Programming with Goals (3)

M. Birna van Riemsdijk, TU Delft, The Netherlands

GOAL slides adapted from MAS course slides by Hindriks

4/24/11

Outline

- GOAL: sensing
- GOAL: instantaneous & durative actions
- GOAL: program structure
- Goal interactions
- Modularity, multiple agents, organizational reasoning

1.

GOAL: Sensing

Sensing

- Agents need sensors to:
  - explore the environment when they have incomplete information (e.g. Wumpus World)
  - keep track of changes in the environment that are not caused by itself
- GOAL agents sense the environment through a perceptual interface defined between the agent and the environment
  - Environment generates percepts
  - Environment Interface Standard: EIS (Hindriks et al.)

Percept Base

- Percepts are received by an agent in its percept base.

- The reserved keyword `percept` is wrapped around the percept content, e.g. `percept(block(a)).`

- Not automatically inserted into beliefs!

Processing Percepts

- The percept base is refreshed, i.e. emptied, every reasoning cycle of the agent.
- Agent has to decide what to do when it perceives something, i.e. receives a percept.
- Use percepts to update agent's mental state:
  - Ignore the percept
  - Update the beliefs of the agent
  - Adopt/drop a new goal

Updating Agent's Mental State

One way to update beliefs with percepts:

- First, delete everything agent believes.
  Example: remove all `block` and `on` facts.

- Second, insert new information about current state provided from percepts into belief base.
  Example: insert `block` and `on` facts for every `percept(block(…))` and `percept(on(…))`.

Assumes that environment is **fully observable** with respect to `block` and `on` facts.
Downside: not very efficient…

Percept Update Pattern

A typical pattern for updating is:

Rule 1

If the agent

- **perceives** block X is on top of block Y, and
- does **not believe** that X is on top of Y

Then **insert** `on(X,Y)` into the belief base.

Rule 2

If the agent

- **believes** that X is on top of Y, and
- does **not perceive** block X is on top of block Y

Then **remove** `on(X,Y)` from the belief base.

Percepts and Event Module

- Percepts are processed in GOAL by means of event rules, i.e. rules in the event module.

```
event module{
 program{
    <…
      rules
    …>
 }
}
```

- Event module is executed every time that agent receives new percepts.

Implementing Pattern Rule 1

INCORRECT!

Rule 1

If the agent
- **perceives** block X is on top of block Y, and
- does **not believe** that X is on top of Y

Then **insert** `on(X,Y)` into the belief base.

```
event module {
 program{
   % assumes full observability.
   if bel(percept(on(X,Y)), not(on(X,Y))) then insert(on(X,Y)).
   …
 }
}
```

Note: percept base is inspected using the bel operator, e.g. `bel(percept(on(X,Y)))`.

Implementing Pattern Rule 1

Rule 1

If the agent **perceives** block X is on top of block Y, and does **not believe** that X is on top of Y, then **insert** `on(X,Y)` into the belief base.

We want to apply this rule ***for all*** *percept instances that match it!*

Content Percept Base

```
percept(on(a,table))
percept(on(b,table))
percept(on(c,table))
percept(on(d,table))
…
```

```
event module {
 program{
   % assumes full observability.
   forall bel(percept(on(X,Y)), not(on(X,Y))) do insert(on(X,Y)).
   …
 }
}
```

Implementing Pattern Rule 2

Rule 2

If the agent

- **believes** that X is on top of Y, and
- does **not perceive** block X is on top of block Y

Then **remove** `on(X,Y)` from the belief base.

```
event module {
 program{
   % assumes full observability.
   forall bel(percept(on(X,Y)), not(on(X,Y))) do insert(on(X,Y)).
   forall bel(on(X,Y), not(percept(on(X,Y)))) do delete(on(X,Y)).
 }
}
```

1. We want that **all** rules are applied!
   By default the event module applies all rules in linear order.
2. Note that none of these rules fires if nothing changed.

Initially… Agent Knows Nothing

- In most environments an agent initially has no information about the state of the environment, e.g. Tower World, Wumpus World, …
- Represented by an empty belief base:

```
beliefs{
}
```

- There is no need to include a belief base in this case in a GOAL agent.
- It is ok to simply have no belief base section.

Summarizing

- Two types of rules:
  - `if <cond> then <action>.`
    is applied at most once (if multiple instances chooses randomly)
  - `forall <cond> do <action>.`
    is applied once *for each* instantiation of parameters that satisfy condition.
- Main module by default:
  - checks rules in **linear order**
  - applies **first** applicable rule (also checks action precondition!)
- Event module by default:
  - Checks rules in **linear order**
  - Applies **all** applicable rules (rules may enable/disable each other!)
- Program section modifiers: `[order=random]`, `[order=linear]`, `[order=linearall]`, `[order=randomall]`
- Built-in actions: insert, delete, adopt, drop.

