Executing Model-based Programs Using Graph-based Temporal Planning

Prof. Brian C. Williams
February 23rd, 2004
16.412/6.834 Cognitive Robotics

based on [Kim, Williams & Abramson, IJCAI01]

Outline

• Model-based Programming
• Cooperative Vehicle Missions
• The Reactive Model-based Programming Language (RMPL)
• Temporal Plan Networks (TPN)
• Activity Planning (Kirk)
• Unifying Activity and Path Planning
Why Model-based Programming?

Leading Diagnosis:
- Legs deployed during descent.
- Noise spike on leg sensors latched by monitors.
- Laser altimeter registers 50ft.
- Begins polling leg monitors to determine touch down.
- Latched noise spike read as touchdown.
- Engine shutdown at ~50ft.

Mars 98:
- Climate Orbiter
- Mars Polar Lander

Create Embedded Languages That Reason on the Fly from Commonsense Models

Model-based Programs Interact Directly with State

Embedded programs interact with plant sensors/actuators:
- Read sensors
- Set actuators

Model-based programs interact with plant state:
- Read state
- Write state

Programmer must map between state and sensors/actuators.

Model-based executive maps between sensors, actuators to states.

Image courtesy of JPL.
Example: The model-based program sets the state to thrusting, and the deductive controller . . . .

- Deduces that thrust is off, and the engine is healthy
- Plans actions to open six valves
- Deduces that a valve failed - stuck closed
- Determines that valves on the backup engine will achieve thrust, and plans needed actions.

RMPL Model-based Program

System Model

- Generates target goal states conditioned on state estimates
- Tracks least cost goal states
- Tracks likely plant states
- Observations
- State goals
- Plant

Titan Model-based Executive

Control Program

- Executes concurrently
- Preempts
- Queries (hidden) states
- Asserts (hidden) state
Modeling Complex Behaviors through Probabilistic Concurrent Constraint Automata

• Complex, discrete behaviors
  • modeled through concurrency, hierarchy and non-determinism.
• Anomalies and uncertainty
  • modeled by probabilistic transitions
• Physical interactions
  • modeled by discrete and continuous constraints

Outline

• Model-based Programming
• Cooperative Vehicle Missions
• The Reactive Model-based Programming Language (RMPL)
• Temporal Plan Networks (TPN)
• Activity Planning (Kirk)
• Unifying Activity and Path Planning
Cooperative Search and Rescue

- High-level vehicle coordination
- Fast Agile Maneuvering

Cooperative Mars Exploration

How do we coordinate heterogeneous teams of orbiters, rovers and air vehicles to perform globally optimal science exploration?
Properties:

- Teams exploit a hierarchy of complex strategies.
- Maneuvers are temporally coordinated.
- Novel events occur during critical phases.
- Quick responses draw upon a library of contingencies.

Self-Adaptive Languages: RAPs [Firby PhD]

