Multi-Agent Systems

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Felix Lindner, and Thorsten Engesser

Summer Term 2017

Course outline

BDI Agent

function BDI-AGENT(*percept*)
    global *beliefs*, *desires*, *intentions*
    beliefs ← UPDATE-BELIEF(*beliefs*, *percept*)
    desires ← OPTIONS(*beliefs*, *intentions*)
    intentions ← FILTER(*beliefs*, *intentions*, *desires*)
    action ← MEANS-END-REASONING(*intentions*)
    beliefs ← UPDATE-BELIEF(*action*)
    return *action*
end function

- BDI agents start out with some beliefs and intentions.
- Intentions are goals the agent has actually chosen to bring about (can be adopted and dropped).
- Beliefs and intentions constrain what the agent desires.
- Together, B, D, and I determine the agent's future intentions.

Signatures of main processes

- The alternatives for action (options) for an agent is a set of desires dependent on the agent's beliefs and its intentions:

$$options : 2^{Bel} \times 2^{Int} \rightarrow 2^{Des}$$

- To select between competing options, an agent uses a filter function. This choice depends on the agent's beliefs, current options (desires), and intentions:

$$filter : 2^{Bel} \times 2^{Des} \times 2^{Int} \rightarrow 2^{Int}$$

$\Rightarrow$Prior intentions serve as input! They provide a filter of admissibility for options, and thereby "provide a [...] purpose for deliberation, rather than merely a general injunction to do the best." (Bratman, 1987, p. 33)

Intentions: Main properties

- Intentions drive means-ends reasoning: If I adopt an intention, I will attempt to achieve it.
- Intentions persist: Once adopted they will not be dropped until achieved, deemed unachievable, or reconsidered.
- Intentions constrain future deliberation: Filter of admissibility. Options inconsistent with current intentions will not be entertained.
- Intentions influence beliefs upon which future practical reasoning is based: Rationality requires that I believe that I can achieve my intentions.

Comparison: Intention vs. Desire

- Desires, similar to intentions, are states of affairs considered for achievement (or actions considered for execution), i.e., basic preferences of an agent.
- Unlike desires, intentions involve a commitment to bringing them about.
- Unlike desires, intentions must be consistent.

(Bratman, 1990, after Wooldridge, p. 67)

My desire to play basketball this afternoon is merely a potential influence of my conduct this afternoon. It must vie with my other relevant desires [. . .] before it is settled what I will do. In contrast, once I intend to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions.

Role in explanations

- "I want to have some icecream, and I believe there is icecream in the freeze, and I choose to have some icecream, therefore, I go to the freeze to get some icecream."
- Each of these three clauses constitutes an adequate explanation.
- Beliefs, desires, and intentions are reason-giving forces.

BDI Frameworks

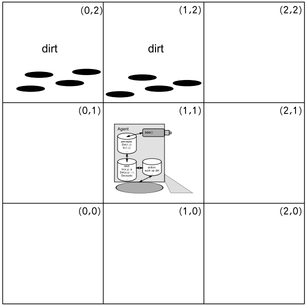
- Just to name a few
  - Jason: http://jason.sourceforge.net/
  - 3APL: https://en.wikipedia.org/wiki/3APL
  - 2APL: http://apapl.sourceforge.net/
  - JADEX: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/
  - GOAL: https://goalapl.atlassian.net/wiki
- Different technologies, e.g., Prolog-style knowledge bases vs. XML files vs. Java Objects
- Different formalizations of BDI, e.g., AgentSpeak, GOAL

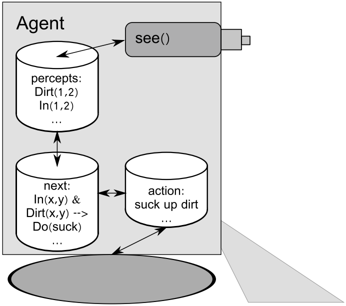
Cognitive Agents in GOAL

- GOAL emphasizes programming cognitive agents.
- Cognitive agents maintain a cognitive state that consists of knowledge and goals.
  - Knowledge: Facts the agent believes are true.
  - Goals: Facts the agent wants to be true.
- Cognitive state is represented in some knowledge representation (KR) language.
- Cognitive agents derive their choice of action from their knowledge and goals.

Example: The Vacuum World

- Percepts: dirt, orientation (N, S, E, W)
- Knowledge: In/2, dirt/0, clean/0. initial KB: In(0, 0), ¬*clean*
- Goal: clean [*Note: clean cannot be perceived but must be inferred!*]
- Actions: suck, step forward, turn right (90°)





Programming language GOAL

- Mind-body metaphor:
  - Agents (mind) are connected to controllable entities (body) living in some environment.
  - Agents receive percepts from the environment through their controlled entities.
  - Agents decide what the controlled entities will do.

