

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 INDIGOLOG 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
 - $do(a, s)$ where a is an action term and s is a situation
- Fluents: predicates/functions whose value changes from situation to situation:

$$Locked(S_0) \wedge \neg Locked(do(dial(2), S_0))$$

- use a distinguished predicate $Poss(a, s)$ to assert that action a is possible to execute in situation s

Knowledge and Strategies

Knowledge and Strategies

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

Knowledge and Strategies

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

A strategy dictates one possible behavior at every situation

$$\begin{array}{ll} \sigma(S_0) = \textit{readComb} & \sigma'(S_0) = \textit{dial}(0) \\ \sigma(\textit{do}(\textit{readComb}, S_0)) = \textit{dial}(1) & \sigma'(\textit{do}(\textit{dial}(0), S_0)) = \textit{open} \\ \vdots & \vdots \end{array}$$

Knowledge and Strategies

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

A strategy dictates one possible behavior at every situation

$$\begin{array}{ll} \sigma(S_0) = \textit{readComb} & \sigma'(S_0) = \textit{dial}(0) \\ \sigma(\textit{do}(\textit{readComb}, S_0)) = \textit{dial}(1) & \sigma'(\textit{do}(\textit{dial}(0), S_0)) = \textit{open} \\ \vdots & \vdots \end{array}$$

3. **Ability:** **Can**(α, s): agent is capable of achieving α 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?

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;

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

- $PATH = Strategy + Situation$

(σ, s) : path obtained from executing strategy σ from situation s

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

$$\begin{array}{ll} \forall \sigma. H(\sigma, \bar{s}, 1) \equiv \phi_1(\sigma, \bar{s}) & \longleftarrow \text{MOST IMPORTANT} \\ \vdots & \\ \forall \sigma. H(\sigma, \bar{s}, k) \equiv \phi_k(\sigma, \bar{s}) & \longleftarrow \text{LESS IMPORTANT} \\ \forall n. n > k \supset \forall \sigma. H(\sigma, \bar{s}, n) \equiv TRUE & \end{array}$$

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

Representing Multiple Goals with Priorities

- $PATH = Strategy + Situation$

(σ, s) : path obtained from executing strategy σ from situation s

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

$$\begin{aligned} \forall \sigma. H(\sigma, \bar{s}, 1) &\equiv \phi_1(\sigma, \bar{s}) && \longleftarrow \text{MOST IMPORTANT} \\ &\vdots && \\ \forall \sigma. H(\sigma, \bar{s}, k) &\equiv \phi_k(\sigma, \bar{s}) && \longleftarrow \text{LESS IMPORTANT} \\ \forall n. n > k &\supset \forall \sigma. H(\sigma, \bar{s}, n) \equiv TRUE \end{aligned}$$

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

$$\begin{aligned} \forall \sigma. H(\sigma, S_0, 1) &\equiv \text{Always}(\neg Exploded, \sigma, S_0) \\ \forall \sigma. H(\sigma, S_0, 2) &\equiv \text{Eventually}(Open, \sigma, S_0) \\ \forall n. n > 2 &\supset \forall \sigma. H(\sigma, S_0, n) \equiv TRUE \end{aligned}$$

Comparing Strategies

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

$$\sigma_1 \succ_n^s \sigma_2 \stackrel{\text{def}}{=} \forall s'. K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\begin{aligned} \mathbf{AsGood}(\sigma_1, \sigma_2, s) &\stackrel{\text{def}}{=} \\ &(\forall n. \sigma_2 \succ_n^s \sigma_1 \supset \sigma_1 \succ_n^s \sigma_2) \vee \\ &\exists m. \sigma_1 \succ_m^s \sigma_2 \wedge \forall i. i < m \supset (\sigma_2 \succ_i^s \sigma_1 \supset \sigma_1 \succ_i^s \sigma_2) \end{aligned}$$

