Evaluating PDDL for programming production cells: – a case study

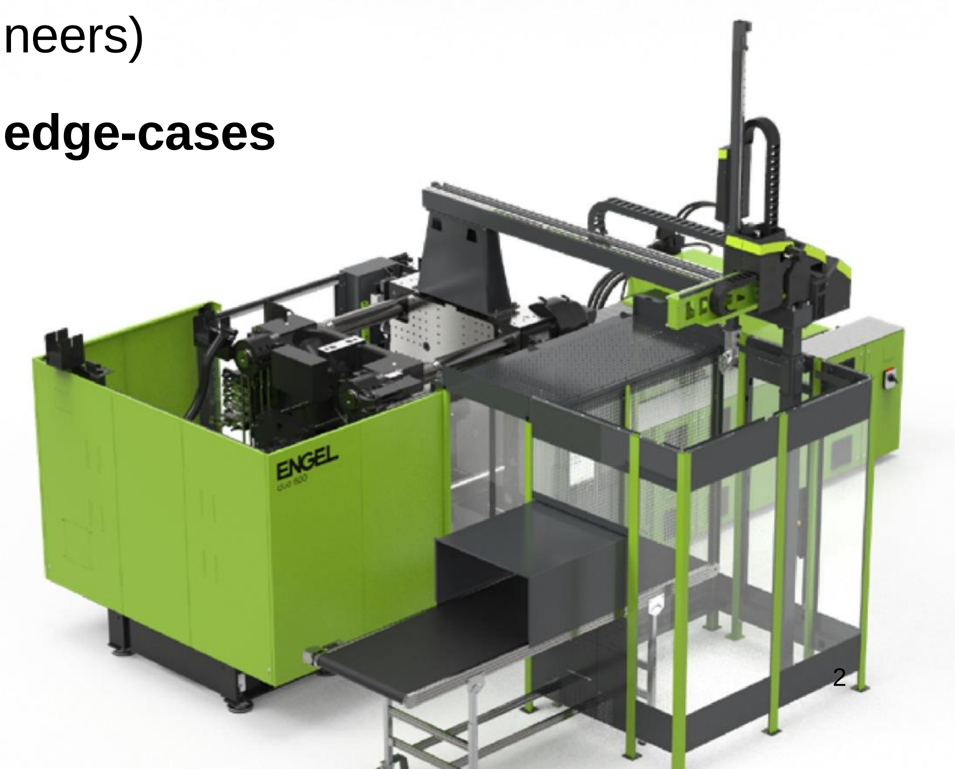
Christoph Mayr-Dorn, Alexander Egyed (Johannes Kepler University, Linz)
Mario Winterer, Christian Salomon, (Software Competence Center Hagenberg GmbH)
Harald Fürschuß, (ENGEL Austria GmbH)

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Motivation

Manufacturing companies are required to **quickly/easily adapt** their **production** to changing demands and innovation.

