From AgentSpeak to C
for Unmanned Aerial Vehicles

Samuel Bucheli\textsuperscript{a}

Department of Computer Science
University of Oxford

May 28, 2015

\textsuperscript{a} joint work with Daniel, Ruben, and Ashutosh
In this Talk

- autonomous agents as hybrid systems,
- safety considerations: DO-178C and MISRA C,
- AgentSpeak as a modeling language for high-level behavior,
- from AgentSpeak to C: examples and experiments,
- further work: translation validation.
Autonomous Agents

hybrid system

- discrete decision making
- abstraction layer
- continuous control
Autonomous Agents

hybrid system

- discrete decision making
- abstraction layer
- continuous control

☑️
Autonomous Agents

hybrid system

- discrete decision making
- abstraction layer
- continuous control
DO-178C
Software Considerations in Airborne Systems and Equipment Certification

System Requirements Allocated to Software

Software Design

Software High Level Requirements

Software Low Level Requirements

Source Code

Executable Object Code

Test Cases

Test Procedures

Test Results

16.2 (req) Functions shall not call themselves, either directly or indirectly.

20.4 (req) Dynamic heap memory allocation shall not be used.
The Autopilot

image from http://fly.historicwings.com/2012/08/george-the-autopilot/
Sample Flight Plan

takeoff

goto 0 0 1 0
goto 1 0 1 0
goto 1 1 1 0
goto 0 1 1 0
goto 0 0 1 0

land
Sample Flight Plan

takeoff

engage rotors, bring aircraft to altitude

goto 0 0 1 0

goto 1 0 1 0

goto 1 1 1 0

goto 0 1 1 0

goto 0 0 1 0

land
Sample Flight Plan

Takeoff:
- goto 0 0 1 0
- goto 1 0 1 0
- goto 1 1 1 0
- goto 0 1 1 0
- goto 0 0 1 0

Land:
- move aircraft towards target until reached
- engage rotors, bring aircraft to altitude
Sample Flight Plan

takeoff

go to 0 0 1 0

land

go to 1 0 1 0

/go to 1 1 1 0

/go to 0 1 1 0

/go to 0 0 1 0

ingage rotors, bring aircraft to altitude

move aircraft towards target until reached

lower aircraft, disengage rotors
AgentSpeak

agent

- initial goal
- belief base
- plan 1
- plan 2
- ...
event
  : condition & condition & ... & condition
<- formula;
  formula;
... 
  formula.
Plans

triggering events the plan is relevant for

: condition & condition & ... & condition
<- formula;
  formula;
  ...
  formula.
event
: condition & condition & ... & condition
<- formula;
   formula;
   ...
   formula.

preconditions when the plan is applicable
Plans

event :

\[- \text{condition} \land \text{condition} \land \ldots \land \text{condition}\]

\[\leftarrow \text{formula};\]

\[\text{formula};\]

\[\ldots\]

\[\text{formula}.\]

actual body of the plan to be executed
 Plans, concretely

```
+!goto(Target) : takenOff
<- sendHover();
+lastTarget(Target);
!completeGoto.

+!completeGoto
: takenOff & myPosition(Pos)
 & lastTarget(Target) & not closeEnough(Pos, Target)
<- Movement = calculateMovement(Pos, Target);
  sendControl(Movement);
  !completeGoto.

+!completeGoto
: takenOff & myPosition(Pos)
 & lastTarget(Target) & closeEnough(Pos, Target)
<- notifyUser("target reached").
```
Semantics

process messages → select event → relevant plans → applicable plans → select plan

clear intents → execute intent → select intent

add intent
Semantics

- process messages
  - select event
    - relevant plans
      - applicable plans
        - select plan
          - add intent
    - clear intents
  - execute intent
    - select intent
Semantics

- Semantics
- Process messages
- Select event
- Relevant plans
- Applicable plans
- Select plan
- Clear intents
- Execute intent
- Select intent
- Add intent
Semantics

- Process messages
- Select event
- Relevant plans
- Applicable plans
- Select plan
- Clear intents
- Execute intent
- Select intent
- Add intent
Semantics

process messages → select event → relevant plans → applicable plans → select plan

clear intents → execute intent → select intent → add intent
Semantics

process messages → select event

relevant plans → applicable plans

select plan

select intent

select intent

execute intent

add intent

clear intents
Semantics

- process messages
- select event
- relevant plans
- applicable plans
- select plan
- clear intents
- execute intent
- select intent
- add intent
Semantics

- process messages
- select event
- relevant plans
- applicable plans
- select plan
- clear intents
- execute intent
- select intent
- add intent
Our Customization

• “run-to-completion” style scheduling,
• only “tail recursion” allowed,
• belief base holds at most one instance of a literal
void next_step(void) {
    updateBeliefs();
    eventt event = get_next_event();
    switch (event.trigger) {
        /* ... */
        case ADD_ACHIEVE_GOTO:
            add_achieve_goto(event.goto_param0);
            break;
        case ADD_ACHIEVE_COMPLETEGOTO:
            add_achieve_completeGoto();
            break;
        /* ... */
    }
}
void add_achieve_goto(positiont param0) {
    /* try first plan */
    if (add_achieve_completeGoto_plan0()) {
        return;
    }

    /* try second plan */
    if (add_achieve_completeGoto_plan1()) {
        return;
    }

    /* ... handle the case where no plan is applicable ... */

    return;
}
bool add_achieve_goto_plan0(positiont param0) {
    positiont Target = param0;
    if (!takenOff_set) { return false; }
    sendHover();
    lastTarget_set = true;
    lastTarget_param0 = Target;
    internal_achieve_completeGoto();
    return true;
}
Experimental Setup

- user interface
- state estimation
- Node (wrapper code)
- UAV
- AI (translated AgentSpeak)
- PID Controller

Connections:
- User command: user interface → Node
- filter state: state estimation → Node
- events: Node → AI
- actions: Node → PID Controller
- takeoff, velocity commands, land: PID Controller → UAV
- navigation data, video: UAV → Node
Test Flight
Test Flight
Test Flight
Test Flight
Test Flight
Test Flight
Test Flight
Test Flight
A Simple Translation Validation

AgentSpeak Code

LTL/BDI Property

MISRA C Code

Maude / K

CBMC

Result

the same?
Conclusions

- translation from agent-oriented programming language to low-level language + translation validation = traceability

- what is the sweet spot for agent-oriented programming languages: expressivity vs. translatability?

- translation validation?