Agent Programming Languages with Declarative Goals

Qualifying Oral Presentation

Shakil M. Khan

Overview

- Motivation
- Agent theories
- Agent architectures
- Agent programming languages
- Declarative goals in agent programs
- Open problems
Background

• Research on intelligent agents –
  – Agent theories: model mental attitudes and how they relate to each other and to the agents’ behavior
  – Agent architectures: focus on resource boundedness
  – Agent programming languages (APLs): programming languages to realize agents

Motivation – Declarative Goals

• One goal of APL research – to show a one-to-one correspondence between the underlying theory and the abstract interpreter
• Primary reason for the gap between the underlying theory and the interpreter:
  – Agent theories use declarative goals (AKA ‘goals-to-be’)
  – APLs use procedural goals, (AKA ‘goals-to-do’ or plans)
  – Declarative goals are more expressive
    • Commitment to a declarative goal vs. to a procedural goal
• Declarative goals have many other practical advantages
Agent Theories – Informational Attitudes

• Informational attitudes (knowledge, belief)
  – Almost always modeled using KD45 logic
  – Two issues:
    • Informational prerequisites of actions (AKA epistemic ability)
    • effects of actions on these attitudes (belief update)
  – Dynamics: belief revision (and contraction) – new information are
    allowed to be inconsistent with existing knowledge

Agent Theories – Motivational Attitudes

• Motivational attitudes (choices/desires, goals, intentions)
  – Intentions are persistent goals – achievement is tracked
  – Act as a filter for adopting newer intentions – focus practical
    reasoning
• Formalization of intentions
  – Mostly defined in terms of goals; sometimes as primitive
  – Often formalized using a normal modal KD logic with linear or
    branching time temporal logic
  – Sometimes as a non-normal modal operator (side-effect-free)
  – Others [Singh-90] specify them in terms of strategies/plans
Agent Theories

- Inter-attitudinal constraints:
  - Realism: agents should not intend something that is believed to be impossible
  - Side-effect-freeness: agents should not intend all the side-effects of their intentions – problematic for normal-modal logic formalizations
- Success Theorems
- Rational Action
- Multiagent Systems:
  - Joint mental attitudes
  - Cooperation and communication

Agent Architectures

- Assumptions:
  - May have incomplete information about the world
  - Are not always able to accurately predict the effects of actions
  - Can deal with external interference
  - Don’t have arbitrary time to deliberate
- Issues include the tradeoff between
  - Commitment vs. intention reconsideration
  - Reactivity vs. deliberation
Agent Architectures – Classification

- Deliberative architectures – very expressive, but computationally complex and not reasonable in a real world
- Reactive architectures – efficient, but ineffective in environments that deviate from those expected by the designer
- Reactive plan execution architectures (e.g. PRS)
- Hybrid architectures

Reactive Plan Execution Architectures – PRS

Diagram showing the components of PRS architecture:
- Belief-base
- Goal-base
- Plan library (Rules)
- Interpretation (Reasoner)
- Intention Structure
- Environment
- Monitor
- Command Generator
Agent Programming Languages – Classification

- Deductive reasoning languages
  - More expressive and strongly grounded into the theory
  - Computationally complex, poor scalability and modularity, no support for distribution of computation
  - E.g.: AGENT0, Concurrent METATEM, ConGolog, FLUX, etc.
- Reactive plan-execution languages
  - Mostly PRS-based
  - Efficient and modular, but limited expressiveness
  - E.g.: AgentSpeak(L), 3APL, CAN

Agent Programming Languages – Issues

- Explicit representation of BDI concepts (beliefs, intentions, etc): some do not (e.g. Golog, Concurrent METATEM)
- Planning with look-ahead search (Golog, CAN-PLAN) vs. reactive plan selection and execution (3APL, AgentSpeak(L))
- Incomplete knowledge and sensing actions (e.g. Golog, FLUX)
- Model of belief update, world dynamics/theory of action (Golog, FLUX)
- Specification of behavior
- Support for communication, interaction protocols (JACK), verification (GOAL, Dribble)
Advantages of Declarative Goals

- Expressiveness – atomic vs. complex formulae as goals
- Decoupling plan success/failure and goal achievement/failure
- Detecting fortuitous achievement of goals
- Roles in communication: goal delegation, handling incoming requests, etc.
- Rational behavior: reasoning about interferences between goals; planning from scratch

APLs with Declarative Goals

- PRS-style APLs w/declarative goals:
  - Store declarative goals and procedural intentions (plans) in two different databases (Dribbble, GD-3APL, CAN-PLAN)
  - Select plans using declarative goals
  - Use goals to decouple goal success/failure from plan success/failure
  - Jadex: treats declarative goals as active components
- Sardiña & Shapiro-03:
  - Users specify a high-level non-deterministic program and a set of prioritized declarative goals
  - Interpreter searches for an execution of this program that maximizes achievement of goals
APLs with Declarative Goals – Issues

• Types of goals handled:
  – Goal types:
    • Achievement goals
    • Maintenance goals
    • Perform goals
    • Query goals
  – State formulae as goals vs. temporal goals
  – Consistent vs. Inconsistent goal-bases
  – Prioritized goals vs. flat representation of goals

APLs with Declarative Goals – Issues

• Consistency of intentions:
  – Intentions (plans) generated by an APL should be consistent with the intended declarative goals
  – Goals should be consistent with each other; plans should be consistent with the goals and with each other
  – Unfortunately, no APL handles this issue
  – Separate databases for plans and goals – makes it even harder to ensure this consistency
APLs with Declarative Goals – Issues

• Representation of means-ends relationship:
  – How to capture the dependency between sub-goals and their parent-goals?
  – Sub-goals (and sub-sub-goals, etc.) need to be dropped when parent-goal succeeds/is dropped
  – Mostly modeled syntactically

Open Problems

• Developing a uniform formalization of intended goals and plans
  – Consistent, side-effect free, introspectable, etc.,
  – Expressive enough to capture conditional, prioritized, temporal intentions
• Modeling dynamics of intention
  – Satisfy properties such as maintaining consistency, unachievable intentions get dropped, etc.
  – Handle general forms of intention contraction and revision
• Managing the means-ends relationship between goals and sub-goals
Open Problems

• Formalizing more realistic look-ahead planning
  – Current formalizations do not ensure consistency between new plan and other concurrent goals/plans
  – How to specify and use search control information to speed-up
• Achieving a balance between deliberation and execution
  – need a way to determine :
    – When employing deliberation (in contrast to just selecting and executing plans) has good payoff
    – How much look-ahead is necessary
• Developing efficient representation and reasoning for :
  – Incomplete knowledge
  – Nested beliefs and goals (required in a multiagent context)

Sardiña & Shapiro Framework
[S&S-03]

• Grounded on an expressive formal action theory – the situation calculus
• Expressive representation of intentions:
  – Temporal (achievement and maintenance goals)
  – Prioritized
• Formalizes an on-demand look-ahead planning operator
[S&S-03] as a Starting Point

- Interpreter searches for an execution of given program – not really goal directed
- Modify to incorporate PRS-style rules to
  - Model goal driven behavior and speed up deliberation
  - Allow arbitrary modification of intended plans
- Formalize:
  - More realistic model of dynamics of intention (currently investigating)
  - Model of dependency between goals and sub-goal (currently investigating)
  - An on-demand look-ahead planning mechanism
  - Model of communication