MOST IMPORTANT

LESS IMPORTANT

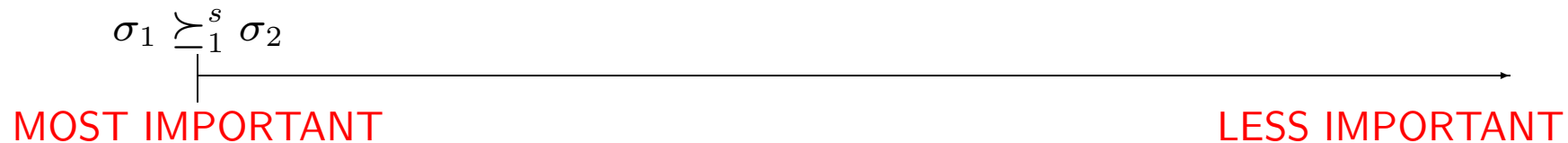
Comparing Strategies

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

$$\sigma_1 \succ_n^s \sigma_2 \stackrel{\text{def}}{=} \forall s'. K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\begin{aligned} \mathbf{AsGood}(\sigma_1, \sigma_2, s) &\stackrel{\text{def}}{=} \\ &(\forall n. \sigma_2 \succ_n^s \sigma_1 \supset \sigma_1 \succ_n^s \sigma_2) \vee \\ &\exists m. \sigma_1 \succ_m^s \sigma_2 \wedge \forall i. i < m \supset (\sigma_2 \succ_i^s \sigma_1 \supset \sigma_1 \succ_i^s \sigma_2) \end{aligned}$$



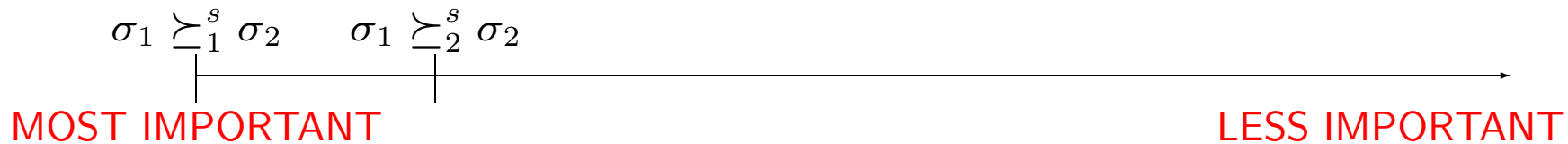
Comparing Strategies

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

$$\sigma_1 \succ_n^s \sigma_2 \stackrel{\text{def}}{=} \forall s'. K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\begin{aligned} \mathbf{AsGood}(\sigma_1, \sigma_2, s) &\stackrel{\text{def}}{=} \\ &(\forall n. \sigma_2 \succ_n^s \sigma_1 \supset \sigma_1 \succ_n^s \sigma_2) \vee \\ &\exists m. \sigma_1 \succ_m^s \sigma_2 \wedge \forall i. i < m \supset (\sigma_2 \succ_i^s \sigma_1 \supset \sigma_1 \succ_i^s \sigma_2) \end{aligned}$$



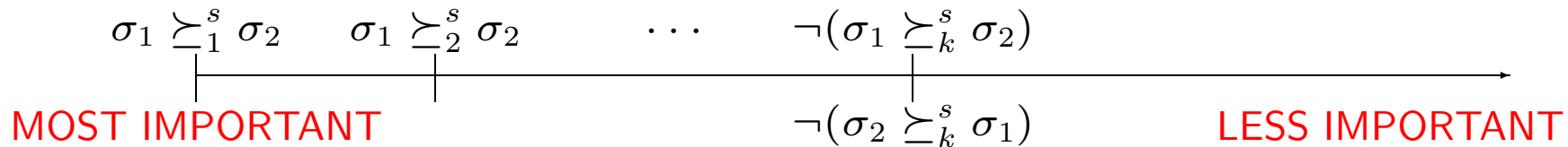
Comparing Strategies

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

$$\sigma_1 \succ_n^s \sigma_2 \stackrel{\text{def}}{=} \forall s'. K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\begin{aligned} \mathbf{AsGood}(\sigma_1, \sigma_2, s) &\stackrel{\text{def}}{=} \\ &(\forall n. \sigma_2 \succ_n^s \sigma_1 \supset \sigma_1 \succ_n^s \sigma_2) \vee \\ &\exists m. \sigma_1 \succ_m^s \sigma_2 \wedge \forall i. i < m \supset (\sigma_2 \succ_i^s \sigma_1 \supset \sigma_1 \succ_i^s \sigma_2) \end{aligned}$$



