# Synthesizing a Lego Forklift Controller in GR(1): A Case Study

Shahar Maoz and Jan Oliver Ringert
Tel Aviv University, School of Computer Science
Supported by ERC StG SYNTECH

# Context

- Reactive Synthesis: obtain correct by construction controller from temporal specifications

- Recent algorithmic advancements show feasibility for mid-size systems, e.g., LTL fragment GR(1) with symbolic polynomial algorithm [PPS06]

**Jan O. Ringert**
Computer Science
Tel Aviv University


# Example GR(1) Specification

Assumption and guarantee specification limited to initial, safety, and justice constraints over environment variables (obstacle) and system variables (lMot, rMot, has2clear):

θ^s ≡ lMot=STOP=rMot & !has2clear

ρ^s ≡ (obstacle=BLOCKED -> !(lMot=FWD=rMot)) &

((lMot=BWD | rMot=BWD) &
obstacle!=CLEAR -> next(has2clear) ) &

(!(lMot=BWD | rMot=BWD) &
obstacle!=CLEAR -> next(has2clear)=has2clear ) &
obstacle=CLEAR -> next(!has2clear)

J^s ≡ lMot=FWD=rMot

θ^e ≡ TRUE

ρ^e ≡ lMot=STOP=rMot -> next(obstacle)=obstacle

J^e ≡ !has2clear

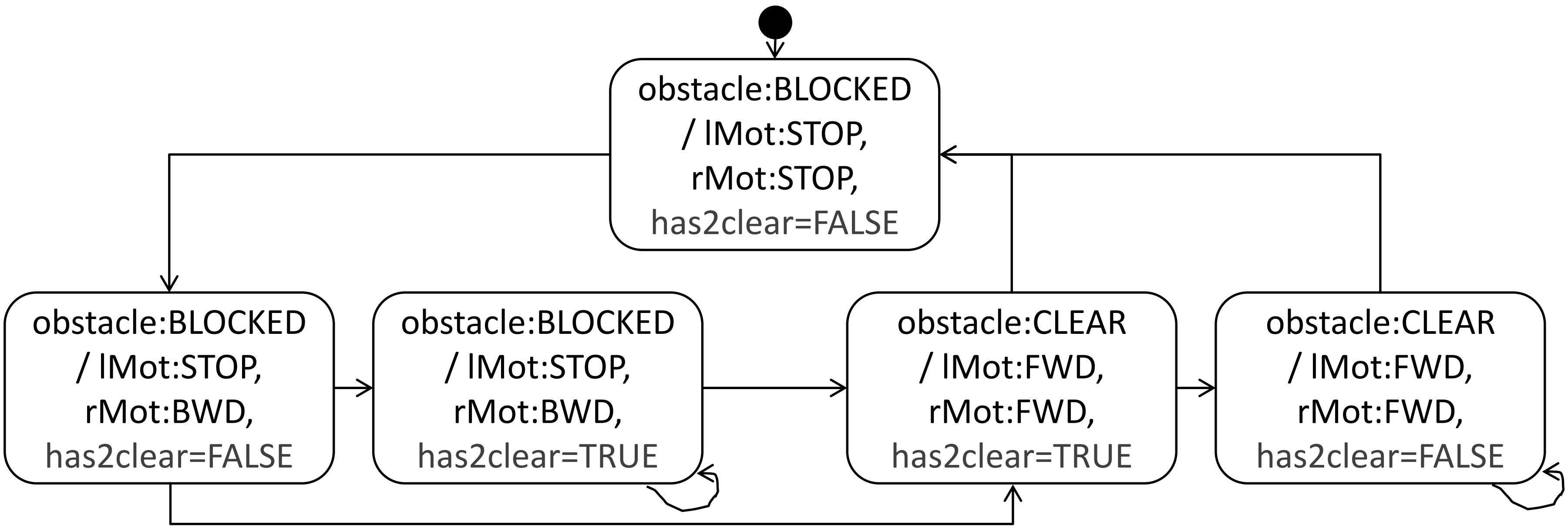
**Jan O. Ringert**
Computer Science
Tel Aviv University


