On the Power of LTL_f in Assured Autonomy

Shufang Zhu
shufang.zhu@cs.ox.ac.uk

May 16, 2023

University of Oxford

Artificial Intelligence

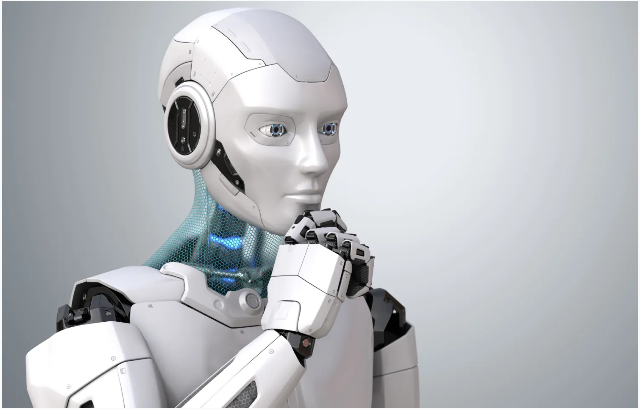
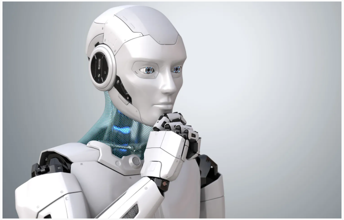


Image from https://www.gevers.eu/blog/artificial-intelligence/video-post/

Artificial Intelligence

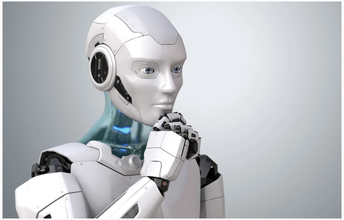
AI aims at devising systems that act autonomously

- Autonomy, one of the grand challenges in AI



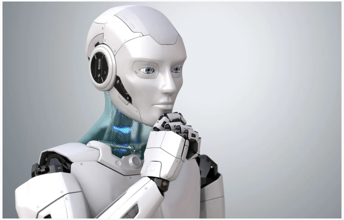
Artificial Intelligence

- Autonomy, one of the grand challenges in AI
  - Autonomous agents/robots, operating in a changing, incompletely known, unpredictable environments



Artificial Intelligence

- Autonomy, one of the grand challenges in AI
  - Agents with the ability of autonomously deliberating how to act to environment changes to achieve a given task



AI aims at devising systems that act autonomously

- AI agents with the ability to self-deliberate its own behaviours carries significant risks
- AI agents with **Assured Autonomy**

AI aims at devising systems that act autonomously

- AI agents with the ability to self-deliberate its own behaviours carries significant risks
- AI agents with **Assured Autonomy**

How to achieve assured autonomy?

- Formal Methods (FM), automated synthesis[1]
  - Both the environment and the task are formally specified
  - Mechanical translation of human-understandable environment and task specifications to a program that is known to meet the task wrt the environment[2]

[1] A. Pnueli, R. Rosner, POPL1989; B. Finkbeiner, 2016
[2] M. Vardi - The Siren Song of Temporal Synthesis, 2018

How to achieve assured autonomy?

- Formal Methods (FM), automated synthesis[1]
  - Both the environment and the task are formally specified
  - Mechanical translation of human-understandable environment and task specifications to a program that is known to meet the task wrt the environment[2]

[1] A. Pnueli, R. Rosner, POPL1989; B. Finkbeiner, 2016
[2] M. Vardi - The Siren Song of Temporal Synthesis, 2018

How to achieve assured autonomy?

- Formal Methods (FM), automated synthesis[1]
  - Both the environment and the task are formally specified
  - Mechanical translation of human-understandable environment and task specifications to a program that is known to meet the task wrt the environment[2]

[1] A. Pnueli, R. Rosner, POPL1989; B. Finkbeiner, 2016
[2] M. Vardi - The Siren Song of Temporal Synthesis, 2018

How to achieve assured autonomy?

- Formal Methods (FM), automated synthesis[1]
  - Both the environment and the task are formally specified
  - Mechanical translation of human-understandable environment and task specifications to a program that is known to meet the task wrt the environment[2]

[1]A. Pnueli, R. Rosner, POPL1989; B. Finkbeiner, 2016
[2]M. Vardi - The Siren Song of Temporal Synthesis, 2018

How to achieve assured autonomy?

- Specification language in FM
  - Linear Temporal Logic (LTL)[3], remarkable applicability
  - Interpreted over infinite traces, relating to non-terminating systems

[3] A. Pnueli, FOCS1977

How to achieve assured autonomy?

- Specification language in FM
  - Linear Temporal Logic (LTL)[3], remarkable applicability
  - Interpreted over infinite traces, relating to non-terminating systems

[3] A. Pnueli, FOCS1977

How to achieve assured autonomy?