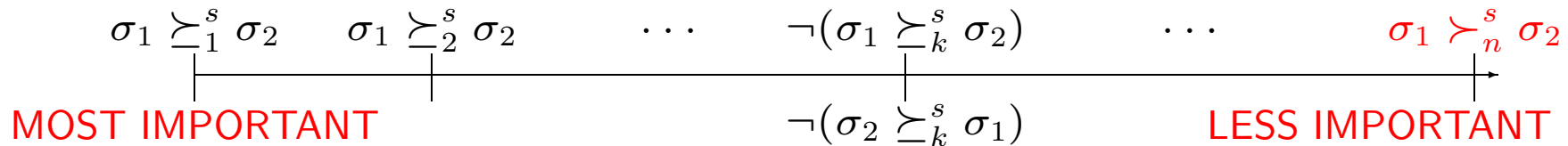
Comparing Strategies

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

$$\sigma_1 \succ_n^s \sigma_2 \stackrel{\text{def}}{=} \forall s'. K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

Next, we combine all levels:

$$\begin{aligned} \mathbf{AsGood}(\sigma_1, \sigma_2, s) &\stackrel{\text{def}}{=} \\ &(\forall n. \sigma_2 \succ_n^s \sigma_1 \supset \sigma_1 \succ_n^s \sigma_2) \vee \\ &\exists m. \sigma_1 \succ_m^s \sigma_2 \wedge \forall i. i < m \supset (\sigma_2 \succ_i^s \sigma_1 \supset \sigma_1 \succ_i^s \sigma_2) \end{aligned}$$



Defining Rational Strategies

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

$$\begin{aligned} \mathbf{CanFollow}(\sigma, s) &\stackrel{\text{def}}{=} \\ &\forall s'. \mathbf{OnPath}(\sigma, s, s') \supset \\ &\quad \exists a \mathbf{Know}(\sigma = a, s') \wedge \mathit{Poss}(\sigma(s'), s') \end{aligned}$$

Defining Rational Strategies

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

$$\begin{aligned} \mathbf{CanFollow}(\sigma, s) &\stackrel{\text{def}}{=} \\ &\forall s'. \mathbf{OnPath}(\sigma, s, s') \supset \\ &\quad \exists a \mathbf{Know}(\sigma = a, s') \wedge \mathit{Poss}(\sigma(s'), s') \end{aligned}$$

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:

$$\begin{aligned} \mathbf{RationalPrio}(\sigma, s) &\stackrel{\text{def}}{=} \\ &\mathbf{Know}(\mathbf{CanFollow}(\sigma), s) \wedge \\ &(\forall \sigma'. \mathbf{Know}(\mathbf{CanFollow}(\sigma'), s) \supset \mathbf{AsGood}(\sigma, \sigma', s)) \end{aligned}$$

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 σ is rational but the agent believes that σ does not guarantee some **achievable** goal ϕ_k . Then, the agent believes that strategy σ has a “**better chance**” of achieving some more important goal than any strategy that guarantees ϕ_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**” ($\Sigma\delta$)

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?

```
proc open_safe
  ( $\pi a.a$ )*;
  (holding(paper))?;
  senseLegible*;      /* sensing action */
  readComb*;         /* sensing action */
  ( $\pi c.dial(c)$  | (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?

