From Means-End Analysis to Proactive Means-End Reasoning

Luca Sabatucci and Massimo Cossentino
The Vision

MUSA RUNNING SYSTEM

GOALS

ACTIVE GOALS

POD

WHAT

HOW
Goal Oriented Requirements

- A goal is a *state of affair* that an actor wants to achieve
The State of the World

• A state of the world \( W^t \) is a dynamic object that describes the current “state of affair” – or better: what the system knows about

• We implement \( W^t \) by employing a set of semantically coherent first order logic facts.

• \( W^t \) describes a closed-world in which everything is not explicitly declared is assumed to be false.
Operative Implementation of Goal

- Goal's TC is the Condition that must hold in $W^t$ in order the agent can actively pursue that goal.
- Goal's FS is the Condition that must hold in $W^t$ in order the goal can be marked as addressed.
- GOALSpec is a language conceived to inject goal specifications in a human-friendly format

**USER- GOAL 01**

WHEN schedule(Usr,Meeting) THE system SHALL PRODUCE canceled(Meeting) OR confirmed(Meeting)

**USER- GOAL 02**

WHEN pending(Meeting) AND meeting_datetime(DT) AND attendee(Meeting,A) THE system SHALL PRODUCE notified(A,Meeting,DT)
AI-Style CAPABILITIES

- The system owns a set of capabilities, i.e. atomic and self-contained actions
- The effect of a capability is an endogenous evolution of $W^t$
- The system is aware of its capabilities
- and it is aware of ‘when’ and ‘how’ to use a capability in order to address a desired result

<table>
<thead>
<tr>
<th>Name</th>
<th>PROPOSAL_MAIL_SENDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputParams</td>
<td>Question : TEXT,</td>
</tr>
<tr>
<td></td>
<td>RESPONSE_ID : STRING</td>
</tr>
<tr>
<td></td>
<td>UserMail : STRING</td>
</tr>
<tr>
<td>OutputParams</td>
<td>NONE</td>
</tr>
<tr>
<td>Constraints</td>
<td>format(UserMail,</td>
</tr>
<tr>
<td></td>
<td>RFC_5322_Address_Specification)</td>
</tr>
<tr>
<td>Pre-Condition</td>
<td>email(Usr, UserMail)</td>
</tr>
<tr>
<td>Post-Condition</td>
<td>notified(Question, Usr)</td>
</tr>
<tr>
<td>Evolution</td>
<td>$evo = { add(notified(Msg, Usr)), add(mailed(UserMail, Question)), add(questioned(Usr, ResponseId)) }$</td>
</tr>
</tbody>
</table>

ABSTRACT DESCRIPTION OF A CAPABILITY
Bridging WHAT and HOW

The PROACTIVE MEANS-END REASONING is the problem of finding the minimal set of capabilities (called PMR Solution) that can fully address a goal model, given the current $W'$. 

WHAT (goal spec)

WHEN pending(Meeting) AND meeting datetime(DT) AND attendee(Meeting,A) THEN THE system SHALL PRODUCE notified(A,Meeting,DT)

HOW (capabilities)

PROPOSAL MAIL SENDER
COLLECT_MAIL_RESPONSES
GOOGLECALENDAR_CHECK

MUSA RUNNING SYSTEM

To Call Participants 
To Check Timetables 
To Provide Meeting Scheduling 
To Schedule Meetings
To Select Schedule
To Choose Location
To Send Mail
To Mail Participants
To Run Website
The PMR Solution

• The Proactive Means-End Reasoning is different from
  – A scheduling problem: it does not require an exact timing of the activities
  – A planning problem: it does not require to create a plan for executing the activities

• The system will contextually evaluate which capability to use, when, and how.
  – The same capability in the PMR_Solution will eventually used 0..n times
The proposed algorithm

• It is based on the ability to discover if a capability can be used for addressing a goal (or contributing to)

• The principle is that of matching Goal’s TC/FS and Capability’s Pre/Post/Evolution

• This is possible if goals and capabilities share
  – The same formalism
  – The same background ontology
The State of World as Common Formalism

- **SUBJECT**
- **TRIGGER CONDITION**
- **FINAL STATE**
- **GOAL**
  - **CONDITION**
  - **STATE OF THE WORLD**
  - **EVOLUTION**

