# Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

Giuseppe De Giacomo

*University of Rome "La Sapienza", Italy*

ESSLLI 2021

Workshop on Automated Synthesis

July 29-30, 2021, ESSLLI 2021

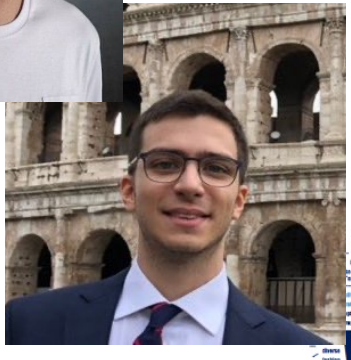
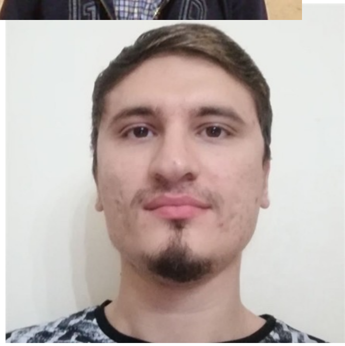
# WhiteMech Group

WhiteMech: Whitebox Self Programming Mechanisms

ERC Advanced Grant

WE ARE HIRING!





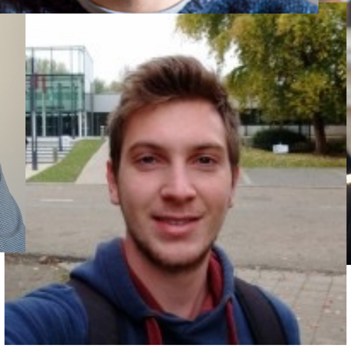
European Research Council

Established by the European Commission







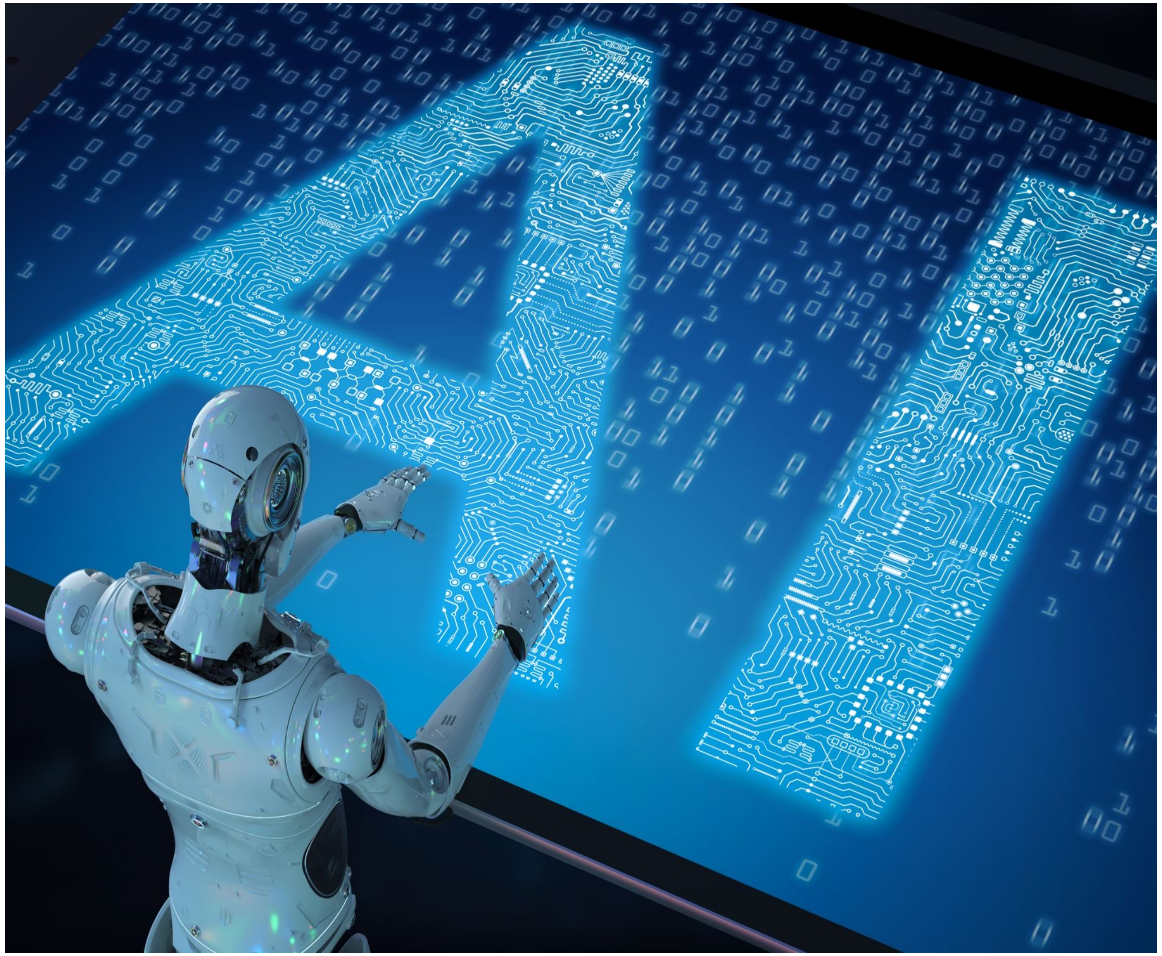


Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

# INTRODUCTION

# Autonomy in AI

- Autonomy is one of the grand objectives of AI.
- Aims at building autonomous agents/robots that operate in changing, incompletely known, unpredictable environments.
- Requires autonomous reasoning and planning capabilities, as well as learning from experience.



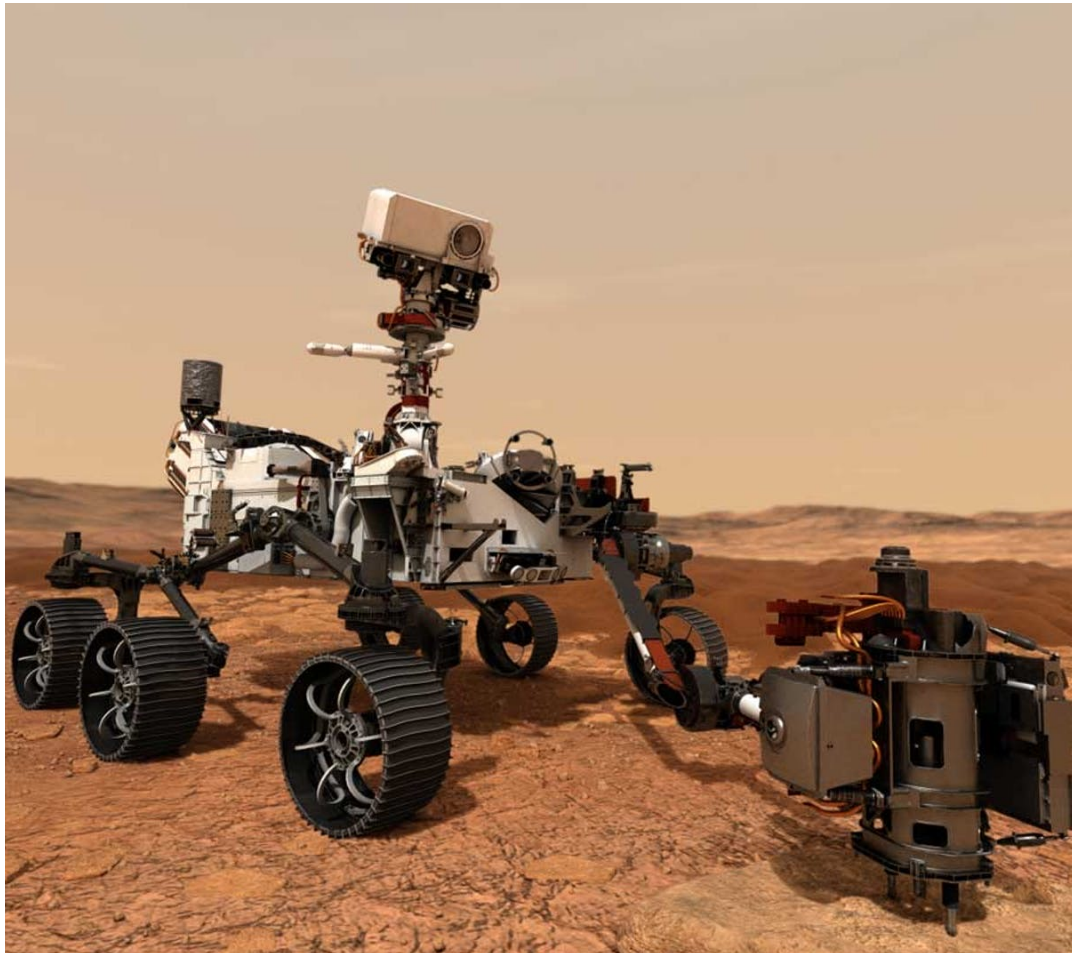
# Space Exploration

Delay in communication requires high-level of autonomy during the mission.

Planning and scheduling for temporal extended goals is a top research topic at NASA.

https://mars.nasa.gov/mars2020/

NASA Science
MARS 2020 MISSION
PERSEVERANCE ROVER

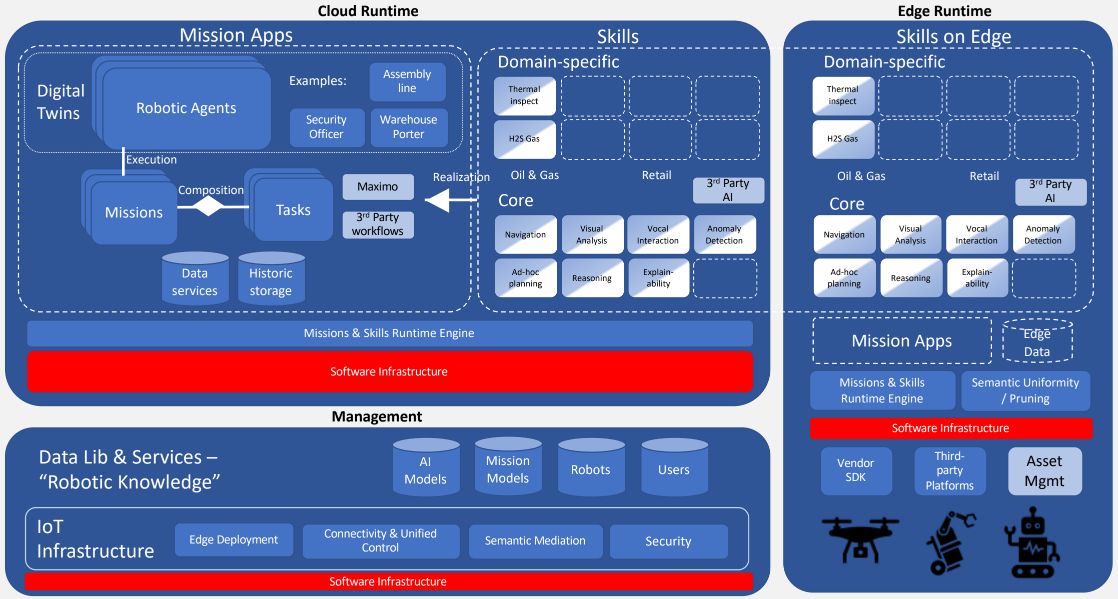


# Autonomous Mobile Robots in Logistics

Complex **multi-robot systems** need highly synchronized behaviours to fulfil their job.

These **robots need autonomously resolve** unexpected clashes.