```
proc open_safe
  ( $\pi a.a$ )*;
  (holding(paper))?;
  senseLegible*;      /* sensing action */
  readComb*;         /* sensing action */
  ( $\pi c.dial(c)$  | (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: $\Delta_{rat}(\delta : \phi_1 > \dots > \phi_n)$

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

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

where δ_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 δ_s is an execution of δ ;
- **respects goals**, i.e., it is rational w.r.t. the goals $\phi_1 > \dots > \phi_n$.

Defining Rational Programs

Induced(δ, σ, s): if any situation reached with the strategy σ from s can be reached with program δ from s .

Equivalent(σ_1, σ_2, s): if σ_1 and σ_2 dictate the same actions from situation s .

UInduced(δ, σ, s): if σ is induced by δ and any other induced strategy σ' is equivalent to it in s .

Recast the strategy **comparison** in the language:

Defining Rational Programs

Induced(δ, σ, s): if any situation reached with the strategy σ from s can be reached with program δ from s .

Equivalent(σ_1, σ_2, s): if σ_1 and σ_2 dictate the same actions from situation s .

UInduced(δ, σ, s): if σ is induced by δ and any other induced strategy σ' is equivalent to it in s .

Recast the strategy **comparison** in the language:

$$\mathbf{AsGood}(\sigma_1, \sigma_2, s) \implies \mathit{AsGood}^L(\sigma_1, \sigma_2, \phi_1 > \dots > \phi_k, s)$$

Defining Rational Programs (cont.)

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

$$\begin{aligned} \mathbf{RationalPrio}(\sigma, \delta, g, s) &\stackrel{\text{def}}{=} \\ &\mathbf{Know}(\mathbf{CanFollow}(\sigma), s) \wedge \\ &[\forall \sigma'. \mathbf{Know}(\mathbf{Induced}(\delta, \sigma') \wedge \\ &\quad \mathbf{CanFollow}(\sigma'), s) \supset \mathit{AsGood}^L(\sigma, \sigma', g, s)] \end{aligned}$$

Defining Rational Programs (cont.)

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

$$\begin{aligned} \mathbf{RationalPrio}(\sigma, \delta, g, s) &\stackrel{\text{def}}{=} \\ &\mathbf{Know}(\mathbf{CanFollow}(\sigma), s) \wedge \\ &[\forall \sigma'. \mathbf{Know}(\mathbf{Induced}(\delta, \sigma') \wedge \\ &\quad \mathbf{CanFollow}(\sigma'), s) \supset \mathit{AsGood}^L(\sigma, \sigma', g, s)] \end{aligned}$$

A plan δ_r is a *rational implementation* of δ if δ_r induces a unique rational strategy:

$$\begin{aligned} \mathbf{DRationalSol}(\delta, \delta_r, g, s) &\stackrel{\text{def}}{=} \\ &\exists \sigma. \mathbf{Know}[\mathbf{UInduced}(\delta_r, \sigma) \wedge \mathbf{Induced}(\delta, \sigma), s] \wedge \\ &\quad \mathbf{RationalPrio}(\sigma, \delta, g, s) \end{aligned}$$

Note: The most important and *implicit* “goal” is to execute the original program!

Rational Program for the Safe Problem

proc *open_safe*

$(\pi a.a)^*$;

$(\text{holding}(\text{paper}))?$;

senseLegible^* ;

readComb^* ;

$(\pi c.\text{dial}(c) \mid (\text{true}))?$;

$(\text{open} \mid \text{call}(\text{locksmith}))$;

end;

$\Delta_{rat}(\text{open_safe} : \mathbf{Always}(\neg \text{Exploded}) > \mathbf{Eventually}(\text{Open}))$

Rational Program for the Safe Problem

proc *open_safe*

($\pi a.a$)^{*};

(holding(paper))?;

senseLegible^{*};

readComb^{*};

($\pi c.dial(c) \mid (true)?$);

(open \mid call(locksmith));

end;

$\Delta_{rat}(open_safe : \mathbf{Always}(\neg Exploded) > \mathbf{Eventually}(Open))$