- **CAPABILITY**
  - **PRE CONDITION**
  - **POST CONDITION**

- **CONDITION**
  - **to test over**

- **wants**
- **is active when**
- **is addressed when**
- **is executable when**
- **is correctly executed when**
- **generates**
- **modifies**

GOAL

SUBJECT

TRIGGER CONDITION

FINAL STATE

GOAL

CAPABILITY

PRE CONDITION

POST CONDITION

CONDITION

STATE OF THE WORLD

EVOLUTION

wants

is active when

is addressed when

is executable when

is correctly executed when

generates

modifies

to test over
The Ontology as Common Background

Meeting

- **Schedule**
- **Initiator**
- **Attendee**
- **Confirmed**
- **Pending**
- **Canceled**
- **MinAttendees**

**Meeting DateTime**

- ISO 8601 DateTime

**Contact Info**

- Email
- Skype Id

**Calendar**

- Timeslot
  - **Free**
  - **Busy**

**User**

**Attendee**

- **Notified**
- **usr**
- **msg**

**Accepted**

**Rejected**
Common Background (II)

USER-GOAL_01
WHEN pending(Meeting) AND meeting datetime(DT) AND attendee(Meeting,A) THE system SHALL PRODUCE notified(Meeting, A)
Planning-Like Space Exploration

**USER-GOAL_01**
WHEN pending(Meeting) AND meeting_datetime(DT) AND attendee(Meeting,A) THE system SHALL PRODUCE
notified(A,Meeting)

<table>
<thead>
<tr>
<th>Name</th>
<th>Calendar_Timeslot_Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>calendar(Usr,UserAccount)</td>
</tr>
<tr>
<td>Post-condition</td>
<td>free(Usr,TimeSlot) OR busy(Usr,TimeSlot)</td>
</tr>
<tr>
<td>Evolution</td>
<td>evo={ add(verified_ts(Usr,TimeSlot)) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Append_Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>free(Usr,TimeSlot)</td>
</tr>
<tr>
<td>Post-condition</td>
<td>busy(Usr,TimeSlot)</td>
</tr>
<tr>
<td>Evolution</td>
<td>evo={ add(notified(Usr,Meeting)) }</td>
</tr>
</tbody>
</table>
Space Exploration (II)

USER-GOAL_01
WHEN pending(Meeting) AND meeting_datetime(DT) AND attendee(Meeting,A) THE system SHALL PRODUCE notified(A,Meeting)

<table>
<thead>
<tr>
<th>Name</th>
<th>Proposal Mail Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>email(Usr,MailAddress)</td>
</tr>
<tr>
<td>Post-condition</td>
<td>questioned(Usr,Meeting)</td>
</tr>
<tr>
<td>Evolution</td>
<td>evo={ add(questioned(Usr,Meeting) ) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Collect Mail Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>email(Usr,MailAddress)</td>
</tr>
<tr>
<td>Post-condition</td>
<td>accepted(Usr,Meeting) OR rejected(Usr,Meeting)</td>
</tr>
<tr>
<td>Evolution</td>
<td>evo={ add(notified(Usr,Meeting) ) }</td>
</tr>
</tbody>
</table>
Final Remarks – Self Adaptation

• Self Adaptation is the result of a loop in which the Proactive Means-End Reasoning is executed every time (with different $W^I$)
  – New goal-model is injected
  – An existing goal changes
  – A capability fails:
    • software failure and exceptions
    • the generated $W$ is different from the expected one
    • the connected resource is no more available
  – New capability is injected
Future Works

• The planning algorithm is inefficient
  – In some circumstances it requires an exponential time to complete.
  – We are planning to explore many strategies for improving it
    • SAT solvers, optimized planning and case base reasoning

• Scalability is limited
  – We are studying a better integration with a Cloud architecture (Open-Stack)

• To date the use of a static ontology enables the agent's
  – it is also a limit when capabilities/goals evolve one independently from the others.
  – In order to enable distributed development-teams, we are integrating linguistic techniques for dealing with
    • conceptual ambiguities and linguistics flaws, similarities and synonyms.
Questions?

https://github.com/icar-aose/MUSA

sabatucci@pa.icar.cnr.it