Sophisticated **AMR platforms** under study in industry




# Smart Manufacturing and Digital Twins

- **Manufacturing as a service** products to be manufactured are not known in advance and each product may differ from the products manufactured immediately before and immediately after it
- Analogies with **Service Composition and Orchestration**: synthesize the orchestrator
- **Digital Twins** platforms offer infrastructure to deploy orchestrations
- **Automated exception handling** is crucial

*G. De Giacomo, M. Vardi, P. Felli, N. Alechina, B. Logan: Synthesis of Orchestrations of Transducers for Manufacturing. AAAI 2018*

*N. Alechina, T. Brázdil, G. De Giacomo, P. Felli, B. Logan, M. Vardi: Unbounded Orchestrations of Transducers for Manufacturing. AAAI 2019*

# Autonomy Requires Reasoning and Learning

- Autonomy requires:
  - reasoning and planning capabilities
  - learning from experience
- Many areas of AI are concern with autonomy:
  - Logics in AI
  - Knowledge representation and reasoning
  - Planning
  - Multi-agent systems
  - Sequential decision making (MDPs)
  - Reinforcement learning
- Recently: some objectives are shared with automated synthesis in formal methods

WhiteMech: Whitebox Self Programming Mechanisms

ERC Advanced Grant


European Research Council

Established by the European Commission

Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

# FUNDAMENTALS

# KR: Have a Model of Environment

Knowledge Representation:

"Equip agent with a model of the environment it acts in"

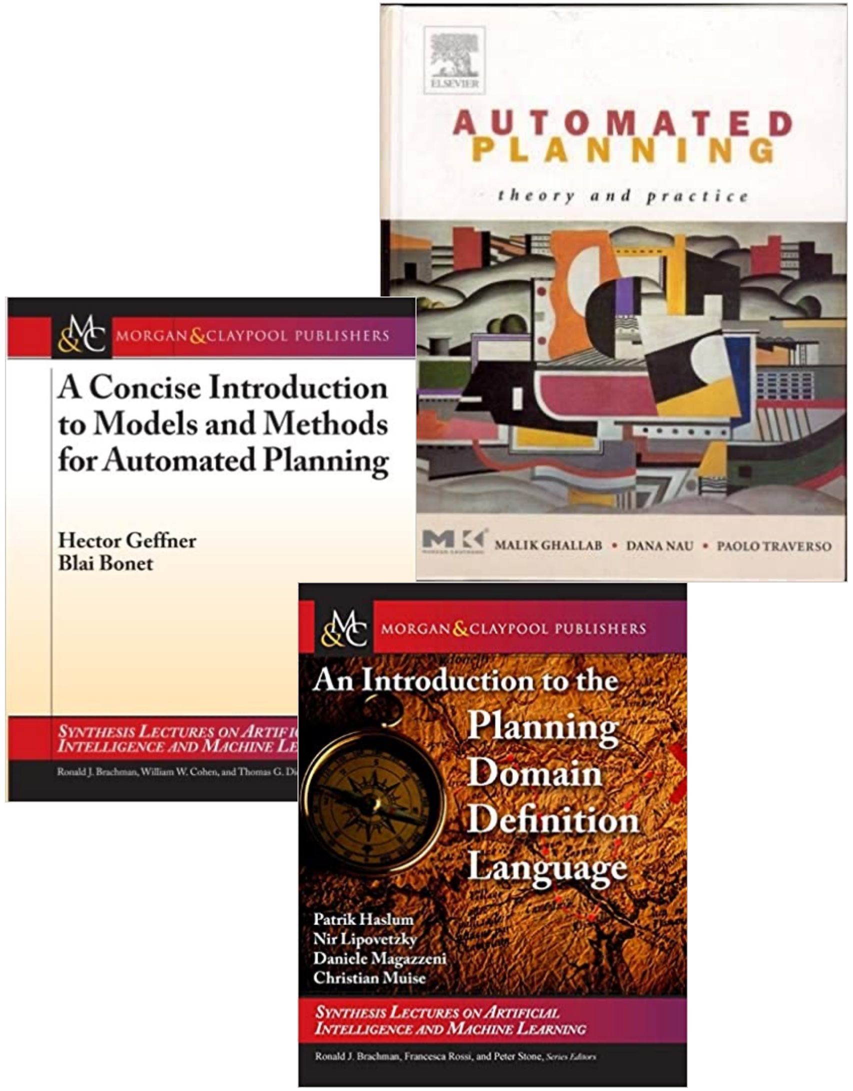
Reasoning about actions (classical view):

- Capture model of the environment with a logical theory
  - Preconditions for agent actions
  - Effects of agent actions + solution to "Frame Problem"
- Multiple interpretations of the theory
  - Multiple possible instantiations of the environment (one of them the correct one, but we do not know which)
- Reasoning (skeptical)
  - based on logical implication

# Planning: Model the Environment as a Transition System

Planning:

- Inherit from KR the idea that environment can be represented in terms of
  - Preconditions for agent actions
  - Effects of agent actions + solution to "Frame Problem"
- But use them as a specification for generating a transition system
  - a single model vs a theory
- Reasoning
  - based on "model checking"
- Two notable cases:
  - Classical planning: everything determined by agent actions
  - Planning in nondeterministic domains (FOND):
    - Agent: instructs actions
    - Env: determine their effects



# MDPs: Model the Environment Stochastically

MDPs:

- Very similar to Planning in a nondeterministic domain (FOND)
  - Agent: instructs actions
  - Env: determine their effects
- But env chooses effects stochastically (vs adversarial)
  - With known and stationary probability distributions.
- It's the framework at the base of Reinforcement Learning

Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

# FOCUS ON FINITE TRACES FOR TASKS

# KR and Planning: Agent Tasks Must Terminate

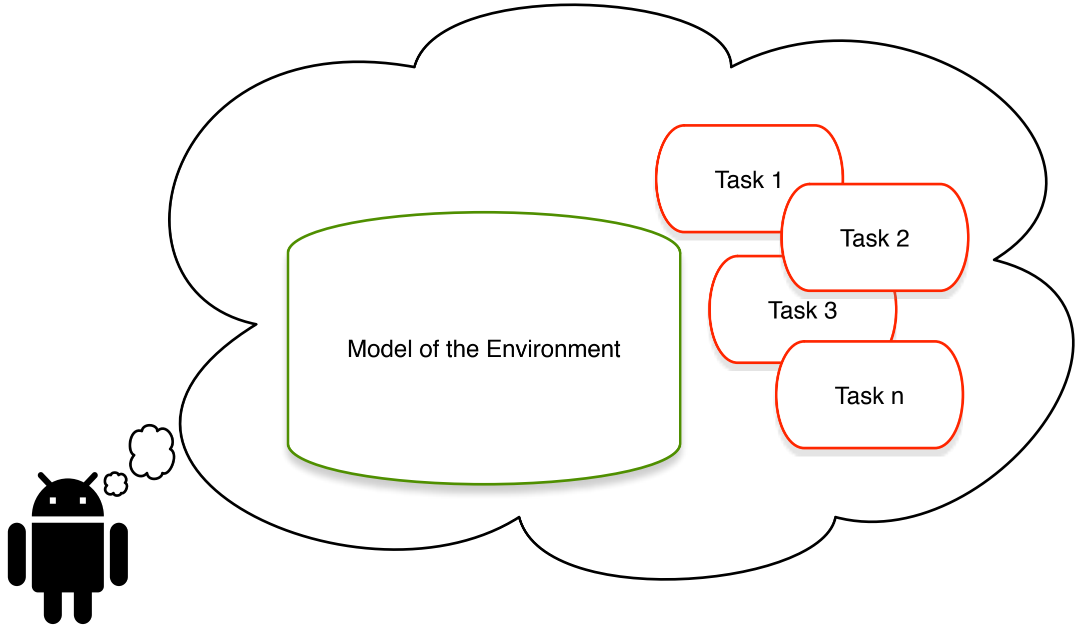
## Planning in AI:

- Is all about having a task specification or "goal" and producing a "plan" (or strategy or policy) to satisfy the task in the environment model.

- Which tasks?
  - A **task that terminates**!
  - Typically, just reaching a certain state in the environment

## Why tasks that terminates?

- Because it is the agent that is planning/reasoning
- If the task would not terminate, the agent would be stuck into doing the same task forever
- But then, why bother with equipping it with a model of the environment and of the task at all?
- Note it is the agent, NOT the designer, who has such models



# Focus on Finite Traces for Tasks

In fact, the Reasoning about Actions and Planning community is adopting temporal logics since a long time often, interpret LTL on finite traces.

- Temporally extended goals [BacchusKabanza96] - infinite/finite
- Temporal constraints on trajectories [GereviniHaslumLongSaettiDimopoulos09 - PDDL3.0 2009] - finite
- Declarative control knowledge on trajectories [BaierMcIlraith06] - finite
- Procedural control knowledge on trajectories [BaierFrizMcIlraith07] - finite
- Temporal specification in planning domains [CalvaneseDeGiacomoVardi02] - infinite
- Planning via model checking - infinite
  - Branching time (CTL) [CimattiGiunchigliaGiunchigliaTraverso97]
  - Linear time (LTL) [DeGiacomoVardi99]

Finite traces also considered in Declarative Business Processes in Business Process Management [vanderAalstPesicSchonenberg2009]

# Linear Time Logic on Finite Traces

## $\text{LTL}_f/\text{LDL}_f$: linear temporal logics on finite traces [DeGiacomoVardi2013]

### $\text{LTL}_f$: linear time temporal logic on finite traces

Interesting questionnaire on easiness of LTL:
https://brown.co1.qualtrics.com/jfe/form/SV_3gBU9j7yap90ICO

Same syntax as standard LTL but interpreted over finite traces

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \textit{next}\,\varphi \mid \textit{eventually}\,\varphi \mid \textit{always}\,\varphi \mid \varphi_1\;\textit{until}\,\varphi_2$$

| Examples: | | | |
|---|---|---|---|
| | *eventually* $A$ | "eventually $A$" | *reachability* |
| | *always* $A$ | "always $A$" | *safety* |
| | *always*$(A \rightarrow$ *eventually* $B)$ | "always if $A$ then eventually $B$" | *reactiveness* |
| | $A$ *until* $B$ | "$A$ until $B$" | *until* |
| | $\neg B$ *until* $A \vee$ *always* $\neg B$ | "$A$ before $B$" | *precedence* |

### $\text{LDL}_f$: linear dynamic logic on finite traces

Same syntax as PDL but interpreted over finite traces

$$\varphi ::= tt \mid A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\rho\rangle\varphi \mid [\rho]\varphi \qquad \rho ::= A \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1;\rho_2 \mid \rho^*$$

Adds the possibility of expressing procedural constraints/goals [Reiter01], [BaierFritzMcIlraith07]:

$$\delta ::= A \mid \varphi? \mid \delta_1 + \delta_2 \mid \delta_1;\delta_2 \mid \delta^* \mid \textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \mid \textbf{while } \phi \textbf{ do } \delta$$

where **if** and **while** are abbreviations: **if** $\phi$ **then** $\delta_1$**else** $\delta_2 \doteq (\phi?;\delta_1) + (\neg\phi?;\delta_2)$ and **while** $\phi$ **do** $\delta \doteq (\phi?;\delta)^*;\neg\phi?$

# Linear Time Logic on Finite Traces

## Example

- "*All coffee requests from person $p$ will eventually be served*":

$$\textit{always}(request_p \rightarrow \textit{eventually}\ coffee_p) \qquad [\mathtt{true}^*](request_p \rightarrow \langle \mathtt{true}^* \rangle coffee_p)$$

- "*Every time the robot opens door $d$ it closes it immediately after*":

$$\textit{always}(openDoor_d \rightarrow \textit{next}\ closeDoor_d) \qquad [\mathtt{true}^*]([openDoor_d]closeDoor_d)$$

- "*Before entering restricted area $a$ the robot must have permission for $a$*":

$$\neg inArea_a\ \textit{until}\ getPerm_a \vee \textit{always}\,\neg inArea_a \qquad \langle(\neg inArea_a)^*\rangle getPerm_a \vee [\mathtt{true}^*]\neg inArea_a$$

- "*Each time the robot enters the restricted area $a$ it must have a new permission for $a$*":

$$\langle(\neg inArea_a{}^*; getPerm_a; \neg inArea_a{}^*; inArea_a; inArea_a^*)^*; \neg inArea_a{}^*\rangle end$$

- "*At every point, if it is hot then, if the air-conditioning system is off, turn it on, else don't turn it off*":

$$[\mathtt{true}^*]\langle \textbf{if}\ (\textit{hot})\ \textbf{then}\ \textbf{if}\ (\neg \textit{airOn})\ \textbf{then}\ \textit{turnOnAir}\ \textbf{else}\ \neg \textit{turnOffAir}\rangle\, true$$

# LTLf to DFA

## Key point

$\mathrm{LTL}_f/\mathrm{LDL}_f$ formulas can be translated into deterministic finite state automata (DFA).

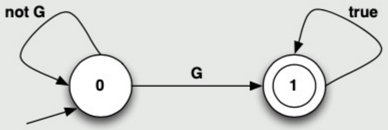
$$t \models \varphi \text{ iff } t \in \mathcal{L}(A_\varphi)$$

where $A_\varphi$ is the DFA $\varphi$ is translated into.

