Robust Task Execution: Procedural and Model-based

Brian C. Williams
16.412J/6.834J
March 14th, 2005

Desiderata: Robust Task-level Execution

Create Languages that are:
- Suspicious
- Monitor intentions and plans
- Self-Adaptive
- Exploits and generates contingencies
- Anticipatory
- Predicts, plans and verifies into future
- State Aware
- Commanded with desired state
- Reasons about and responds to failure

Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
- Model-based Reactive Planning

Robust Task Execution: RAPS [Firby PhD]

- RAPS Monitors Success Against Spec

```
(define-rap (move-to thing place)
  (succeed (LOCATION thing place))
  (method
    (context (and (LOCATION thing loc)
                  (not (= loc UNKNOWN)))
    (task set
      (10 (goto loc)) (TRUCK-LOCATION loc) for t1)
      (11 (pickupthing)(TRUCK-HOLDING thing) for t2)
      (TRUCK-HOLDING thing) for t3)
      (12 (goto place) (TRUCK-LOCATION place) for t3)
      (13 (putdown thing))))
  (method
    (context (LOCATION thing UNKNOWN))
    (task set
      (10 (goto WAREHOUSE))))
```

Robust Task Execution: RAPS [Firby PhD]

- RAPS Exploits contingencies by performing functionally redundant method selection

```
(define-rap (move-to thing place)
  (succeed (LOCATION thing place))
  (method
    (context (and (LOCATION thing loc)
                  (not (= loc UNKNOWN)))
    (task set
      (10 (goto loc)) (TRUCK-LOCATION loc) for t1)
      (11 (pickupthing)(TRUCK-HOLDING thing) for t2)
      (TRUCK-HOLDING thing) for t3)
      (12 (goto place) (TRUCK-LOCATION place) for t3)
      (13 (putdown thing))))
  (method
    (context (LOCATION thing UNKNOWN))
    (task set
      (10 (goto WAREHOUSE))))
```
Robust Task Execution: RAPS [Firby PhD]

- 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 to deal with contingencies and opportunities.

  Methods selected reactively
  - Model-predictive dispatch
  - Goals explicitly observable and controllable
  - Model-based execution

Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
  - Model-based Programming
  - Temporal Plan Networks (TPN)
  - Activity Planning (Kirk)
  - Unifying Activity and Path Planning
- Model-based Reactive Planning

Example: Cooperative Mars Exploration

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

Example: Cooperative Mars 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.
- Selected contingencies must respect timing constraints.

Reactive Model-based Programming

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

- 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

Corridor 2

Corridor 1

Rendezvous

Rendezvous

Rescue Area

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

Conditionality

and Preemption:

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

Temporal Constraints:

RMPL for Group-Enroute

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

Sequentiality:

Concurrency:
Non-deterministic choice:

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

How do we provide fast, temporally flexible planning for contingent method selection?

- Graph-based planners support fast planning.
- ... but plans are totally order.
- Desire flexible plans based on simple temporal networks (e.g., Constrain-based Interval Planning).

How do we create temporally flexible plan graphs?

- Augment simple temporal networks with activities & choice.
- Temporal plan network (TPN).

Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
  - Model-based Programming
  - Temporal Plan Networks (TPN)
  - Activity Planning (Kirk)
  - Unifying Activity and Path Planning
- Model-based Reactive Planning
Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
  - Model-based Programming
  - Temporal Plan Networks (TPN)
  - Activity Planning (Kirk)
  - Unifying Activity and Path Planning
- Model-based Reactive Planning

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

Generates Schedulable Plan

To Plan:
- Instantiate Group-Enroute
- Trace Trajectories
- Check Schedulability
  - Satisfy and Protect Asks
• Find paths from start-node to end-node

• Not a decision-node: Follow all outarcs

• Not a decision-node: Follow all outarcs

• Decision-node: Select a single outarc

• Not a decision-node: Follow all outarcs
**Trace Trajectories**

- Continue

**Trace Trajectories**

- Not a decision-node: Follow all outarcs

**Trace Trajectories**

- Continue

**Check Schedulability**

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

**Check Schedulability**

- Example: Inconsistent
Trace Alternative Trajectories

- Backtrack to choice

How Do We Handle Asks?

<table>
<thead>
<tr>
<th>Group-Enroute</th>
<th>Group Traverse</th>
<th>Group Wait</th>
<th>Group Transmit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4, 6]</td>
<td>[4, 6]</td>
<td>[4, 6]</td>
<td>[4, 6]</td>
</tr>
</tbody>
</table>

Unconditional planning approach:
- Guarantee satisfaction of asks 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
How do we optimally select activities and paths?

Background: Can perform global path planning using Rapidly-exploring Random Trees (RRTs) (la Vallee).

Approach:
1. Search for globally optimal activity and path plan by
   - unifying TPN & RRT graphs, and
   - by searching hybrid graph best first.
2. Refine plan using receding horizon control.

Group Traverse sub-activity:
- Traverse through waypoints 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

Path 2

X
obs

X
init

X
goal

RRT: Example
Path 1

X
init

X
goal

RRT: Example
Path 1

X
init

X
goal

RRT: Example
Path 1

X
init

X
goal

RRT: Example
Path 1

X
init

X
goal

RRT: Example
Path 1

X
init

X
goal
RRT: Example

Path 1

Common Node

Path 1

RRT: Example

Path 1

Model-Predictive Dispatch

Goal: Fast, robust, temporal execution with contingencies, in uncertain environments.

Solution: Model-Predictive Dispatch, a middle ground between non-deterministic programming 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.