- RAPS programs monitor goals and plan activities

```lisp
(define-rap
  (index (move-to ?thing ?place))
  (succeed (LOCATION ?thing ?place))
  (method
    (context (and (LOCATION ?thing ?loc)
                  (not (= ?loc UNKNOWN)))
    (task-net
      (t0 (goto ?loc) ((TRUCK-LOCATION ?loc) for t1))
      (t1 (pickup ?thing)((TRUCK-HOLDING ?thing) for t2)
        ((TRUCK-HOLDING ?thing) for t3))
      (t2 (goto ?place) ((TRUCK-LOCATION ?place) for t3))
      (t3 (putdown ?thing))))
  (method
    (context (LOCATION ?thing UNKNOWN))
    (task-net
      (t0 (goto WAREHOUSE))))
)```
Self-Adaptive Languages: RAPs [Firby PhD]

• RAPS Programs recover by selecting from functionally redundant methods

\[
\text{define-rap}
\begin{align*}
\text{index (move-to \( \text{?thing} \) \( \text{?place} \))} \\
\text{succeed (LOCATION \( \text{?thing} \) \( \text{?place} \))} \\
\text{method}
\end{align*}
\begin{align*}
\text{context}(\text{and (LOCATION \( \text{?thing} \) \( \text{?loc} \))} \\
\text{not (= \( ?\text{loc} \) UNKNOWN)))} \\
\text{task-net}
\begin{align*}
(t0 \text{ (goto \( ?\text{loc} \)) ((TRUCK-LOCATION ?loc) for t1))} \\
(t1 \text{ (pickup ?thing))((TRUCK-HOLDING ?thing) for t2)} \\
((\text{TRUCK-HOLDING ?thing) for t3}) \\
(t2 \text{ (goto \( ?\text{place} \)) ((TRUCK-LOCATION ?place) for t3}) \\
(t3 \text{ (putdown ?thing))))
\end{align*}
\end{align*}
\begin{align*}
\text{method}
\end{align*}
\begin{align*}
\text{context (LOCATION \( ?\text{thing} \) UNKNOWN))} \\
\text{task-net}
\begin{align*}
(t0 \text{ (goto WAREHOUSE))))
\end{align*}
\end{align*}
\]

Self-Adaptive languages: RAPS

• RAPS Exploits contingencies by performing functionally redundant method selection
  – Methods are chosen based on the current situation.
  – If a method fails, another is tried instead.
  – Tasks do not complete until satisfied.
  – Methods can include monitoring subtasks that deal with contingencies and opportunities.

• Limitations
  • Goals must be explicitly observable
  • Methods selected reactively
  ➢ Method selection may dig itself into a hole.

Create Languages with planner-like capabilities
Control Sequencer

Deductive   Controller

Environment  Model

CommandsObservations

Selects consistent threads of activity from redundant methods

Schedules and Dispatches Activities Dynamically

location goals

location estimates

Tracks location

Finds least cost paths

Executive

• pre-plans activities
• pre-plans paths
• dynamically schedules [Tsmardinós et al.]

RMPL Model-based Program   Kirk Model-based Executive

Control Program

• Executes concurrently
• Preempts
• non-deterministic choice
• $A[l,u]$ timing
• $A$ at $l$ location

Environment Model

Executive

• pre-plans activities
• pre-plans paths
• dynamically schedules [Tsmardinós et al.]

Outline

• Model-based Programming
• Cooperative Vehicle Missions
• Reactive Model-based Programming Language (RMPL)
• Temporal Plan Networks (TPN)
• Activity Planning (Kirk)
• Unifying Activity and Path Planning
Reactive Model-based Programming

Idea: Describe team behaviors by starting with a rich concurrent, embedded programming language (RMPL, TCC, Esterel):

- c
- If c next A
- Unless c next A
- A, B
- Always A

- Sensing/actuation activities
- Conditional execution
- Preemption
- Full concurrency
- Iteration

- Add temporal constraints:
  - A [l,u]
  - Timing

- Add choice (non-deterministic or decision-theoretic):
  - Choose {A, B}
  - Contingency

Example Enroute Activity:

Enroute

- Rendezvous
- Corridor 1
- Corridor 2
- Rescue Area
Group-Enroute()[l,u] = {
choose {
    do {
        Group-Traverse-
        Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[l*90%,u*90%];
        } maintaining PATH1_OK,
    do {
        Group-Traverse-
        Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[l*90%,u*90%];
        } maintaining PATH2_OK
    };
    {
        Group-Transmit(OPS,ARRIVED)[0,2],
        do {
            Group-Wait(HOLD1,HOLD2)[0,u*10%]
        } watching PROCEED
    }
}

Activities:
RMPL for Group-Enroute

Group-Enroute() \{ l, u \} = \{
choose \{
   do \{
      Group-Traverse-Path(PATH1_1, PATH1_2, PATH1_3, RE_POS) \{ l*90\%, u*90\% \};
      \textit{maintaining PATH1_OK},
      do \{
         Group-Traverse-Path(PATH2_1, PATH2_2, PATH2_3, RE_POS) \{ l*90\%, u*90\% \};
         \textit{maintaining PATH2_OK}
      \};
   \};
   \{
      Group-Transmit(OPS, ARRIVED) \{ 0, 2 \},
      do \{
         Group-Wait(HOLD1, HOLD2) \{ 0, u*10\% \}
      \} \textit{watching PROCEED}
   \}
\}
Group-Enroute() \([l,u]\) = \{
choose \{
  do {
    Group-Fly-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS) \([l*90\%,u*90\%]\);