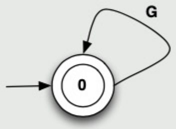
*NB: DFA canical after minimization!*

## Example (Automata for some $\mathrm{LTL}_f/\mathrm{LDL}_f$ formulas)

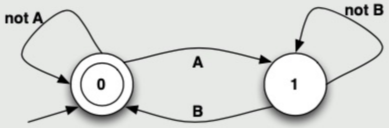
*eventually* $G$



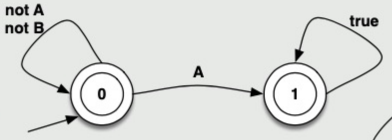

*always* $G$



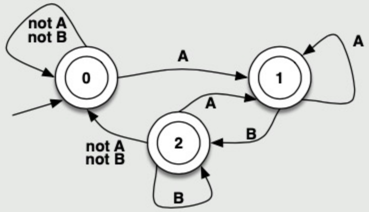

*always*$(A \rightarrow$ *next eventually* $B)$




$\neg B$ *until* $A \vee$ *always* $\neg B$ *"A before B"*




$\langle(\neg B^*; A; \neg B^*; B; B)^*; \neg B^*\rangle end$
*"each time new $A$ before $B$"* ($A$ and $B$ not true simultaneously)



# FOND Planning/Synthesis for LTLf Goals

## FOND for $\mathrm{LTL}_f$ goals

### Algorithm: FOND for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals

1: Given a FOND domain $\mathcal{D}$ and an $\mathrm{LTL}_f/\mathrm{LDL}_f$ goal $\varphi$
2: Compute DFA $A_\varphi$ for $\varphi$ (double exponential)
3: Compute product of $\mathcal{D}$ and $A_\varphi$ (polynomial)
4: Synthesize winning strategy for DFA game (linear)
5: Return strategy

### Theorem ([DeGiacomoRubinIJCAI18])

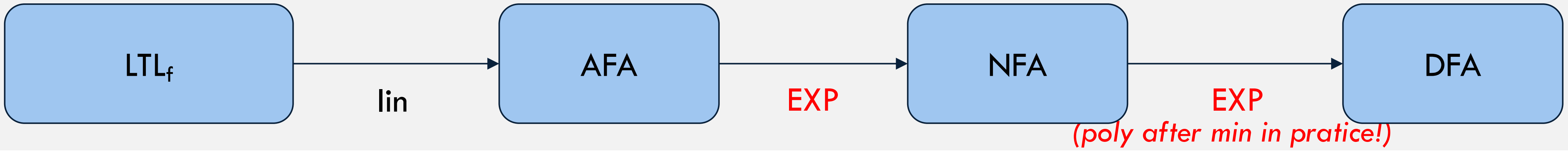
*FOND for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals is:*

- *EXPTIME-complete in the domain (assuming a logarithmic representation as in PDDL);*
- *2EXPTIME-complete in the goal.*

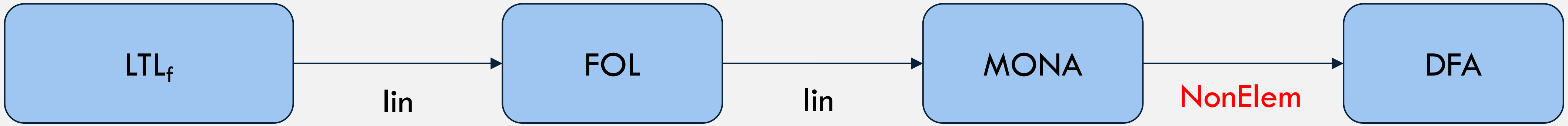
Note we have **separated cost** in the **model** (the doman) from that in the **task** (the goal)!
*(cf. data vs query complexity [ChandraHarel1980], [Vardi1982], [AbiteboulHullVianu1995])*

# LTLf to DFA (Advanced)

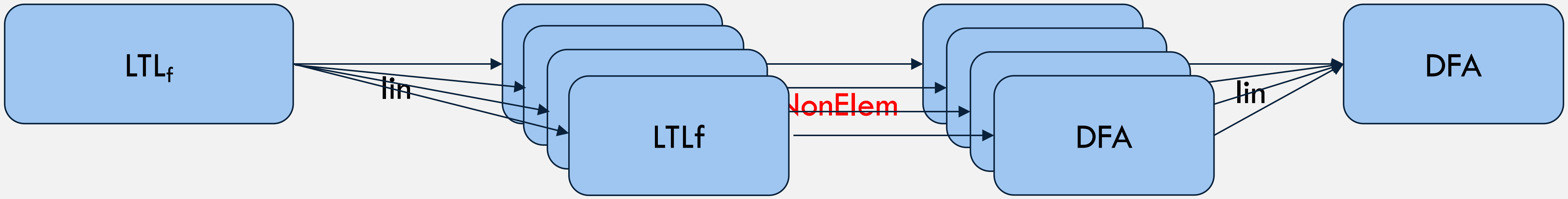
Monolitic tight bounds: [DeGiacomoVardi IJCAI2013/2015]




Monolitic via MONA [Zhu et al. IJCAI 2017]




Compositional [Bansal et al. AAAI2020], [DeGiacomoFavorito ICAPS2021]




Better in practice!

Online tool available! Monolitic via MONA < http://ltlf2dfa.diag.uniroma1.it>; Compositional <http://lydia.whitemech.it>

# LTLf to DFA (Advanced)

Use planning for doing deteminization on the fly [Camacho et al ICAPS 2018]

$$\mathrm{LTL}_f \xrightarrow{\text{lin}} \mathrm{AFA} \xrightarrow{\text{EXP}} \mathrm{NFA} \xrightarrow[\text{EXP}]{\text{PDDL/on the fly}} \mathrm{DFA}$$

On the fly forward fashion [Xiao et al. AAAI2021], [DeGiacomo et al. 2022 submitted]

$$\mathrm{LTL}_f \xrightarrow{\text{exploit formula semantics to obtain DFA on the fly in forward fashion}} \mathrm{DFA}$$

Based on “next normal form” or “progression” [BacchusKabanzaAAAI1998]:

eventually Red iff Red or next eventually Red

Important: transition must be “symbolic” i.e., propositional formulas

*… but cannot translate to PDDL (since 2EXP instead of 1EXP)*

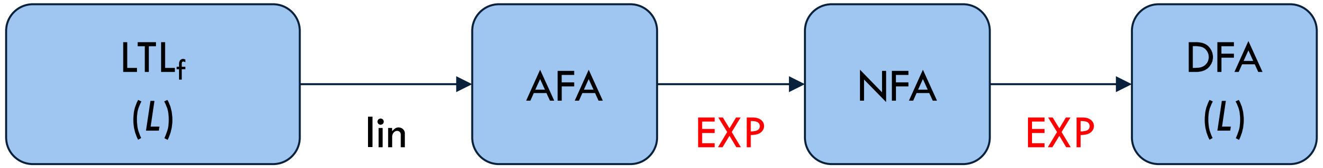
# Pure Past LTLf

## Pure past temporal specifications on finite traces

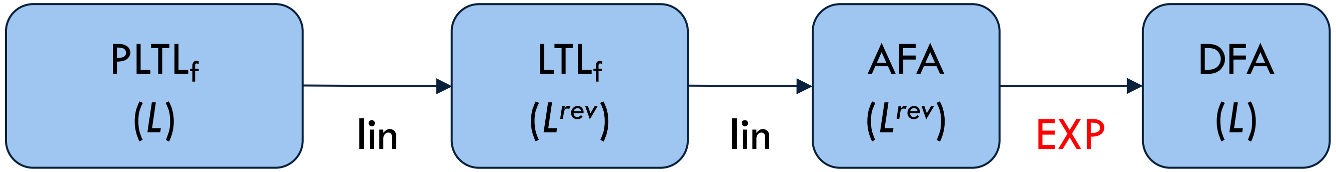
- Sometimes specifications are *easier* and *more natural* to express referring to the past
  ["The Glory of the Past" LichtensteinPnueliZuck1985]
  - Non-Markovian models [Gabaldon 2011]
  - Non-Markovian rewards in MDPs [Bacchus et al. 1996]
  - Normative properties in multi-agent systems [FisherWooldridge2005], [Knobbout et al. 2016], [Alechina et al. 2018]

- This is very convenient because we do have an exponential computational advantages in this cases

Pure Future Specs




Pure Past Specs



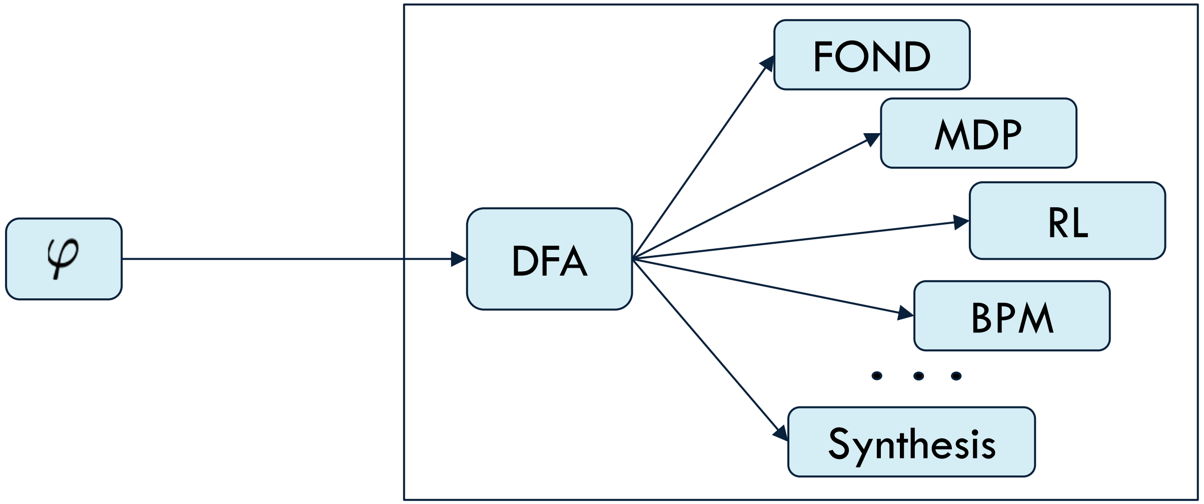

*Given an AFA of k states for language L, there exists a DFA of at most $2^k$ states for language $L^{reverse}$ [Chandra et al. 1981]*

G. De Giacomo, A. Di Stasio, F. Fuggitti, and S. Rubin.
Pure-Past Linear Temporal and Dynamic Logic on Finite Traces. IJCAI 2020 Survey Track.

# Several Applications of LTLf Specs

## Many Applications:

- FOND planning for temporally extended goals
- MDP with non-Markovian rewards
- Reinforcement Learning for non-Markovian tasks
- Declarative Process Specification in BPM
- Several forms of Synthesis



# MODELS OF THE ENVIRONMENT

# Classic KR, Planning, MDPs Focuses on Markovian Models

Classically AI focuses on

Markovian models of the environment:

- Environment is in a state
- Agent actions effects (and preconditions) depend only on the current state
- History of how we got in a certain state plays no role
- Action effects manifest at the very next state

# Nondeterministic Planinning Domains are Markovian

## FOND for $\mathrm{LTL}_f$ goals

### Algorithm: FOND for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals

1: Given a $\boxed{\text{FOND domain } \mathcal{D}}$ and an $\mathrm{LTL}_f/\mathrm{LDL}_f$ goal $\varphi$
2: Compute $\mathrm{DFA}$ $A_\varphi$ for $\varphi$ (double exponential)
3: Compute $\boxed{\text{product of } \mathcal{D} \text{ and } A_\varphi}$ (polynomial)
4: Synthesize winning strategy for $\mathrm{DFA}$ game (linear)
5: Return strategy

### Theorem ([DeGiacomoRubinIJCAI18])

*FOND for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals is:*

- *EXPTIME-complete in the domain (assuming a logarithmic representation as in PDDL);*
- *2EXPTIME-complete in the goal.*

Note we have **separated cost** in the **model** (the doman) from that in the **task** (the goal)!
*(cf. data vs query complexity [ChandraHarel1980], [Vardi1982], [AbiteboulHullVianu1995])*

# Beyond Markovian Models

This Markovian view is not foundational and can be challenged!

… and it has been challenged in literature:

- Non-Markovian action theories in Reasoning about Actions (e.g., in the Situation Calculus) [GabaldonAIJ2011]
  - Effects depend on the past history (safety properties)

- Trajectory constraints in Planning [CimattiPistoreRoveriTraversoAIJ2003]
  - Planning domain is a transition system/game arena
  - But in reacting to agent actions the environment has to fulfill certain temporal rules (originally forms of fairness)

Artificial Intelligence 175 (2011) 25–48

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Non-Markovian control in the Situation Calculus ☆

Alfredo Gabaldon

*Center for Artificial Intelligence, New University of Lisbon, Lisbon, Portugal*

ARTICLE INFO

*Article history:*
Available online 3 April 2010

*Keywords:*
Reasoning about actions
Situation Calculus

ABSTRACT

In reasoning about actions, it is commonly assumed that t
satisfies the *Markov Property*: the executability conditions and
are fully determined by the present state of the system. Th
Reiter's Basic Action Theories in the Situation Calculus. In
Basic Action Theories by removing the Markov property restr
to directly axiomatize actions whose effects and executability c
past and even alternative, hypothetical situations. We then ge
operator, which is the main computational mechanism used for r
Theories, so that it can be used with non-Markovian theories.



