Factored Symbolic Approach to Reactive Planning

Seung H. Chung
Brian C. Williams
June 7, 2005

Motivation for Reactive Planning

- Reason for Planning
  - Anomalies
    - Environmental
    - System
  - May require repair and/or reconfiguration capabilities.

- Reason for onboard reactive planning
  - Time-critical situations
  - Communication time delays
  - Situation in which no communication available

Model-base Programs Interact Directly with States

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

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

Complexity: Programmer must map between state and sensors/actuators.
Simplification: Model-based executive maps between state and sensors/actuators.

Increase Robustness through Model-based Programming

- Raised Level of Abstraction
  - Code in terms of desired state evolution
  - Fewer lines of code
  - Less chance of introducing bugs

- Executable Specification
  - Increase robustness by synthesizing executable code from the verified specification
  - Models are the specification of the system
  - Model-based Executive operates on the model, i.e. the specification
  - The model-based embedded program is guaranteed to meet the specification

Model-based Executive: Deductive Controller

Model Reconfiguration

control nondeterministic system in a nondeterministic environment

Model-based Programming of Intelligent Embedded Systems and Robotic Explorers
[Williams et al., IEEE'03]
Past Approaches to Planning

- **General-purpose Planner**
  - Generates a sequence of commands that achieves the goal.
  - A sequence of commands lacks robustness within nondeterministic system and environment.
  - Replanning is expansive.

- **Universal Planner**
  - Maps all possible initial states to the appropriate actions.
  - State explosion problem

Assume:
- $x$ components
- $n$ number of states per component

Number of system states: $O(x^n)$

- Must replan if the goal state changes.

Recent Advances in Reactive Planning: BDD-based Universal Planning

- Ordered Binary Decision Diagrams (BDD)
  - Compact representation of Boolean functions
  - Efficient algorithms for operating on Boolean functions

- Symbolic Model Checking
  - Use of BDDs for model checking
  - Reduce the state explosion problem
  - Has been very successful [Burch et al., IEEE’90]

- Recognized the similarity of Symbolic Model Checking and Planning [Cimatti et al., ECP’97]
  - BDD-based Universal planners have been developed:
    - Strong Plan, Strong Cyclic Plan, Optimistic Planning, Etc.

Recent Advances in Reactive Planning: Burton [Williams & Nayak, IJCAI97]

- Goal-directed plan: $(\text{Current State, Goal State}) \rightarrow \text{Action}$
  - Introduced a decomposition technique that enables subgoal serialization (i.e. in essence, applies a divide-and-conquer approach to reactive planning).
    - Mitigate the state space explosion problem.
    - Enable a compact encoding of a goal-directed plan.
  - Only applicable to a limited subset of a planning problems (i.e. cannot generate a plan for a system with interdependent components).

Factored Symbolic Approach to Reactive Planning

- Unify the two complementary approaches:
  - Address the state space explosion problem at the global level through decomposition: divide-and-conquer
  - Address the state space explosion problem at the subproblem level through BDD-based planning
  - Extend the decomposition technique of [Williams & Nayak, IJCAI97] to problems with interdependent components.
  - Extend the BDD-based Universal planning technique to generate a goal-directed plan.

Outline

- Spacecraft telecommunication system
- Model: Concurrent automata
- Decomposing the Problem: Transition dependency graph
- Reactive Plan for a Subproblem: Goal-directed plan
- Reactive Plan for the Problem: Decomposed goal-directed plan
- Executing the Plan

Telecommunication Subsystem Example

- Computer
  - Controls the devices and sends data to the devices.
- Bus Controller
  - Routes the commands and the data to the appropriate devices.
- Transmitter
  - Generates a signal that corresponds to the data to be transmitted.
- Amplifier
  - Amplifies the signal and transmits it to an antenna.
Concurrent Automata (CA)

- Synchronous
  - Assume that each automaton performs a single state transition at each time step.
- Interleaved execution within a time step
  - A single main processor executes synchronous activities by interleaving.
  - Devices are not synchronized.

Interdependent Components

- Turning the transmitter on or off can generate a noise (i.e. transient signal).
- The transient signal may damage the amplifier.
- The amplified transient signal may damage other devices downstream of the amplifier.
- Constraint on the system:
  - The amplifier must be turned off before the transmitter can be turned on or off.
  - The transmitter must be turned on before the amplifier can be turned on.

BDD Encoding of a Concurrent Automaton

Transition Dependency Graph

- Transition Dependency Graph (TDG)
  - Vertex: for each automaton
  - Edge $(v, u)$: if a transition of the automaton $v$ is conditioned on the state of automaton $u$.
- Use Strongly Connected Components (SCC) algorithm to find the cyclic components.
- Compose SCC concurrent automata
  - New TDG is acyclic.
  - Serialize the subgoals in the inverse topological ordering.

Subgoal Serialization

- Goal:
  - Bus Controller = on
  - Transmitter/Amplifier #1 = (on, on)
  - Transmitter/Amplifier #2 = (off, off)
- Solve each subgoal sequentially in the inverse topological order

Composing Strongly Connected CA

- Compose all automata into a single automaton
  \[ R_{SCC} = \bigwedge_i R_i \]
**Interdependent Concurrent Transitions**

- One Transition Missing!