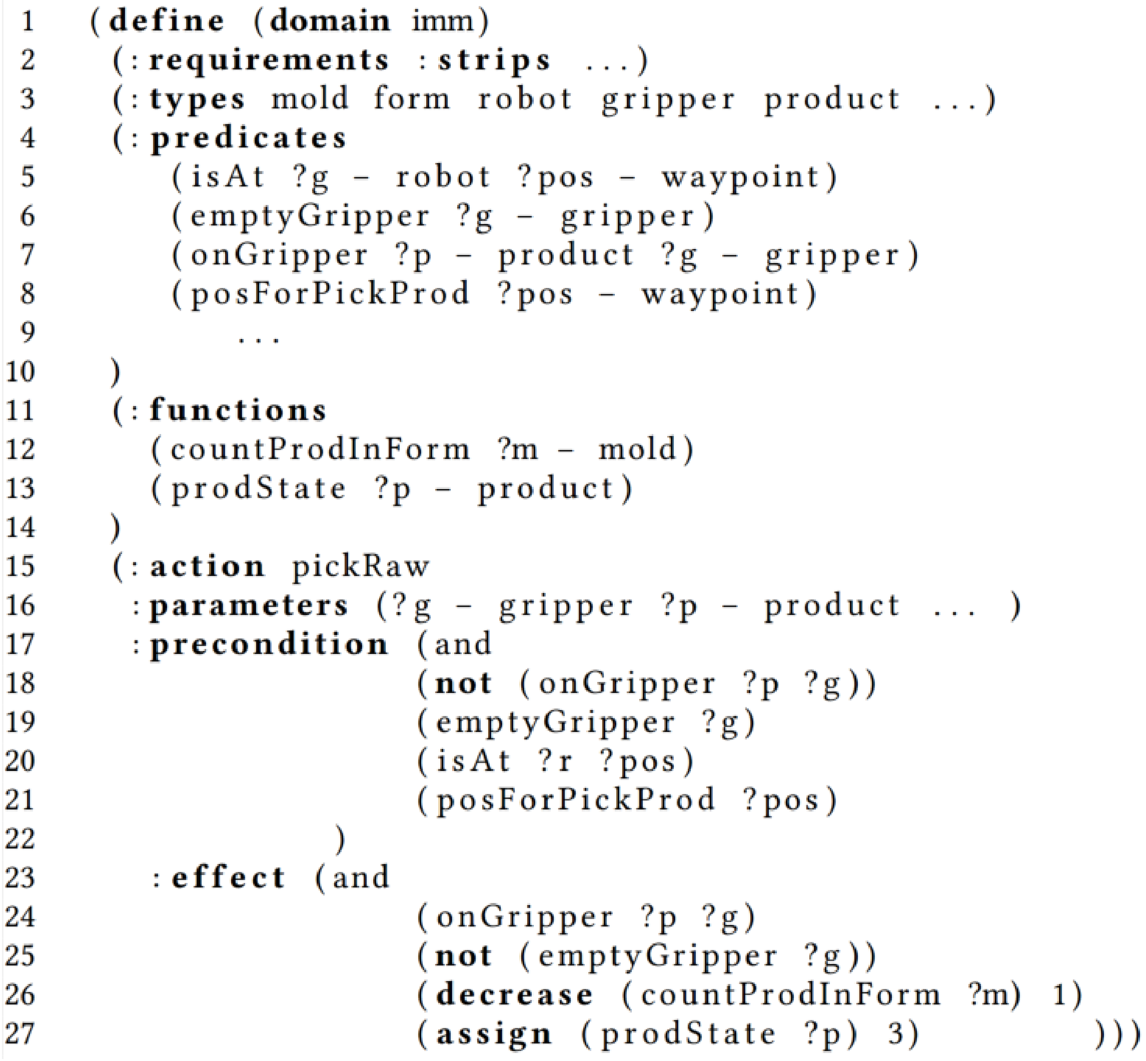
- Need to **frequently reprogram robots** and machines on the shop floor
- Involves defining the **interaction with other shop floor participants** (robots/machines/humans).
- (Re)programming is often done **by end-users** (e.g., domain engineers)
- Complex interactions/sequences require **extensive handling of edge-cases**
  - Hard to get right
  - Hard to understand
  - Hard to reuse