Since the 1960's when John McCarthy's papers (in particular the 1969 paper with Pat Hayes) a
Situation Calculus, researchers have been studying and working on this language for reasoning abou
Situation Calculus, one of John's many great inventions, is the topic of this paper and I am delighted
to make a contribution to a special issue in John's honor.

**1. Introduction**

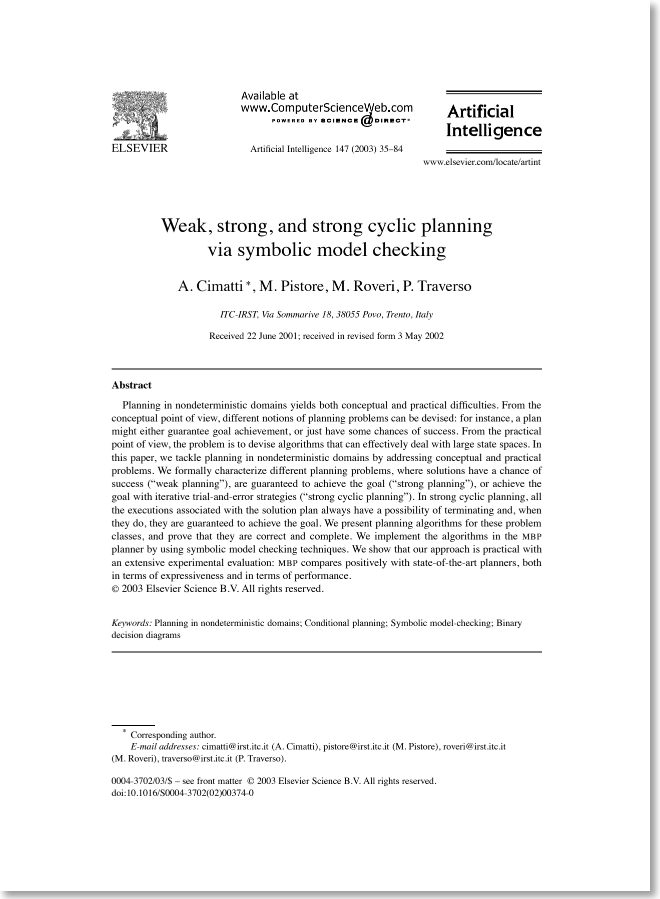
An assumption commonly made in formalisms for reasoning about the effects of actions is the sc
the executability of an action and its effects are entirely determined by the current state or situati
Basic Action Theories [2], a Situation Calculus [3,4] based axiomatization, define the value of a flu
of an action in terms of a formula that can only talk about the situation in which the action v
preconditions of an action are specified by formulas with the same restriction. In this paper we
Theories by removing this restriction. The generalized theories will allow the executability conditio
action to depend not only on what holds when the action is to occur, but also on whether certain c
at different points in the past and even alternative hypothetical evolutions of the system.

As an example, imagine a robot that works in a biological research facility with different safety-le
is such that a material will be considered contaminated after the robot touches it if the robot has b
or has directly been in contact with a hazardous material, and has not been to the disinfection st
effect of touching the material depends on the history of robot activities. We could also imagin
execute the action $open(Entrance, Lab1)$ if $temp(Lab1) > 30$ was ever true since the last time *closed(L*
The latter is an example of an action with non-Markovian preconditions.

In simple scenarios, it is not difficult to extend a theory to preserve the necessary history
variables, especially when the domain is finite. But in complex domains it may not be obvious

☆ A preliminary abstract of this paper appeared in Proc. of AAAI'02 (A. Gabaldon (2002) [1]).
*E-mail address:* ag@di.fct.unl.pt.

0004-3702/$ – see front matter 
doi:10.1016/j.artint.2010.04.012



Available at
www.ComputerScienceWeb.com

Artificial Intelligence 147 (2003) 35–84

Artificial Intelligence

www.elsevier.com/locate/artint

Weak, strong, and strong cyclic planning
via symbolic model checking

A. Cimatti *, M. Pistore, M. Roveri, P. Traverso

*ITC-IRST, Via Sommarive 18, 38055 Povo, Trento, Italy*

Received 22 June 2001; received in revised form 3 May 2002

**Abstract**

Planning in nondeterministic domains yields both conceptual and practical difficulties. From the conceptual point of view, different notions of planning problems can be devised: for instance, a plan might either guarantee goal achievement, or just have some chances of success. From the practical point of view, the problem is to devise algorithms that can effectively deal with large state spaces. In this paper, we tackle planning in nondeterministic domains by addressing conceptual and practical problems. We formally characterize different planning problems, where solutions have a chance of success ("weak planning"), are guaranteed to achieve the goal ("strong planning"), or achieve the goal with iterative trial-and-error strategies ("strong cyclic planning"). In strong cyclic planning, all the executions associated with the solution plan always have a possibility of terminating and, when they do, they are guaranteed to achieve the goal. We present planning algorithms for these problem classes, and prove that they are correct and complete. We implement the algorithms in the MBP planner by using symbolic model checking techniques. We show that our approach is practical with an extensive experimental evaluation: MBP compares positively with state-of-the-art planners, both in terms of expressiveness and in terms of performance.


*Keywords:* Planning in nondeterministic domains; Conditional planning; Symbolic model-checking; Binary decision diagrams

* Corresponding author.
*E-mail addresses:* cimatti@irst.itc.it (A. Cimatti), pistore@irst.itc.it (M. Pistore), roveri@irst.itc.it (M. Roveri), traverso@irst.itc.it (P. Traverso).

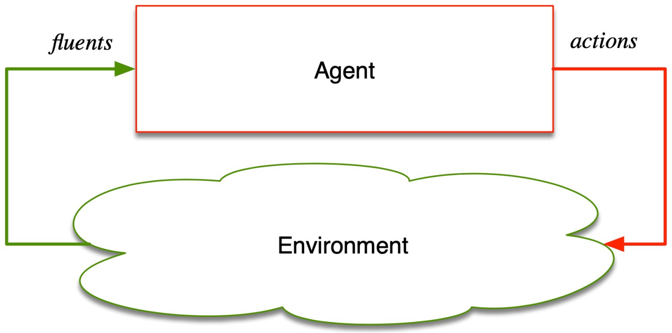
0004-3702/03/$ – see front matter 
doi:10.1016/S0004-3702(02)00374-0

Back to the Basics:

# Agent and Environment Behaviors must be Processes!

- Define **alphabet**
  - actions for the agent
  - fluents for the environment

- Behaviors (aka strategies/policies/protocols/plans) must be **processes** [AbadiLamportWolper89]
  - Functions that chooses the next move on the base of the history so far.



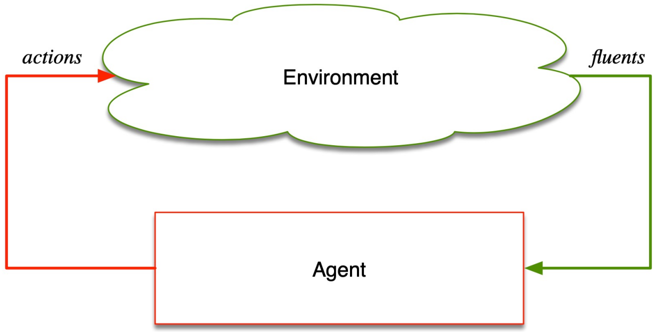

**Agent Behavior**

$$\sigma_a : (fluents)^* \to actions$$

where

- $(fluents)^*$ denotes the **history** of what observed so far by the agent *(a finite sequence of fluents configurations)*
- $actions$ denotes the **next action** that the agent does

*Every program/process has this form! [AbadiLamportWolper89].*




**Environment Behavior**

$$\sigma_e : (actions)^* \to fluents$$

where

- $(actions)^*$ denotes the **history** of what observed so far by the environment *(a finite sequence of agent actions)*
- $fluents$ denotes the **next effects** that the environment brings about.

*Every program/process has this form! [AbadiLamportWolper89].*

# Domains as Environment Specifications

## Domain

- Planning considers the agent acting in a **(nondeterministic) domain**
- The domain is a **model of how the environment works**
- That is, it is a **specification of the possible environment behaviors**


$$[\![Dom]\!] = \{\sigma_e | \sigma_e \text{ compliant with } Dom\}$$

*The presence of domain is a crucial point of planning since the beginning!*

## Planning in nondeterministic domains

Given an $\text{LTL}_f$ task $Goal$ for the agent, and a domain $Dom$ modeling the environment

$$\text{Find agent behavior } \sigma_a \text{ such that } \forall \sigma_e \in [\![Dom]\!].trace(\sigma_a, \sigma_e) \models Goal$$

# General LTL Properties as Environment Specifications

We can we use **LTL/LTLf specify the environment**, through the notion of **realizability**

## Environment specifications in LTL

Let $Env$ be an $\text{LTL}/\text{LTL}_f$ formula over action and fluents.

$$[\![Env]\!] = \{\sigma_e | \forall \sigma_a . trace(\sigma_a, \sigma_e) \models Env\}$$

i.e $Env$ denotes all environment behaviors that play according to the specification whatever is the agent behavior.

## Synthesis with environment model in $\text{LTL}/\text{LTL}_f$

Given an $\text{LTL}/\text{LTL}_f$ task $Task$ for the agent, and an $\text{LTL}/\text{LTL}_f$ environment specification $Env$:

Find agent behavior $\sigma_a$ such that $\forall \sigma_e \in [\![Env]\!] . trace(\sigma_a, \sigma_e) \models Task$

# General LTL Properties as Environment Specifications

But **not every LTL/LTLf formula** can be used to specify the environment, it needs to be **"consistent"**

## Consistent environment specifications

Is any $\mathrm{LTL}/\mathrm{LTL}_f$ formula a valid environment specification? No, $Env$ needs to be "consistent"!:

$$[\![Env]\!] \neq \emptyset \qquad \text{i.e. } \exists\sigma_e.\forall\sigma_a.trace(\sigma_a, \sigma_e) \models Env$$

For example "eventually agent does action $dec$"

$$\textit{eventually } dec$$

is not a valid specification of the environment, since the agent might decide not to do $dec$.

# General LTL Properties as Environment Specifications

## Solve synthesis

To find **agent strategy realizing** $Task$ **under the environment specification** $Env$, we can use standard LTL/LTL$_f$ synthesis for

$$Env \rightarrow Task$$

## Theorem ([AminofDeGiacomoMuranoRubinICAPS2019])

*Let $Task$ be a agent task and $Env$ be a consistent LTL/LTL$_f$ environment specification. Then*

1. *There **exists** agent strategy realizing $Task$ in $Env$ iff there **exists** an agent strategy realizing $Env \rightarrow Task$, i.e.,*

$$\exists \sigma_a . \forall \sigma_e \in [\![Env]\!] . trace(\sigma_a, \sigma_e) \models Task \;\; iff \;\; \exists \sigma_a . \forall \sigma_e . trace(\sigma_a, \sigma_e) \models Env \rightarrow Task$$

2. ***Every** agent strategy realizing $Env \rightarrow Task$ is a agent strategy realizing $Task$ in $Env$, i.e.,*

*for all $\sigma_a$ we have:* $\forall \sigma_e . trace(\sigma_a, \sigma_e) \models Env \rightarrow Task$ *implies* $\forall \sigma_e \in [\![Env]\!] . trace(\sigma_a, \sigma_e) \models Task$

*but not viceversa!*

## Theorem

*Solving LTL/LTL$_f$ synthesis under environment specification is 2EXPTIME-complete.*

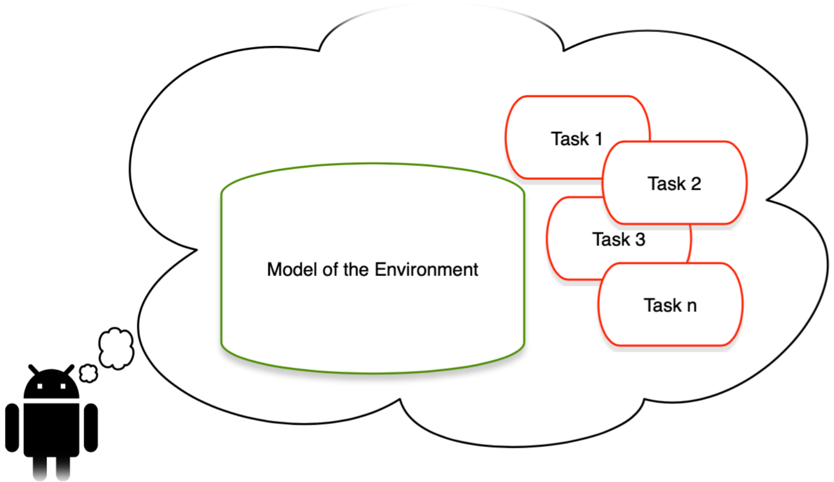
# Results on Synthesis under Environment Specifications

