sigma, 1, s)\} \). Then,

\[ \mathcal{H} \cup \{(1), (\forall \sigma, s. \text{CanFollow}(\sigma, s))\} \models \forall \sigma, s. \text{RationalPrio}(\sigma, s) \equiv \text{Rational}(\sigma, s) \]

**Theorem 2.** For any formula \( \phi(\text{asf, now}) \):

\[ \{(1)\} \models \forall s, n, \sigma_\phi. \text{OGoal}(\text{Eventually}(\phi), n, s) \land \text{Achieve}(\phi, \sigma_\phi, s) \land \text{RationalPrio}(\sigma, s) \land \neg \text{Know}(\text{Eventually}(\phi, \sigma, \text{now}), s) \supset \exists n'. n' < n \land P^\neq(\sigma, \sigma_\phi, n', s) \]
Issues

Q: What happens if there is no “rational solution” for a search block?

A: There is no legal transition for the search block. However, other branches of the program may proceed.

Q: What if the goals are impossible to satisfy?

A: If the agent believes that no possible strategy can satisfy the goals, then any strategy is as good as any other strategy. Hence, committing to any of them would be rational.

Q: Does this account of goals suffer from the side-effect problem?

A: Yes, every logical consequence of a goal is also a goal (at the same level).

Q: Can the agent have impossible goals?

A: As far as this paper is concerned, yes. We have not addressed the problem of goal dynamics or how the agent comes to acquire her desires.