    } maintaining PATH1_OK,
  do {
    Group-Fly-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS) \([l*90\%,u*90\%]\);
    } maintaining PATH2_OK
};

{ }

Group-Transmit(OPS,ARRIVED) \([0,2]\),
do {
  Group-Wait(HOLD1,HOLD2) \([0,u*10\%]\)
} watching PROCEED
}
• How do we provide fast, temporally flexible planning?
• Graph-based planners support fast planning.
• … but plans are totally order.
• Desire flexible plans based on simple temporal networks (e.g., HSTS, Muscetola et al.).

How do we create temporally flexible plan graphs?
• Generalize simple temporal networks (temporal plan network TPN).
Outline

- Model-based Programming
- Cooperative Vehicle Missions
- Reactive Model-based Programming Language (RMPL)
- Temporal Plan Networks (TPN)
- Activity Planning (Kirk)
- Unifying Activity and Path Planning

Enroute Activity:

- Start with flexible plan representation
Enroute Activity:

- Start with flexible plan representation

Enroute Activity:

- TPN representation of Enroute activity and sub-activities
Enroute Activity:

• Add conditional nodes

Enroute Activity:

• Add temporally extended, symbolic constraints
Outline

- Cooperative Vehicle Missions
- Model-based Programming
- Reactive Model-based Programming Language (RMPL)
- Temporal Plan Networks (TPN)
- Activity Planning (Kirk)
- Unifying Activity and Path Planning

Planning Group-Enroute

To Plan:
- Instantiate Group-Enroute
Planning Group-Enroute

To Plan:
• Instantiate Group-Enroute
• Add External Constraints (Tells)

Generates Schedulable Plan

To Plan:
• Instantiate Group-Enroute
• Add External Constraints

Trace consistent trajectories
• Check Schedulability
• Satisfy and Protect Asks
Planning Example

• Find paths from start-node to end-node

Planning Example

• Not a decision-node: Follow all outarcs
Planning Example

• Not a decision-node: Follow all outarcs
Planning Example

• Decision-node: Select a single outarc

Planning Example

• Not a decision-node: Follow all outarcs
Planning Example

• Continue

Planning Example

• Not a decision-node: Follow all outarcs
Planning Example

- Continue
Temporal Constraint Consistency

• Don’t test consistency at each step.
• Only when a path induces a cycle, check for negative cycle in the STN distance graph

Example: Inconsistent
Temporal Constraint Consistency

• Backtrack to choice

Temporal Constraint Consistency

• Complete paths
How Do We Handle Asks?

Unconditional Planning:
- Guarantee satisfaction at compile time.
- Treatment similar to causal-link planning

Satisfying Asks
- Compute bounds on activities.
- Link ask to equivalent, overlapping tell.
- Constrain tell to contain ask.
Avoiding Threats

- Identify overlapping **Inconsistent** activities.

Symbolic Constraint Consistency

- Promote or demote
Outline

• Cooperative Vehicle Missions
• Model-based Programming
• Reactive Model-based Programming Language (RMPL)
• Temporal Plan Networks (TPN)
• Activity Planning (Kirk)
• Unifying Activity and Path Planning

How do we optimally select activities and paths?

Current Research:

• Perform global path planning using Rapidly-exploring Random Trees (RRTs) (la Valle).

• Search for globally optimal plan by unifying TPN & RRT graphs, and by searching hybrid graph best first.

• Refine plan using receding horizon control based on CLP (Krishnan, Williams), or MILP (How) or hybrid maneuver automata (Frazzoli, Dahleh, Feron).
Enroute Activity:

- Closer look at Group Traverse sub-activity

---

Group Traverse sub-activity:

- Traverse through **way points** to science target
Group Traverse sub-activity:

• One obstacle between nodes 4 and 5
• Two obstacles between nodes 6 and 7

Group Traverse sub-activity:

• Non-explicit representations of obstacles obtained from an incremental collision detection algorithm
RRT: Example

Planner considers rovers taking Path 1:
RRT: Example

Path 1

\( x_{\text{init}} \)

\( x_{\text{goal}} \)

\( X_{\text{obs}} \)
RRT: Example

Path 1

---

RRT: Example

Path 1
RRT: Example

Path 1
RRT: Example

Path 1

Common Node

X_{init} 4

X_{obs}

X_{goal} 5
RRT: Example

Path 1

RRT: Example

Path 1
Model-based Cooperative Programming


Solution: New middle ground between embedded programming, task decomposition execution, and temporal planning.

• Rich embedded language, RMPL, for describing complex concurrent team strategies extended to time and contingency.
• Kirk Interpreter “looks” for schedulable threads of execution before “leaping” to execution.
• Temporal Plan Network provides a flexible, temporal, graph-based planning paradigm built upon Simple Temporal Nets.
• Global optimality achieved by unifying activity planning and global kino-dynamic path planning.