$\delta_0 \stackrel{\text{def}}{=} \text{pickup(paper); dial(1) ; open}$

If combination num is known to be 1

Rational Program for the Safe Problem

```
proc open_safe
  ( $\pi a.a$ )*;
  (holding(paper))?;
  senseLegible*;
  readComb*;
  ( $\pi c.dial(c) \mid (true)?$ );
  (open  $\mid$  call(locksmith));
end;
```

$\Delta_{rat}(open_safe : \mathbf{Always}(\neg Exploded) > \mathbf{Eventually}(Open))$

$\delta_0 \stackrel{\text{def}}{=} \text{pickup(paper); dial(1) ; open}$ **If combination num is known to be 1**

$\delta_1 \stackrel{\text{def}}{=} \text{pickup(paper); senseLegible;}$ **If safe may explode and comb. is unknown**
if PaperLegible **then** {readComb; dial(Comb); open}
 else call(locksmith)

Rational Program for the Safe Problem

```
proc open_safe
  ( $\pi a.a$ )*;
  (holding(paper))?;
  senseLegible*;
  readComb*;
  ( $\pi c.dial(c) \mid (true)?$ );
  (open  $\mid$  call(locksmith));
end;
```

$\Delta_{rat}(open_safe : \mathbf{Always}(\neg Exploded) > \mathbf{Eventually}(Open))$

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

$\delta_1 \stackrel{\text{def}}{=} \text{pickup(paper); senseLegible;}$ If safe may explode and comb. is unknown
 if PaperLegible **then** {readComb; dial(Comb); open}
 else call(locksmith)

$\delta_2 \stackrel{\text{def}}{=} \text{pickup(paper); senseLegible;}$ If safe never explodes and comb. is unknown
 if PaperLegible **then** {readComb; dial(Comb); open}
 else {dial(1); open}

Conclusions

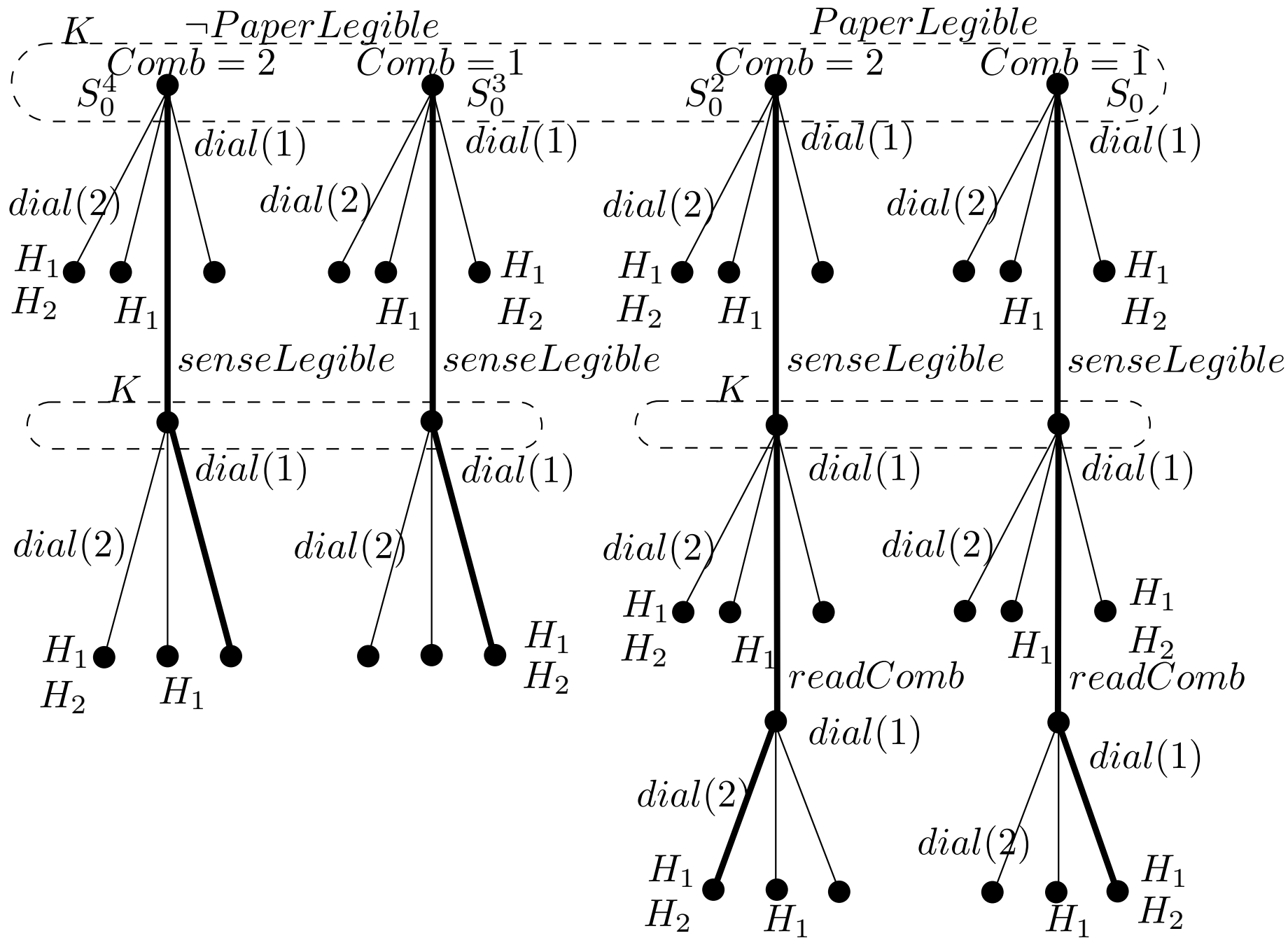
What we have done here:

- 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 σ and s :

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

Formula α is **eventually** true in the path defined by σ and s :

$$\mathbf{Eventually}(\alpha, \sigma, s) \stackrel{\text{def}}{=} \exists s^* . \mathbf{OnPath}(\sigma, s, s^*) \wedge \alpha(\sigma, s^*)$$

Formula α is **always** true in the path defined by σ and s :

$$\mathbf{Always}(\alpha, \sigma, s) \stackrel{\text{def}}{=} \forall s^* . \mathbf{OnPath}(\sigma, s, s^*) \supset \alpha(\sigma, s^*)$$

Useful Definitions II

A situation s' is in the situation sequence defined by σ and s :

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

A non-deterministic program **induces** strategies:

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

Two strategies may be **equivalent** at a particular situation:

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

A program may **uniquely** induce a strategy:

$$\mathbf{UInduced}(\delta, \sigma, s) \stackrel{\text{def}}{=} \mathbf{Induced}(\delta, \sigma, s) \wedge (\forall \sigma'. \mathbf{Induced}(\delta, \sigma', s) \supset \mathbf{Equivalent}(\sigma, \sigma', s))$$

INDIGOLOG Semantics

Based on transition systems

Trans(δ, s, δ', s')

means that can make transition $(\delta, s) \rightarrow (\delta', s')$ by executing a single primitive action or test.

Final(δ, s)

means that computation can be terminated in (δ, s)

Do(δ, s, s')

means that there is an execution of δ starting in s and terminating in s'

$$Do(\delta, s, s') \stackrel{\text{def}}{=} \exists \delta', s'. Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s')$$

where $Trans^*$ is the reflexive transitive closure of $Trans$.

Theorems

$$P(\sigma_1, \sigma_2, n, s) \equiv \sigma_1 \succeq_n^s \sigma_2 \wedge (\forall n'. n' < n \wedge P(\sigma_2, \sigma_1, n', s) \supset P(\sigma_1, \sigma_2, n', s)) \quad (1)$$

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

$$\mathcal{H} \cup \{(1), (\forall \sigma, s. \mathbf{CanFollow}(\sigma, s))\} \models \forall \sigma, s. \mathbf{RationalPrio}(\sigma, s) \equiv \mathbf{Rational}(\sigma, s)$$

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

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

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.