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.

Restraining bolts cannot rely on the internals of the agent they control.
The controlled agent is not built to be controlled by the restraining bolt.
Can we build restraining bolts?
YES! By mixing formal specifications and Reinforcement learning!

Soon: AI for Restraining Bolts

Giuseppe De Giacomo (DIAG, Sapienza Univ. Roma)
ERC Advanced Grant
WhiteMech:
White-box Self Programming Mechanisms

Large project (2.5 M€ – 5 years) on KR, generalized planning, synthesis and RL starting in Nov. 2019

Hiring Senior Postdocs, Junior Postdocs and PhD Students in the next 5 years!
Outline

1. White-box Self Programming Mechanisms
2. Need a New Generation of AI-Based Dynamic Systems
3. Starting Point: Reasoning about Actions and Planning
4. Reasoning about Actions and Planning + Reactive Synthesis
5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts
6. Lifting to First Order State Representations
7. Conclusions
Consider the following scenario.\(^1\)

After a long week-end, the human supervisor inspects the manufacturing system and notices that a production line has slowed down significantly, though it is still producing.

She queries the system on what it is doing. The system reveals to her the revised process, which is avoiding the use of the production island 176-176 by repurposing the tools in island 176-671 and sending items there.

She then queries why the system has reprogrammed itself to do so. The system answers by showing that on Sunday 11:43pm the island 176-176 started to produce an unacceptable percentage of defective items, based on tests performed during production.

So, instead of shutting down the production line, the system reprogrammed the process to achieve the specified objectives (quality and throughput) at best, by moving the fabrication of the items to island 176-671 and reconfiguring the tools there to do so.

\(^1\) Liberally adapted from a true story.
In the scenario above we have:

- The smart manufacturing system has **sensing and acting capabilities** (detecting defective items, reconfiguring tools parameters), which possibly use Machine Learning themselves.

- The system has the ability to **monitor and detect faulty executions** and acquire **knowledge** about objects of interest, their properties and their relationships.

- The system has **self-deliberation** abilities that it can use to **re-program by itself** its behavior, i.e., its logic, **without human intervention**.

- The system must be **white-box**, i.e., it can be **queried** about the **circumstances** and **specifications** it reprogrammed itself for. Moreover it can be queried on whether its **self-deliberated way to proceed** meets dynamic properties of interest to the human supervisor (and possibly even facilitate a counterfactual/“what-if” analysis).
“With great power comes great responsibility”:

- Introducing advanced forms of self-deliberation/self-programming calls for being **guarded by human guided specifications and oversight**.

- The resulting behavior must be **white-box**, i.e., **comprehensible in human terms** to make the dynamic systems **trustworthy** [CastelfranchiFalcone10, Neumann17, EUAIplan19].

- Here we concentrate on the notion of **queryability**: in every moment the system can be queried for the status of its specifications, and on whether its behavior meets any dynamic property of interest to the human supervisors.

- **Queryability** is related to explainability, which is per se a research issue [ExplainableAI@IJCAI19], but queryability abstracts away from how to present the answers.

---

*Observe that this kind of transparence, accountability and explainability is strongly advocated by a large part of the AI community [RussellDT15] as well as the CS community [ACMStatement17], and taken up by DARPA within the context of machine learning, through the “Explainable Artificial Intelligence (XAI)”, and by EU through its “Coordinated Plan on Artificial Intelligence” [EUAIplan19]. These issues are also related to Explicability, Predictability, Legibility, etc. [ChakrabortiKulkarniSreedharanSmithKambhampatiICAPS19]*.
White-box Self Programming Mechanisms

Summarizing, several application areas, such as smart manufacturing, IoT, robotic-process automation, process aware information systems, are showing interest in developing a **new generation dynamic systems**, which we may call

**White-box Self Programming Mechanisms**

- We are interested in dynamic systems that rather then being explicitly “programmed”, **reason and deliberate autonomously**, while **learning** from their interaction with their environment and humans there in.

- Crucially, empowering systems with self-deliberation capabilities carries significant risks and therefore we must be able to **“balance power with safety”**. For this reason such dynamic systems need to be (at least) **“queryable”**.

*AI plays a central role in developing such a systems!*
Outline

1. White-box Self Programming Mechanisms

2. Need a New Generation of AI-Based Dynamic Systems

3. Starting Point: Reasoning about Actions and Planning

4. Reasoning about Actions and Planning + Reactive Synthesis

5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts

6. Lifting to First Order State Representations

7. Conclusions
A dynamic system in its general form is a function

\[(\text{input})^* \rightarrow \text{output}\]

where

- \((\text{input})^*\) denotes the history of what observed so far (a finite sequence of inputs)
- \(\text{output}\) denotes the next action that the system does

Every program has this form!
Such function is typically predefined!
An AI-based dynamic system in its general form is a function \((\text{observation})^* \rightarrow \text{action}\) (cf, Leslie Kaelbling’s IJCAI’19 talk)

where

- \((\text{observation})^*\) denotes the history of what observed so far \((\text{a finite sequence of observations})\)
- \text{action} denotes the next action that the system does

