Intentional Learning and Reconsideration in Rational Agents
Modus Operandi (for Artificial Intelligence)

- Philosophy (Intentionality)
  - Formal Methods (BDI Logics)
  - Agent Oriented Programming (AgentSpeak(L))
Philosophical Background

Bratman
Practical Reasoning

Dennett
Intentional Stance

Searle
Speech Acts
Intentionality 1: Intentional Stance / Representation
Intentionality 2: Speech Acts / Communication

- Assertives
- Directives
- Commisives

Agent

Desires

Intentions

Beliefs

Environment

Intend ☝️

Believe ☝️

Promise ☝️

Declare ☝️

Environment

lunes 7 de noviembre de 11
Intentionality 3: Practical Reasoning

• How to do the right thing? vs Epistemic Reasoning.

• The input to such process are Beliefs, Desires and Intentions.

• Intentions are piles of partial, hierarchical, consistent and coherent plans.

• We will be able to deal with temporal aspects of practical reasoning as: Future Intentions, Persistence, Reconsideration and Commitment.

• Policy based Reconsideration is composed by rules determining when to reconsider; if you are a single-minded agent.
Commitment \((B^{KD45}D^{KD}\uparrow^{KD}_{CTL})\) [Rao, 1998]

- Blind Commitment

\[
\text{INTEND}(A\diamond \phi) \implies A(\text{INTEND}(A\diamond \phi) \cup (\text{BEL}(\phi)))
\]

- Rational Commitment (Single-Minded)

\[
\text{INTEND}(A\diamond \phi) \implies A(\text{INTEND}(A\diamond \phi) \cup (\text{BEL}(\phi) \lor \neg\text{BEL}(E\diamond \phi)))
\]

- Emotional Commitment (Open-Minded)

\[
\text{INTEND}(A\diamond \phi) \implies A(\text{INTEND}(A\diamond \phi) \cup (\text{BEL}(\phi) \lor \neg\text{DES}(E\diamond \phi)))
\]
Agent Architectures -> Agent Oriented Programming

dMARS

lunes 7 de noviembre de 11
A simple scenario

Blocks World

INT on(b,c)
Beliefs

6 on(b, a).
7 on(a, table).
8 on(c, table).
Beliefs and Rules

6  on(b,a).
7  on(a,table).
8  on(c,table).

clear(table).
4  clear(X) :- not(on(_,X)).
Beliefs, Rules, and Test Goals (?)

on(b,a).
on(a,table).
on(c,table).

clear(table).
clear(X) :- not(on(_,X)).

?clear(b) => true     ?on(a,X) => {X\b}
Achieve Goals as Desires (!)
Planes (Trigger Event : Context <- Body)

26 \(+!on(X,Y) : on(X,Y). // already achieved\)
27 \(+!on(X,Y) <- !clear(X); !clear(Y); move(X,Y).\)
Plans

Context

Conjunction of First Order Literals

26

+!on(X,Y) : on(X,Y). // already achieved

27

+!on(X,Y) <- !clear(X); !clear(Y); move(X,Y).
Plans

Sequence of Goals, Actions and/or Beliefs updates.
Intentions

+!on(a,c) <- !clear(a); !clear(c); move(a,c).

+!clear(a) : ?on(b,a) <- ?clear(W); move(b,W).

+!on(a,c) <- !clear(c); move(a,c).
AgentSpeak(L): Configurations [Bordini, 2007]

\[ c = \langle ag, C, M, T, s \rangle \]

Where:

\[ ag = \langle bs, ps \rangle \]

\[ C = \langle I, E, A \rangle \]

\[ M = \langle In, Out, SI \rangle \]

\[ s \in Labels \]
AgentSpeak(L): Semantics
AgentSpeak(L): Process Messages

on(b,a).
on(a,table).
on(c,table).
+put(b,c),
+on(b,a)...

