Controller Synthesis for Golog Programs over Finite Domains with Metric Temporal Constraints

Till Hofmann, Gerhard Lakemeyer
Knowledge-Based Systems Group
RWTH Aachen University

At A Glance

The Problem

Executing a GOLOG program on a real robot is challenging:

- Programs typically only specify abstract behavior
- Need to account for hardware and software details

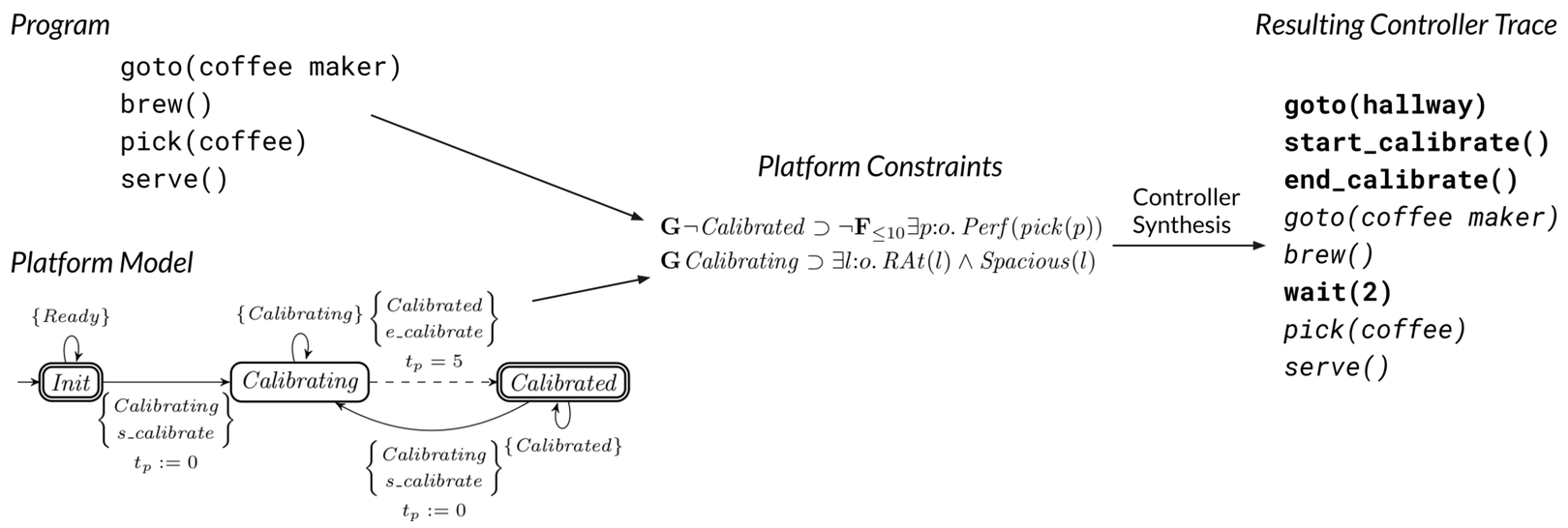
→ Need to satisfy additional constraints during execution

The Solution

- Model robot platform with timed automata
- Connect GOLOG program and robot platform with metric temporal constraints
- Transform program into a timed automaton representation
- Synthesize a controller that executes the original program and controls the platform

→ Resulting controller satisfies all platform constraints while executing the program

Synthesis Overview



1. A Simple Carrier Bot Basic Action Theory (BAT)

- Simple robot with two durative actions: `pick` and `goto`
- Durative actions modeled with start and end actions
- Predicate $Perf(a)$ indicates that the robot is performing action a
- Cannot perform actions simultaneously
- Precondition axioms:

$$\begin{aligned}&\Box Poss(a) \equiv\\ &\exists s{:}o \exists g{:}o.\ a = s\_goto(s,g) \wedge \neg\exists a'{:}a.\ Perf(a')\\ &\vee \exists s{:}o \exists g{:}o.\ a = e\_goto(s,g) \wedge Perf(goto(s,g))\\ &\vee \exists o{:}o, l{:}o.\ a = s\_pick(o) \wedge \neg\exists a'{:}a.\ Perf(a')\\ &\quad \wedge RAt(l) \wedge At(o,l)\\ &\vee \exists o{:}o.\ a = e\_pick(o) \wedge Perf(pick(o))\end{aligned}$$

