The Algorithmic Principles of Verifiable Autonomy of Vehicles

Prof Sandor M Veres
Department of Automatic Control and Systems Engineering
University of Sheffield
Outline of algorithmic principles

• Engineering: automatic feedback control using sensors and actuators (cruise, climate, ABS, controls, lane keeping, reverse parking, etc.) – adaptive control solutions

• Use of lidar, radar and cameras – image analysis: identifying and locating fixed objects and estimating the state of other agents and vehicles – rich information set!

• Stepping out of the feedback (reactive) control system paradigm – sensors generate abstract statements (predicates) logic, judgement and decisions are needed

• Environment and situations are complex: non-reactive systems are needed -> deliberative agents
Classical three layer agent architecture (deliberative)

- **Utility Abstractor**
  - (functional relations, future prediction)

- **Basic Abstractor**
  - (signal processing, recognition of objects and basic relationships)

- **Sensors**
  - (+attention control)

- **Deliberator**
  - (decision making in abstracted, symbolic form)

- **Sequencer**
  - (translating symbolic actions into detailed skill sequence)

- **Controller**
  - (executor of actions in feedback loops of skills)

*Environment*
Sensing and abstractions (example)

Sensing the lane for autonomous platoon driving under occlusion and poor visibility - a number of statements emerge:

The dashed lines on the side are the lane separator lines. The car in front follows the lane. The distance to the car in front is 4m. The car in front breaks. Someone is driving in the parallel lane. Sensing is too poor for platooning. etc.
Controllers (= Skills)

- **Utility Abstractor**
  (functional relations, future prediction)
- **Basic Abstractor**
  (signal processing, recognition of objects and basic relationships)
- **Sensors**
  (+attention control)
- **Deliberator**
  (decision making in abstracted, symbolic form)
- **Sequencer**
  (translating symbolic actions into detailed skill sequence)
- **Controller**
  (executor of actions in feedback loops of skills)

Environment
**Controllers – (Robot) Skills**

*The controller* consists of one or more *threads of computation* that implement one or more feedback control loops, *tightly coupling sensors to actuators.*

Usually the controller *contains a library of hand-crafted control dynamics* (e.g. transfer functions, neural networks, state feedback controllers, etc.) called *primitive behaviours or skills.*

For instance, “*keep to steady cruise*”, “*follow the lane*”, “*turn left or right*”, “*slow down*”, “*stop*”, “*re-start*”, “*accelerate*”, etc.

The *movement and dynamics* applied by the controller *can be changed* at run time by the decision making deliberator.
Sequencer (= Planner of complex movements)

- **Utility Abstractor** (functional relations, future prediction)
- **Basic Abstractor** (signal processing, recognition of objects and basic relationships)
- **Sensors** (+attention control)
- **Deliberator** (decision making in abstracted, symbolic form)
- **Sequencer** (translating symbolic actions into detailed skill sequence)
- **Controller** (executor of actions in feedback loops of skills)

*Environment*
The Sequencer

The sequencer's job is to select which primitive behaviour (i.e. feedback controller) the Controller should use at a given time, and to supply parameters for the behaviour.

For instance “go to destination” requires the activation of a number of primitive behaviours, similarly for “join the motorway”, “keep cruising”, “go around obstacle”, “park on the roadside”, “change lanes”, etc.

The sequencer must be able to respond conditionally to the current situation, whatever it might be.
Deliberator & Utility Abstractor

Utility Abstractor (functional relations, future prediction)

Basic Abstractor (signal processing, recognition of objects and basic relationships)

Sensors (+attention control)

Sequencer (translating symbolic actions into detailed skill sequence)

Controller (executor of actions in feedback loops of skills)

Environment
The Deliberator Layer

*The deliberator* performs computations such as planning and other search-based algorithms. Usually these are *complex NP hard* but can also include polynomial-time algorithms.

The deliberator runs as **one or more separate threads of control:** idea generation, goal setting, planning, runtime verification and high level monitoring of execution of plans.

The key architectural feature of the deliberator is that several **behaviour transitions can occur** between the time a deliberative algorithm is invoked and the time it produces decisions.
Agent based approaches in robots
BDI Agents for Deliberation

- Belief revision function
- High-level planning: options selection
- Desire (sub-goal) selection/filtering
- Deliberation: intention commitment
- Execution
Rational Agents – more sophistication!
Agent programming principles

Ontology

>agent body model
  @total mass: scalar
  @inertial matrix: matrix
  @force to thruster conversion matrix: matrix
  @torque to thruster conversion matrix: matrix
  @gyro to body frame conversion matrix: matrix
  @gyro bias estimates: vector
  @motor rotation bias: scalar
  @operational motors: vector
  @battery voltage: double
  @bearing disturbance: double

>agent control parameters
  @position gain: double
  @velocity gain: double
  @attitude gain: double
  @omega gain: double

>state information
  >>state transition matrix: matrix
  >>gyro acceleration data: vector
  >>gyro angular rates: vector
  >>vision position estimate: vector
  >>vision velocity estimate: vector
  >>vision orientation estimate: quaternion
  >>current state vector: vector
  >>desired state vector: vector
  >>current circumstances
    @battery voltage: double
    @bearing disturbance: double

>control signal: vector
>position: double
>>starting point

>time period: double
>timed path: matrix
>>timed circle manoeuvre

Agent program:

INITIAL BELIEFS AND GOALS

~System is initialized.
~Mission is active.

!Take initial actions.

INITIAL ACTIONS

Initialize system.
Determine starting location P0 from negotiation with other connected agents and available positions defined within file ‘C:\start_points.txt’.
Generate timed circle manoeuvre Tc from file ‘C:\trn1.txt’ based on starting location P0.

PERCEPTION PROCESSES

Monitor the following Booleans:

Arrived at starting location P0.
Buddy is too near without reason.
The Jason agent programming language provides symbolic operations which communicate with a number of executive processes (for perception and action) through abstractions.
Agent programming practice

Development

sEnglish Publisher

Compile sentence consistency

Check sentence consistency

sEnglish Project

Compile ontology

Match templates

Populate meanings

Generate *.asl file

Build m-files

Compile to executable option

Application

Executive processes for perception and action

Jason Process (Java)

Modified Jason based decision making with realtime external context testing and perception updates.
Requirements of a sound algorithmic solution for autonomous cars

• Safe decisions need speed of sensing and completing situation analysis at all levels
• To be able to safely guess other road users move and behaviour (not to hold up traffic)
• Move bravely within own physical boundaries of protection (reachability set analysis)
• Be polite and do not hold up other road user’s progress (anticipate intentions)
• Anticipate danger from the roadside (pedestrians, joining cars, rolling objects, etc.)
The verification/certification problem

• Robust dynamical (spatial temporal) modelling of sensors and actuators
• Hierarchical abstractions to reduce complexity
• Scenario and failure analysis by dedicated (super) computers
• Legal safety requirements to be checked by computers
• Human-car interactions, social acceptability analysed by multi-agent modelling/verification
Example: platooning verification

Physical Processes: sensing/control and abstractions

Agent verification

The car always does the best decision under its sensing and physical constraints
Conclusions

• Algorithmic approaches to autonomous car decisions have been reviewed
• Most promising are “anthropomorphistic” approaches to programming
• Legal interpretability of car decisions are important – can be ensured in a BDI approach
• Good (human) car user experience is important to be maintained by algorithmic solutions – long period of work expected
Thank You
-
Any questions?