Beliefs

Events
AgentSpeak(L): Select an Event

+l!put (b,c)

SelEv → SelEv₁ → Rel₁ → AppPl

ProcMsg → Cirlnt₁ → Cirlnt₂ → Cirlnt

SelEv → SelEv₂ → Rel₂ → SelInt

SelInt → SelInt₁ → SelInt₂

SelInt → SelAppl

SelInt → ExtEv

SelInt → IntEv

Cirlnt → TestGl₁ → TestGl₂

Cirlnt → AddBel

Cirlnt → DelBel

Cirlnt → Action

Cirlnt → ExecInt

Cirlnt → AddIM
AgentSpeak(L): Select Relevant Plans

SelEv -> Rel1 -> AppPl
SelEv1 -> Rel2

ProcMsg

SelInt1
SelInt2
SelInt3

AchvGl

ClrInt

TestGl1
TestGl2
AddBel
DelBel

Action

Rel

SelAppl

AddIM

SelEv

ClrInt

2
2

Rel

SelEv

1
1

SelInt

1

ExtEv

IntEv

SelAppl

2

SelEv

2

Appl

2

Appl

1

SelEv

2

SelEv

1

ProcMsg

ClrInt

1

ClrInt

3

AchvGl

{X\b, Y\c}

@put1
+!put(X, Y) : on(X, Y).

@put2
+!put(X, Y) <- !clear(X);
!clear(Y);
move(X, Y).

lunes 7 de noviembre de 11
AgentSpeak(L): Select Applicable Plans

on(b,a).
on(a,table).
on(c,table).

.+put(X,Y) : on(X,Y).

.+put(X,Y) : true
  <- !clear(X);
  !clear(Y);
  move(X,Y).

\{X\b,Y\c\}

lunes 7 de noviembre de 11
AgentSpeak(L): Select one Applicable Plan
AgentSpeak(L): Add an Intention

\[ C'_i = C_i \cup < \texttt{@put2 \{X\b, Y\c\}, T} > \]
AgentSpeak(L): Select an Intention

\[ C_I = C_I \cup \langle \text{@put2} \{X\b, Y\c\}, T \rangle \]

\[ \text{@put2} \]
\[ +!\text{put}(X,Y) \leftarrow \neg \text{clear}(X); \]
\[ \neg \text{clear}(Y); \]
\[ \text{move}(X,Y). \]

\[ CI = CI \cup \langle \text{@put2} \{X\b, Y\c\}, T \rangle \]

lunes 7 de noviembre de 11
AgentSpeak(L): Execute the Intention

@put2
+!put(X,Y) ← !clear(X);
!clear(Y);
move(X,Y).

{X\b, Y\c}
AgentSpeak(L): Events are added

![Diagram of AgentSpeak(L) events](attachment:image.png)
AgentSpeak(L): If everything is ok...

@put2
+!put(X,Y) <- move(X,Y).

{X\b, Y\c}
AgentSpeak(L): ... But what if moves fails?

• The full intention fails!

• It is desirable to prevent such failures by learning a new context for the given plan, so that it won’t be selected again when it is not applicable (Intentional Learning):

\[
+!\text{put}(X,Y) : \text{clear}(X) \& \text{clear}(Y) \leftarrow \text{move}(X,Y).
\]

• It is desirable to learn abandon policies for Reconsidering intentions before they actually fail:

\[
\begin{align*}
\text{drop}(\text{put}(X,Y)) & \leftarrow \text{intend}(\text{put}(X,Y)) \& \text{not clear}(X). \\
\text{drop}(\text{put}(X,Y)) & \leftarrow \text{intend}(\text{put}(X,Y)) \& \text{not clear}(Y).
\end{align*}
\]
AgentSpeak(L): Reconsideration

- Formally: Agents must be single minded committed:

\[
\text{INTEND}(A\lozenge \phi) \implies A(\text{INTEND}(A\lozenge \phi) \cup (\text{BEL}(\phi) \lor \neg \text{BEL}(E\lozenge \phi)))
\]

- Where after BDI_{CTL}, \(A\) = for all possible future, \(\lozenge\) = eventually, and \(U\) = until.