- Specification language in FM
  - Linear Temporal Logic (LTL)[3], remarkable applicability
  - Interpreted over infinite traces, relating to non-terminating systems
- AI agents are not dedicated to a single task all their life but are supposed to accomplish one task after another

[3] A. Pnueli, FOCS1977

How to achieve assured autonomy?

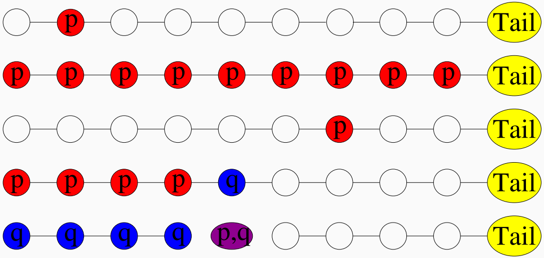
- Specification language in FM
  - Linear Temporal Logic (LTL)[3], remarkable applicability
  - Interpreted over infinite traces, relating to non-terminating systems
- Specification language for AI agents
  - Linear Temporal Logic on finite traces (LTL_f)[4]

[3] A. Pnueli, FOCS1977
[4] G. De Giacomo, M. Vardi, IJCAI2013

Linear Temporal Logic over Finite Traces[5]

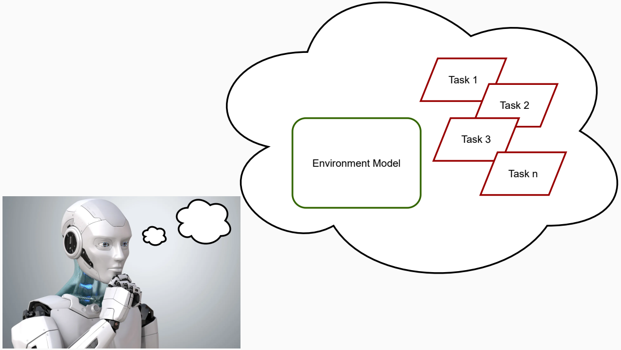
- finite set of atomic propositions {p, q}.
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$.
- temporal connectives:

$\mathcal{X}p$	NEXT TIME
$\Box p$	ALWAYS
$\Diamond p$	EVENTUALLY
$p\mathcal{U}q$	UNTIL
$p\mathcal{R}q$	RELEASE



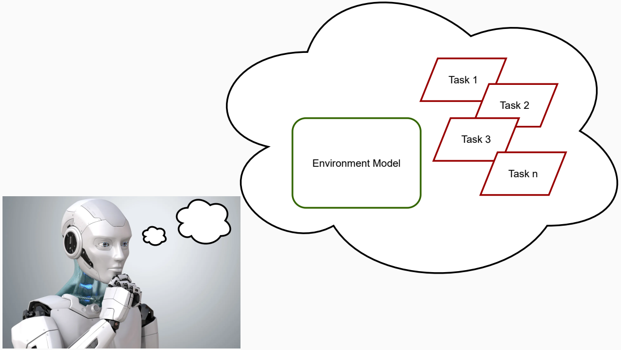
[5] Finite but no specific bound.

On the Power of LTL_f in Assured Autonomy



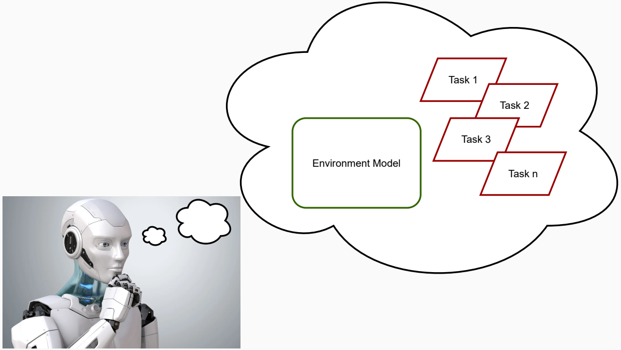
Env model: Specification of environment's behaviors

On the Power of LTL$_f$ in Assured Autonomy



Env model: planning domain, LTL/LTL$_f$ formula, $\emptyset$

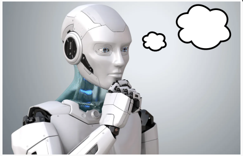
On the Power of LTL_f in Assured Autonomy

