Agent Deliberation in an Executable Temporal Framework (2011)

by Michael Fisher

Ian Helmke
Daniel Kreymer
Karl Wiegand
9/21/2012
Agenda

- Motivation
- Definitions
- Prior Work
- Contributions:
  a. A Propositional Temporal Logic
  b. Examples and Implementation in Prolog
  c. Representation and Ordering of Goals
  d. Correctness Arguments of Deliberation
- Summary
General Motivation

- Advances in networking and grid computing
- Widespread adoption of multi-agent systems
- Distributed systems w/o central control
- Autonomous agents that:
  - adapt to unforeseen circumstances
  - negotiate/cooperate with other agents
  - overcome problems
  - progress towards goals
Field-Specific Motivation

● **State of the art:**
  ○ Theories of agency, negotiation, and cooperation
    ■ BDI and KARO
  ○ Most agent systems are developed directly
  ○ Agent shells exist (e.g. JACK Intelligent Agents)

● **Current issues:**
  ○ Lots of implementations, but no clear semantics
  ○ Lack of clarity regarding how deliberation works
  ○ Real systems are unrelated to agent theories

● **Goal:**
  ○ Help bridge gap between theory and implementation
Definitions

● **Agent:**
  "an autonomous software component, communicating and cooperating with other agents in order to achieve common goals"

● **Autonomous:**
  "responsible for deliberating over a range of possibilities and for deciding on their own course of action"

● **Deliberation:**
  "changing the order in which the agent attempts to achieve goals"
Definitions, cont.

- **Agent Theory:**
  "a logical basis for representing deliberative agents"

- **Agent Programming Language:**
  "an intuitive language for implementing deliberative agents"

- **Agent Software Engineering:**
  "developing agent-based systems based on executable specifications"
Prior Work: BDI

- **BDI: Beliefs - Desires - Intentions**
  - **Beliefs**: What does the agent think about the world?
    - Includes inference rules
    - Enables forward-chaining
  - **Desires**: What does the agent want to accomplish?
    - Also called goals
    - Must be internally consistent (e.g. live vs. die)
  - **Intentions**: What has the agent decided to do?
    - Includes plans, which can include sub-plans
  - **Events**: What triggers the agent to update its BDI?
Prior Work: KARO

- Knowledge
  - Epistemic modal logic using Kripke structures

- Abilities vs. Opportunities
  - What can an agent do, in general?
  - What can the agent do right now?

- Results
  - Expected results based on knowledge
  - Real results based on sensor data

- Motivation
  - Wishes, goals, and commitments
Prior Work - JACK

- Java framework for modeling and executing agents
  - Language for modeling agents
    - Distinct from a logic
  - Framework for executing agents
- Uses BDI to represent agents
- Deliberation is opaque
- Language is distinct from a logical representation
A Propositional Temporal Logic

Temporal operators:

\[ \Diamond \varphi \] is satisfied now if \( \varphi \) is satisfied at some moment in the future

\[ \Box \varphi \] is satisfied now if \( \varphi \) is satisfied at all moments in the future

\[ \varphi U \psi \] is satisfied now if \( \varphi \) is satisfied from now until a future moment when \( \psi \) is satisfied

\[ \diamond \varphi \] is satisfied now if \( \varphi \) is satisfied at the next moment in time
Temporal Logic Basics

A model $\sigma$ is a sequence of moments or states

$$\sigma = s_0, s_1, s_2, s_3, \ldots$$

where each state $s_i$ is a set of propositions that are satisfied in the $i$th moment in time; and

$$\langle \sigma, i \rangle \models \varphi$$

denotes the truth of formula $\varphi$ in model $\sigma$ at moment $i \in \mathbb{N}$ in time. If there is some $\sigma$ such that $\langle \sigma, 0 \rangle \models \varphi$, then $\varphi$ is satisfiable. If $\langle \sigma, 0 \rangle \models \varphi$ for all models, then $\varphi$ is valid and written $\models \varphi$. 
Temporal Logic Details

\[ \langle \sigma, i \rangle \models p \iff p \in s_i \quad \text{[where } p \in \mathcal{P} \text{]} \]

\[ \langle \sigma, i \rangle \models \text{true} \]

\[ \langle \sigma, i \rangle \models \text{start} \iff i = 0 \]

\[ \langle \sigma, i \rangle \models \varphi \land \psi \iff \langle \sigma, i \rangle \models \varphi \text{ and } \langle \sigma, i \rangle \models \psi \]

\[ \langle \sigma, i \rangle \models \varphi \lor \psi \iff \langle \sigma, i \rangle \models \varphi \text{ or } \langle \sigma, i \rangle \models \psi \]

\[ \langle \sigma, i \rangle \models \neg \varphi \iff \langle \sigma, i \rangle \not\models \varphi \]

\[ \langle \sigma, i \rangle \models \Diamond \varphi \iff \text{there exists a } k \in \mathbb{N} \text{ such that } k \geq i \text{ and } \langle \sigma, k \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \Box \varphi \iff \text{for all } j \in \mathbb{N}, \text{ if } j \geq i \text{ then } \langle \sigma, j \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \varphi U \psi \iff \text{there exists a } k \in \mathbb{N}, \text{ such that } k \geq i \text{ and } \langle \sigma, k \rangle \models \psi \]

\[ \quad \text{and for all } j \in \mathbb{N}, \text{ if } i \leq j < k \text{ then } \langle \sigma, j \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \varphi W \psi \iff \text{either } \langle \sigma, i \rangle \models \varphi U \psi \text{ or } \langle \sigma, i \rangle \models \Box \varphi \]
Example: Searcher Agent