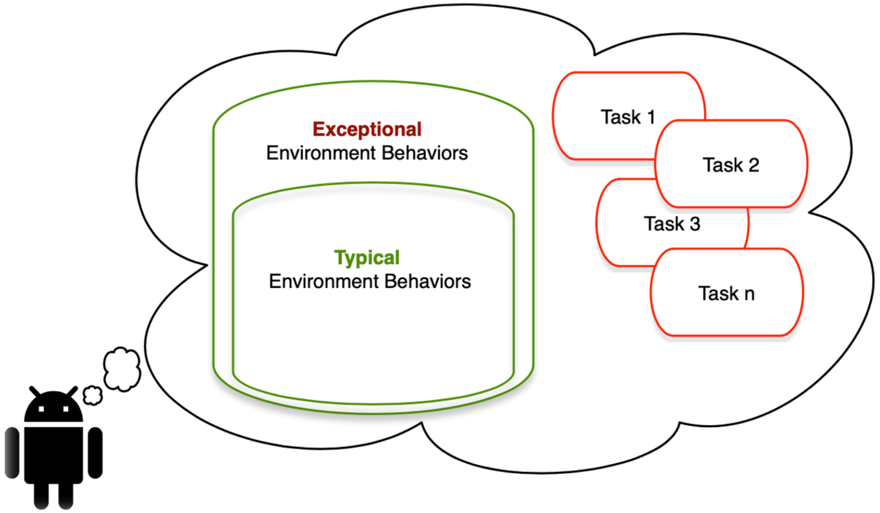
- FOND planning for LTLf tasks
  - Strong: these are simple Markovian Safety properties [DeGiacomoRubinIJCAI2018]
  - Stochastic fairness: as FOND strong cyclic planning, but on an arena that is obtained from domain D and Task [DeGiacomoRubinIJCAI2018], [Aminof et al. ICAPS 2020]

- Env: Safe, coSafe, GR(1), Live
  - Env = Safe: Safe implies Task iff not Safe or Task. But not Safe is LTLf so this is LTLf synthesis
  - Env = Simple Fairness and Stability: Use task to generate arena, then play for single nested fixpoint [Zhu et al. AAAI2020]
  - Env = Safe & coSafe: reduction to deterministic Buchi automata [Camachio et al 2018], use Safe, coSafe and Task to generate arena, then play for single nested fixpoint [De Giacomo et al. KR2020]
  - Env = Safe & GR(1): reduction to GR(1), use Task and Safe to generate arena, then play GR(1) game (double nested fixpoint) [De Giacomo et al. IJCAI2021]
  - Env = Live & Safe: reduction to Live implies LTLf, solvable by LTL synthesis, needed for (hopefully small) Live [De Giacomo et al. KR 2020]

- Env = Live & Safe + agent MUST stop!
  - Agent stops env irrelevant: drop Live, and solve Safe implies Task (LTLf synthesis) [De Giacomo et al KR2021]
  - When agent stops env can continue to evolve: the agent cannot act anymore, though some AgtSafe must be maintained! Find by model checking "agent safe states" where AgtStafe can be maintained without doing anything, then solve Safe implies Task& "at agent safe states" (LTLf synthesis) [De Giacomo et al KR2021]

# Multiple Models of the Environment

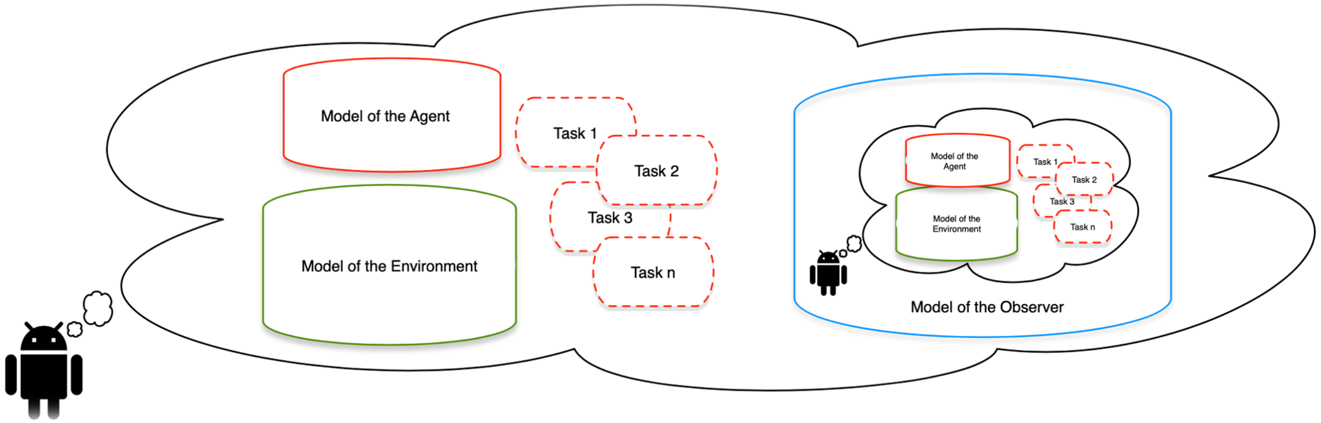



- Model of the **environment (i.e., agent's world)** does **not** need to be **monolitic**,
- Have **multiple models** emphasizing **different facets of the environment** itself.
- For example, separate
  - **Typical** environment behaviors, from
  - **Exceptional** environment behaviors
- **Plan/synthesize** over them with **different guarantees** (e.g., strict **fulfilment** vs. **best effort**)

[Aminof et al. IJCAI2020], [CiolekD'IppolitoPozancoSardinaICAPS2020], [Amonof et al. IJCAI2021], [Amoniof et al. KR2021]

# Model of the Observer




- The model that the agent uses for acting may be:
  - **learned** from data
  - **too detailed** to be narrated
  - expressed in **alien terms**
- As a result may actions of the agent **not be understandable** to the observer (say a human)
- Need for **model of the observer** [ChakrabortiKulkarniSreedharanSmithKambhampatiICAPS2019], [ChakrabortiSreedharanKambhampatiIJCAI2020]
- **Use model of the observer when determining what to do**, so that the agent behavior is **understandable** or can be **explained** to the observer

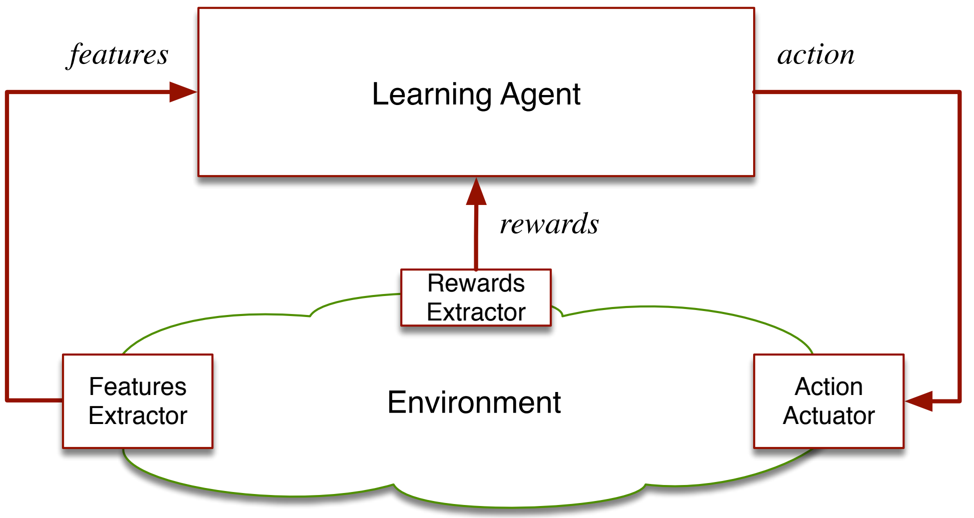
Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

# MERGING REASONING AND LEARNING

# Learning Agents and Reasoning Agents

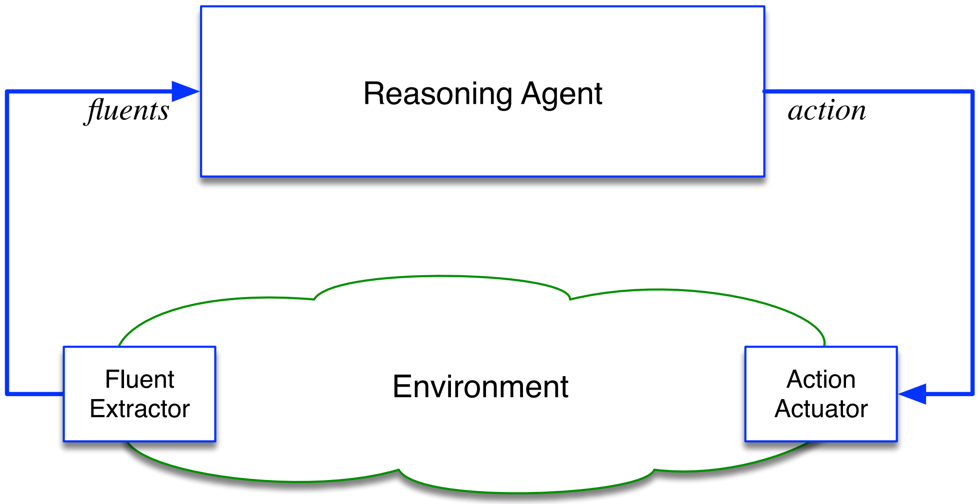
## Learning agent:

- Senses and acts on the environment
- Gets rewards when right
- Does reinforcement learning




## Reasoning agent:

- Senses and acts on the environment
- Has models of its environment and tasks
- Does reasoning and planning

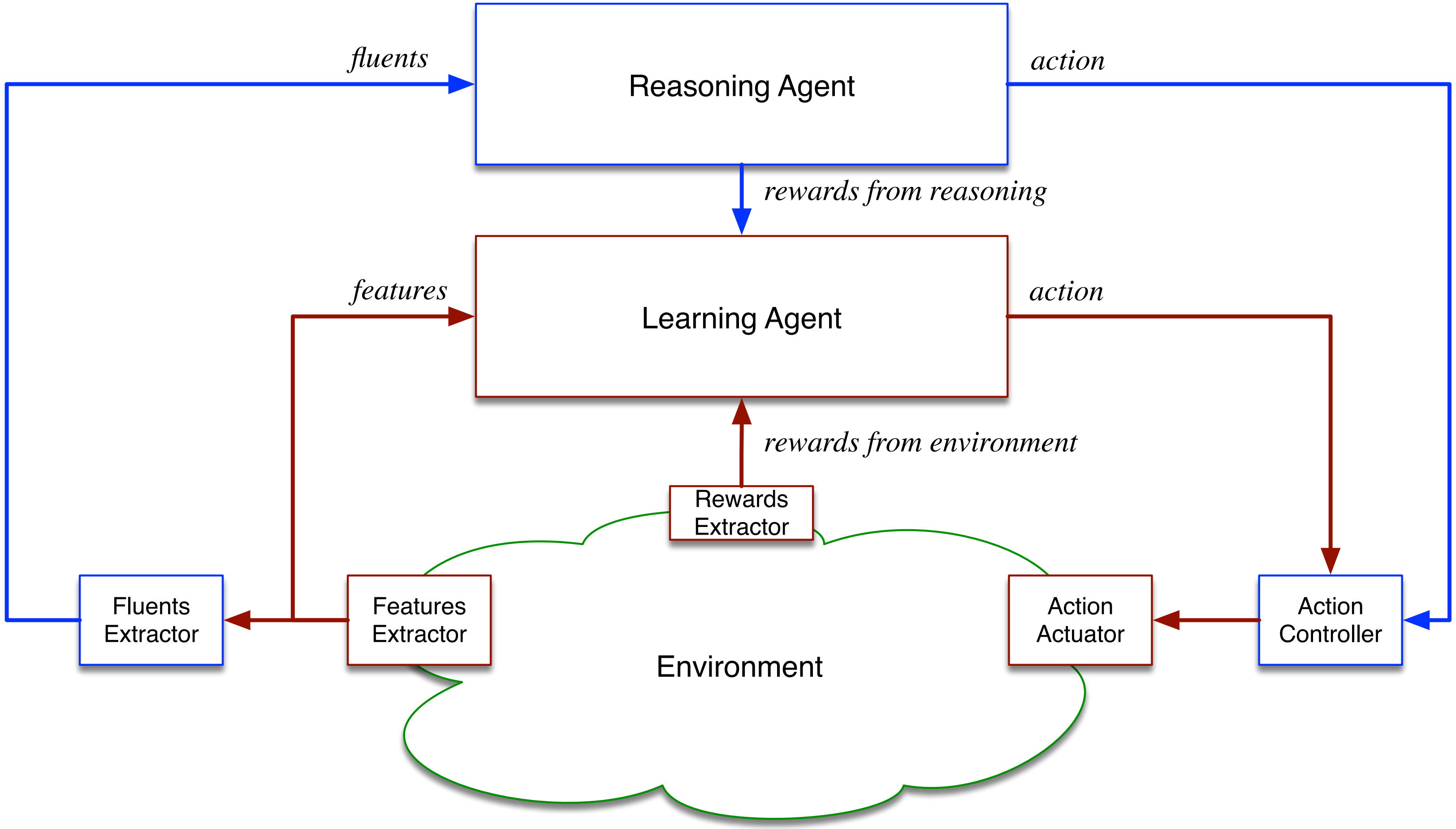


# Merging Learning and Reasoning

Merging:

- Learning agent
  - Does reinforcement learning
  - Possibly deep reinforcement learning

- Reasoning agent
  - Does reasoning
  - Possibly on temporal specification as in formal methods



# MDPs with Logic-based non-Markovian Rewards

## MDPs with non-Markovian rewards

- **Learning agent:** $\mathcal{M} = (S_{ag}, A_{ag}, Tr_{ag}, \cancel{R_{ag}})$

  MDP without rewards

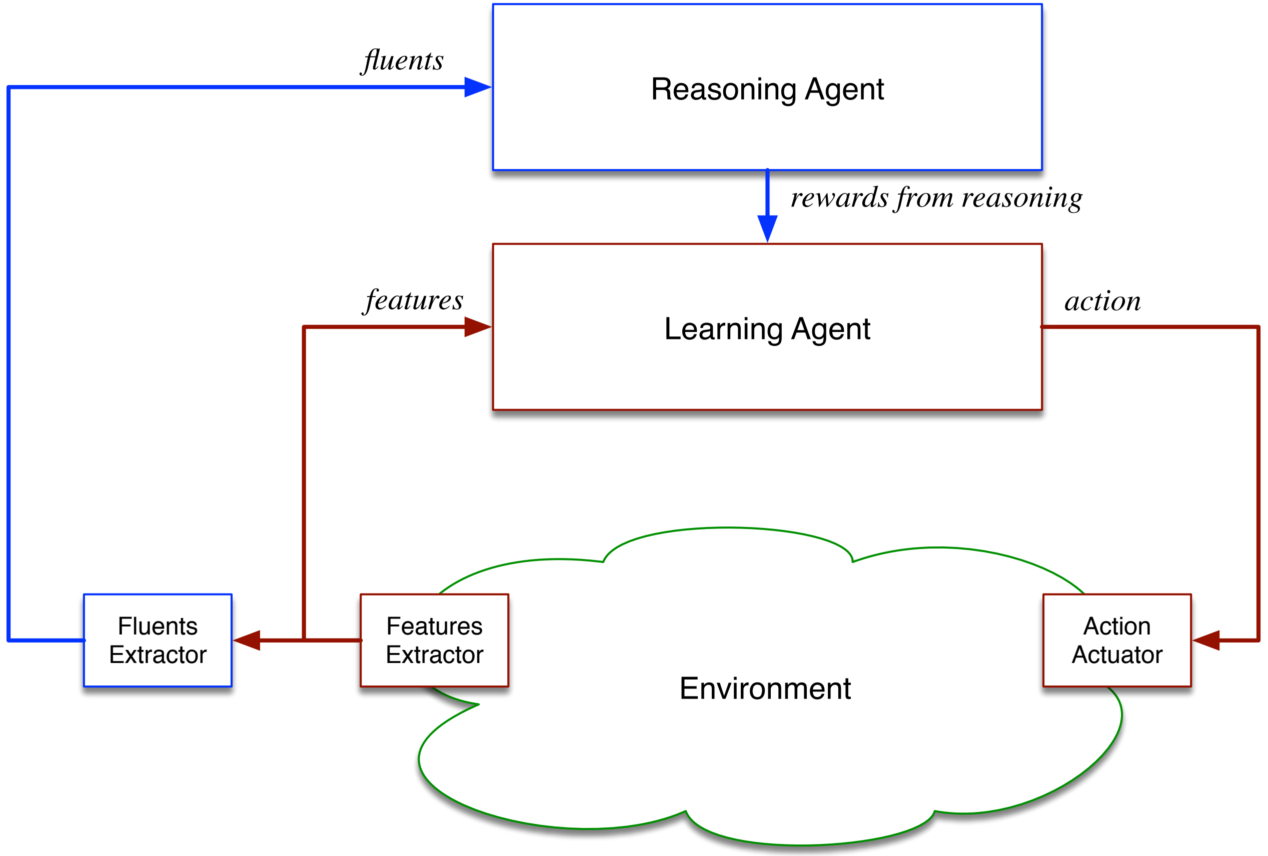
- **Reasoning agent:** $\mathcal{R} = (\mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^m)$

  $\varphi_i$ in LTLf/LDLf ⇒ $\overline{R}_{ag} : (S_{ag}, A_{ag})^* \to \mathbb{R}$

  non-Markovian rewards!

- Mapping between $S_{ag}$ and $\mathcal{L}$

We can define equivalent MDP over an extended state space and do standard RL





*M. Littman. Programming agents via rewards. (Invited talk) IJCAI 2015.*

*R. Brafman, G. De Giacomo, F. Patrizi. LTLf /LDLf non-Markovian rewards. AAAI 2018.*

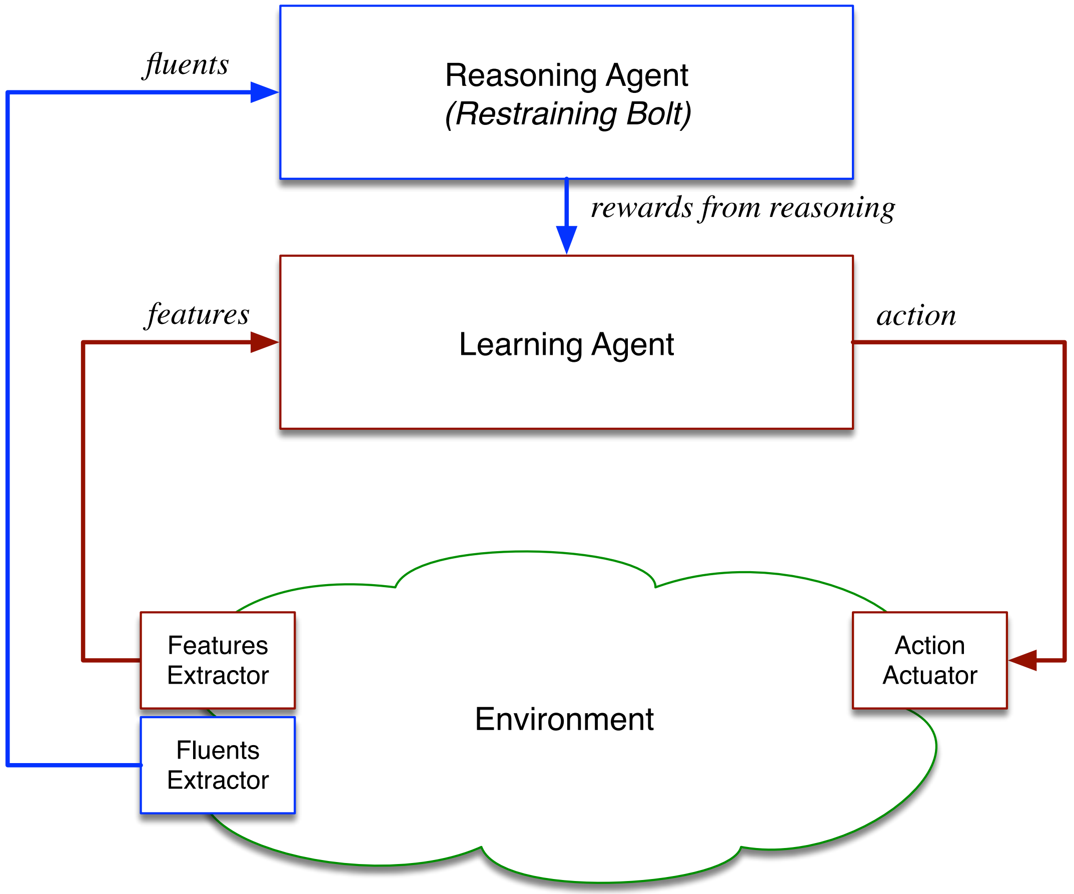
*A. Camacho, R. Icarte, T. Klassen, R. Valenzano, S. McIlraith. LTL and Beyond: Formal Languages for Reward Spec. in RL. IJCAI 2019.*

# Restraining Bolts as Reasoning Agents

## Double state representation (restraining bolts)

- Learning agent: $\mathcal{M} = (S_{ag}, A_{ag}, Tr_{ag}, \cancel{R_{ag}})$
  MDP without rewards
- Reasoning agent: $\mathcal{R} = (\mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^m)$
  $\varphi_i$ in LTLf/LDLf ⇒ $\overline{R}_{ag} : (S_{ag}, A_{ag})^* \rightarrow \mathbb{R}$
  non-Markovian rewards!
- ~~Mapping between $S_{ag}$ and $\mathcal{L}$~~

We can define equivalent MDP over an extended state space and do standard RL




*G. De Giacomo, M. Favorito, L. Iocchi, and F. Patrizi. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. ICAPS 2019.*

# Restraining Bolts

*https://www.starwars.com/databank/restraining-bolt*

## RESTRAINING BOLT

A restraining bolt is a small cylindrical device that restricts a droid's actions when connected to its systems. Droid owners install restraining bolts to limit actions to a set of desired behaviors.

Two distinct representations of the environment

One for the agent
- by the designer of the agent

One for the restraining bolt
- by the authority imposing it

# Restraining Bolts as Reasoning Agents

We can define equivalent MDP over an extended state space and do standard reinforcement learning

RL with $\mathrm{LTL}_f/\mathrm{LDL}_f$ restraining specifications for learning agent $M = \langle S, A, Tr_{ag}, R_{ag}\rangle$ and restraining bolt $RB = \langle \mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^m\rangle$

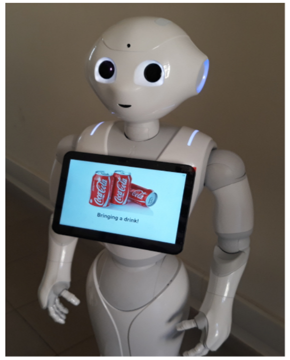
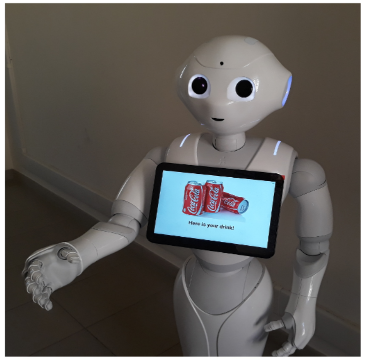
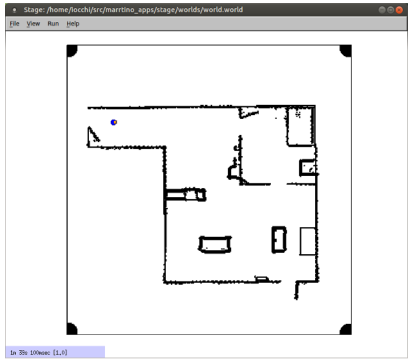
- **Transform each $\varphi_i$ into** DFA $\mathcal{A}_{\varphi_i} = \langle 2^{\mathcal{L}}, Q_i, q_{io}, \delta_i, F_i\rangle$ over fluents evaluations $\mathcal{L}$ with states $Q_i$ and final states $F_i \subseteq Q_i$.
- **Do classical RL over a new MDP** $M' = \langle Q_1 \times \cdots \times Q_m \times S, A, Tr'_{ag}, R'_{ag}\rangle$
- **Thm: the optimal policy $\rho'_{ag}$ learned for $M'$ is an optimal policy of the original problem.**[a]

[a] Crux of the result: the reward function depends on features and automata states, not on fluents

$$R'_{ag}(q_1, \ldots, q_m, s, a, q'_1, \ldots, q'_m, s') = \sum_{i: q'_i \in F_i} r_i + R_{ag}(s, a, s')$$

*G. De Giacomo, M. Favorito, L. Iocchi, and F. Patrizi. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. ICAPS 2019.*

# Example: Cocktail Party

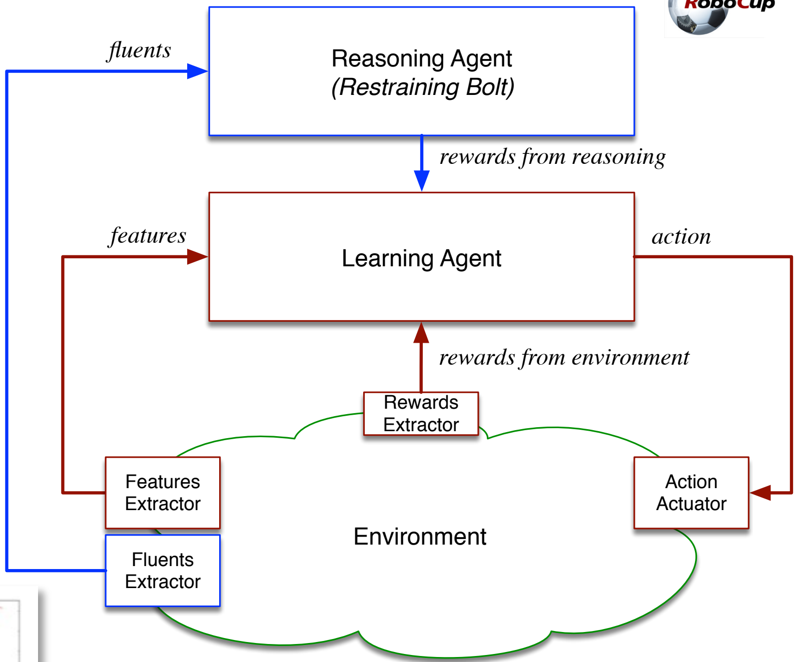


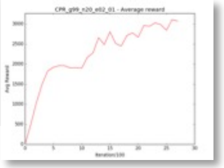
## Learning Agent

- Features: robot's pose, location of objects (drinks and snacks), and location of people
- Actions: move in the environment, can grasp and deliver items to people
- Rewards: robot's navigation, deliver task is completed.

