tBurton: Model-based Temporal Generative Planning

David Wang
PI: Brian Williams

August 1st, 2012, Keck Institute for Space Studies
Generative Planning

initial state:

goals:

actions/model:

plan:
Applications

Intent Recognition

Service Library

- filetransfer()
- ssh()
- securetransfer()

Service Composition

Planner
Model-based Executive

Program the nominal behavior, leave the exceptional behavior to the executive.
Motivation

Overview

Example
Problem Traits
(Desired Expressivity)

• Time and concurrency are important
• Devices can transition automatically after timed intervals (timed light switch).
• Transitions in one device can cause transitions in another. (throwing a breaker switch, can turn off an appliance)
• Actions do not have to be reversible
Specify desired system states at desired times for desired durations.

An intuitive, timed finite state machine representation for system behaviors.
Planner Features

- Time: Least-commitment planning through Temporal Networks
- Robust Execution: Use Finite State Machines (Timed Concurrent Constraint Automata)
- Natural Encoding: Causal Graph Decomposition
- Fast

Continuous Operation

Risk-aware & Probabilistic Transitions

Parallel/Distributed
Read: “Event B must occur between lowerbound and upperbound time after A”
Temporal Flexibility
(Temporal Least Commitment)

Would you rather operate using the plan:

- 12:00pm – walk to subway station
- 12:15pm – take subway to airport
- 12:30pm – waste time at airport
- 2:30pm – get on plane
- 4:45pm – catch a taxi

Or:

Starting at 12:00pm

- walk to the subway station [10,20] min
- take subway to airport [10,20]
- waste time [120,180]
- fly on plane [120,180]
- catch a taxi [0,inf]
Wouldn’t it be great if…

There is an order in which to resolve those constraints.

LARGE PROBLEM

Only pair-wise complicating constraints.

decompose

only has sequential plans*
State of the art generative planners use heuristics to guide the search.
tBurton

Ideas:
- Concurrent Constraint Automata Encoding
- **Decomposition based on Causal Graphs**
- Goal Regression Search
- Decomposed Prime Implicate Generation
- Systematic Goal Orderings by Topological Sort
- Incremental Temporal Consistency Checking
Why is this a good idea?

• Threat-resolution in goal-regression planning is the slowest part.
  – Causal graphs remove a lot of constraint processing

• Heuristic forward-search planners are good at solving cycles in a causal graph.
  – Current search strategies throw away cycles.
Why is this innovative?

• Output temporal, least commitment plan.
• Increased expressiveness of system models.
• Explicit use of Causal Graph.
tBurton

- Motivation
- Overview
- Example
Example

Logistics Example:
• Objective: move packages from locations in cities to other locations.
• Planes move *between* cities
• Trucks move *in* cities
tBurton’s Encoding

Model: Timed Concurrent Constraint Automata (TCCA)
tBurton’s Encoding

Model: Timed Concurrent Constraint Automata

Package1 TCCA

\[
\text{Truck2.state=loc}5 \text{ and cmd=unload\_truck}3\_\text{loc}5 [0,1]
\]

\[
\text{Truck2.state=loc}5 \text{ and cmd=load\_truck}3\_\text{loc}5 [0,1]
\]
Use a causal graph:

Benefits:
- Decomposes/composes the problem into planning tasks suitable for heuristic based searches.
- Captures information, to prioritize planning goals.
  i.e. to move a package, we must first determine the route the package must travel, then move the trucks and planes accordingly.
Causal Graph Decomposition

Causal graph – captures device dependencies

Benefits:
- Decomposes/composes the problem into planning tasks suitable for heuristic based searches.
- Captures information, to prioritize planning goals.
  i.e. to move a package, we must first determine the route the package must travel, then move the trucks and planes accordingly.
Example

Model: TCCA with Causal Graph:

Initial State:
- pkg1.state = loc4
- truck1.state = loc2
- plane1.state = loc4

Goal:
- [0, inf]
- [2, 3]
- pkg1.state = loc8
Example

```
24
package 1
truck 1
plane 1
[0, inf]
trk1.state = loc8
pkg1.state = trk1
[5, 10]
[2, 3]
load - truck1
pkg1.state = trk1
trk1.state = loc8
pln1.state = loc6
```
Take-aways

• A model-based executive can alleviate the task of handling off-nominal behavior.

• The **product of an algorithm can support resiliency** as well as the algorithm. (Temporal networks allow flexible execution)

• We can **exploit the structure of engineered systems** to quickly plan for a reasonably expressive models.

• The plans produced reflect the **concurrent** nature of systems.