Research Questions

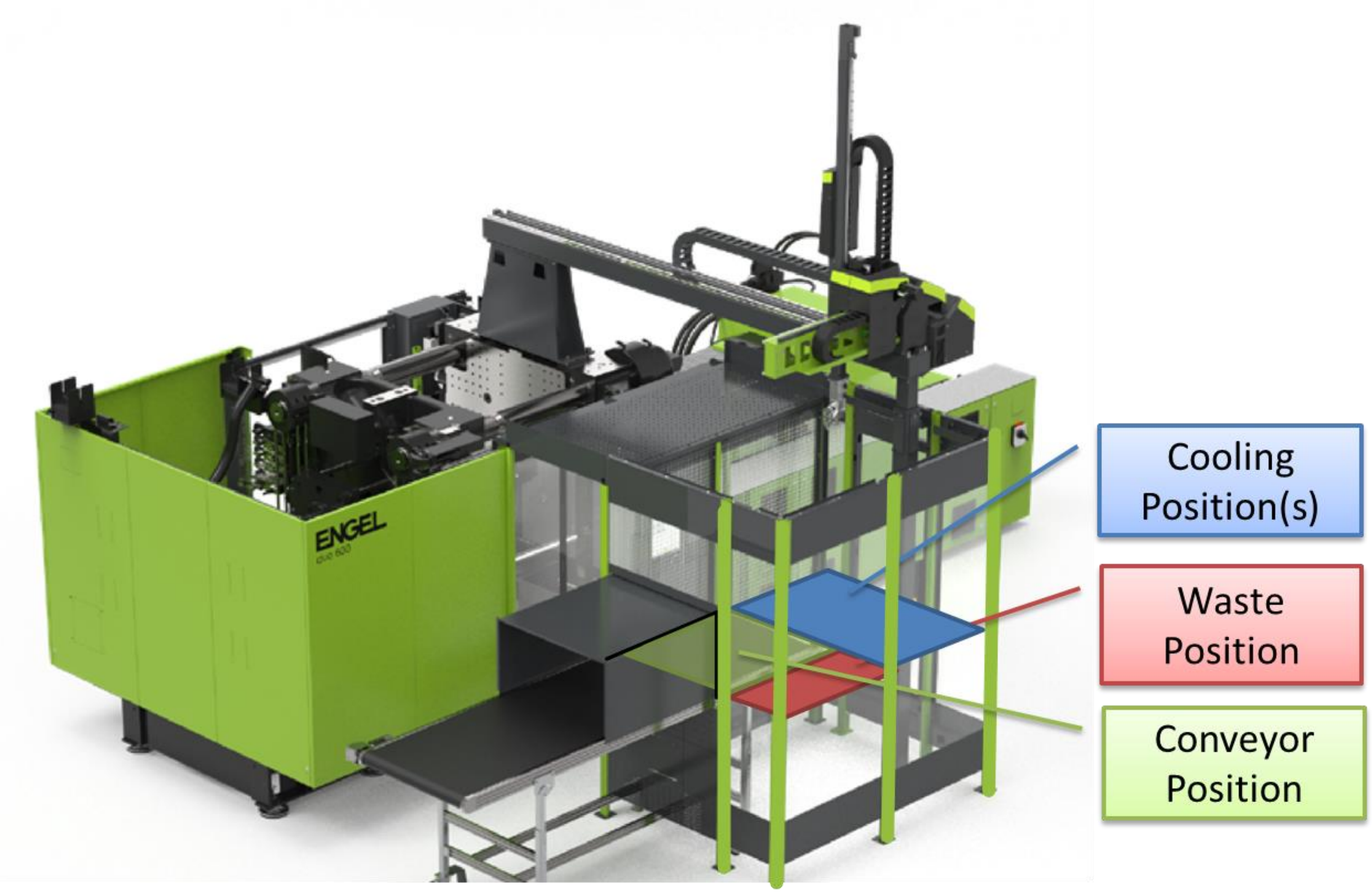
- How can we enable engineers to focus on their domain knowledge and limit detailed implementation work?
  - Engineers have detailed product know-how
  - Are aware of production stages, goals
  - Pre and post conditions of production steps

