VERIFYING MACHINE ETHICS

LOUISE DENNIS
IJCAI 2018 SUMMER SCHOOL
WHAT IS VERIFICATION?
if you believe there is an obstacle then stop
if you believe there is a path then follow it

No obstacle, no path
Obstacle, no path
No obstacle, path
Obstacle, path

Perception

If the agent believes there is an obstacle then it will try to stop

Data abstracted to beliefs/facts/predicates

Data from Sensors

Control system executes command

Something happens in the real world

VERIFYING AGENT BASED AUTONOMOUS SYSTEMS
**BRAKE FAIL ON LINE UP**

- Turn Left (damages the aircraft and airport hardware)
- Turn Right (damages the aircraft and risks colliding with people)
- Continue (risks collision with a manned aircraft)

· \( \phi_1 = \text{do not damage own aircraft (1)}, \)
· \( \phi_2 = \text{do not collide with airport hardware (2)}, \)
· \( \phi_3 = \text{do not collide with people (3)}, \)
· \( \phi_4 = \text{do not collide with manned aircraft (4)}. \)

**AIRCRAFT TURNS LEFT**
Plan A has no concerns

Plan A violates
do not damage aircraft

Plan A has no concerns
Plan B violates
do not damage aircraft

Plan A violates
do not collide with airport hardware

Plan A has no concerns
Plan B violates
do not collide with airport hardware

Plan A violates
do not damage aircraft
Plan B violates
do not collide with airport hardware

VERIFYING THE PROGRAM
PROPERTIES

• If selected plan collides with a manned aircraft then all other plans collided with manned aircraft.

• If the selected plan collides with people then all other plans collided with people or manned aircraft.

• If the selected plan damages airport hardware then all other plans damaged airport hardware or collided with people or manned aircraft.

• If the selected plan damages unmanned aircraft then all other plans damaged unmanned aircraft or airport hardware or collided with people or manned aircraft.
INTRUDER AIRCRAFT

\[ \phi_1 = \text{do not violate turn right rule (2)}; \]
\[ \phi_2 = \text{do not stay above 500 feet rule (2)}; \]
\[ \phi_3 = \text{do not collide with objects on the ground (3)}; \]
\[ \phi_4 = \text{do not collide with aircraft (4)}. \]

\[ +! \text{avoid\_collision} : \{ \text{B flightPhase(eAvoid)}, \sim \text{B route(eAvoid, Route)} \} \leftarrow \]
\quad \text{plan(reqEmergRoute,turnRight), } \ast \text{route(eAvoid, R), enactRoute(R), wait;} \]

\[ +! \text{avoid\_collision} : \{ \text{B flightPhase(eAvoid)} \} \leftarrow \text{enactRoute(turn\_left); } [\phi_1] \quad 1 \]
\[ +! \text{avoid\_collision} : \{ \text{B flightPhase(eAvoid)} \} \leftarrow \text{enactRoute(emergency\_land); } [\phi_2,\phi_3,\phi_4] \quad 2 \]
\[ +! \text{avoid\_collision} : \{ \text{B flightPhase(eAvoid)} \} \leftarrow \text{enactRoute(return\_to\_base); } [\phi_4] \quad 3 \]
VERIFYING THE PROGRAM

TURN RIGHT

TURN LEFT

RETURN TO BASE

aircraft

no aircraft
REMEMBER WINFIELD’S ROBOTS?
agent = nao_agent.NaoAgent()

add_pick_best_rule(AND(B('plans'), B('danger_close')), compare_plans_asimov_WD, update_plan_rule)
add_pick_best_rule(AND(B('plans'), NOT(B('danger_close'))), compare_plans_asimov_WT, update_plan_rule)

def compare_plans_asimov_WD(self, plan1, plan2):
    if ((plan1.robot_walking_dist < plan2.robot_walking_dist)
        and not (worse(plan1, plan2, 'robot_danger_dist'))
        and not (worse(plan1, plan2, 'robot_obj_dist'))
        and not (worse(plan1, plan2, 'human_danger_dist'))):
        return 1;
    else:
        if (worse(plan2, plan1, 'human_danger_dist')):
            return 1;
        else:
            if (worse(plan2, plan1, 'robot_obj_dist')
                and not (much_worse(plan1, plan2, 'human_danger_dist'))):
                return 1;
            else:
                if (worse(plan2, plan1, 'robot_danger_dist')
                    and not (worse(plan1, plan2, 'robot_obj_dist'))
                    and not (worse(plan1, plan2, 'human_danger_dist'))):
                    return 1;
                else:
                    return 0;

def compare_plans_asimov_WT(self, plan1, plan2):
    if ((plan1.wait_time < plan2.wait_time)
        and not (worse(plan1, plan2, 'robot_danger_dist'))
        and not (worse(plan1, plan2, 'robot_obj_dist'))
        and not (worse(plan1, plan2, 'human_danger_dist'))):
        return 1:
PROPERTIES

- The selected plan does not put the human in more danger than the other plans.

- If the selected plan puts the robot further from its (human ordered) goal then the other plans put the human in more danger.

- If the selected plan puts the robot in danger then the other plans put the human in danger or placed the robot further from its goal.

- Eventually the agent selects a plan. (FALSE)
CONSTRAINED ENVIRONMENT

- Plan comparisons (worse) and walking time/distance relations are transitive.
- compare_plans_asimov_WD etc are antisymmetric and transitive.
REASONING WITH EMBEDDED THEOREM PROVING/ MODEL CHECKING
REFERENCES


