Strategy Synthesis for First-Order Agent Programs over Finite Traces

35th Nordic Workshop on Programming Theory

Till Hofmann, Jens Claßen

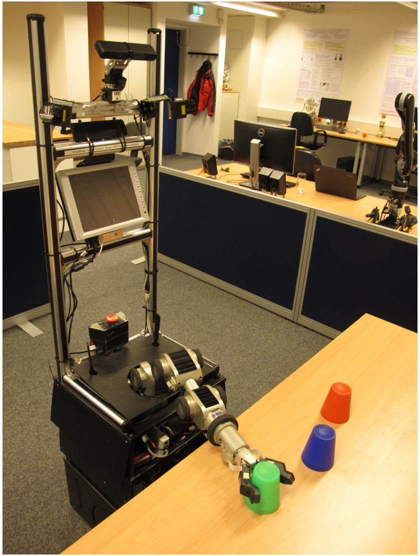
November 6, 2024

Roskilde University
Chair of Computer Science 6 (Machine Learning and Reasoning)
RWTH AACHEN UNIVERSITY

High-Level Reasoning on Robots

What is high-level reasoning in cognitive robotics?

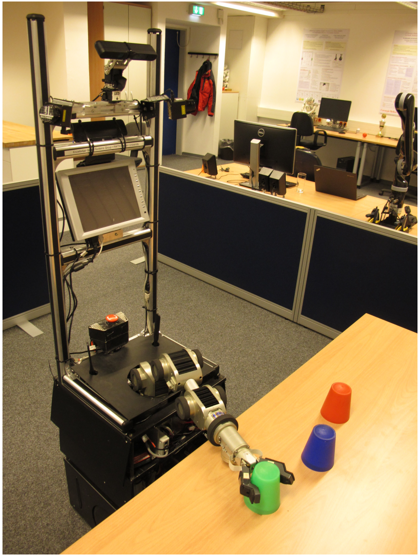
- Given: A skilled robot capable of doing *primitive* actions, e.g., `pick`, `goto`
- Goal: Determine what the robot is supposed to do
- Different methods how to accomplish this, e.g.,
  - task planning (i.e., heuristic search)
  - agent programs
  - reactive synthesis



High-Level Reasoning on Robots

What is high-level reasoning in cognitive robotics?

- Given: A skilled robot capable of doing *primitive* actions, e.g., `pick`, `goto`
- Goal: Determine what the robot is supposed to do
- Different methods how to accomplish this, e.g.,
  - task planning (i.e., heuristic search)
  - agent programs
  - reactive synthesis



The Situation Calculus and $\mathcal{ES}$

- Situation calculus (McCarthy and Hayes 1969):
  Formal framework based on First-order logic for describing dynamically changing worlds
- $\mathcal{ES}$: A modal variant of the situation calculus (Lakemeyer and Levesque 2010)
  - Modality $[\alpha]\phi$: ϕ *holds after performing action* α
  - Modality $\Box\phi$: ϕ *holds after any sequence of actions*
- All changes in the world are caused by *actions*
- *Fluent*: relation that may change its value from situation to situation
- A *situation* $z \in \mathcal{Z}$ is a sequence of actions describing the current state of the world, e.g.,

$$z = goto(room_1), load(cup_1, room_1), goto(kitchen), unload(cup_1)$$

- A *world* $w \in \mathcal{W}$ assigns a truth value to every fluent in every possible situation:

$$w : \mathcal{P}_F \times \mathcal{Z}^* \rightarrow \{0, 1\}$$

Basic Action Theories

A *basic action theory* (BAT) axiomatizes the world:

- The initial situation $\mathcal{D}_0$ describes the (possibly incomplete) initial state, e.g.:

$$At(kitchen) \land \neg \exists x \, OnRobot(x)$$

- The *precondition axiom* $\mathcal{D}_{\text{pre}}$ states when each action can be performed:

$$\square \, \text{Poss}(unload(x)) \equiv OnRobot(x) \land At(kitchen)$$

- The successor state axioms (SSAs) $\mathcal{D}_{\text{post}}$ describe how actions change fluents:

$$\square [a] OnRobot(x) \;\equiv\; \exists y. \, a = load(x, y) \lor OnRobot(x) \land a \neq unload(x)$$

We can use a first-order theorem prover to check entailment, e.g.,:

$$\mathcal{D} \models [goto(room_1)][load(cup_1, room_1)] \exists x \, OnRobot(x)$$