**Interface Definition:**

In: new_search, add_resources, terminate
Out: found, need_resources

**Internal Definition (partial):**

```plaintext
start -> ~searching
new_search -> ♦ searching
(searching ^ new_search) ->
  ○ (found v need_resources)
```
Implementation: Overview

● Basic problems:
  ○ When $\Diamond \varphi$ is executed, the system must attempt to ensure that $\varphi$ eventually becomes true.
  ○ Eventualities may not be able to be satisfied immediately, so we need to keep a list of all outstanding eventualities.

● Basic approach:
  ○ Forward-chain to determine eventualities
  ○ At each state, try to satisfy as many eventualities as possible, starting with the oldest
  ○ Set a limit on the age, in steps, of eventualities, and backtrack when eventualities are stagnant
3 sets of clauses: Initial, Step, and Eventuality

1. Assign symbols to Initial; label as $S_0$ and let $E_0 = \emptyset$.
2. Given $S_i$ and $E_i$, construct $S_{i+1}$ and $E_{i+1}$:
   a. Construct Step constraints using $\bigcirc$ clauses
   b. Forward-chain all the $\Diamond$ clauses for $E_{i+1}$
   c. For each eventuality in $E_{i+1}$, beginning with the oldest, check for consistency with Step.
   d. Choose an assignment $S_{i+1}$ consistent with Step.
2. Constructing $S_{i+1}$ and $E_{i+1}$ (cont.):
   
   e. [Loop Check] If an eventuality within $E_{i+1}$ has occurred identically in the previous $N$ states, then fail and backtrack to a previous choice point; if none are available, then terminate.
   
   f. go to (2)

- Loop check prevents endless oscillation
- Multiple, simultaneous activities allows:
  - Reaction: immediate actions to stimuli
  - Planning: Longer term processes in background
Deliberation: Vehicle Navigation

• Agent has:
  ○ information about local terrain
  ○ information concerning target destinations
  ○ motivations:
    ■ get to a destination
    ■ avoid obstacles
    ■ continue moving until a destination is reached
    ■ etc.

• Agent must deliberate over:
  ○ actions to take (e.g. movement)
  ○ new goals to generate (e.g. add a new destination)
  ○ revision of its current goals
Deliberation: Dining and Wishing

- Human-like agent with goals:
  - eat_lunch
  - sleep
  - be_famous

- Potential deliberation:
  - try (be_famous); fail (?) and choose another goal
  - try (eat_lunch); fail (?) and create goal make_lunch
  - try (make_lunch)
  - etc.

- Need capability to dynamically reorder goals
- Basic system uses fixed strategy: oldest first
Deliberation: Definition

- Deliberation can be modeled as:
  - Input: List of eventualities
  - Output: List of eventualities

- Can be smarter and also take past history:
  \[ E_{i+1} = \text{priority\_function}(E_i, \text{History}) \]

- Compared to BDI:

  \[
  \text{Desires} = [\diamond \text{be\_famous}, \diamond \text{sleep}, \diamond \text{eat\_lunch}, \diamond \text{make\_lunch}]
  \]

  \[
  \text{Intentions} = [\diamond \text{be\_famous}, \underline{\diamond \text{eat\_lunch}}, \diamond \text{sleep}, \diamond \text{make\_lunch}]
  \]

  \[
  \text{Attempt} = [\underline{\diamond \text{make\_lunch}}, \diamond \text{eat\_lunch}, \diamond \text{sleep}, \underline{\diamond \text{be\_famous}}]
  \]
Deliberation: Correctness

● Problem of Fair Ordering:
  ○ We want to make sure no eventuality is "starved" and never attempted because other eventualities are continually prioritized in front.

● Basic Solution:
  ○ Attempt the oldest eventuality first.

● Problem of Finiteness:
  ○ We want to make sure that we don't have eventualities that exist forever.

● Basic Solution:
  ○ Try every eventuality on every execution.
  ○ Backtrack if an eventuality is stagnant after N turns.
Summary

- Many theories and implementations:
  - BDI
  - KARO
  - JACK Intelligent Agents

- No formal engineering approaches

- This paper ties theory with implementation:
  - A propositional temporal logic
  - Examples and implementation in Prolog
  - Representation and ordering of goals
  - Correctness arguments of deliberation

- Future work will include multi-agent support