Operational Semantics of Goal Models in Adaptive Agents

Mirko Morandini¹, Loris Penserini² and Anna Perini¹

¹FBK IRST – Center for Information Technology, Trento, Italy
²Department of Computer Science, Universiteit Utrecht, NL
Outline

- Background and motivations
  - Goal models in software engineering
  - Goals in agent-oriented programming
  - Work objective
- Semantics for goal models at run-time
  - Semantics for “leaf”-goals [Riemsdijk08]
  - Semantics for goals in goal trees
- A small example
- Conclusions & future work
Goal models in Sw. Engineering

- from Goal-Oriented Requirements Engineering
  - Capture stakeholders' objectives
  - Analyse and structure them
  - Decompose goals, identify alternatives
  - Identify tasks (plans/capabilities) to perform, to achieve a goal

- used in KAOS, i*, many AOSE methodologies: Tropos, Prometheus, MaSE, Ingenias,… but most AOSE methodologies “loose” the concept of goal in the later development phases!

Research question:

How to use this knowledge to shift decision making (evaluation of alternatives) from design- to run-time, to gain in autonomy, for the development of adaptive and fault-tolerant systems?
Goals in agent-oriented programming

- Jason, 2APL, Jadex, Jack:
  - BDI-architecture: Goals, Plans, Beliefs
  - Represent “operationalised” goals, with possible plans to achieve them (goal model “leaf level”).
  - Plans can contain activities to execute and other goals to achieve.
  - Various goal types for a specific run-time behaviour (achieve, maintain, perform,…) [Dastani06]

Research question:

How can we deal with goal models at run-time?
From goal models to run-time

Maintain goal models also at implementation and run-time!

**Previous work**

- **Tropos4AS**: extends the AOSE methodology **TROPOS** for modelling properties of adaptive systems [Morandini08]:
  - goal types
  - conditions to the environment
- **t2x**: automated mapping of Tropos4AS goal models to Jadex BDI agents [PenseriniAAMAS07]
Work Objective

- Goal models in most AOSE methodologies, but “lost” in the later development phases
- Agent languages: goals, but no support for goal structures
- We have an (informal) mapping of goal models to code

Try to formalise the intended behaviour of the satisfaction process for a goal model!

Goal models at run-time – motivation:
- Maintain high-level design information and traceability of the requirements
- Use this knowledge to shift design decisions (evaluation of alternatives) to run-time to gain in autonomy, for the development of adaptive and fault-tolerant systems
Semantics for leaf goals [Riemsdijk08]


Unified representation of operational semantics for the different goal types available in current agent programming languages.

- Abstract architecture for goals
  - possible goal states
  - operational semantics defined by transition rules

\[ \langle c, \text{ACTIVATE} \rangle \in E \quad B \models c \]
\[ \langle B, g(C, E, \text{SUSPENDED}, \epsilon) \rangle \rightarrow \langle B, g(C, E, \text{ACTIVE}, \epsilon) \rangle \]

Formalisation of common goal types

e.g.: “Achieve-goal” with satisfaction condition \( s \) and failure condition \( f \):

\[ A'(s, f) \equiv g(\{\}, \{\langle s \lor f, \text{DROP} \rangle, \langle \text{true}, \text{ACTIVATE} \rangle \}) \]
Semantics for non-leaf goals

Challenges:

- Semantics for goal AND-OR decompositions,
- Interplay between subgoal satisfaction and the satisfaction of the achievement conditions for different goal types,
- Customisable formalisation to capture different satisfaction behaviours.
Semantics for non-leaf goals

Extend [Riemsdijk08] for non-leaf goals in goal models

"Active" state extended to

- "Active, deliberate" (AD): get applicable subgoals
- "Active, undefined" (AU): subgoal achievement taking place, result still undefined
- "Active, succeeded" (AS): "provisional" success state. Subgoal achievement succeeded, evaluate goal achievement conditions
- "Active, failed" (AF): "provisional" failure state. Subgoal achievement failed, evaluate goal achievement conditions

Transition rules – example for OR:

In state AU, try to achieve a subgoal, if it succeeds, go to AS
The behaviour of the different goal types can be defined by defining the conditions linked to the transition actions.

\[ E: \text{conditions evaluated when the list of subgoals to achieve is empty} \]
\[ C: \text{conditions evaluated when the list of subgoals is not empty} \]