**Self-deliberation**: such a function is computed/refined/changed while the system/agent is in execution!

*The agent thinks and acts*
AI-based Self-Deliberating Dynamic Systems

Many areas of AI share this basic notion of dynamic system with forms of first-person self-deliberation:

- Knowledge Representation and Reasoning, in particular Reasoning about Actions [Reiter2001]
- Planning [GhallabNauTraverso2016], [GeffnerBonet2013]
- MDPs/Sequential Decision Making [Puterman1994]
- Reinforcement Learning [SuttonBarto2018],[Bertsekas2019]
- …
**Fluents**: are high-level properties (human comprehensible), and are used to specify

- A model of the world (planning domain/environment specifications)
- A goal (task specification)

**Self-deliberating**: On the base of the model of the world, automatically synthesize agent dynamics/behavior such that all its execution on the environment (which satisfies its specification) brings about the goal.

**Queryable**: the model of the world, the goal and the plan are expressed in a form that is understandable for humans. Though it is an active area of research on how to present it to humans [ExplainableAI@IJCAI19]
AI-based Self-Deliberating Dynamic Systems: Reinforcement Learning

**Reinforcement Learning** *(an example of model-free AI – cf. Adnan Darwiche’s IJCAI’19 talk):*

\[(\text{features})^* \rightarrow \text{action}\]

- **Features**: these are possibly “low-level” properties *(possibly far from human thinking)*, in fact learning the features themself is a trendy area of research in RL *(cf. Zhi-Hua Zhou’s IJCAI’19 talk)*.
- **Rewards**: form a sort of implicit temporally extended task specification *(due to accumulation)*.
- **Self-deliberating**: automatically learn an agent dynamics/behavior “a policy” such that its execution on the environment maximize the expected rewards over time.
- **Not queryable**: if the features are not high-level, e.g. are learned themself from the data. *(But, if features are not high-level, does it really need to?)*
Outline

1. White-box Self Programming Mechanisms
2. Need a New Generation of AI-Based Dynamic Systems
3. Starting Point: Reasoning about Actions and Planning
4. Reasoning about Actions and Planning + Reactive Synthesis
5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts
6. Lifting to First Order State Representations
7. Conclusions
Reasoning about Actions and Planning is a good starting point for addressing White-box Self Programming Mechanisms.

- Based on model of the world in terms of fluents and actions.

- Models form the bases for human-comprehensibility:
  - Fluents and actions describe the domain of interest in a high-level terminology, readily accessible to humans.
  - Models and the plans can be queried through verification/model checking/logical reasoning.
  - Furthermore they provide the basis for performing counterfactual/what-if analysis by checking properties of interest after placing constraints on the system and/or the environment.

- Well developed solvers (there is an entire ICAPS community working on them)
  - Best solvers for reachability, adversarial reachability and "fair adversarial" reachability.
  - Can be adapted also for solving safety.
Reasoning about Actions and Planning

However, **Reasoning about Actions and Planning need to be extended in several directions** for addressing White-box Self Programming Mechanisms. Specifically we need

- **Expressive goal specifications**, which go beyond classical reachability goals, e.g., temporal logic specifications.

- **Expressive model specifications**, which go beyond classical Markovian models: e.g., effects that depend on the past, effect that happen eventually in the future as well as other forms of Non-Markovian.

- **Sound and complete algorithms**, but also **optimal wrt computational complexity** and **practically well-behaved**, to handle these forms of Reasoning about Actions and Planning.

  Synergies with Synthesis in Formal Methods!

- Support **integration of model-based and model-free approaches**, such as Reinforcement Learning.

  Currently, integrating model-based and model-free approaches is a main issue in AI!

- Moreover we need **lift to first-order logical specification** of the state, which is common in reasoning about actions, but virtually absent in planning and in formal methods.

  Synergies Database Theory!
Outline

1. White-box Self Programming Mechanisms
2. Need a New Generation of AI-Based Dynamic Systems
3. Starting Point: Reasoning about Actions and Planning
4. Reasoning about Actions and Planning + Reactive Synthesis
5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts
6. Lifting to First Order State Representations
7. Conclusions
Program Synthesis in Formal Methods

Program Synthesis

**Basic Idea:** “Mechanical translation of human-understandable task specifications to a program that is known to meet the specifications.” [Vardi - The Siren Song of Temporal Synthesis 2018]

**Classical vs. Reactive Synthesis:**
- **Classical:** Synthesize transformational programs [Green1969], [WaldingerLee1969], [Manna and Waldinger1980]
- **Reactive:** Synthesize programs for interactive/reactive ongoing computations (protocols, operating systems, controllers, robots, etc.) [Church1963], [HarelPnueli1985], [AbadiLamportWolper1989], [PnueliRosner1989]

Reactive Synthesis (cf. Hadas Kress-Gazit’s IJCAI’19 talk)

- Reactive synthesis is by now equipped with an **elegant and comprehensive theory** [EhlersLafortuneTripakisVardi2017], [Finkbeiner2018]