High-Level Programming with Golog

loop:
  while $\exists x.\, OnRobot(x)$ **do**
    $\pi x : \{d_1, \ldots, d_m\}.\, unload(x)$;
  $\pi y : \{r_1, \ldots, r_n\}.\, goto(y)$;
  while $\exists x.\, DirtyDish(x, y)$ **do**
    $\pi x : \{d_1, \ldots, d_m\}.\, load(x, y)$;
  $goto(kitchen)$

GOLOG: High-level agent programming language

- Imperative language tailored to specify agent behavior
- Atomic instructions: $\mathcal{ES}$ actions
- Allows nondeterministic constructs:
  - $\pi(x)$ Pick a value for variable x
  - δ^* Repeat sub-program δ
  - $\delta_1 | \delta_2$ Choose between δ_1 and δ_2
  - $\delta_1 \| \delta_2$ Interleaved concurrency
  - **loop** and **while** can be defined as macros
- New modality $[\delta]\phi$:
  after every possible execution of program δ, ϕ holds

What is missing?

- GOLOG allows expressing abstract behavior in an expressive language
- *However:* Assumes *angelic nondeterminism*, i.e., agent may choose which branch to follow
- Hence, all changes in the worlds follow the agents choices
- Unrealistic in many real-world settings:
  - Actions may have unintended effects, e.g., dropping a cup while moving
  - Humans may interfere and change the world, e.g., placing a new dirty dish on the table
  - The agent may need to *react* to requests $\rightarrow$ temporal goals

Program Realization as Synthesis

- Idea: Also model uncontrollable behavior as part of the agent program, e.g., during execution, a new dirty dish may appear in any room:

  loop: $\pi x : \{d_1, \ldots, d_m\}, y : \{r_1, \ldots, r_n\}.\, \mathit{newDish}(x, y)$
- Partition all actions into *controllable* and *uncontrollable* actions
- Program realization now becomes a *synthesis task*: Determine a successful execution of the program while considering all possible environment choices
- Here: Goal given as LTL formula Φ, interpreted over finite traces, e.g.,

$$\mathcal{F}\,\mathcal{G}\,\neg\exists x, y.\, \mathit{DirtyDish}(x, y)$$

- Task: Given a Golog program $\mathcal{P} = (\mathcal{D}, \delta)$ and a partitioning of the actions, find a *policy* π:
  - π must follow the program δ
  - π must allow all possible environment actions
  - Every π trace must satisfy Φ

Solving the Synthesis Problem: Main Challenges

1. First-order logic and thus $\mathcal{ES}$ is undecidable
2. Check the satisfaction of the temporal goal Φ
3. Determine a strategy that executes the program δ and satisfies Φ

Solving the Synthesis Problem: Main Challenges

1. First-order logic and thus $\mathcal{ES}$ is undecidable

$\rightarrow$ Need to find a decidable fragment

2. Check the satisfaction of the temporal goal Φ

3. Determine a strategy that executes the program δ and satisfies Φ

Solving the Synthesis Problem: Main Challenges

1. First-order logic and thus $\mathcal{ES}$ is undecidable

$\rightarrow$ Need to find a decidable fragment

2. Check the satisfaction of the temporal goal Φ

$\rightarrow$ Split Φ into two parts:

   1. Sub-formulas that are satisfied in the current state
   2. Sub-formulas that must be satisfied in some future state

3. Determine a strategy that executes the program δ and satisfies Φ

Solving the Synthesis Problem: Main Challenges

1. First-order logic and thus $\mathcal{ES}$ is undecidable

$\rightarrow$ Need to find a decidable fragment

2. Check the satisfaction of the temporal goal Φ

$\rightarrow$ Split Φ into two parts:

   1. Sub-formulas that are satisfied in the current state
   2. Sub-formulas that must be satisfied in some future state

3. Determine a strategy that executes the program δ and satisfies Φ

$\rightarrow$ Game-theoretic approach: construct a finite game arena and label recursively

Step 1: Decidable Fragment

Zarrieß and Claßen 2016 describe a decidable fragment of $\mathcal{ESG}$ for verification:

1. The base logic is restricted to the two-variable fragment of FOL with counting (C^2)
2. Successor state axioms must be acyclic; i.e., if an effect on F depends on some fluent F', then any effect on F' may not depend on F
3. The pick operator π may only pick from a finite set of ground terms

Step 1: Decidable Fragment

Zarrieß and Claßen 2016 describe a decidable fragment of $\mathcal{ESG}$ for verification:

1. The base logic is restricted to the two-variable fragment of FOL with counting (C^2)
2. Successor state axioms must be acyclic; i.e., if an effect on F depends on some fluent F', then any effect on F' may not depend on F
3. The pick operator π may only pick from a finite set of ground terms