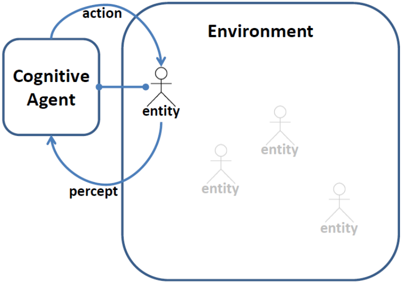
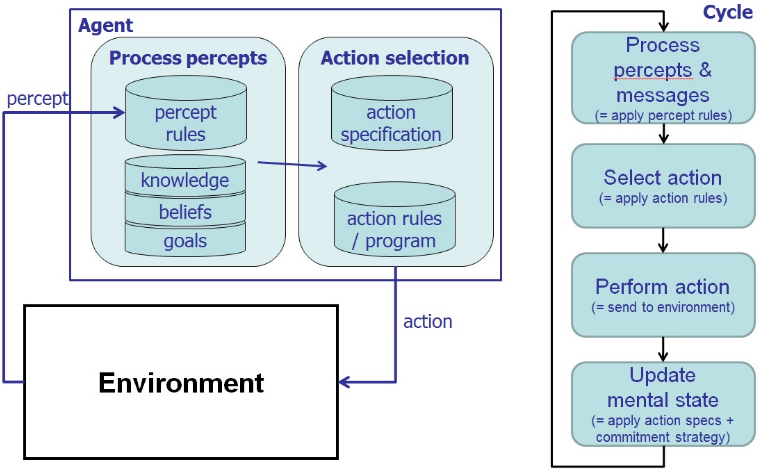


Fig.: Source [1]

- Controlled entities: A car in a Nagel-Schreckenberg-Simulation, a bot in Unreal Tournament, a robot, ...

GOAL Execution Cycle



Knowledge-based Systems: Motivation

```
printColor(snow) :- !, write(“It’s white”).
printColor(grass) :- !, write(“It’s green”).
printColor(soccerGround) :- !, printColor(grass).
printColor(X) :- write(“Hello world”).
```

versus

```
color(snow, white).
color(grass, green).
color(X, Y) :- madeOf(X, Z), color(Z, Y).
madeOf(soccerGround, grass).
printColor(X) :- color(X, Y), !, write(“It’s “), write(Y), write(“.”).
printColor(X) :- write(“Hello world”).
```

⇒Single- vs. multi-purpose, Cognitive penetrability

Example originally from Brachmann & Levesque (2004)

Declarative Knowledge Bases

- Classical formalism for knowledge representation: First-Order Logic (FOL)
- Ontological assumption: World consist of objects and relations between these objects.
- FOL syntax
  - Predicate Symbols: Beautiful/1, MotherOf/2, Between/3
  - Terms:
    - Constant Symbols: john, mary, cat-7
    - Function Symbols: f, g, ...
    - Variables: x, y, ...
  - Quantifiers: $\forall, \exists$
  - Connectives: $\wedge, \vee, \neg, \rightarrow$, ...
- “Maria is the mother of John’s (only) girlfriend.”
  - $motherOf(maria, girlfriend(john))$
  - $\forall X [girlfriend(X, john) \rightarrow motherOf(maria, X)]$

FOL Semantics

- Structures: $\mathcal{A} = (U, I)$
  - Variables and Constants: $I(c) \in U$
  - Function Symbols (n-ary): $I(f) : U^n \rightarrow U$
  - Predicate Symbols (n-ary): $I(P) \subseteq U^n$
- Example
  - $\mathcal{A}_1 = (U_1 = \{maria, susi, john\}, I_1), I_1(maria) = maria, I_1(susi) = susi, I_1(john) = john, I_1(girlfriend) = \{john \mapsto susi\}, I_1(motherOf) = \{(maria, susi)\}$
  - $\mathcal{A}_2 = (U_2 = U_1, I_2)$ similar to $\mathcal{A}_1$ but $I_2(motherOf) = \{(maria, john)\}$
  - We want to be able to say that structure $\mathcal{A}_1$ is a model for the formula $motherOf(maria, girlfriend(john))$ and $\mathcal{A}_2$ is not.

Models and Satisfiability

- $\mathcal{A} \models P(t_1, \ldots, t_n)$ iff. $(I(t_1), \ldots, I(t_n)) \in I(P)$
- $\mathcal{A} \models \neg\varphi$ iff not $\mathcal{A} \models \varphi$
- $\mathcal{A} \models (\varphi \wedge \psi)$ iff $\mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$
- $\mathcal{A} \models (\varphi \vee \psi)$ iff $\mathcal{A} \models \varphi$ or $\mathcal{A} \models \psi$ (or both)
- $\mathcal{A} \models \exists x(\varphi)$ iff $\mathcal{A}_{[x/d]} \models \varphi$ for some $d \in U$
- $\mathcal{A} \models \forall x(\varphi)$ iff $\mathcal{A}_{[x/d]} \models \varphi$ for all $d \in U$

- $\mathcal{A}$ is a model of φ iff $\mathcal{A} \models \varphi$.
- φ is satisfiable iff $\mathcal{A} \models \varphi$ for some $\mathcal{A}$.
- φ is valid iff $\mathcal{A} \models \varphi$ for all $\mathcal{A}$.
- φ entails ψ iff every model of φ is also a model of ψ.