- Are planning languages such as PDDL and/or HDDL practical to this end?
  - RQ1: To create efficient production sequences?
  - RQ2: How easy are they to be extended for changing production scenarios?



```
(define (domain imm)
  (:requirements :strips ...)
  (:types mold form robot gripper product ...)
  (:predicates
     (isAt ?g - robot ?pos - waypoint)
     (emptyGripper ?g - gripper)
     (onGripper ?p - product ?g - gripper)
     (posForPickProd ?pos - waypoint)
        ...
  )
  (:functions
    (countProdInForm ?m - mold)
    (prodState ?p - product)
  )
  (:action pickRaw
   :parameters (?g - gripper ?p - product ... )
   :precondition (and
                 (not (onGripper ?p ?g))
                 (emptyGripper ?g)
                 (isAt ?r ?pos)
                 (posForPickProd ?pos)
             )
    :effect (and
                 (onGripper ?p ?g)
                 (not (emptyGripper ?g))
                 (decrease (countProdInForm ?m) 1)
                 (assign (prodState ?p) 3)         )))
```

Case Study: multi-stage molding



- Multi-stage sequence C:
  - Pick solidified part
  - Place for cooling
  - Take cooled insertion component
  - (Pick solidified final part from previous run)
  - (Pick solidified intermediary part from previous run)
  - Insert component into mold
  - Place solidified final part
- Complexity due to distinction between initial and subsequent runs, failures of picking/placing, and restarting from previous run in unknown state: multiple cooling locations, two mold forms, amount of grippers

Results – Support for adaptation

3 modeling variants: PDDL+ for time-aware sequences, PDDL cost for some optimization, HDDL for hierarchical task structuring

	PDDL+ based	PDDL cost based	HDDL based
Structure:	10 actions, 3 proc., 6 events	11 actions	11 actions, 10 tasks, 23 methods
LoC:	~300	~205	~490
Structure	12 actions, 4 proc. , 7 events	14 actions	14 actions, 12 tasks, 29 methods
LoC:	~365	~280	~660
New/diff LoC	65/15	70/20	170/80

Results – Support for adaptation

scch { }

3 modeling variants: PDDL+ for time-aware sequences, PDDL cost for some optimization, HDDL for hierarchical task structuring

	PDDL+ based	PDDL cost based	HDDL based
Structure:	10 actions, 2 proc., 6 events	11 actions	11 actions, 10 tasks, 2[illegible] methods
LoC:	~300	~205	~490
Structure	12 actions[illegible]	14 actions	14 actions, 12 tasks, 29 methods
LoC:	~[illegible]	[illegible]	~660
New/diff LoC	65/15	70/20	170/80