- Successor state axioms:

$$\begin{aligned}&\Box[a] RAt(l) \equiv \exists s{:}o.\ a = e\_goto(s,l))\\ &\qquad \vee RAt(l) \wedge \neg\exists s'{:}o\, \exists g'{:}o.\ a = s\_goto(s',g')\\ &\Box[a] At(p,l) \equiv At(p,l) \wedge a \neq s\_pick(p)\\ &\Box[a] Holding(p) \equiv a = e\_pick(p) \vee Holding(p)\\ &\Box[a] Perf(a') \equiv\\ &\qquad \exists s{:}o \exists g{:}o.\ [a = s\_goto(s,g)] \vee \exists o\,[a = s\_pick(o)]\\ &\qquad \vee Perf(a') \wedge \neg\exists s{:}o \exists g{:}o\,[a = e\_goto(s,g)]\\ &\qquad\qquad \wedge \neg\exists p{:}o\,[a = e\_pick(p)]\end{aligned}$$

- Initial situation:
  - two locations m_1 and m_2, one object o_1
  - robot at m_1, object o_1 at m_2
  - m_1 has enough room for arm calibration

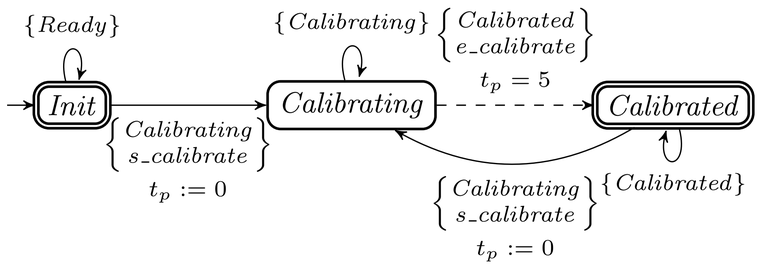
$$\begin{aligned}\Sigma_0 = \{&\\ \forall x{:}o.\ RAt(x) &\equiv (x = m_1),\\ \forall x{:}o\, \forall y{:}o.\ At(x,y) &\equiv (x = o_1 \wedge y = m_2),\\ \forall x{:}o.\ Spacious(x) &\equiv (x = m_1),\\ \tau_o(x) &\equiv (x = m_1 \vee x = m_2 \vee x = o_1),\\ \tau_a(a) &\equiv (a = s\_goto(m_1,m_2) \vee \ldots \vee a = e\_pick(o_1))\}\end{aligned}$$

- Abstract program to move and pick up the object:

$$\begin{aligned}&\pi l_o.\ At(o_1,l_o)?;\\ &\quad s\_goto(m_1,l_o); e\_goto(m_1,l_o); s\_pick(o); e\_pick(o);\end{aligned}$$

2. Platform Models

- Each platform component is a timed automaton (TA)
- A robot arm with three states:
  `Init`, `Calibrating`, and `Calibrated`
- Calibration takes exactly 5 seconds



3. Platform Constraints

1. If the robot's arm is not calibrated, it must not perform a pick action in the next 10 seconds, i.e., it must calibrate the arm before doing `pick`:

$$\mathbf{G}\neg Calibrated \supset \neg\mathbf{F}_{\leq 10}\exists p{:}o.\ Perf(pick(p))$$