- The reasons for not believing that in some future (E) eventually \(\phi\) are to be learned, and they cause that, when selected, the intention towards \(\phi\) is being dropped instead of executed.
AgentSpeak(L): Intentional Learning & abandon
Problems 1: The Learning Method

• What kind of learning method can these agents use, given that:

  • They use First-Order representations for Beliefs, Desires and Intentions.
  
  • The feedback for learning is the success or failure of executed Intentions.
  
  • The learning target (plan contexts) is possible conjunctive (it was in dMARS).
  
  • They have background knowledge to be exploited while learning.
  
Problems 2: Commitment

• But, what kind of commitment follows an AgentSpeak(L) agent in Jason? Ups, where are my Intentional and Temporal Operators?

• $\text{CTL}_{\text{AgentSpeak(L)}}$ [Guerra et al., 2008, 2009] A BDI-temporal Logic grounded on the Operational Semantics of Jason/AgentSpeak(L).

  • Agents satisfy the non-infinite deferral axiom: They eventually drop their intentions.

  • Agents do not satisfy the blind-commitment axiom: It is possible for them to abandon their intention for $\Phi$ without believing $\Phi$.

  • Agents satisfy a limited version of the single-minded commitment: Explicit reasons to abandon an Intention are not represented anywhere, but they can be learned as rules.
Learning: Training Examples

begin(model(1))
  succ.
  intend(put,b,c).
  on(b,a).
  on(a,table).
  on(c,table).
  on(z,table).
end(model(1))

begin(model(2))
  fail.
  intend(put,b,c).
  on(b,a).
  on(a,table).
  on(c,table).
  on(z,c).
end(model(2))
Learning: Language Bias

\[
\begin{align*}
\text{rmode}(\text{clear}(+\text{V1})). \\
\text{rmode}(\text{on}(+\text{V1}, +\text{V2})). \\
\text{rmode}(\text{on}(+\text{V2}, +\text{V1})). \\
\end{align*}
\]

\[
\begin{align*}
\text{clear}(A). \\
\text{clear}(B). \\
\text{on}(A, B). \\
\text{on}(B, A). \\
\end{align*}
\]

\[
\begin{align*}
\text{\textless Q = intend(put, A, B).}
\end{align*}
\]
Learning: Background Knowledge

clear(X) :- not(on(_,X)).
clear(table).
Algorithm 1 Top-down Induction of Logical Decision Trees.

1: procedure BUILDTREE(E,Q) ▷ E is a set of examples, Q a query
2: ▷ best max information gain
3: ∈ Qb := best(ρ(∈ Q)) ▷ E.g., No information gain obtained
4: if stopCriteria(∈ Qb) then
5: T := leaf(majority_class(E))
6: else
7: Conj ← Qb \ Q
8: E1 ← {e ∈ E | e ∧ B |= Qb}
9: E2 ← {e ∈ E | e ∧ B ¥= Qb}
10: buildTree(Left, E1, Qb);
11: buildTree(Right, E2, Q)
12: T ← nodei(Conj, Left, Right)
13: return T ▷ The built tree
14: end procedure
Learning: Logical Decision Trees

intend(put, A, B), clear(A)

+!put(X,Y) : clear(X) & clear(Y)
<- move(X,Y).

clear(B)  failure

success  failure

drop(put(X,Y)) :- .intend(put(X,Y)) & not clear(X).
drop(put(X,Y)) :- .intend(put(X,Y)) & not clear(Y).
Problems 3: Implementation [Guerra et al., 2010]

```
InternalAction
+ suspendIntention(): boolean
+ execute(TransitionSystem, Unifier, Term[]): Object

DefaultInternalAction
+ suspendIntention(): boolean
+ execute(TransitionSystem, Unifier, Term[]): Object

getCurrentCtxt
+ execute(TransitionSystem, Unifier, Term[]): Object

getCurrentBels
+ execute(TransitionSystem, Unifier, Term[]): Object

getCurrentInt
+ execute(TransitionSystem, Unifier, Term[]): Object

getLearnedCtxt
+ execute(TransitionSystem, Unifier, Term[]): Object

changeCtxt
+ execute(TransitionSystem, Unifier, Term[]): Object

setTilde
+ execute(TransitionSystem, Unifier, Term[]): Object

execTilde
+ execute(TransitionSystem, Unifier, Term[]): Object

addDropRule
+ execute(TransitionSystem, Unifier, Term[]): Object

setLearningMode
+ execute(TransitionSystem, Unifier, Term[]): Object

setSMLearningMode
+ execute(TransitionSystem, Unifier, Term[]): Object
```
Implementation (JILDT): Extending Plans