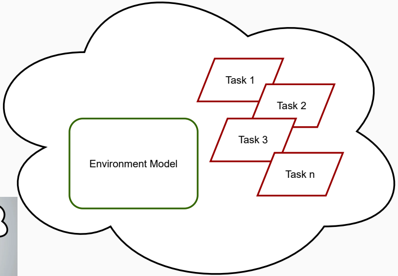


Env model: planning domain, LTL/LTL_f formula, $\emptyset$

Agent task: Specification of desired task/goal

On the Power of LTL_f in Assured Autonomy

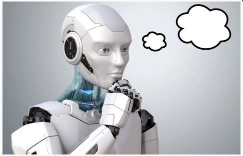


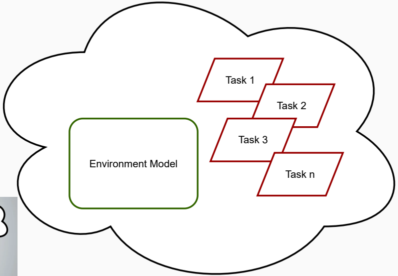


Env model: planning domain, LTL/LTL_f formula, $\emptyset$

Agent task: LTL_f formula

On the Power of LTL_f in Assured Autonomy





Env model: planning domain, LTL/LTL_f formula, $\emptyset$

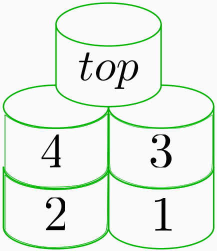
Agent task:

Obtain: An agent strategy that is **guaranteed to realize** the task wrt the environment

Synthesized Program (Strategy)

- at every time step
- make an action
- for every response from the env model
- the combined play (trace consists of moves from both env. and agn.)
- satisfies φ

Synthesized Program (Strategy)



1 Action move block to $L2$

  1.1 **Response** do-nothing

    1.1.1 Action move block to $L1$

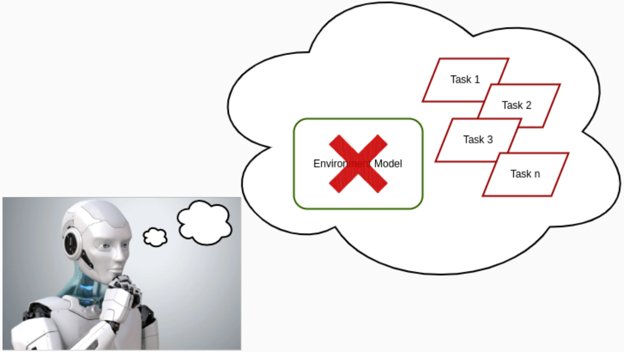
      ...

  1.2 **Response** remove block from $L2$

    1.2.1 Action move block to $L2$

      ...

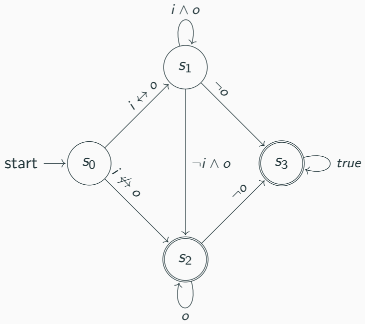
LTL$_f$ Synthesis



- **Given:** agent task φ
- **Obtain:** agent strategy **guaranteed to realize** φ against the environment

LTL_f Synthesis and Reachability Game[6]

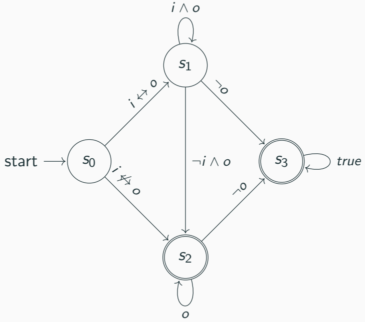
- Key point: LTL_f φ and corresponding Deterministic Finite Automata (DFA)
  - A trace π satisfies φ iff π is accepted by the DFA



[6] G. De Giacomo, M. Vardi, IJCAI2013

LTL_f Synthesis and Reachability Game[6]

- Key point: LTL_f φ and corresponding Deterministic Finite Automata (DFA)
  - A trace π satisfies φ iff π is accepted by the DFA

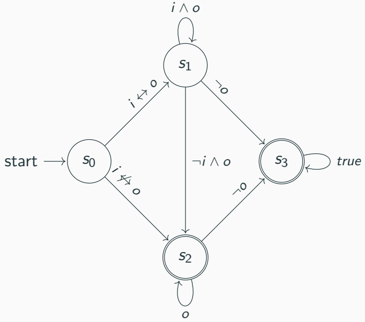


Adversarial reachability

[6] G. De Giacomo, M. Vardi, IJCAI2013

LTL$_f$ Synthesis and Reachability Game[6]

- Key point: LTL$_f$ φ and corresponding Deterministic Finite Automata (DFA)
  - A trace π satisfies φ iff π is accepted by the DFA



Adversarial reachability

- $W_0 = \{s_2, s_3\}$
- $W_1 = \{s_2, s_3, s_1\}$, $\omega(s_1) = \neg o$
- $W_2 = \{s_2, s_3, s_1, s_0\}$, $\omega(s_0) = o$
- $W_3 = W_2$, fixpoint!