## Restraining Bolt (Reasoning Agent)

- Rewards: serve exactly one drink and one snack to every person, and do not serve alcoholic drinks to minors
- Fluents: identity and age of people, and received items *(uses Microsoft Cognitive Services Face API to provide information)*

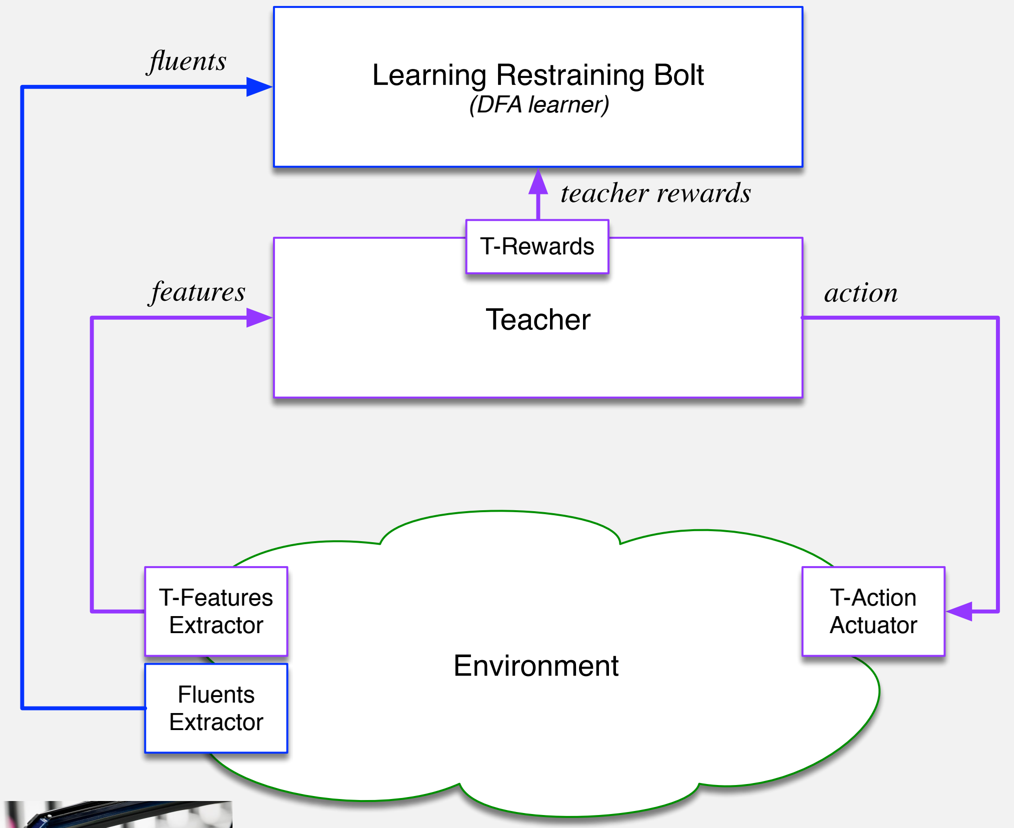


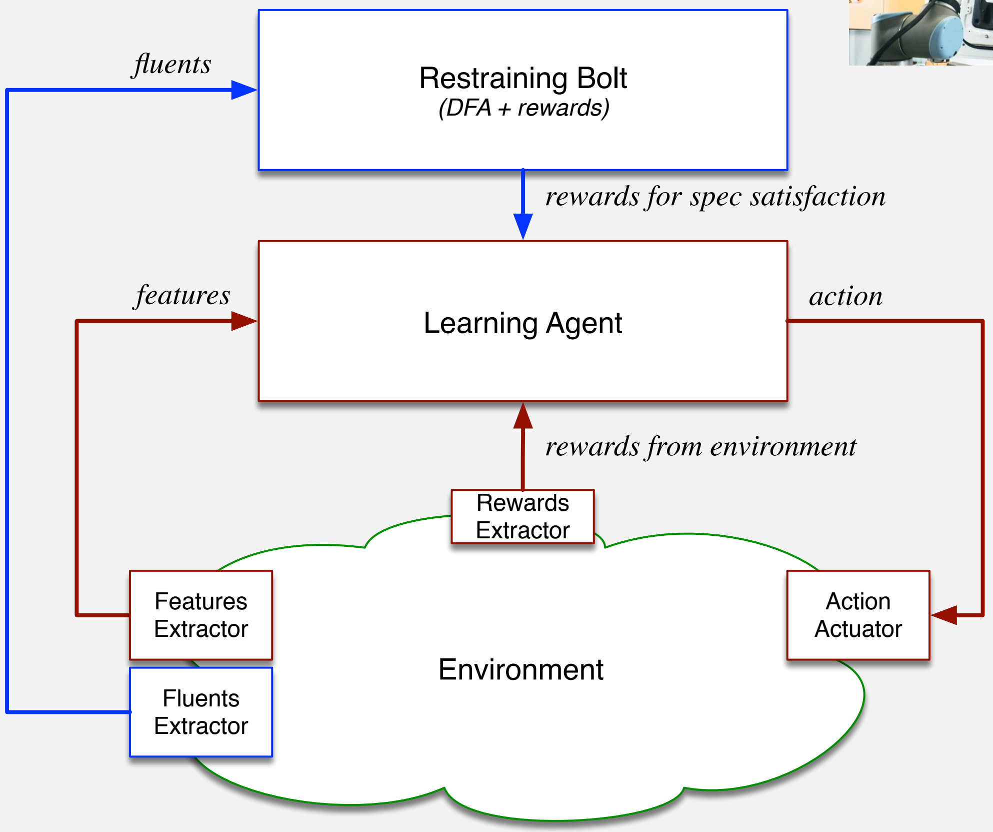
*https://sites.google.com/diag.uniroma1.it/restraining-bolt*

# Extensions: Imitation Learning

## Learn a Restraining Bolt



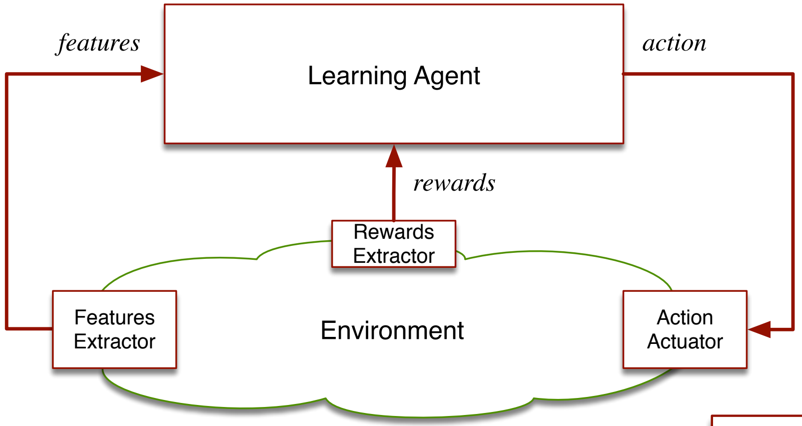

## Use the Restraining Bolt

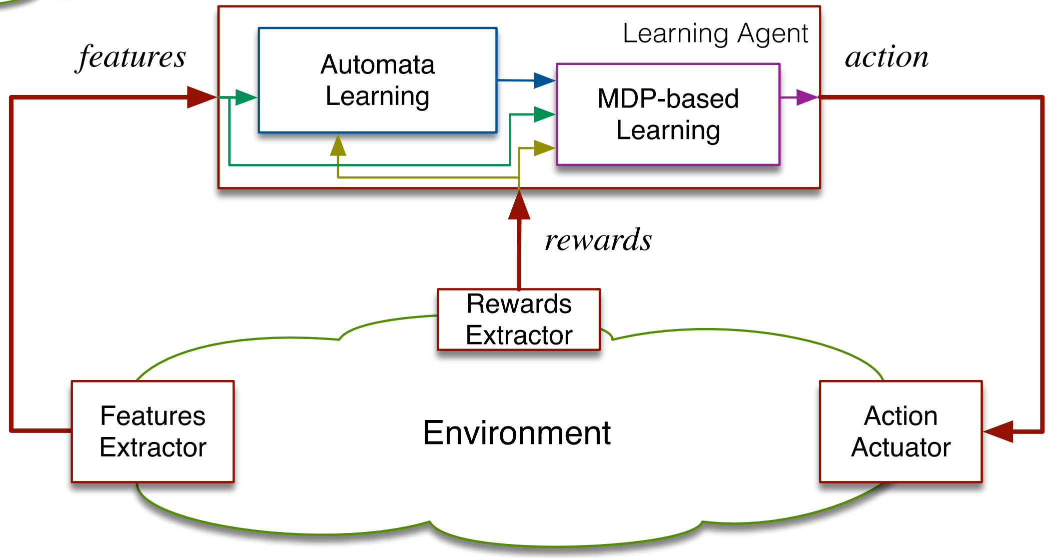



G. De Giacomo, M. Favorito, L. Iocchi, and F. Patrizi.
Imitation Learning over Heterogeneous Agents with Restraining Bolts. ICAPS 2020.
*https://whitemech.github.io/Imitation-Learning-over-Heterogeneous-Agents-with-Restraining-Bolts*

# Reinforcement Learning in non-Markovian Domains

- Reinforcement Learning is typically based on MDPs, i.e. on state-based domains
- Can we do handle non-Markovian dynamics (i.e., depending on the history) without postulating a priori existence of hidden variable, as in  POMDPs?
- Use Regular Decision Processes (RDP) instead of MDPs
- Reinforcement Learning on RDPs requires simultaneously learning an automaton for the dynamics and an optimal policy wrt rewards:
  - Polynomial PAC-learnability
  - With no prior knowledge




*R. Brafman, G. De Giacomo. Regular Decision Processes: A Model for Non-Markovian Domains. IJCAI 2019.*

*A. Ronca, G. De Giacomo. Efficient PAC Reinforcement Learning in Regular Decision Processes. IJCAI 2021*

Reactive Synthesis, Planning and Reinforcement Learning in Linear Temporal Logic on Finite Traces