2.

GOAL: Instantaneous & Durative Actions

Instantaneous versus Durative

- Instantaneous actions
  Actions in the *Blocks World* environment are instantaneous, i.e. they do not take time. *Wumpus World* actions are of this type as well.

- Durative actions
  Actions in the *Tower World* environment take time. *When a GOAL agent sends an action to such an environment, the action will not be completed immediately.*

Durative Actions and Sensing

- While durative actions are performed an agent may receive percepts.
- Useful to monitor progress of action.
- UT2004 Example:

  Other bot is perceived while moving.

Specifying Durative Actions

- delayed effect problem
- solution: "no" postcondition
  - results of action are handled by event rules
- Postcondition may be "empty": `post { }`
- Better practice is to indicate that you have not forgotten to specify it by using `post { true }`.

3.

GOAL: Program Structure

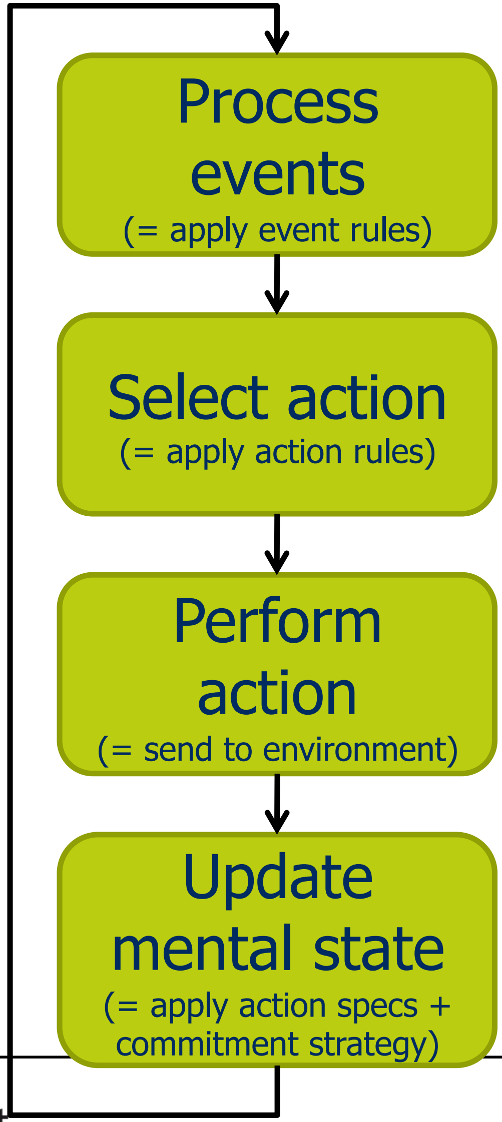
Structure of GOAL Program

```
init module {
  <initialization of agent>
}

main module{
  <action selection strategy>
}

event module {
 <percept processing>
}
```

GOAL Interpreter Cycle



Also called reasoning or deliberation cycle.

GOAL's cycle is a classic sense-plan-act cycle.

Sections in Modules

1. knowledge{...}
2. beliefs{...}
3. goals{...}
4. program{...}
5. actionspec{...}

- Init module: all sections optional, globally available
- Main & event module: 2 not allowed; 4 obligatory; 1,3,5: optional

- At least event or main module should be present

4.

Goal Interactions

Positive and Negative Interactions

- Agents may have multiple goals that could interact when trying to pursue these goals simultaneously
- Positive interaction
  - benefits can be obtained from simultaneous pursuit