Prolog

- GOAL uses Prolog as knowledge representation formalism
  - Based on Horn fragment of First-Order Logic, programs evaluated by logical proof. Prolog adds procedures for arithmetics and input/output handling.
- Knowledge Base
  - Rules: motherOf(maria, X) :- girlfriend(X, john).
  - Facts: girlfriend(susi, john).
- Sample Queries
  - ? :- motherOf(maria, susi). yes
  - ? :- motherOf(maria, bernie). no
  - ? :- motherOf(maria, C). C = susi.
  - ? :- motherOf(maria, susi), not(motherOf(maria, bernie)). yes

Prolog: SDL Resolution by Example (Idea)

- Program: motherOf(maria, X) :- girlfriend(X, john).
  girlfriend(susi, john).
- Query: ? :- motherOf(maria, susi).
- Remember Entailment: The program entails the query iff every model of the program is a model of the query, $Program \models Query$.
  - I.e., none of the program's models is a model of the negation of the query.
  - I.e., there is no model for the conjunction of the program and the negation of the query.
  - Thus, the entailment can be proven by showing that $Program \wedge \neg Query \models \perp$

Prolog: SDL Resolution by Example (Proof)

- Program: motherOf(maria, X) :- girlfriend(X, john).
  girlfriend(susi, john).
- Query: ? :- motherOf(maria, susi).
- Show: $\{\neg gi(X, j) \vee mo(m, X), gi(s, j), \neg mo(m, s)\} \models \perp$

Prolog: SDL Resolution by Example (Proof)

UNI FREIBURG

- Program: motherOf(maria, X) :- girlfriend(X, john). girlfriend(susi, john).
- Query: ? :- motherOf(maria, susi).
- Show: $\{\neg gi(X,j) \vee mo(m,X), gi(s,j), \neg mo(m,s)\} \models \bot$

1. $\neg mo(m,s), \neg gi(X,j) \vee mo(m,X) \leadsto_{[X/s]} \neg gi(s,j)$
2. $\neg gi(s,j), gi(s,j) \leadsto_{[]} \bot$

- $\Rightarrow$There is no structure that is a model for both the program and the negation of the query. $\Rightarrow$Every model of the program will not be a model for the negation of the query. $\Rightarrow$Every model of the program will be a model of the unnegated query. $\Rightarrow$The program entails the query.

Prolog: Queries with Variables

- Program: motherOf(maria, X) :- girlfriend(X, john).
  girlfriend(susi, john).
- Query: ? :- motherOf(maria, C).
- Idea: Ask for answer(C) :- motherOf(maria, C)
- $\{\neg gi(X,j) \vee mo(m,X), gi(s,j), \neg mo(m,C) \vee answer(C)\}$

Prolog: Queries with Variables

- Program: motherOf(maria, X) :- girlfriend(X, john).
  girlfriend(susi, john).
- Query: ? :- motherOf(maria, C).
- Idea: Ask for answer(C) :- motherOf(maria, C)
- $\{\neg gi(X,j) \vee mo(m,X), gi(s,j), \neg mo(m,C) \vee answer(C)\}$

1. $\neg mo(m,C) \vee answer(C), \neg gi(X,j) \vee mo(m,X) \leadsto_{[C/X]}$
   $answer(X) \vee \neg gi(X,j)$
2. $answer(X) \vee \neg gi(X,j), gi(s,j) \leadsto_{[X/s]} answer(s)$

Back to GOAL

- We will try to avoid programming in Prolog, but we will make use of it:
  - Adding facts to the agent's KB: **insert**(at(1, 2))
  - Removing facts from the agent's KB: **delete**(at(1,1))
  - Adding goals to the agent's KB: **adopt**(at(7,7))
  - Asking what the agent believes about some fact:
    - Am I at position (1,1)? **bel**(at(1, 1))
    - Where am I? **bel**(at(X, Y))
  - Writing rules:
    - **forall bel**(at(X1, Y1)), **percept**(at(X2, Y2)) **then** **delete**(at(X1, Y1) **+** **insert**(at(X2, Y2)).
    - **if bel**(at(1, 1), lectureAt(1, 1)), **not**(**goal**(enlightened)) **then** sleep.
    - **If goal**(at(X1, Y1)), **bel**(at(X1, Y2), Y2 > Y1, D is Y2 - Y1) **then** goNorth(D).
  - For more, study the GOAL manual [1], `https://goalapl.atlassian.net/wiki/`.

Live Programming

- We want to do first steps towards programming an agent for the Wumpus world.

1. Ontology Design
   1. Identify percepts
   2. Identify environment actions
   3. Design an ontology to represent the agent's environment
   4. Identify the goals of agents
2. Strategy Design
   1. Write event rules
   2. Write action specifications
   3. Determine action selection strategy
   4. Write decision rules

Course outline

Literature

- Hindriks, K. V., Programming Cognitive Agents in GOAL, Technical Manual, 2017, https://goalapl.atlassian.net/wiki/.
- Brachmann, R. J. & Levesque, H. J., Knowledge Representation and Reasoning, 2004, Morgan Kaufmann Publishers.
- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, second edition, Prentice Hall, 2003.