# GR(1) Synthesis Problem

- GR(1) synthesis: Find controller (if one exists) that implements specification controlling system variables:

$$(\theta^e \wedge G\rho^e \wedge \bigwedge_{0<i\leq m} GFJ_i^e) \rightarrow (\theta^s \wedge G\rho^s \wedge \bigwedge_{0<i\leq n} GFJ_i^s)$$

- A controller implementing above specification:



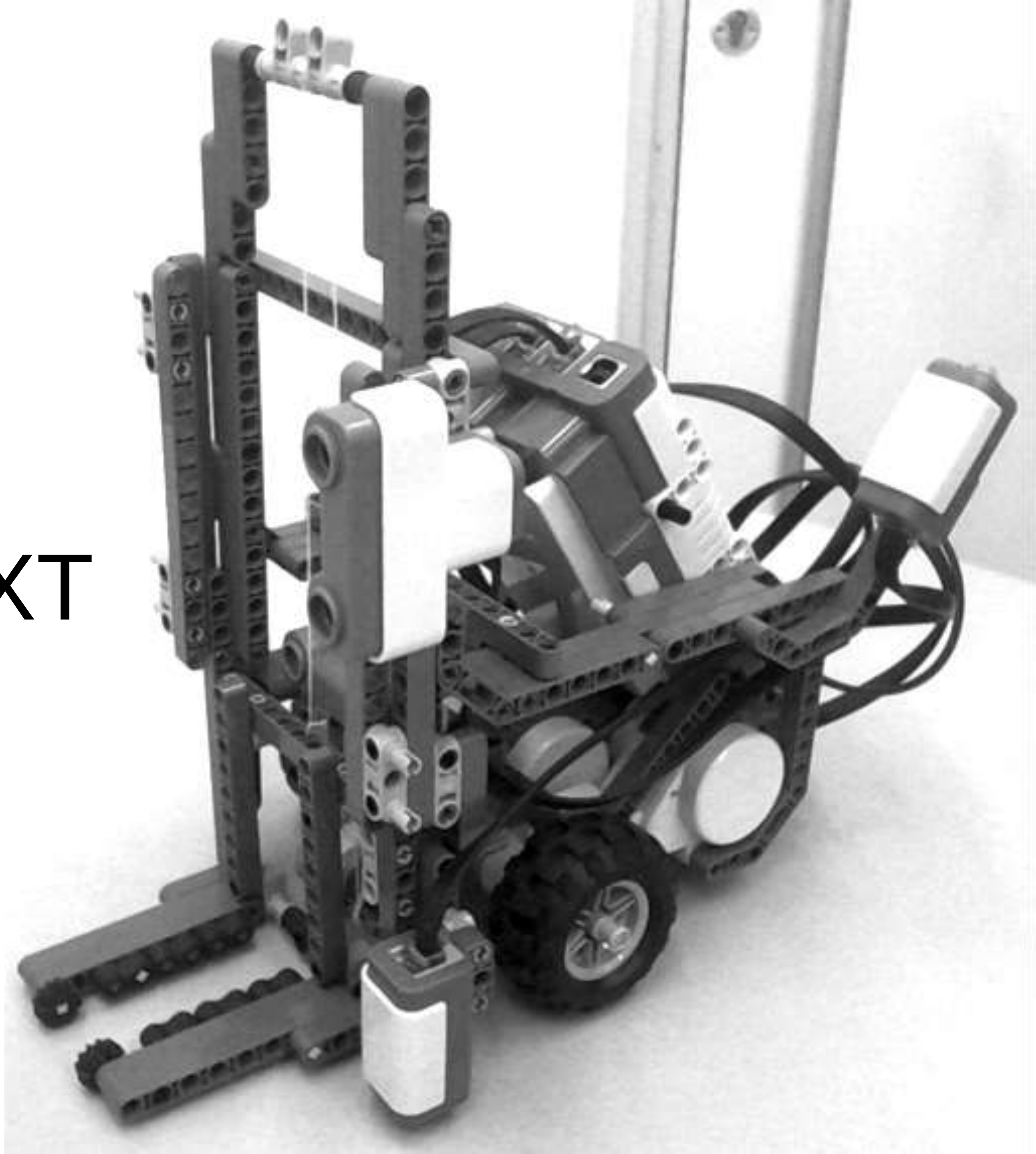
# Need for Case Study

- Despite recent theoretical and algorithmic achievements, reactive synthesis is far from being used by software engineers in practice