**Simultaneous Commanding**

- Both the transmitter and the amplifier depend on one another for the transition.
- The transmitter must be commanded “off” and the amplifier must be commanded “on” precisely at the same time.
- Due to concurrency via interleaving, simultaneous commanding cannot be guaranteed.
- If the amplifier were commanded on first, and then the transmitter is commanded off, the amplifier can be damaged.

**Assuring Proper Execution of Interdependent Transitions**

- Enforce concurrency as interleaving:
  - For a given transition, the interdependent state constraints become the pre- and post-conditions.
  - No change to all other automata that are not independent.

**Amplifier**

- Assuring Proper Execution of Interdependent Transitions
  - Inconsistencies are automatically detected when conjoining the transition relations in BDDs.
  - Similar to the Graphplan mutual exclusion rule.
    - Interference:
      - One transition deletes the precondition and/or effect of another.
    - Competing Needs:
      - Inconsistent preconditions

**Goal-directed Plan**

- Executing a goal-directed plan guarantees:
  - Progress toward the goal.
  - Finite number of actions to achieve the goal.
  - Optimal (shortest) trajectory under nominal conditions.

**Computing Goal-Directed Plan:**

- Iteratively search backward breath-first for the goal-directed rules.
  - Find (s, a, s’) that can reach s’ within 1 step
  - Find (s, a, s’) that can reach s’ within 2 steps
  - ...
  - Find (s, a, s’) that can reach s’ within n steps
Generating Goal-Directed Plan

- \( (s, a, s') \) that can reach \( s' \) within 1 step

Transition Relation

Generating Goal-Directed Plan

- \( (s, a, s') \) that can reach \( s' \) within 2 steps
  - To the previous GDP add \( (s, a, s') \) that can reach \( s' \) in 2 steps:
    - \( s \): current state
    - \( s' \): goal state that can be reached in 2 steps
    - \( a \): first control action that must be commanded to eventually reach \( s' \)

Generating Goal-Directed Plan

- \( (s, a, s') \) that can reach \( s' \) within 3 steps
  - To the previous GDP add \( (s, a, s') \) that can reach \( s' \) in 3 steps:
    - \( s \): current state
    - \( s' \): goal state that can be reached in 3 steps
    - \( a \): first control action that must be commanded to eventually reach \( s' \)

Generating Goal-Directed Plan

- \( (s, a, s') \) that can reach \( s' \) within 4 steps
  - No new \( (s, a, s') \) exists that can reach \( s' \) in 4 steps.
  - When the fixed-point is reached, generating the plan is complete.

Generating Goal-Directed Plan

- \( (s, a, s') \) that can reach \( s' \) within 4 steps

Computing Decomposed Goal-directed Plan

- For each automaton compute a GDP.

Size of DGDP

- Given
  - Number of concurrent automata: \( n \)
  - Average number of states in each automaton: \( m \)
  - Number of strongly connected components: \( l \)
  - Average number automata in a strongly connected component: \( w \)
  - Number of states for one composed automation: \( O(mw) \)
  - Size of a GDP: \( O(nm^2w) \)
  - Size of DGDP: \( O(kw^2) \)
- Approximately linear in the number of components
  - Assume \( m \) and \( w \) are constant.
  - \( O(kw^2) \) is linear in \( k \).
- Use of BDD makes each GDP even more compact.
Execution

- Achieve subgoals incrementally in the inverse topological order → subgoal serialization ordering
  1. Transmitter/Amplifier #2
  2. Transmitter/Amplifier #1
  3. Bus Controller

Execution Example

- Current State:
  - B = off
  - T/A #1 = (off, off)
  - T/A #2 = (off, off)

- Goal State:
  - B = off
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Current State:
  - B = on
  - T/A #1 = (off, off)
  - T/A #2 = (off, off)

- Goal State:
  - B = on
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Current State:
  - B = off
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Goal State:
  - B = off
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Current State:
  - B = on
  - T/A #1 = (off, off)
  - T/A #2 = (off, off)

- Goal State:
  - B = on
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Current State:
  - B = off
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)

- Goal State:
  - B = off
  - T/A #1 = (on, on)
  - T/A #2 = (off, off)
DGDP Execution Capability

- Time complexity of one execution cycle.
  - GDP rule lookup is polynomial execution.
  - DGDP execution is \( O(l) \), where \( l \) is the number of GDPs.

- DGDP is capable of real-time repair and reconfiguration.
  - Repair capability is necessary when anomalies occur during execution time (e.g. T/A #1 fails into a reparable state).
  - Reconfiguration capability is necessary when goal-states change quickly (e.g. turn on T/A #2 instead).

Conclusion

- Factored Symbolic approach to Reactive Planning enables:
  - Compact decomposed goal-directed plan compilation through:
    - Decomposition
    - BDD encoding
  - Real-time execution capabilities:
    - Reactive repair
    - Reactive reconfiguration
    - Approximately linear in the number of components

- Possible Extension
  - Add probability and utility using ADD