With these restrictions, we can define a finite abstraction of a Golog program $\mathcal{P} = (\mathcal{D}, \delta)$:

- A *characteristic graph* is a finite representation of the program expression δ
- Due to restriction 3, only finitely many ground actions $\mathcal{A}$ may occur
- With 2, a program may only accumulate finitely many effects
  $\rightarrow$ collect them in a set $\mathfrak{E}^{\mathcal{D},\mathcal{A}}$

Step 1: Decidable Fragment by Means of Finite Abstraction

- Finally, the *context* $\mathcal{C}(\mathcal{P})$ is a **finite** set of formulas consisting of:
  - sentences in the initial theory
  - context conditions in the SSAs
  - formulas in guards and termination conditions of the program
  - $\mathcal{ES}$ sub-formulas in the temporal goal Φ
- There are infinitely many worlds $w \in \mathcal{W}$ with $w \models \mathcal{D}$
- However, we only have finitely many effects $\mathfrak{E}^{\mathcal{D},\mathcal{A}}$ and a finite context $\mathcal{C}(\mathcal{P})$

→ Define an equivalence relation among worlds where two worlds are equivalent if they satisfy the same context formulas after the same sequence of actions

- Each equivalence class is represented by the *type*:

$$\mathrm{type}(w) \doteq \{(\psi, E) \mid w \models \mathcal{R}[E, \psi]\}$$

"ψ holds in w after applying E"

context condition from $\mathcal{C}(\mathcal{P})$

effect from $\mathfrak{E}^{\mathcal{D},\mathcal{A}}$

Step 2: Tracking the satisfaction of Φ

- LTLf is like LTL, but interpreted over finite traces (De Giacomo and Vardi 2015)
- Here: same syntax, but replacing propositions with $\mathcal{ES}$ fluent sentences

$$\Phi ::= \phi \mid \Phi \wedge \Phi \mid \mathcal{X}\,\Phi \mid \Phi\,\mathcal{U}\,\Phi$$

- We adopt two notions from (Li et al. 2020):
  1. *Tail Normal Form* (TNF): introduce explicit proposition Tail that marks the last state of a trace, e.g.,

$$\operatorname{tnf}(\Phi_1\,\mathcal{U}\,\Phi_2) \doteq (\neg \mathit{Tail} \wedge \operatorname{tnf}(\Phi_1))\,\mathcal{U}\,\operatorname{tnf}(\Phi_2)$$

  2. *neXt Normal Form* (XNF): transform formulas such that outermost temporal operator is $\mathcal{X}$, e.g.,

$$\operatorname{xnf}(\Phi_1\,\mathcal{U}\,\Phi_2) \doteq \operatorname{xnf}(\Phi_2) \vee (\operatorname{xnf}(\Phi_1) \wedge \mathcal{X}(\Phi_1\,\mathcal{U}\,\Phi_2))$$

Step 2: Tracking the satisfaction of Φ

- With XNF, we can interpret an LTLf formula Φ as propositional formula Φ^p
- Each temporal formula is a proposition
- If Φ is satisfiable, then so is Φ^p
- Determine propositional assignments for Φ^p using a SAT solver
- Split propositional assignment P intro three parts:
  1. $L(P) = \{l \mid l \in P$ is a literal other than $(\neg)\mathit{Tail}\}$
  2. $X(P) = \{\theta \mid \mathcal{X}\,\theta \in P\}$
  3. $T(P) = \top$ if $\mathit{Tail} \in P$ and $T(P) = \bot$ otherwise

Putting Things Together: The Game Arena

- We construct a *game arena* $\mathbb{A}_{\mathcal{G}}^{\Phi}$ that captures the execution of $\mathcal{P}$ while tracking the satisfaction of Φ
- Each state is of the form

$$s = (\tau, E, \{(\chi_i, \theta_i)\}_i, \rho)$$

accumulated effects (E), *remaining program* (ρ), *type of the world* (τ), $X(P_i)$ *and* $T(P_i)$ *of some prop. assignment* P_i (χ_i, θ_i)

- There is a transition $s_1 \xrightarrow{\alpha} s_2$ from $s_1 = (\tau, E, A_1, \rho_1)$ to $s_2 = (\tau, E_2, A_2, \rho_2)$ if
  - the program can transition from ρ_1 to ρ_2 by doing action α
  - E_2 are the effects resulting from applying α in τ on effects E_1
  - $(\chi_2, \theta_2) \in A_2$ if for some $(\chi_1, \theta_1) \in A_1$, there is a propositional assignment P for $\mathrm{xnf}(\bigwedge \chi_1^p)$ such that

$$\theta_1 = \bot \quad \{(\psi, E_2) \mid \psi \in L(P)\} \subseteq \tau \quad \chi_2 = X(P) \quad \theta_2 = T(P)$$