Strategy $\omega : Win \to 2^{\mathcal{O}}$

[6] G. De Giacomo, M. Vardi, IJCAI2013

Explicit to Symbolic

Drawback of explicit DFA:

The explicit DFA can have double-exponential states

Symbolic LTL_f synthesis framework[7]

Basic idea: binary encoding of state representation, exp fewer variables

[7] **S. Zhu**, L. M. Tabajara, J. Li, G. Pu, M. Vardi, IJCAI2017

Explicit to Symbolic

Drawback of explicit DFA:

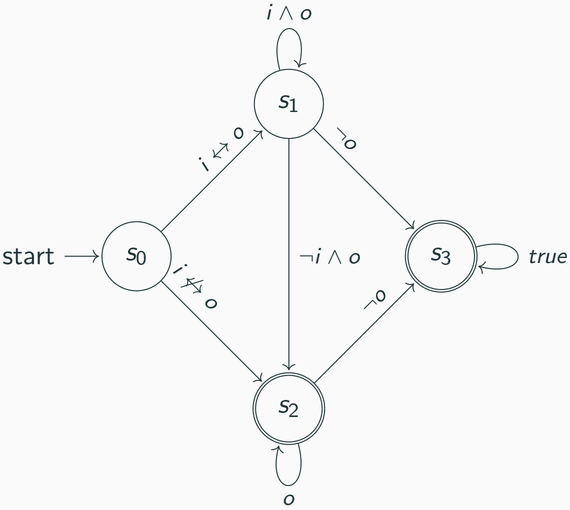
The explicit DFA can have double-exponential states

Symbolic LTL_f synthesis framework[7]

Basic idea: binary encoding of state representation, exp fewer variables

[7] **S. Zhu**, L. M. Tabajara, J. Li, G. Pu, M. Vardi, IJCAI2017

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation



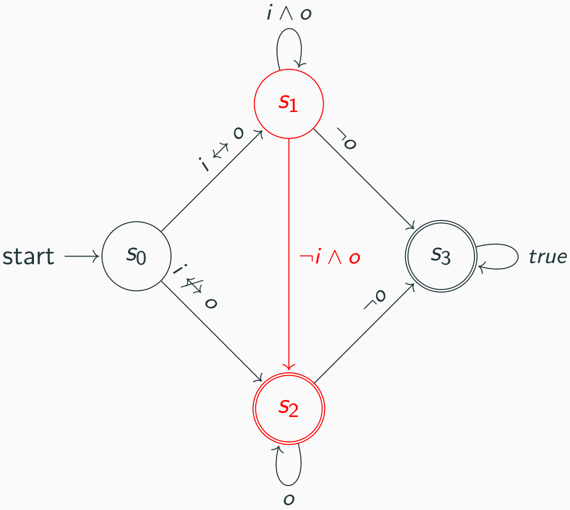
State variables: $\mathcal{Z} = \{z_0, z_1\}$

Transition function:

$\{\eta_z = \mathcal{Z} \times \mathcal{I} \times \mathcal{O} \to \{0,1\} \mid z \in \mathcal{Z}\}$

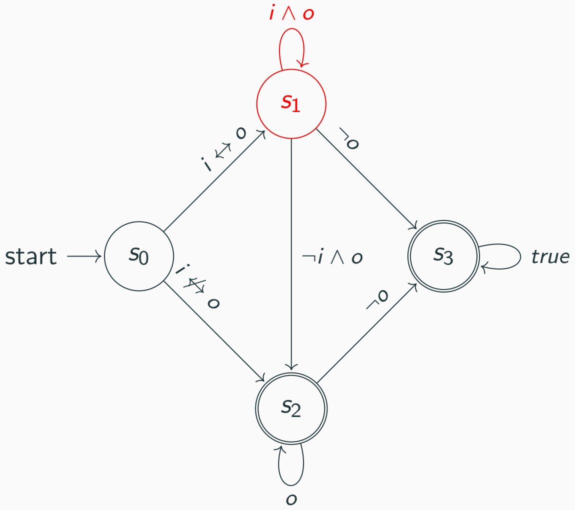
$\eta_z(Z, I, O) \in \{0,1\}$

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation



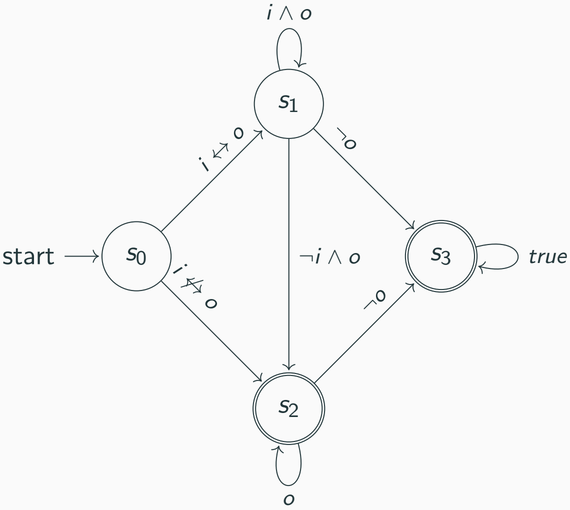
- $\underbrace{(\neg z_0, z_1}_{s_1(01)}, \neg i, o) \rightarrow \underbrace{z_0, \neg z_1}_{s_2(10)}$
- $\eta_{z_0}(\neg z_0, z_1, \neg i, o)$ evaluates to *true*
  $\eta_{z_1}(\neg z_0, z_1, \neg i, o)$ evaluates to *false*