- Questions to investigate:
  - What are challenges faced when using a GR(1) synthesis tool?
  - Is the use of LTL specification patterns helpful?
  - Is it easy to understand reasons for unrealizability?
  - Do successfully synthesized controllers work as expected? If not, how can one understand why?

- On a wider scale: What is required to make a synthesis specification language and synthesis tools available and accepted by reactive systems software engineers?

Forklift
LEGO NXT
Robot



**Jan O. Ringert**
Computer Science
Tel Aviv University


# Case Study Materials

- GR(1) synthesis algorithm implemented in JTLV by Sa'ar (including counterstrategy synthesis)
- AspectLTL input language by Maoz and Sa'ar [MS11]
- Traceability implementation between controller and specification Maoz and Sa'ar [MS13]
- Catalog of LTL specification pattern templates in GR(1) from [MR15]
- Lego NXT forklift with LeJOS and code generation

# Case Study Task

- Automatically synthesize controller for forklift ready for code generation and deployment

- Informal requirements:
  1. Do not run into obstacles.
  2. Only pick up or drop cargo at stations.
  3. Do not attempt to lift cargo if cargo is lifted.
  4. Always keep on delivering cargo.
  5. Never drop cargo at the station where it was picked up.
  6. Stop moving if emergency off switch is pressed.

# Specification Variants

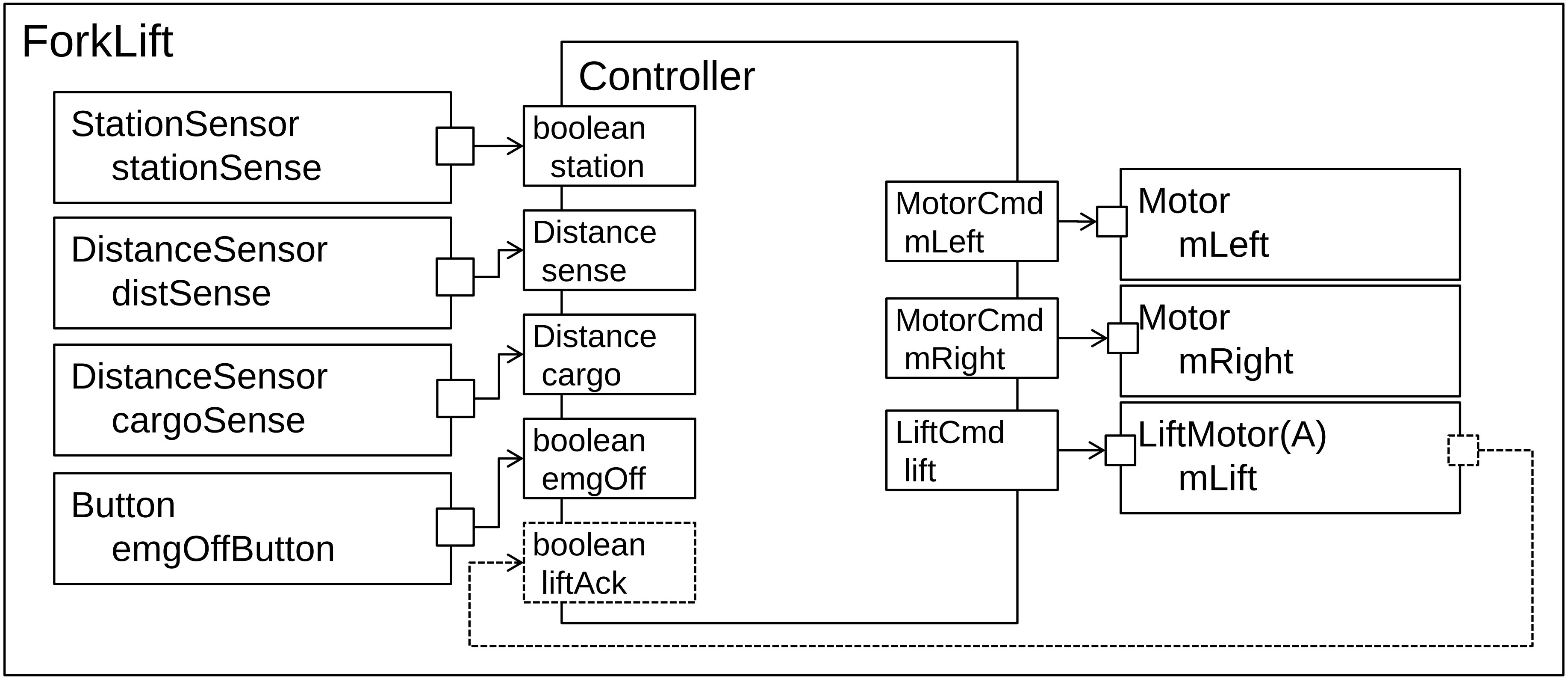
- Two variants for different scheduling of component execution

