Executing Reactive, Model-based Programs through Graph-based Temporal Planning

Phil Kim and Brian C. Williams,
Artificial Intelligence and Space Systems Labs
Massachusetts Institute of Technology

Mark Abramson
Draper Labs

Outline

• Cooperative Vehicle Missions
• Model-based Programming
• Reactive Model-based Programming Language (RMPL)
• Temporal Plan Networks (TPN)
• Activity Planning (Kirk)
• Optional: Hybrid Activity/Path Planning

Cooperative Mars Exploration

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

MIT Cooperative Vehicle Testbed

• Distributed Satellites: Spheres, Spheres, TechSat21
• Aerobots: Indoor blimps
• Sensing: distributed, wireless sensor net
MIT Cooperative Vehicle Testbed

- Distributed Satellites: Spheres, Spheres, TechSat21
- Aerobots: Indoor blimps
- Sensing: distributed, wireless sensor net
- Rovers: 1 ATRV Sr., 3 ATRV Jr

Outline

- Cooperative Vehicle Missions
- Model-based Programming
- Reactive Model-based Programming Language (RMPL)
- Temporal Plan Networks (TPN)
- Activity Planning (Kirk)
- Optional: Hybrid Activity/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.

Create Embedded Languages That Reason 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.

Cooperative Model-based Programming

- How do we specify the allowed behaviors of cooperative robotic networks? (RMPL)
- How do we command cooperative networks? (this talk)
- How do we monitor cooperative networks? (next talk)
Outline

- Cooperative Vehicle Missions
- Model-based Programming
- Reactive Model-based Programming Language (RMPL)
- Temporal Plan Networks (TPN)
- Activity Planning (Kirk)
- Optional: Hybrid Activity/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

Add temporal constraints:
- A [a]

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

Properties:
- Teams are focused to a hierarchy complex strategies.
- Maneuvers are temporally coordinated.
- Novel events occur during critical phases.
- Quick response draws upon a library of contingencies.

RMPL for Group-Enroute

```
Group-Enroute() [l,u] = {
    choose { 
        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
    }
    }
}
```

Example Scenario

Example Enroute Activity:

RMPL for Group-Enroute

```
Group-Enroute() [l,u] = {
    choose { 
        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
    }
    }
}
```
RMPL for Group-Enroute

Group-Enroute()

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

}\}

Conditionality and Preemption:

Sequentiality:

Concurency:

Temporal Constraints:

Non-deterministic choice:

RMPL Interpreter

• Dynamically selects among alternate executions, satisfies open conditions and checks schedulability,
• Selects execution times, monitors outcomes and plans contingencies.
  RMPL Program
  • Kirk
  • Idea
  • Titan

Model of Networked Embedded Vehicles

Mode Estimation

Reactive Planning

Observables

Commands

• How do we provide fast, temporally flexible planning?
• Graph-based planners support fast planning.
• … but plans are totally ordered.
• 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).
Kirk: Reactive Temporal Planner

- RMPL Compiler
  - Temporal Plan Network (TPN) with STN
  - Represents all RMPL executions
  - Selects schedulable execution threads of TPN
  - Plan = Execution threads related by Simple Temporal Net

Outline

- Cooperative Vehicle Missions
- Model-based Programming
- Reactive Model-based Programming Language (RMPL)
- Temporal Plan Networks (TPN)
- Activity Planning (Kirk)
- Optional: Hybrid Activity/Path Planning
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)
- Optional: Hybrid Activity/Path Planning

**Planning Example**

- Find paths from start-node to end-node

**Generates Schedulable Plan**

To Plan:
- Instantiate Group-Enroute
- Add external constraints
- Trace consistent trajectories
- Check schedulability
- Satisfy and protect asks

**Planning Group-Enroute**

To Plan:
- Instantiate Group-Enroute
- Add external constraints (Tells)

**To Plan:**
- Instantiate Group-Enroute

**Group-Enroute**

- Generates schedulable plan

**Group-Enroute**

- Instantiates Group-Enroute

**Group-Enroute**

- Completes planning

**Group-Enroute**

- Completes planning

**Group-Enroute**

- Completes planning

**Group-Enroute**

- Completes planning
Planning Example

• Not a decision-node: Follow all outarcs

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

Planning Example

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

Temporal Constraint Consistency

- Example: Inconsistent

Temporal Constraint Consistency

- Backtrack to choice
Temporal Constraint Consistency

- Complete paths

How Do Handle Asks?

- Guaranee 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)
- Optional: Hybrid Activity/Path Planning
Enroute Activity:

*Closer look at Group Traverse sub-activity*

Traverse 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*

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.*

*Perform local kino-dynamic path planning along path segments using hybrid maneuver automata (Frazzoli, Dahleh, Feron).*

RRT: Example

Path 1

Path 2
Assume rovers take Path 1:

\[ X_{\text{init}} \rightarrow X_{\text{goal}} \]

RRT: Example

Path 1

RRT: Example

Path 1

RRT: Example

Path 1

RRT: Example

Path 1

RRT: Example

Path 1

RRT: Example

Path 1
- Kirk demonstrated a cooperative scenario using UAV simulation.
- Development on Multi-Rover testbed currently in progress.
- Distributed hybrid activity/path planning in progress.
- Selected for NASA Space Technology 7, Phase A
- together with IDEA (Muscettola)
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.
- Interpreter “leaps” through flexible execution (Nicola talk).
- Current work towards unifying activity planning, global path planning and kino-dynamics.