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation



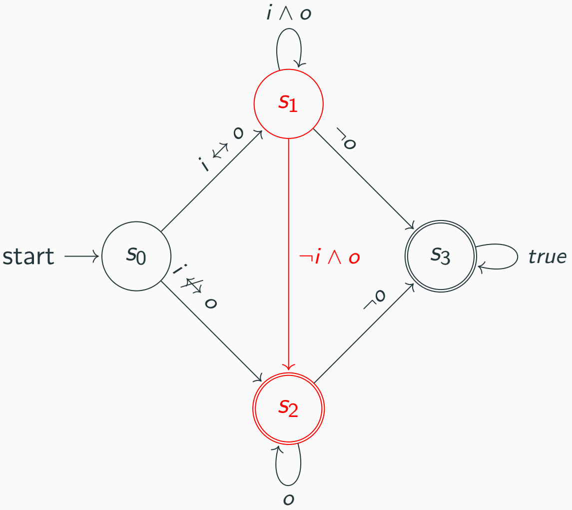
- $(\underbrace{\neg z_0, z_1}_{s_1(01)}, i, o) \rightarrow \underbrace{\neg z_0, z_1}_{s_1(01)}$
- $\eta_{z_0}(\neg z_0, z_1, i, o)$ evaluates to *false*
  $\eta_{z_1}(\neg z_0, z_1, i, o)$ evaluates to *true*

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation



- $\eta_{z_0}(\neg z_0, z_1, \neg i, o)$ evaluates to *true*
  $\eta_{z_1}(\neg z_0, z_1, \neg i, o)$ evaluates to *false*
- $\eta_{z_0}(\neg z_0, z_1, i, o)$ evaluates to *false*
  $\eta_{z_1}(\neg z_0, z_1, i, o)$ evaluates to *true*
- ...

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation

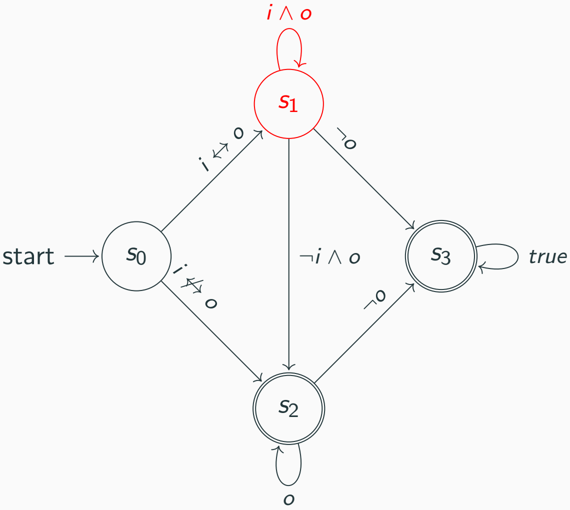


Only transitions evaluated to *true*

- $\eta_{z_0}(\neg z_0, z_1, \neg i, o)$ evaluates to *true*
- ~~$\eta_{z_0}(\neg z_0, z_1, i, o)$ evaluates to *false*~~
- ...

$$\eta_{z_0} = (\neg z_0 \wedge z_1 \wedge \neg i \wedge o) \vee \ldots$$

Symbolic LTL$_f$ Synthesis – Symbolic DFA Representation



Only transitions evaluated to *true*

- ~~$\eta_{z_1}(\neg z_0, z_1, \neg i, o)$ evaluates to *false*~~
- $\eta_{z_1}(\neg z_0, z_1, i, o)$ evaluates to *true*
- ...

$$\eta_{z_1} = (\neg z_0 \wedge z_1 \wedge i \wedge o) \vee \ldots$$

Symbolic LTL$_f$ Synthesis – Symbolic Reachability Game

Reachability game on symbolic DFA $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{Z}, \iota, \eta, f)$

- A Boolean formula w over $\mathcal{Z}$ for winning states
- A Boolean formula t over $\mathcal{Z} \cup \mathcal{O}$ for (winning state, winning output) pairs

Symbolic LTL$_f$ Synthesis – Symbolic Reachability Game

Reachability game on symbolic DFA $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{Z}, \iota, \eta, f)$

- $w_0 = f$ every accepting state is a winning state
- $t_0 = f$ the task is accomplished (*true*) after reaching accepting states