@put
+!put(X,Y) : true <-
  jildt.getCurrentInt(I);
  jildt.getCurrentBels(Bs);
  +intending(I,Bs);
  move(X,Y);
  -intending(I,Bs);
  +example(I,Bs,succ).
Implementation (JILDT): When to learn?

@put_failCase
-!put(X,Y) : intending(put(X,Y), Bs) <-
  -intending(I,Bs);
  +example(I,Bs,fail);
  !learning(put);
  +example_processed.
Implementation (JILDT): Learning

@learning
+!learning(P): true <-
   .print("Trying to learn a better context...");
   jildt.setTilde(P);
   jildt.execTilde(false,false);
   jildt.getLearnedCtxt(P,LC,F);
   !learningTest(P,LC,F).

lunes 7 de noviembre de 11
A simple experiment: \(+!\text{put}(X,Y) \leftarrow \text{move}(X,Y)\).
Results: Rationality, Noise, and Latency.

The JILDT library provides the extensions to AgentSpeak (L) required for defining Intentional learning agents. Using the library, it was easy to implement a single-minded class of agents. We obtained a better understanding of the inductive method that will enable us to improve it. For instance, using the initial query of Tilde to avoid the use of lookaheads, reducing the number of candidates while computing the logical decision tree. Experimental results suggest that rationality is enhanced if the training examples represent not only the beliefs of the agent when the intention was adopted but also what was believed when it failed in order to minimize the effects of the latency in noise perception. Also, incremental versions of the inductive method are now envisioned, as well as social learning protocols exploiting distributed knowledge. A logic for intention revision based on Intentional learning is also envisaged. Multi-modal decision making problems offers more elaborated scenarios for experimentation on Intentional learning. In this problems, the agents decide collectively what to do based on different points of view. The preferences about these points of view can be expressed as plans and so be the subject of reconsideration based on experiences.
Results: Modified failure examples.
Current and Future Work

• JILDT is going to be released as a public library under GNU license, after some debugging and redesign (execTilde becomes a set of plans, and the action is broken in more atomic internal actions: Different strategies for learning, i.e., selecting best candidates, etc.) [González-Alarcón, et al. 2011]

• JILDT is going to be tested within a more realistic scenario: Sharing vehicles to reduce CO2 emissions [J-MADeM Grimaldo, et al. 2011, 2012]


• A discretization method is possibly required for the urban mobility scenario: an open problem given that training examples arrives incrementally and they are scarce. [Guerra, et al. 2012] : Learning from Data Streams, Randomized Binary Search Trees.
Current and Future Work

• Social Learning

Examples = \( E^+ \cup \{ e^- \} \)

\[ \forall e \, H \models e \]

\[ H \ s.t. \ \forall e \left\{ \begin{array}{l}
H \land e \land B \models + \\
H \land e \land B \not\models -
\end{array} \right. \]
References


References


Dedicated to

Amal El Fallah & Henry Soldano

& My students at the Maestría en IA
Contact

Dr. Alejandro Guerra Hernández

Professor - SNI Nivel I - Perfil Deseable

Universidad Veracruzana
Facultad de Física e Inteligencia Artificial
Departamento de Inteligencia Artificial
Sebastián Camacho No. 5
Xalapa, Ver., México 91000
T (+52) 228 817-29-57
F (+52) 228 817-28-55

www.uv.mx/aguerra
aguerra@uv.mx • aguerrahdz@gmail.com