Similar actions/predicates across the variants

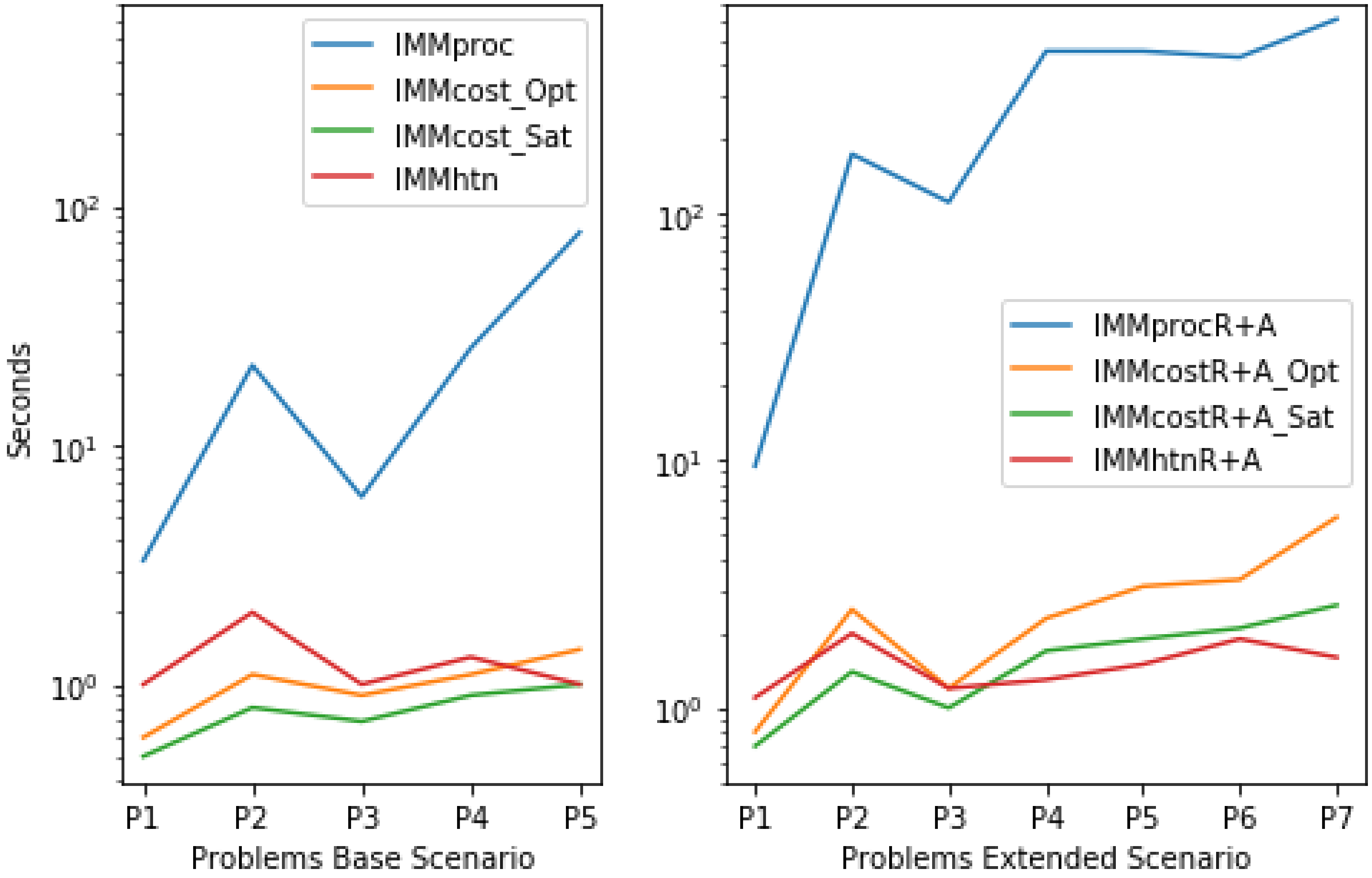
Same type and extent of adaptation across the variants

JOHANNES KEPLER UNIVERSITY LINZ

Results - Performance

Problem scenarios with increasing product instances and starting conditions.

Models for optimal sequences (i.e., considering time) don’t scale.



Discussion and Conclusions

RQ1: feasible but not practical yet

- HDDL/PDDL require tracking each product individually → need to obtain a cyclic plan for continuous production (without constant replanning)
- Difficult transition from start-up to continuous phase
- Planning duration is not practical for efficient (i.e., time aware) sequences
- HDDL solver is sensitive to problem order
- Perhaps process mining (BPM community) can bring some inspiration here

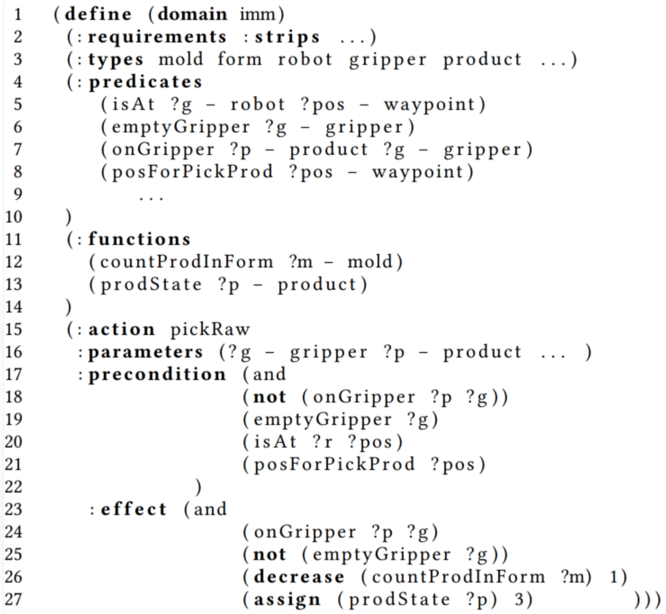
RQ2: advanced engineering support needed