Symbolic LTL$_f$ Synthesis – Symbolic Reachability Game

$t_{i+1} = t_i \vee (\neg w_i \wedge \forall I. w_i(\eta))$

- (Z, O) satisfies t_i
- Z was not yet a winning state, and for every I we can move from Z to an already-identified winning state

Symbolic LTL_f Synthesis – Symbolic Reachability Game

$$t_{i+1} = t_i \lor (\neg w_i \land \forall I . w_i(\eta))$$
$$w_{i+1} = \exists O . t_{i+1}$$

- Z satisfies w_i
- Z was not yet a winning state, and there exists O such that for every I we can move from Z to an already-identified winning state

Symbolic LTL$_f$ Synthesis – Symbolic Reachability Game

Reachability game on symbolic DFA $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{Z}, \iota, \eta, f)$

- $w_{i+1} \equiv w_i$, fixpoint w_∞

Symbolic LTL_f Synthesis – Abstract Winning Strategy

Function $\omega : \text{Win} \rightarrow 2^{\mathcal{O}}$

- Input: winning state s
- Output: winning output O of s

Symbolic LTL$_f$ Synthesis – Abstract Winning Strategy

Function $\omega : \text{Win} \rightarrow 2^{\mathcal{O}}$

- Input: winning state s
- Output: winning output O of s

We have Boolean formula t over $\mathcal{Z} \cup \mathcal{O}$

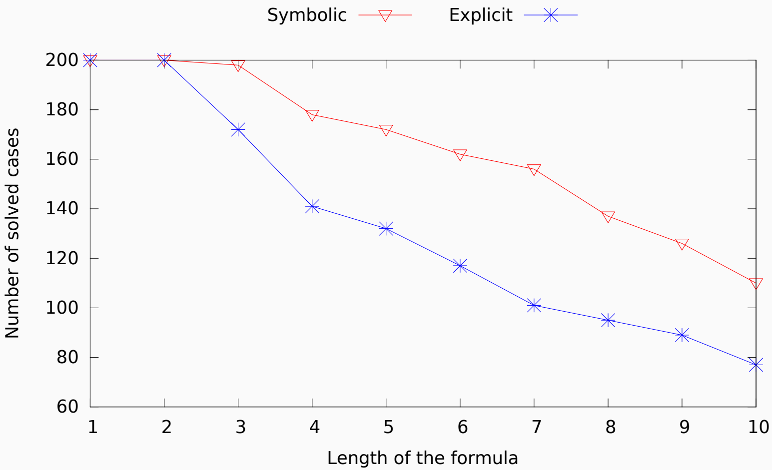
- $(Z \cup O) \models t$ iff Z is a winning state and O is a winning output of Z

Symbolic LTL$_f$ Synthesis – Abstract Winning Strategy

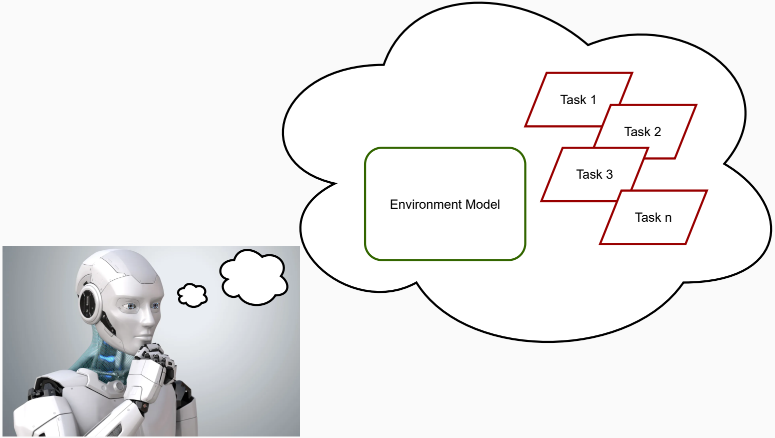
t over $\mathcal{Z} \cup \mathcal{O}$ as the input formula to a Boolean synthesis procedure

- function $\tau : 2^{\mathcal{Z}} \rightarrow 2^{\mathcal{O}}$

Symbolic LTL$_f$ Synthesis



Symbolic LTL$_f$ Synthesis with Env Model



Symbolic LTL_f Synthesis with Env Model

Synthesis with Environment Models

- Markovian environment behaviours
  - Planning domain[8]
- Non-Markovian environment behaviours, e.g., specified in LTL formulas
  - Simple Fairness and Stability[9]
  - Generalized Reactivity (1) and Safety[10]
  - General LTL formula[11]

[8] K. He, A. M. Wells, L. E. Kavraki, M. Vardi, ICRA2019
[9] **S. Zhu**, G. De Giacomo, G. Pu, M. Vardi, AAAI2020
[10] G. De Giacomo, A. Di Stasio, L. M. Tabajara, M. Vardi, **S. Zhu**, IJCAI2021
[11] G. De Giacomo, A. Di Stasio, M. Vardi, **S. Zhu**, KR2020