2. If the robot is calibrating its arm, it must be at a location that provides enough space for doing so, i.e., a *Spacious* location:

$$\mathbf{G}\, Calibrating \supset \exists l{:}o.\ RAt(l) \wedge Spacious(l)$$

Major Steps

1. Write an abstract GOLOG program
2. Model the robot platform with timed automata
3. Define platform constraints connecting program and platform
4. Transform the program into a timed automaton
5. Apply MTL Controller Synthesis to obtain a controller

Timed $\mathcal{ESG}$ [Hofmann and Lakemeyer, 2018]

- Modal variant of the Situation Calculus
- Extends $\mathcal{ESG}$ [Claßen and Lakemeyer, 2008] with metric time
- Combines MTL and Situation Calculus

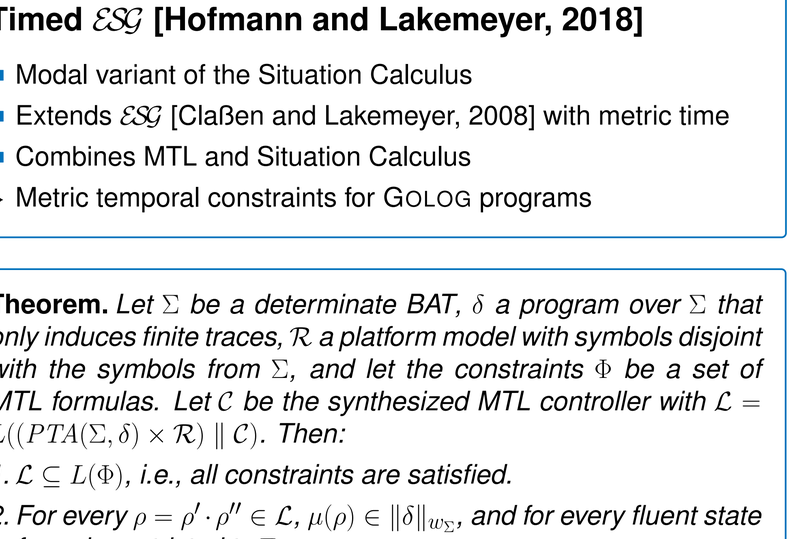
→ Metric temporal constraints for GOLOG programs

Theorem. *Let Σ be a determinate BAT, δ a program over Σ that only induces finite traces, $\mathcal{R}$ a platform model with symbols disjoint with the symbols from Σ, and let the constraints Φ be a set of MTL formulas. Let $\mathcal{C}$ be the synthesized MTL controller with $\mathcal{L} = L((PTA(\Sigma,\delta) \times \mathcal{R}) \parallel \mathcal{C})$. Then:*

1. *$\mathcal{L} \subseteq L(\Phi)$, i.e., all constraints are satisfied.*
2. *For every $\rho = \rho' \cdot \rho'' \in \mathcal{L}$, $\mu(\rho) \in \|\delta\|_{w_\Sigma}$, and for every fluent state formula restricted to Σ:*

$$\rho' \models \alpha \Leftrightarrow w_\Sigma, \mu(\rho') \models \alpha$$

Resulting Controller

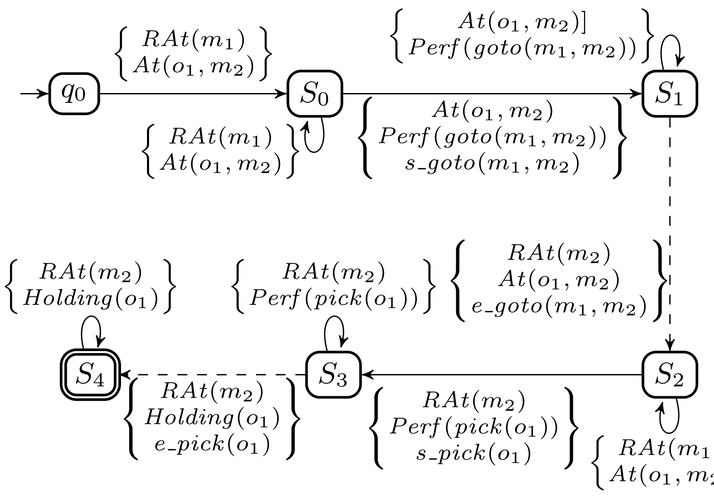


4. TA Representation of a Program

- Convert a GOLOG BAT into a TA
- Each transition is labeled with an action (if any) and all fluents that are true after the action
- A trace in the TA corresponds to a trace of the program