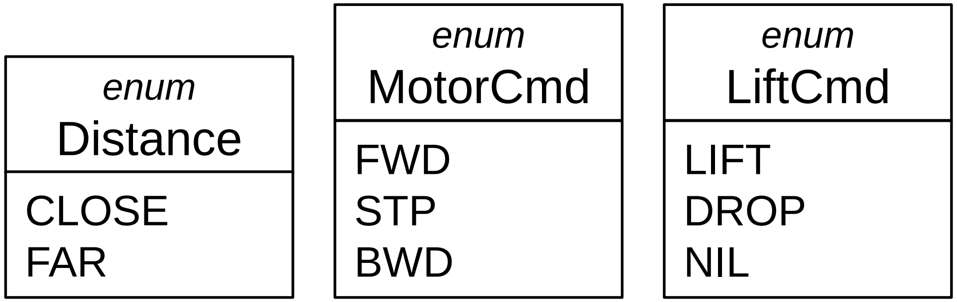
- V1.Delay: execution waits for completion of actions before reading new sensor values
  - cycle: sense-compute-act and 2000ms delay

- V2.Continuous: controller waits for completion of actions (called “continuous control” by Raman et al. [RPK13])
  - cycle: sense-compute-act no delay

**Jan O. Ringert**
Computer Science
Tel Aviv University


# Components and Types Overview



# Examples from the Specification

```
ASSUMPTION -- station does not change when stopping
  G (stopping -> station = next(station));

ASSUMPTION -- at most blocked twice between stations
  After (!atStation) have at most two (lowObstacle) until (atStation); --P15
```

```
GUARANTEE -- emergency stop
  G (emgOff -> (stopping & lift=NIL));

GUARANTEE -- main constraint to always eventually deliver cargo
  G F ((lift = DROP) | emgOff | lowObstacle);
```

# Manually Added Auxiliary Variables

- Manual addition of auxiliary variable to make property explicit
  - approach from [BJP+12]: add variables on system side, as complete and deterministic observers

Example

```
VAR -- new auxiliary variable to "remember" when cargo is loaded
  spec_loaded : boolean;
GUARANTEE
  !spec_loaded; -- initial value false
  G (lifting -> next (spec_loaded)); -- set loaded when lifting
  G (dropping -> ! next (spec_loaded)); -- unset loaded when dropping
  G (lift = NIL -> next (spec_loaded) = spec_loaded); -- preserve value
```