**Achieve-Goal**

\[
A(s, f) \equiv g(E, C), \quad \text{with } E = H \cup \{\neg s \lor f, \text{FAIL}\}
\]
\[
\text{and } C = H \cup \{(f, \text{FAIL}), (\neg s, \text{RETRY})\}
\]
\[
H = \{(true, \text{ACTIVATE}), (f, \text{DROPFAILURE}), (s, \text{SUCCEED}), (s, \text{DROPSUCCESS}), (\neg s, \text{REACTIVATE})\}
\]

**Perform-Goal**

\[
P \equiv g(E, C), \quad \text{with } E = C = \{\text{true, ACTIVATE}, (\text{true, DROPFAILURE}), (\text{true, DROPSUCCESS})\}
\]
A small example

Cleaner Robot:
Should clean a room, with satisfaction condition “floor clean”.

A scenario:
- Robot cleans the floor, achieving “dryCleaning”.
- Sweeping performed, still some dirt spots on the floor! The agent tries “wetCleaning”.
- Cleaning fails, because it runs out of water, → but dirty area already cleaned, → top goal “clean room” achieved with success!
Satisfy the achieve-goal **clean room**

3) deliberate\((g, B)\) gives back subgoals \textit{wetCleaning} \((WC)\) and \textit{dryCleaning} \((DC)\)

\[ \langle B, g(C, E, AD, \emptyset) \rangle \rightarrow \langle B, g(C, E, AU, deliberate(g, B)) \rangle \]

\[ \text{[deliberateE]} \]

4) \textit{dryCleaning} performed with success

\[ disp(G, DC) \rightarrow G \cup \{ DC \} \quad \langle B, G \cup \{ DC \} \rangle \rightarrow \langle B', G \rangle \]

\[ \langle B, disp(G, DC) \rangle \rightarrow \langle B', G \rangle \quad B' \models success(DC) \]

\[ \langle B, g(C, E, AU, \{ DC, WC \}) \rangle \rightarrow \langle B', g(C, E, AS, \{ WC \}) \rangle \]

5) still some dirt spots on the floor! only \(\neg s, \text{RETRY}\) is satisfied.

\[ \Gamma \neq \emptyset \quad \langle c, \text{RETRY} \rangle \in C \quad B \models c \]

\[ \langle B, g(C, E, AS, \Gamma) \rangle \rightarrow \langle B, g(C, E, AU, \Gamma) \rangle \quad \text{[Retry]} \]

6) \textit{wetCl.} is pursued and fails, but condition “floor clean” is now true.

\[ \neg \exists \langle d, \text{FAIL} \rangle \in C. (B \models d) \quad \langle c, \text{SUCCEED} \rangle \in E \quad B \models c \]

\[ \langle B, g(C, E, AU, \emptyset) \rangle \rightarrow \langle B, g(C, E, AS, \emptyset) \rangle \quad \text{[cond-succeedE]} \]

\[ g(C, E, AS, \emptyset) \in G \quad \langle c, \text{DROPSUCCESS} \rangle \in E \quad B \models c \]

\[ \langle B, G \rangle \rightarrow \langle B \cup \text{success}(g), G \setminus \{ g(C, E, AS, \emptyset) \} \rangle \quad \text{[drop-successE]} \]

7) Finally the goal is dropped and its success is annotated in the belief base.
Conclusions & Future Work

- We formalised the run-time behaviour of non-leaf goals, defining the interplay between goal decompositions and goal types.
- The proposed ‘abstract architecture’ can be used to define various goal types and achievement/failure handling behaviours.
- Maintain high-level design information and traceability of the requirements.
- Shift decisions (evaluation of alternatives) from design to run-time to gain in autonomy, for the development of adaptive and fault-tolerant systems.
- The operational semantics can be a starting point:
  - to formalise a mapping from goal models to software agents,
  - to implement a middle layer for goal models in AOP frameworks,
  - for validation and simulation of goal models at design time.
- Goal models at run-time also provide a basis for run-time goal acquisition and goal model modification.
Thank you!

Questions and suggestions are welcome!
Further readings & references


- [Dastani06] M. Dastani et al., Goal types in agent programming. AAMAS06, 2006.
- Some transitions guided by transition actions (Succeed, Fail, Retry,…) linked to a condition $c$, evaluated on the agent’s belief $B$.

**Example transition rules for OR-decomposition**

- in state AU: try to achieve a subgoal, if it fails, remain in AU.

- in state AU, try to achieve a subgoal, if it succeeds, go to AS

- in AU or AF, if success condition $c$ is true and failure condition $d$ false, go to AS

- in AU, if no more subgoals to achieve and success condition true, go to AF.