Mission control of autonomous vehicles based on time logic framework

Danilo F. Bassi
Outline

• Introduction:
  – Autonomous vehicle
  – Multilevel Control
  – Mission Control

• Formulation of logic paradigm for mission control

• Synthesis of a dynamic controller

• Application of mission controller

• Conclusions
Introduction

Autonomous vehicle (mobile robot)

- Vehicle: mobility
  • UGV
  • UAV
  • etc.
- Autonomy: minimize or eliminate human participation in the control
- Purpose: accomplish a desired task or mission
- Problem: provide full autonomous control given an unknown and unstructured environment
Introduction

Control of autonomous vehicles

• Multilevel control
  – Hierarchical approach: a divide-to-conquer method for solving complex problems in a structured way.
  – Higher levels have an abstraction of lower levels.
  – Commands are given to lower levels to be translated into more detailed actions.

• Three level control for robotic systems:
  – Mission level: solves a task (highest level).
  – Navigation level: solves a movement.
  – Locomotion level: produces motion.
Mission specification and control

- **Objective**: to satisfy requirements or commands from the user, given *some* assumptions
- High level of abstraction
- It is a control: specification is given (reference) and outcome is always verified (feedback)
- **Solution**:
  - Find the set of actions that system will perform to satisfy the requirements
  - Select the best action to accomplish the mission
Formulation of logic paradigm

Logic control paradigm

- Logic programming: use mathematical logic for computer programming
- Declarative paradigm: goals are stated and the system find the right actions
- It is close to natural language: easy and precise to formalize goals
- Solution can be shown to be correct by construction (theorem proving)
- Controller is a logical consequence of the symbolic model of the system combined with the control specification
**Formulation of logic paradigm**

**Logic formalism for mission control**

- The logic formalism can capture the nature of mission control: initial conditions, goals.
- Natural human interface: a natural language statement is formalized as logical formula.
- Discrete: the mission problem is transformed into a discrete control problem.
- Use of modal logic for dealing with time dependency of dynamic systems.
- A transition system: a finite state machine is used for control.
Temporal Logic
- Temporal logic have been proposed as the specification language (linear temporal logic)
- System of logic that deals with modalities that qualify the truth in term of time
- New operators are added to the traditional Boolean logic:
  • Until: $U$
  • Next: $X$ (neXt)
  • Always: $G$ (Globally)
  • Eventually: $F$ (Future)
Formulation of logic paradigm

Temporal Logic time diagrams

<table>
<thead>
<tr>
<th>Time:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \phi_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \phi_1 \cup \phi_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \phi )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( X\phi )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( G\phi )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( F\phi )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Formulation of logic paradigm

Definitions

• **Abstract model:** a simplified symbolic model of a dynamic system. Sufficient details to define desired missions of the system:
  - Symbolic states: Q
  - Symbolic inputs: U
  - Transition function: $\delta$
  - Finite State Machine (FSM): formed by previous elements

• **Atomic propositions:** are logic assertions about the state of the system (true or false), a function of the state.
Formulation of logic paradigm

Definitions

- **Discrete transition system**: $M = (Q, Q_0, U, \delta, \Pi, \models)$, with $Q$ the set of states, $Q_0 \subseteq Q$ set of initial states, $U$ the set of system inputs, $\delta: Q \times U \mapsto 2^Q$ the transition function, $\Pi$ set of atomic propositions and $\models \subseteq Q \times \Pi$ the satisfaction relation.

- **Run**: a run is sequence of states that satisfy the transition relation: $q^0, q^1, q^2 \ldots$ with $q^i \in Q$, $q^0 \in Q_0$, $q^{i+1} \in \delta(q^i, u^i)$ $\forall$ $i \geq 1$ and $u^i \in U$.

- **Syntax**: A LTL formula $\phi \in L_{\text{LTL}}(\Pi)$, belongs to the linear temporal language defined over $\Pi$ that satisfies the grammar: $\phi ::= p | \neg \phi | \phi \lor \phi | \phi U \phi | X \phi$, with $\phi$ formula and $p \in \Pi$ proposition, $\neg$ negation, $\lor$ or, $U$ until, $X$ next.
Formulation of logic paradigm

Linear Temporal Logic (LTL) Semantics

• The semantics gives the meaning of a formula: it is defined over a run $\sigma$ that satisfies the formula $\phi$ at position $i$, denoted by $\sigma(i) \models \phi$.

• The semantics is defined as follows:

\[
\begin{align*}
\sigma(i) \models p & \quad \text{iff} \quad (q^i, p) \in \models \quad \text{(or} \quad q^i \models p) \\
\sigma(i) \models \neg \phi & \quad \text{iff} \quad \sigma(i) \not\models \phi \\
\sigma(i) \models \phi_1 \lor \phi_2 & \quad \text{iff} \quad \sigma(i) \models \phi_1 \quad \text{or} \quad \sigma(i) \models \phi_2 \\
\sigma(i) \models \phi_1 U \phi_2 & \quad \text{iff} \quad \exists j \geq i \quad \text{such that} \quad \exists \sigma(j) \models \phi_2 \quad \text{and} \quad \sigma(k) \models \phi_1, \quad \forall k \in [i, j-1] \\
\sigma(i) \models X\phi & \quad \text{iff} \quad \sigma(i+1) \models \phi
\end{align*}
\]

Where $p$ is an atomic proposition, and $\phi, \phi_1, \phi_2$ are formulas.

• A run $\sigma$ satisfies a formula $\phi$, $\sigma \models p$, if it does at the initial position of the sequence: $\sigma(0) \models p$. 