# CONCLUSIONS

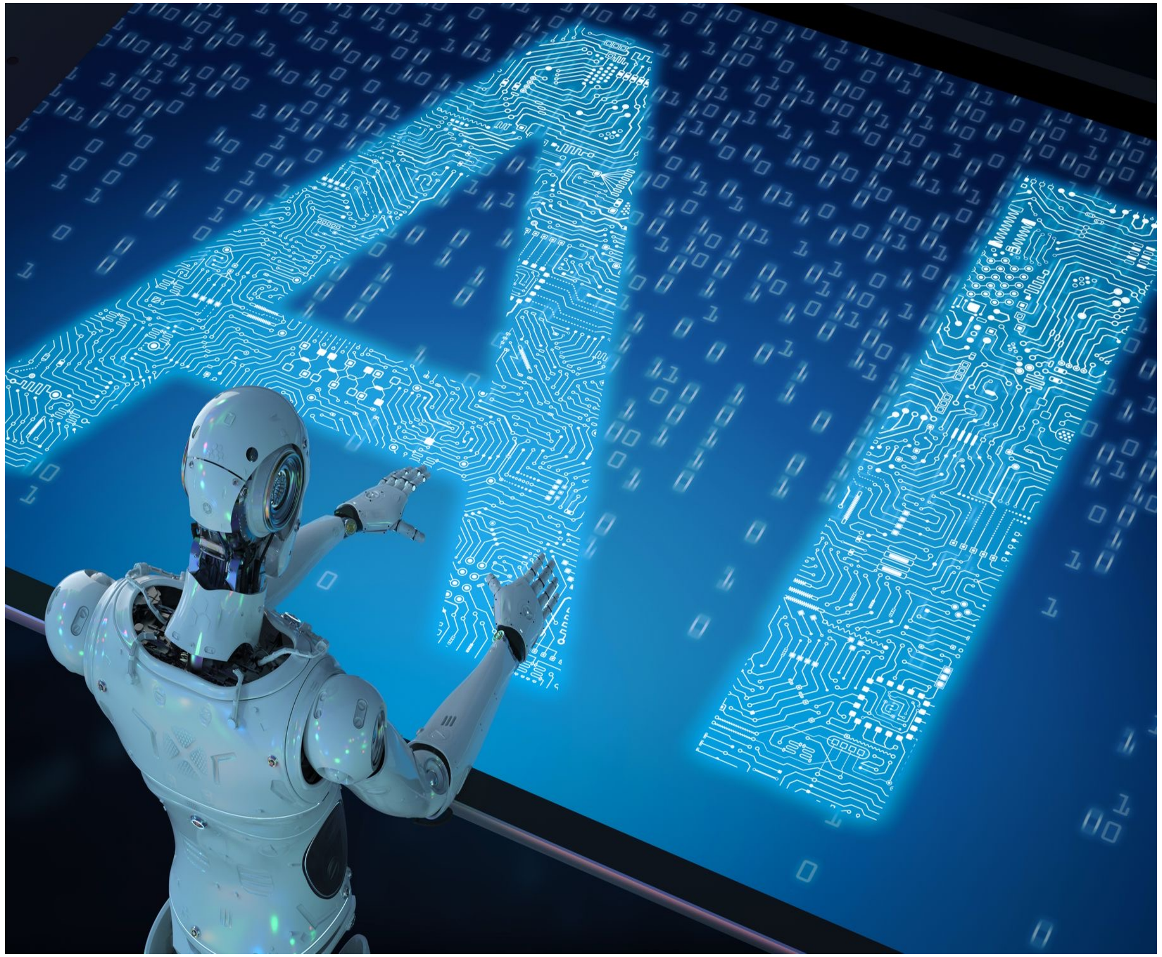
# Conclusions

- Autonomy is one of the grand objectives of AI

- Important advancements from synergies among different areas of AI and CS:
  - Knowledge representation and reasoning
  - Planning
  - Multi-agent systems
  - Sequential decision making (MDPs)
  - Reinforcement learning
  - Formal methods

- Merging reasoning and learning is one of the most important challenges for autonomy in AI. Encouraging results are available

- One more thing. Goal Formation (where do the goal come from?) related to Goal Reasoning: obedient agents, rebellious agents, and agents that change mind through interaction.



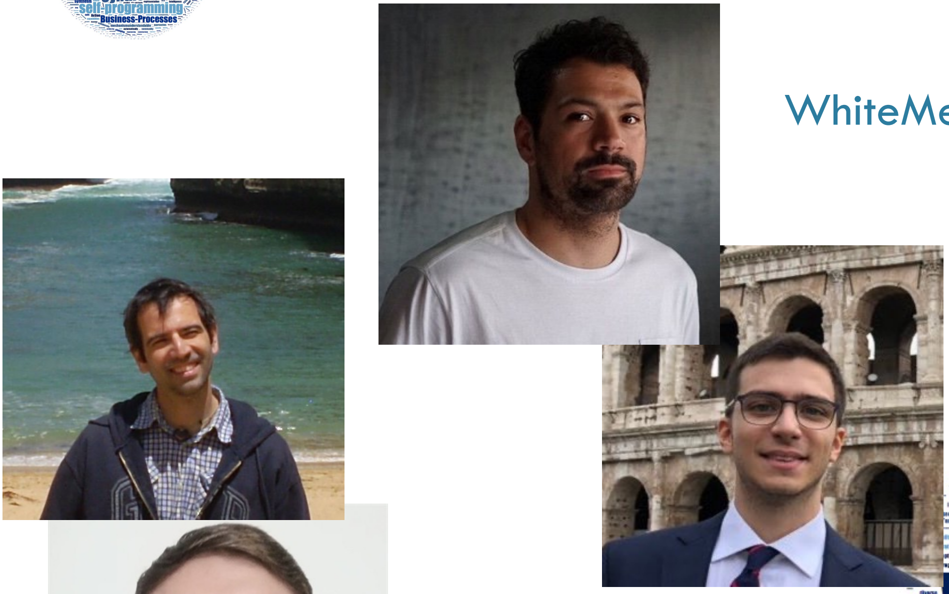

G. De Giacomo, Y. Lesperance. Goal Formation through Interaction in the Situation Calculus: A Formal Account Grounded in Behavioural Science. AAMAS 2020. *(talk available on underline.io)*

# WhiteMech Group

WhiteMech: Whitebox Self Programming Mechanisms

ERC Advanced Grant

WE ARE HIRING!




# References (WhiteMech)


- Synthesizing Best-effort Strategies under Multiple Environment Specifications, Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, Sasha Rubin. *KR 2021*.
- Synthesis with Mandatory Stop Actions, Giuseppe De Giacomo, Antonio Di Stasio, Giuseppe Perelli, Shufang Zhu. *KR 2021*.
- Finite-Trace and Generalized-Reactivity Specifications in Temporal Synthesis, Giuseppe De Giacomo, Antonio Di Stasio Lucas M. Tabajara, Moshe Vardi, Shufang Zhu. *IJCAI 2021*.
- Efficient PAC Reinforcement Learning in Regular Decision Processes, Alessandro Ronca, Giuseppe De Giacomo. *IJCAI 2021*.
- Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up, Benjamin Aminof, Giuseppe De Giacomo, Sasha Rubin. *IJCAI 2021*.
- Compositional Approach to Translate LTLf/LDLf into Deterministic Finite Automata, Giuseppe De Giacomo, Marco Favorito. *ICAPS 2021*.
- Two-Stage Technique for LTLf Synthesis Under LTL Assumptions, Giuseppe De Giacomo, Antonio Di Stasio, Moshe Y. Vardi, Shufang Zhu. *KR 2020*.
- Temporal Logic Monitoring Rewards via Transducers , Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi, Alessandro Ronca. *KR 2020*.
- Synthesizing strategies under expected and exceptional environment behaviors, Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, Sasha Rubin. *IJCAI 2020*.
- Pure-Past Linear Temporal and Dynamic Logic on Finite Traces, Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, Sasha Rubin. *IJCAI 2020 (Survey Track)*.
- Stochastic Fairness and Language-Theoretic Fairness in Planning in Nondeterministic Domains, Benjamin Aminof, Giuseppe De Giacomo, Sasha Rubin. *ICAPS 2020*.
- Imitation Learning over Heterogeneous Agents with Restraining Bolts, Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi. *ICAPS 2020*.
- Goal Formation through Interaction in the Situation Calculus: A Formal Account Grounded in Behavioral Science, Giuseppe De Giacomo, Yves Lesperance. *AAMAS 2020*.
- LTLf Synthesis with Fairness and Stability Assumptions, Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, Moshe Y. Vardi. *AAAI 2020*.
- Planning for LTLf/LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains, Ronen Brafman, Giuseppe De Giacomo. *IJCAI 2019*.
- Regular Decision Processes: A Model for Non-Markovian Domains, Ronen Brafman, Giuseppe De Giacomo. *IJCAI 2019*.
- Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications, Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi. *ICAPS 2019*.
- Planning under LTL Environment Specifications, Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, Sasha Rubin. *ICAPS 2019*.
- Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals, Giuseppe De Giacomo, Sasha Rubin. *IJCAI 2018*.
- LTLf/LDLf Non-Markovian Rewards, Ronen Brafman, Giuseppe De Giacomo, Fabio Patrizi. *AAAI 2018*.
- LTLf and LDLf Synthesis Under Partial Observability, Giuseppe De Giacomo, Moshe Vardi. *IJCAI 2016*.
- Synthesis for LTL and LDL on finite traces, Giuseppe De Giacomo, Moshe Vardi. *IJCAI 2015*.
- Linear Temporal Logic and Linear Dynamic Logic on Finite Traces, Giuseppe De Giacomo, Moshe Vardi. *IJCAI 2013*.

# References (Others)


- Shengping Xiao, Jianwen Li, Shufang Zhu, Yingying Shi, Geguang Pu, Moshe Y. Vardi: On-the-fly Synthesis for LTL over Finite Traces. AAAI 2021: 6530-6537
- Shufang Zhu, Lucas M. Tabajara, Geguang Pu, Moshe Y. Vardi: On the Power of Automata Minimization in Temporal Synthesis. CoRR abs/2008.06790(2020)
- Lucas M. Tabajara, Moshe Y. Vardi: LTLf Synthesis under Partial Observability: From Theory to Practice. GandALF 2020: 1-17
- Suguman Bansal, Yong Li, Lucas M. Tabajara, Moshe Y. Vardi: Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications. AAAI 2020: 9766-9774
- Daniel Alfredo Ciolek, Nicolás D'Ippolito, Alberto Pozanco, Sebastian Sardina: Multi-Tier Automated Planning for Adaptive Behavior. ICAPS 2020: 66-74
- Tathagata Chakraborti, Sarath Sreedharan, Subbarao Kambhampati: The Emerging Landscape of Explainable Automated Planning & Decision Making. IJCAI 2020: 4803-4811
- Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, Sheila A. McIlraith: LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. IJCAI 2019: 6065-6073
- Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E. Smith, Subbarao Kambhampati: Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior. ICAPS 2019: 86-96
- Natasha Alechina, Tomás Brázdil, Giuseppe De Giacomo, Paolo Felli, Brian Logan, Moshe Y. Vardi: Unbounded Orchestrations of Transducers for Manufacturing. AAAI 2019: 2646-2653
- Natasha Alechina, Brian Logan, Mehdi Dastani: Modeling Norm Specification and Verification in Multiagent Systems. FLAP 5(2): 457-490 (2018)
- Alberto Camacho, Meghyn Bienvenu, Sheila A. McIlraith: Finite LTL Synthesis with Environment Assumptions and Quality Measures. KR 2018: 454-463
- Giuseppe De Giacomo, Moshe Y. Vardi, Paolo Felli, Natasha Alechina, Brian Logan: Synthesis of Orchestrations of Transducers for Manufacturing. AAAI 2018: 6161-6168
- Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, Moshe Y. Vardi: Symbolic LTLf Synthesis. IJCAI 2017: 1362-1369
- Alfredo Gabaldon: Non-Markovian control in the Situation Calculus. Artif. Intell. 175(1): 25-48 (2011)
- Jorge A. Baier, Christian Fritz, Sheila A. McIlraith: Exploiting Procedural Domain Control Knowledge in State-of-the-Art Planners. ICAPS 2007: 26-33
- Jorge A. Baier, Sheila A. McIlraith: Planning with First-Order Temporally Extended Goals using Heuristic Search. AAAI 2006: 788-795
- Michael Fisher, Michael J. Wooldridge: Temporal Reasoning in Agent-Based Systems. Handbook of Temporal Reasoning in Artificial Intelligence 2005: 469-495
- Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, Yannis Dimopoulos: Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. Artif. Intell. 173(5-6): 619-668 (2009)
- Alessandro Cimatti, Marco Pistore, Marco Roveri, Paolo Traverso: Weak, strong, and strong cyclic planning via symbolic model checking. Artif. Intell. 147(1-2): 35-84 (2003)
- Diego Calvanese, Giuseppe De Giacomo, Moshe Y. Vardi: Reasoning about Actions and Planning in LTL Action Theories. KR 2002: 593-602
- Ray Reiter: Knowedge in Action, Mit Press 2001
- Giuseppe De Giacomo, Moshe Y. Vardi: Automata-Theoretic Approach to Planning for Temporally Extended Goals. ECP 1999: 226-238
- Alessandro Cimatti, Fausto Giunchiglia, Enrico Giunchiglia, Paolo Traverso: Planning via Model Checking: A Decision Procedure for AR. ECP 1997: 130-142
- Fahiem Bacchus, Froduald Kabanza: Planning for Temporally Extended Goals. AAAI/IAAI, Vol. 2 1996: 1215-1222
- Orna Lichtenstein, Amir Pnueli, Lenore D. Zuck: The Glory of the Past. Logic of Programs 1985: 196-218