- A state is *final* if ρ is in a terminating configuration
- A state is *accepting* if $(\emptyset, \top) \in A$, i.e., Φ is satisfied

$\Rightarrow$ Solve the game to determine strategy

Example with 1 room and 1 cup

- Agent:

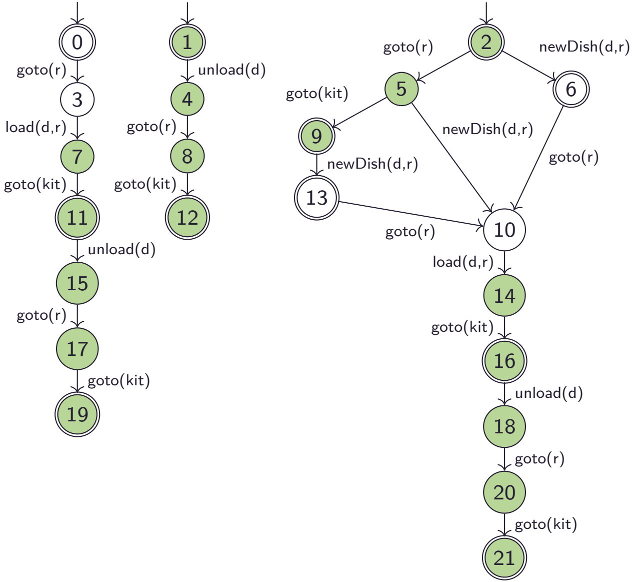
  loop:
  while $\exists x.\ OnRobot(x)$ **do**
  $\pi x : \{d_1\}.\ unload(x);$
  $\pi y : \{r_1\}.\ goto(y);$
  while $\exists x.\ DirtyDish(x, y)$ **do**
  $\pi x : \{d_1\}.\ load(x, y);$
  $goto(kitchen)$

- Environment:

  loop: $\pi x : \{d_1\}, y : \{r_1\}.\ newDish(x, y)$

- Initial state: $At(kitchen)$
- Goal:

$$\mathcal{F}\,\mathcal{G}\,\neg\exists x, y.\ DirtyDish(x, y)$$



Conclusion

- GOLOG is an expressive agent programming language based on first-order logic
- Assumption so far: The agent is under complete control
- More realistic view: Agent acts in a partially controllable environment

→ Program realization is now a synthesis task with an LTL_f goal

- Approach:
  - Finite abstraction of the infinite program configuration space
  - Use a game-theoretic approach to determine a policy

⇒ Resulting policy guarantees to satisfy the goal, independent of the environment's choices

Appendix

Computing a Strategy

- Based on the finite game arena $\mathbb{A}_{\mathcal{G}}^{\Phi}$, determine a *terminating* and *winning* strategy
  terminating: The agent must eventually terminate by not choosing any actions
  winning: In every terminating state, the temporal goal Φ must be satisfied
- In principle, we can just start with the final+accepting states $\mathcal{S}_F \cap \mathcal{S}_A$ and label bottom up
- Problem: Even in a final+accepting state, the environment may continue and eventually lead into bad states

→ *Guess* a subset $H \subseteq \mathcal{S}_F \cap \mathcal{S}_A$

- Check whether there is a strategy that enforce each play to end in H
- Label nodes bottom up with $\top / \bot$
- Any strategy that remains in the $\top$-labeled sub-graph is a terminating and winning strategy

Bibliography (I)

- De Giacomo, Giuseppe and Moshe Y. Vardi (2015). "Synthesis for LTL and LDL on Finite Traces". In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, pp. 1558–1564.
- Lakemeyer, Gerhard and Hector J. Levesque (2010). "A semantic characterization of a useful fragment of the situation calculus with knowledge". In: *Artificial Intelligence* 175.1, pp. 142–164.
- Li, Jianwen et al. (Dec. 2020). "SAT-based Explicit LTLf Satisfiability Checking". In: *Artificial Intelligence* 289, p. 103369. DOI: 10.1016/j.artint.2020.103369.
- McCarthy, John and Patrick J. Hayes (1969). "Some Philosophical Problems from the Standpoint of Artificial Intelligence". In: *Machine Intelligence* 4, pp. 463–502.
- Zarrieß, Benjamin and Jens Claßen (2016). "Decidable Verification of Golog Programs over Non-Local Effect Actions". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, pp. 1109–1115.

Till Hofmann, Jens Claßen

Copenhagen, Denmark, November 6, 2024