Symbolic LTL_f Synthesis with Env Model

Synthesis of LTL_f with environment model in LTL

Step-1: task in LTL_f, abstract winning region of the agent task in LTL_f

Step-2: environment model in LTL, with respect to the winning region

Symbolic LTL_f Synthesis with Env Model

Synthesis of LTL_f with environment model in LTL

Step-1: task in LTL_f, abstract winning region of the agent task in LTL_f

Step-2: environment model in LTL, with respect to the winning region

Practically diminish the difficulty of reasoning the mix of LTL/LTLf specifications

Symbolic LTL_f Synthesis: Limitation

- Backward fixpoint computation on constructed DFA
- **Pros:** Computing the winning region of the task in LTL_f
  - Keep the expressiveness of environment models
  - Maintain the simplicity of reasoning LTL_f specifications
- **Cons:** double-exponential blowup of LTL_f-to-DFA construction

Symbolic LTL_f Synthesis: Limitation

- Backward fixpoint computation on constructed DFA
- **Pros:** Computing the winning region of the task in LTL_f
  - Keep the expressiveness of environment models
  - Maintain the simplicity of reasoning LTL_f specifications
- **Cons:** double-exponential blowup of LTL_f-to-DFA construction

Symbolic LTL$_f$ Synthesis: Limitation

- Backward fixpoint computation on constructed DFA
- **Pros:** Computing the winning region of the task in LTL$_f$
  - Keep the expressiveness of environment models
  - Maintain the simplicity of reasoning LTL$_f$ specifications
- **Cons:** double-exponential blowup of LTL$_f$-to-DFA construction

Symbolic LTL_f Synthesis: Limitation

- Backward fixpoint computation on constructed DFA
- **Cons:** double-exponential blowup of LTL_f-to-DFA construction
  - Limits the scalability in Markovian Decision Process (MDP)-solving problems, e.g., planning with LTL_f tasks

Forward LTL_f Synthesis

– Diminish the double-exponential blowup practically

– Synthesis on the fly[8]

- Abstract a strategy while constructing the DFA
- Construct the complete DFA only in the worst case

[8] G. De Giacomo, M. Favorito, J. Li, S. Xiao, M. Vardi, **S. Zhu**, IJCAI2022

Forward LTL_f Synthesis

- Diminish the double-exponential blowup practically
- Synthesis on the fly[8]
  - Abstract a strategy while constructing the DFA
  - Construct the complete DFA only in the worst case

[8] G. De Giacomo, M. Favorito, J. Li, S. Xiao, M. Vardi, **S. Zhu**, IJCAI2022

Forward LTL_f Synthesis: DFA construction

Construct search space on-the-fly via formula progression

- LTL_f formula φ, as a DFA state, what happens **now** (label), what should happen **next** accordingly (successor state)
- $\varphi = a\,\mathcal{U}\,b$, *a stays true until b holds*

Forward LTL_f Synthesis: DFA construction

Construct search space on-the-fly via formula progression

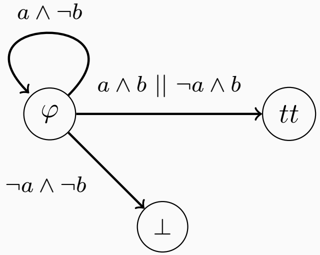
- LTL_f formula φ, as a DFA state, what happens **now** (label), what should happen **next** accordingly (successor state)
- $\varphi = a\,\mathcal{U}\,b \equiv b \vee (a \wedge \mathcal{X}(a\,\mathcal{U}\,b))$, *a stays true until b holds*

Forward LTL$_f$ Synthesis: DFA construction

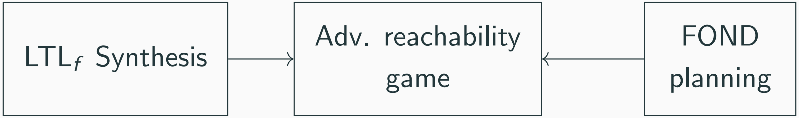
Construct search space on-the-fly via formula progression

- LTL$_f$ formula φ, as a DFA state, what happens **now** (label), what should happen **next** accordingly (successor state)
- $\varphi = a\,\mathcal{U}\,b \equiv b \vee (a \wedge \mathcal{X}(a\,\mathcal{U}\,b))$, *a stays true until b holds*

- **now** $= a \wedge b$, **next** $= tt$
- **now** $= a \wedge \neg b$, **next** $= a\,\mathcal{U}\,b$
- **now** $= \neg a \wedge b$, **next** $= tt$
- **now** $= \neg a \wedge \neg b$, **next** $= \bot$



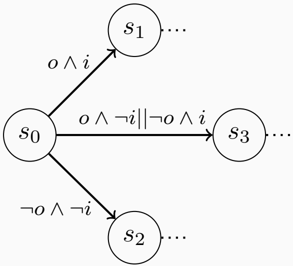
Forward LTL_f Synthesis: Abstract Strategy