# Statistics of Specifications

| | V1.Delay | V2.Continuous |
|---|---|---|
| Assumptions | 7 (1xρ, 5xP26, P15) | 9 (2xρ, 6xP26, P15) |
| Guarantees | 12 (1xθ, 8xρ, 1xJ, P09, P20) | 14 (1xθ, 8xρ, 1xJ, P09, P20) |
| Manual Aux. variable | `loaded` | `loaded, waitingForLifting` |
| Statespace (i/o, manual, pattern) | $2^{10} * 2^{1} * 2^{12}$ | $2^{11} * 2^{2} * 2^{13}$ |
| Synthesis time | 0.2sec realizability<br>1.8sec controller constr. | 0.7sec realizability<br>1.3sec controller constr. |
| States of controller | 3412 | 2888 |

# Observation: Differences between Variants

- Differences of specifications V1.Delay vs. V2.Continuous
  - added manual auxiliary variable `waitingForLifting`
  - 2 new assumptions: ack is eventually sent after lifting command, ack is only sent when waiting
  - 2 new guarantees: no new lifting command while waiting, no driving while waiting
  - neither assumptions nor guarantees removed

- Surprisingly few differences!
  - reason 1: V2.Continuous developed based on V1.Delay
  - reason 2: V1.Delay already used many response patterns and obtained feedback from sensors for driving actions

# Observation: Auxiliary Variables and Patterns

- Manually adding auxiliary variables helpful!
  - help to make properties explicit for writing and reading
  - an alternative past LTL formula for the 4 occurrences of auxiliary variable `loaded` is PREV (lift!=DROP SINCE lift=LIFT)

- Support for patterns helpful!
  - otherwise impossible/very difficult to express temporal properties
  - patterns gave us better confidence that spec matches intention
  - Specifically P26 (response pattern) very helpful for assumption on environment reactions

- Patterns come at a price: make up more than half of variables in synthesis problem

# Challenge 1: Environment Specification

- Case A: Specified environment stronger than real environment behavior

```
ASSUMPTION
   G (turning -> next(sense=CLEAR));
```

- Case B: Real environment behavior difficult/impossible to formulate
  - e.g., assumption to find cargo on every station

- Case C: Bad sensor readings violate assumptions

# Challenge 2: Undesired Realizable Case

- Consider following excerpt of early specification

```
ASSUMPTION
   G (stopping -> station = next(station));
ASSUMPTION
   Globally (forwarding) leads to (!station);
```

- Synthesized controller sometimes stops forever
  - Found stopping-loop in controller (80 states): Why is it there?

- Approach: trace controller states to specification
  - three cases: satisfy justice of system, work towards justice of system, prevent justice of environment

- For large controller with thousands of states manual inspection of traceability information not feasible

# Challenge 3: Unrealizable Case

- Consider following excerpt of the specification

  ASSUMPTION
  G (!next(liftAck) | waitingForLifting);
  GUARANTEE
  G (liftAck -> (lift!=NIL | next(!waitingForLifting)));

- Specification is unrealizable, no controller can satisfy it

- Approach: synthesize counterstrategy and understand reason by interactively playing intended strategy

- For large controller with thousands of states
  - very long runs of manual execution necessary, and
  - from ~6.000 states code of counterstrategy does not compile

# Conclusion

- GR(1) synthesis for forklift controller possible
  - synthesis times fast in general (few seconds)

- Aux vars and patterns in specification language helpful
  - support making properties explicit
  - increase expressiveness at cost of adding variables
  - increase confidence that specification meets intention

- Still many challenges – promising works exist and require evaluation
  - understanding reasons for synthesized behavior, e.g., reporting
  - unrealizability, e.g., core by Cimatti et al., traces by Könighofer et al., assumption generation by Alur et al.
  - more robust synthesized behavior, e.g., bounded Ehlers & Topcu

# Available Materials

- Full specifications V1.Delay and V2.Continuous: http://smlab.cs.tau.ac.il/syntech/forklift/
- Shahar Maoz & Jan O. Ringert: GR(1) Synthesis for LTL Specification Patterns. In: ESEC/FSE 2015
- Catalog of Dwyer LTL specification pattern templates in GR(1): http://smlab.cs.tau.ac.il/syntech/patterns/