Mission Control Logic Formulation

• Given a Discrete Transition System \((Q, Q_0, U, \delta, \Pi, \models)\),

• Specification: a mission is specified as a LTL formula \(\phi\) defined over \(\Pi\) (set of atomic propositions):
  \(\phi \in \mathcal{L}_{LTL}(\Pi)\)

• Feasible solutions: are the runs that satisfy the formula \(\phi\), the set \(\sum_\phi = \{\sigma \mid \sigma \models \phi\}\)

  \[\sigma = q_0 q_1 \ldots q_i \in Q, \ q_0 \in Q_0, \ q_{i+1} \delta, (u_i, q_i) \ \forall \ i \geq 1, \ u_i \in U\]
Development of logic paradigm

Mission Control formulation steps

• **Identify** the real dynamic system and its environment: and find an abstraction (FSM)

• **Find** a set of primitive propositions : $\Pi$. This set will be the base for the specification language.

• **Identify** the discrete transition function $\delta$

• **Formulate** the desired mission specification as a LTL formula $\phi$ defined over $\Pi$

• **Solve** the corresponding feasibility or optimality discrete control problem: find a control sequence $u$ that produces a state sequence $q_0q_1...q_i$ which satisfies the given formula
Examples of formulas for mission control

• Goal reaching and obstacle avoidance
  - Mission formula $\phi = \Diamond F \text{goal} \land G \neg (\lor_i \text{obstacle}_i)$.
  - Semantics: eventually ($F$) reach the destination (goal) and never ($G\neg$: always not) go over any obstacle ($\lor_i \text{obstacle}_i$).

• Fault tolerance and recovery
  - Mission formula $\phi = FG (\Psi \lor (\Psi U (\phi_f \land \rho_f)))$.
  - Semantics: $\Psi$ normal operation, $\phi_f$ is the occurrence of a specific fault $f$, $\rho_f$ recovery procedure for fault $f$.

• Target seeking and following
  - Mission formula $\phi = G(\Psi U (\theta \land \rho))$.
  - Semantics: always do seeking a target ($\Psi$), until one is found ($\theta$) and then continue following that target ($\rho$).
Synthesis of a mission controller

Mission control solutions for LTL formulation

• Main problem
  - Find a control sequence given a LTL formula specification

• Issues
  - Is it possible to satisfy the formula?
  - How to select a solution (best one?)

• Approaches:
  - Static
  - Dynamic
Synthesis of a mission controller

Static approaches

• Most of actually known solutions for control synthesis from LTL formula specification are static and fixed.
• The appeal of the static approach: full solution can be found beforehand and that the feasibility can be checked.
• Typically two automata are defined: one for verifying the acceptance of the formula (Büchi automaton) and the controller, after the first automaton is combined with the FSM model.
• Issues of static approaches:
  - High complexity of Büchi automaton synthesis (exponential space with respect to formula size).
  - The solution essentially fixed, whereas the robotic environment is dynamic and variable.
Synthesis of a mission controller

Formulation of Dynamic approach

- The solution need not be computed in full, but can be found in a more progressive manner.
- This approach follows the general framework of the dynamic programming
  - The solution is built from an actual (present) state.
  - Once a partial solution is found, it is stored, so there is no repetition previous calculations.
  - Optimality principle is used: namely for any optimal problem, each part of the problem needs to be optimal.
- Drawback: no guarantee of finding a solution (or it may need to be improved), may require to do look ahead search
Synthesis of a mission controller

Developed dynamic approach (sketch)

- Definition of control states as a pairs (q, φ): system states and TLT formula
- Definition of control transitions (qi, φi) → (qi+1, φi+1): combines system transitions and formula or successors
- Finding resolution states (q, True) or (q, False): in these states the control automaton will halt
- The control problem is reduced to finding a path between an actual state and a resolution state.
- For finite satisfaction, the end state must be (q, True)
- The resolution paths are stored
- The required input (u) is deduced from transition function δ
Application of mission controller

• Mission control over a synthetic discrete system (automaton)
  - Arbitrary automaton and TLT formula
  - Small system: detailed solution can be followed and analysed

• Robotic task solution
  - A specific robotic system: Robotic Cleaner Vehicle
Application of mission controller

- Synthetic discrete system
Application of mission controller

Robotic Cleaner

- **Setting:** rectangular area, divided in 6 sectors. Each sector has a determined units of “dirt”

- **Vehicle:** ground robot, with limited environment sensor (clean or dirty), and one unit loading capability (empty or loaded)

- **Mission:** to clean different sectors, and bring everything in one area. This mission shall be defined as a TLT formula
Application of mission controller

Motion FSM

Action FSM

Definitions

- Motion Commands  \{right, left, up, down\}
- Action Commands  \{load, dump\}
- Logical Propositions  \{cleaned, loaded, position1, position2,\ldots\}
- Finite State Machine (FSM): formed by combining motion FSM and action FSM (product automaton)
Conclusions

• The Logic Paradigm solution to the problem of mission control is based on the satisfiability of a linear time logical (LTL) formula over a discrete system (FSM):
  - LTL is very expressive to deal with dynamic and time relations
  - Logic paradigm satisfaction translates a specification into actions
• A novel solution for the LTL formula satisfaction problem was developed.
• The main contribution of this research is a dynamic formulation that progressively builds the required structures and functions while running the machine.
Conclusions

- This is a practical solution, with lower computational complexity that other approaches (static). This is very convenient where they are h/w or process constraints.
- Complex robotic mission can be expressed by the LTL language and then solved by this dynamic approach.
- This powerful paradigm could be also used in many other applications where automatic actions are looked to satisfy given requirements.