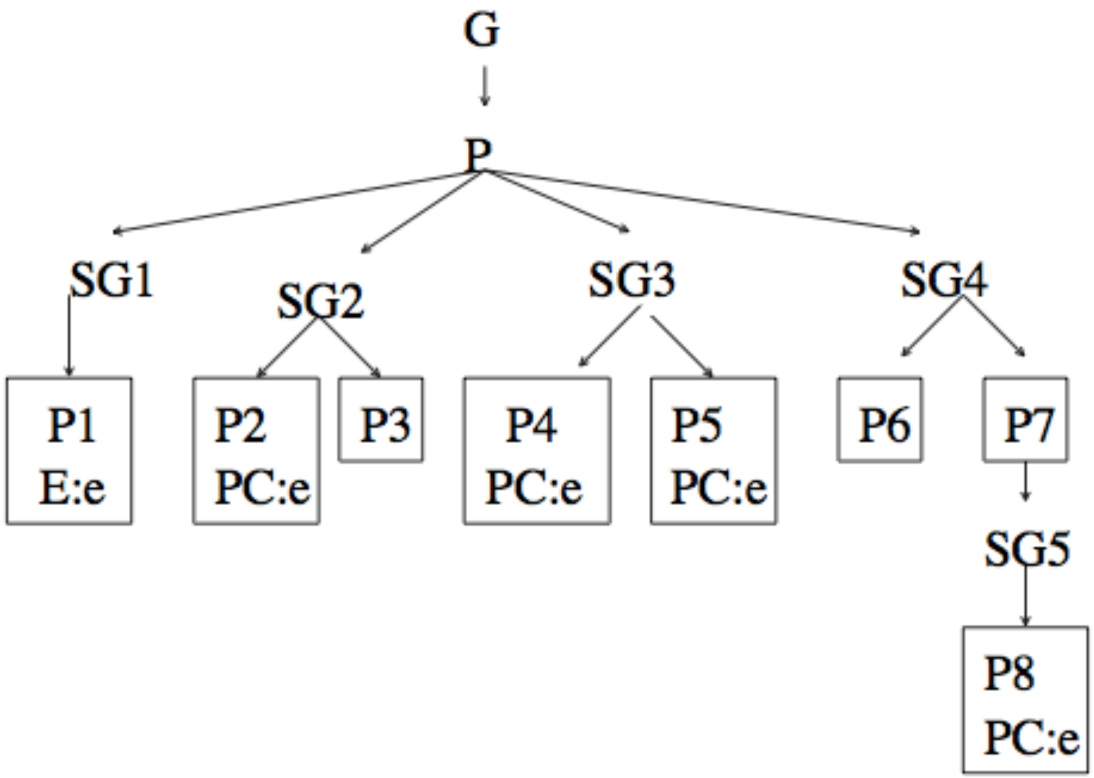
John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting and exploiting positive goal interaction in intelligent agents. In *AAMAS '03: Proceedings of The 2nd International Conference on Autonomous Agents and Multiagent Systems, pages 401-408, 2003.*

- Negative interaction
  - pursuit of one goal negatively impacts possibility to pursue other goal; conflicts between goals
  - BDI logics: goals are consistent...

Negative Interactions (1)

John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *IJCAI '03: Proceedings of the International Joint Conference on Artificial Intelligence, pages 721-726, 2003.*

- Goal representation: goal-plan tree



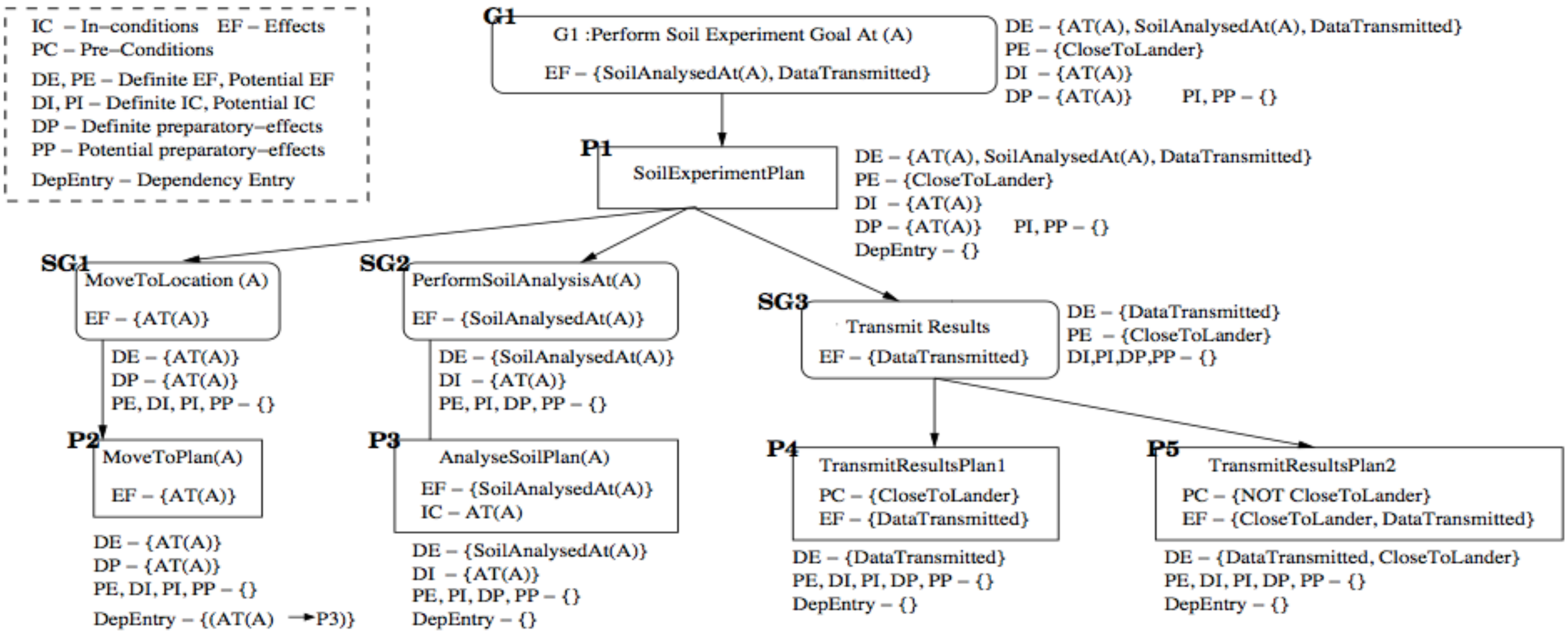
Negative Interactions (2)