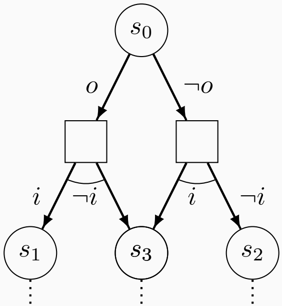


- FOND planning, state space only single-exponential
- LTL_f synthesis, state space is double-exponential
- Existing planners cannot directly solve LTL_f synthesis on-the-fly

Forward LTL$_f$ Synthesis: Abstract Strategy

LTL$_f$ Synthesis as AND-OR graph search





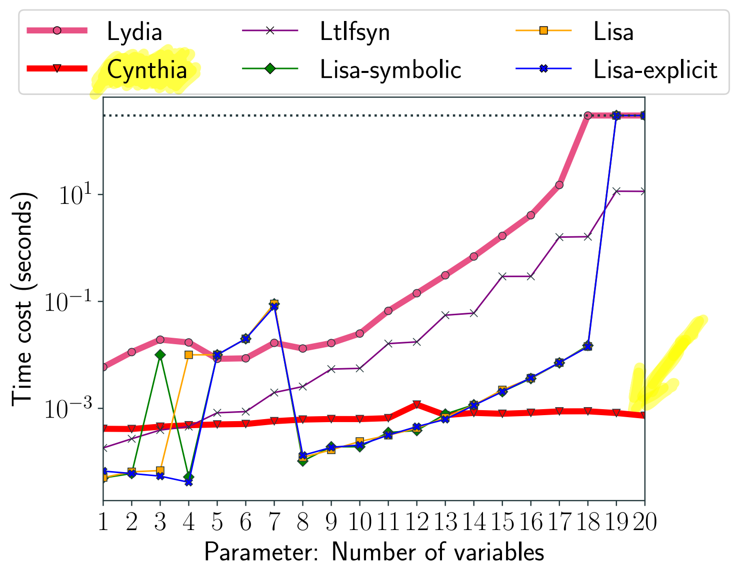
LTL_f Synthesis as AND-OR Graph Search

Knowledge compilation techniques, e.g., Sentential Decision Diagrams (SDDs)[9]

- Compress labels leading to the same nodes, reduce the branching factor

[9] A. Darwiche, IJCAI2011

Experimental Results on Until-Patterns



Forward LTL_f Synthesis

- LTL_f synthesis adopting AND-OR graph search
- Uninformed search, promising synthesis performance
- Move from uninformed search to informed search exploiting heuristics

Conclusions

Assured Autonomy through LTL_f synthesis

- Backward symbolic LTL_f synthesis
  - Synthesize strategy based on the winning region computation
  - Separate the reasoning of the environment model and the agent task
- Forward LTL_f synthesis adopting AND-OR graph search
  - Synthesize strategy on-the-fly, without computing the winning region
  - Applicable to MDP-solving problems, e.g., planning with LTL_f tasks

Assured Autonomy with Resilience

Resilience: the ability to recover from unexpected circumstances

- "Creating resilient systems means thinking hard in advance about what could go wrong and incorporating effective countermeasures into designs."[10]

[10] W. A. Galston. WSJ, March 10, 2020.

Assured Autonomy with Resilience - Challenges

How to appropriately model the contingencies?

How to handle contingencies?

Assured Autonomy with Resilience - Directions

1. Structured model to describe contingencies
   - Combining Markovian and non-Markovian dynamics
2. Contingencies in the environment behavior
3. Contingencies in the agent behavior

Assured Autonomy with Resilience - Resilience Against Env Contingencies

- **Best-Effort strategy**: a program to handle both expected and contingent environment dynamics[11]
  - Symbolic best-effort LTL_f synthesis, both env and task are in LTL_f[12]
- **Maximally permissive strategy**: all possible strategies to meet the task
  - Maximally permissive strategy of LTL_f specifications[13]

[11] B. Aminof, G. De Giacomo, S. Rubin, IJCAI2021

[12] G. De Giacomo, G. Parretti, **S. Zhu**, GenPlan2022

[13] **S. Zhu**, G. De Giacomo, IJCAI2022

Assured Autonomy with Resilience - Resilience Against Agn Contingencies

- **Expected move**, optional tasks
  - Complete optional agent tasks while guaranteeing mandatory tasks[14]
- **Unexpected move**, an agent with a trembling hand
  - 2-player game becomes 2.5-player game

[14] **S. Zhu**, G. De Giacomo, KR2022

Questions?

Assured Autonomy through LTL_f synthesis

- Backward LTL_f synthesis
- Forward LTL_f synthesis

Assured Autonomy with resilience

- Structured specification model
- Resilience against environment contingencies
- Resilience against agent contingencies