We define the TA $PTA(\Sigma,\delta) = (S, q_0, \to, F)$ as follows:

1. $q_0 \xrightarrow{P,\emptyset,\emptyset} (\langle\rangle, \delta)$ with $P = \{f_i \in \mathcal{P}_\Sigma \mid w_\Sigma[f_i, \langle\rangle] = 1\}$
2. $(z,\delta) \xrightarrow{P\cup\{a\},\emptyset,\emptyset} \left(z \cdot a, \delta'\right)$ iff $\left(z^0,\delta\right) \overset{w_\Sigma}{\Rightarrow} \left((z\cdot a)^0, \delta'\right)$ and $P = \{f_i \in \mathcal{P}_\Sigma \mid w_\Sigma[f_i, (z\cdot a)^0] = 1\}$
3. $(z,\delta) \xrightarrow{P,\emptyset,\emptyset} (z,\delta)$ with $P = \{f_i \in \mathcal{P}_\Sigma \mid w_\Sigma[f_i, z] = 1\}$
4. $(z,\delta) \in F$ iff $\langle z^0, \delta\rangle \in \mathcal{F}^{w_\Sigma}$



5. MTL Controller Synthesis [Bouyer et al., 2006]

- Given an alphabet $P = P_C \cup P_E$ partitioned into a set of *controllable* actions P_C and a set of *environment* actions P_E
- A *plant* $\mathcal{P}$ over P is a deterministic TA
- A μ-controller for $\mathcal{P}$ is a deterministic STS $\mathcal{C}$ over a symbolic alphabet based on $(P, X_\mathcal{P} \cup X_\mathcal{C})$ with granularity μ and satisfying:
  1. $\mathcal{C}$ does not reset the clocks of the plant: $q_\mathcal{C} \xrightarrow{a,g,Y} q'_\mathcal{C}$ implies $Y \subset X_\mathcal{C}$,
  2. $\mathcal{C}$ does not restrict environment actions: if $\sigma \in L(\mathcal{P} \parallel \mathcal{C})$ and $\sigma\,(e,t) \in L(\mathcal{P})$ with $e \in P_E$, then $\sigma \cdot (e,t) \in L(\mathcal{P} \parallel \mathcal{C})$
  3. $\mathcal{C}$ is non-blocking: if $\sigma \in L(\mathcal{P} \parallel \mathcal{C})$ and $\sigma\,(a,t) \in L(\mathcal{P})$ and $\sigma \cdot (a,t) \in L(\mathcal{P})$, then $\sigma \cdot \left(b,t'\right) \in \mathcal{L}^*(\mathcal{P} \parallel \mathcal{C})$ for some $b \in P$ and $t' \in \mathbb{Q}$
  4. all states of $\mathcal{C}$ are accepting.
- A μ-controller $\mathcal{C}$ *controls* $\mathcal{P}$ against the specification of desired behaviors Φ iff $L(\mathcal{P} \parallel \mathcal{C}) \subseteq L(\Phi)$

Restrictions

- BAT is *determinate*, i.e., initial state is completely known
- Finite domain: BAT axioms list all objects and actions
- Program must terminate (no infinite traces)
- Platform actions must not interfere with the abstract program

Knowledge-Based Systems Group

RWTH AACHEN UNIVERSITY

Acknowledgements

This work was supported by the German Research Foundation (DFG) under Grant GL-747/23-1.