Interference can occur:

- When an in-condition is made false while a plan or goal is executing, causing the plan or goal to fail.
- When a previously achieved effect is made false before a plan or goal that relies on it begins executing, preventing the goal or plan from being able to execute.

Negative Interactions (3)

Interaction tree: annotate GPT with interaction summaries



Goals in Conflict (1)

M. Birna van Riemsdijk, Mehdi Dastani, John-Jules Ch Meyer. Goals in Conflict: Semantic Foundations of Goals in Agent Programming. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(3):471-500. 2009. © Springer-Verlag.

- Logic-based representation of goal conflicts
- Goal base: set of propositional formulas
  - three semantics for deriving G ϕ
- Goal base: set of goal inference rules $\beta, \kappa^+, \kappa^- \Rightarrow \phi$
  - derive goal ϕ; agent believes β; has goals κ+; conflict with goals κ-
  - translation to default rules:

$$\kappa^+ = \{\phi_1, \ldots, \phi_m\} \quad \kappa^- = \{\psi_1, \ldots, \psi_n\}$$

$$t(\kappa^+, \kappa^- \Rightarrow \chi) = \{\phi_1 \wedge \ldots \wedge \phi_m : \neg\psi_1, \ldots, \neg\psi_n, \chi / \chi\}$$

Goals in Conflict (2)

- Goal base: set of goal inference rules $\beta, \kappa^+, \kappa^- \Rightarrow \phi$
  - derive goal ϕ; agent believes β; has goals κ+; conflict with goals κ-
  - translation to default rules
  - Gϕ follows from goal base iff ϕ follows from one of the extensions of resulting default theory
  - Conflicting goals in different extensions
  - Conflict is user-defined, not only inconsistency

5.

Modularity, Multiple Agents, Organizational Reasoning

Modularity in Agent Programming

- Central issue in software engineering
- Increased understandability of programs
- Busetta et al. (ATAL'99): capability
  - cluster of components of a cognitive agent
- Braubach et al. (ProMAS'05): extension of capability notion
- Van Riemsdijk et al. (AAMAS'06): goal-oriented modularity
  - idea: modules encapsulate information on how to achieve a goal
  - dispatch (sub)goal to module

Modules in GOAL

- User-defined modules, next to init, main and event
- Idea: focus attention on (part of) goal
- Use action rules to call module
  - if goal condition in action rules, corresponding goals become (local) goals of module
  - different exit policies: after doing one action; when local goals have been achieved; when no actions can be executed anymore; using explicit exit-module action
- See also Hindriks (ProMAS'07)

Multi-Agent System in GOAL

- .mas2g file: launch rules to start multiple agents
- action `send(Receiver, Content)` to send messages
  - mailbox semantics: inspected using `bel` operator
- declarative, imperative and interrogative "moods"
- Hindriks, Van Riemsdijk (ProMAS'09): communication semantics based on mental models

Organization-Aware Agent Programming

- How to program organization-aware agents?

Not Discussed

- Novel goal feature
  - ```
    listall L <- <goal condition>
    do {<action rules}.
    ```
- Empirical Software Engineering for Agent Programming
  - see PRIMA'09, PRIMA'10 (Van Riemsdijk & Hindriks)
    ```

Summary

- Sensing
  - percept base
  - inspect using bel(percept(...))
- Goal interactions
  - positive & negative interaction
  - framework for conflicting goals based on default logic
- Modularity
  - modules to focus on and achieve goals