- Changes have cascading effects (not just adding of code),
  - limiting impact/scoping difficult to achieve
  - difficult to understand what is impacted
  - HDDL: Difficult to understand applicable constraints in each step
- Support for testing: wrong logic, wrong test setup, solver limits?
- Support for deadlock detection

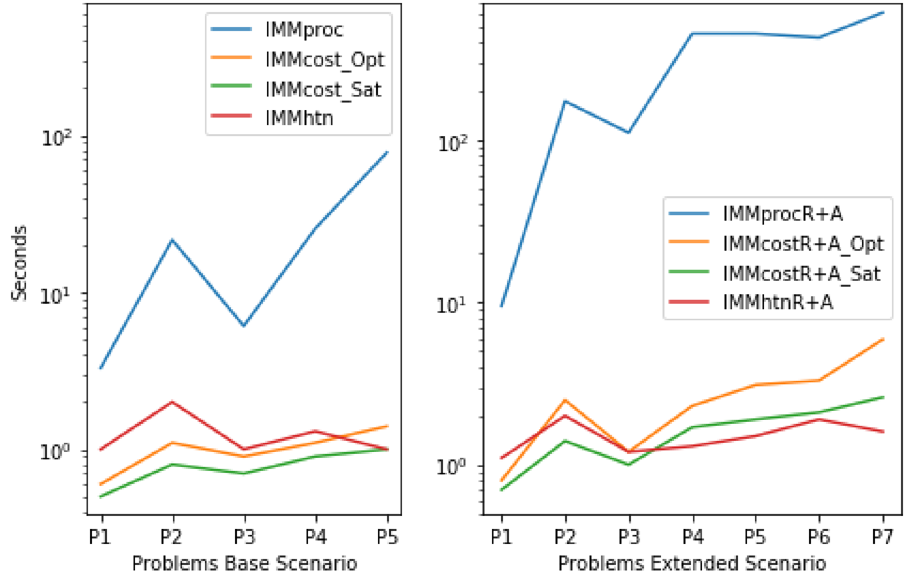
Thanks for your attention. Questions?

Supporting Online Material: https://figshare.com/s/8315f52edb597fb7836a

scch { }



```
(define (domain imm)
 (:requirements :strips ...)
 (:types mold form robot gripper product ...)
 (:predicates
    (isAt ?g - robot ?pos - waypoint)
    (emptyGripper ?g - gripper)
    (onGripper ?p - product ?g - gripper)
    (posForPickProd ?pos - waypoint)
      ...
 )
 (:functions
   (countProdInForm ?m - mold)
   (prodState ?p - product)
 )
 (:action pickRaw
  :parameters (?g - gripper ?p - product ... )
  :precondition (and
                (not (onGripper ?p ?g))
                (emptyGripper ?g)
                (isAt ?r ?pos)
                (posForPickProd ?pos)
            )
   :effect (and
                (onGripper ?p ?g)
                (not (emptyGripper ?g))
                (decrease (countProdInForm ?m) 1)
                (assign (prodState ?p) 3)        )))
```



JOHANNES KEPLER
UNIVERSITY LINZ

FFG
Promoting Innovation.

The research reported in this paper has been funded by BMK, BMDW, and the State of Upper Austria in the frame of the COMET Programme managed by FFG.