- Reactive synthesis is conceptually **related** to **planning in fully observable nondeterministic domains (FOND)** [DeGiacomoVardi2015], [DeGiacomoVardi2016], [DeGiacomoRubin2018], [CamachoTriantafillouMuiseBaierMcIlraith2017], [CamachoMuiseBaierMcIlraith2018], [CamachoBienvenuMcIlraith2019]
Planning and Reactive Synthesis

Planning in Fully Observable Nondeterministic domain

- **fluent**s $F$ (propositions) – controlled by the environment
- **actions** $A$ (actions) – controlled by the agent
- **domain** $D$ – specification of the dynamics
- **goal** $G$ – propositional formula on fluents describing desired state of affairs to be reached

Planning = game between two players

- **arena**: the domain
- **players**: the agent and the environment
- **game**: agent tries to force eventually reaching $G$ no matter how other environment behave
- **Plan** = agent-strategy $(2^F)^* \rightarrow A$ to win the game

Algorithms

EXPTIME-complete. But we have very good algorithms. *(The entire ICAPS community involved!)*

Reactive Synthesis

- **inputs** $X$ (propositions) – controlled by the environment
- **outputs** $Y$ (propositions) – controlled by the agent
- **domain** – not considered
- **goal** $\varphi$ – arbitrary LTL (or other temporal logic specification) on both $X$ and $Y$

Synthesis = game between two players

- **arena**: unconstraint! clique among all possible assignments for $X$ and $Y$
- **players**: the agent and the environment
- **game**: agent tries to force a play that satisfies $\varphi$ no matter how other environment behave.
- **Winning strategy** = agent-strategy $(2^X)^* \rightarrow 2^Y$ to win the game.

Algorithms

2EXPTIME-complete. But we only have non-scalable algorithms. *(In spite of 30 years of research!)*
Focus on finite traces!
Synthesis for general linear time logic (LTL) specifications does not scale.

Solving reactive synthesis

Algorithm for LTL synthesis

Given LTL formula $\varphi$
1: Compute corresponding Buchi Nondeterministic Aut. (NBW) (exponential)
2: Determinize NBW into Deterministic parity Aut. (DPW) (exp in states, poly in priorities)
3: Synthesize winning strategy for parity game (poly in states, exp in priorities)
Return strategy

Reactive synthesis is 2EXPTIME-complete, but more importantly the problems are:
- The determinization in Step 2: no scalable algorithm exists for it yet.
  - From 9-state NBW to 1,059,057-state DRW [AlthoffThomasWallmeier2005]
  - No symbolic algorithms
- Solving parity games requires computing nested fixpoints (possibly exp many)

Focus on finite traces!

Giuseppe: “We should consider synthesis for finite traces specifications.”
Moshe: “But that is easy.”
Focus on finite traces!

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]
LTL$_f$/LDL$_f$: Linear Temporal Logics on Finite Traces [DeGiacomoVardi2013]

**LTL$_f$: linear time temporal logic on finite traces**

Same syntax as standard LTL but interpreted over finite traces

$$\varphi ::= A | \neg \varphi | \varphi_1 \land \varphi_2 | \Diamond \varphi | \Box \varphi | \varphi_1 U \varphi_2$$

Examples:

- $\Diamond A$ 
- $\Box A$ 
- $\Box (A \supset \Diamond B)$ 
- $A \cup B$ 
- $\neg B U A \lor \Box \neg B$

Comments:

- "eventually $A$" 
- "always $A$" 
- "always if $A$ then eventually $B$" 
- "$A$ until $B$" 
- "$A$ before $B$"

**LDL$_f$: linear dynamic logic on finite traces**

Same syntax as PDL but interpreted over finite traces

$$\varphi ::= tt | A | \neg \varphi | \varphi_1 \land \varphi_2 | \langle \rho \rangle \varphi | [\rho] \varphi$$

$$\rho ::= A | \varphi? | \rho_1 + \rho_2 | \rho_1; \rho_2 | \rho^*$$

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

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

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

- “All coffee requests from person $p$ will eventually be served”:
  \[
  \Box (\text{request}_p \supset \Diamond \text{coffee}_p) \quad [\text{true}^*](\text{request}_p \supset (\text{true}^*) \text{coffee}_p)
  \]

- “Every time the robot opens door $d$ it closes it immediately after”:
  \[
  \Box (\text{openDoor}_d \supset \circ \text{closeDoor}_d) \quad [\text{true}^*]([[\text{openDoor}_d] \text{closeDoor}_d])
  \]

- “Before entering restricted area $a$ the robot must have permission for $a$”:
  \[
  \neg \text{inArea}_a \cup \text{getPerm}_a \vee \Box \neg \text{inArea}_a \quad ((\neg \text{inArea}_a)^*) \text{getPerm}_a \vee [\text{true}^*] \neg \text{inArea}_a
  \]

- “Each time the robot enters the restricted area $a$ it must have a new permission for $a$”:
  \[
  ((\neg \text{inArea}_a)^*; \text{getPerm}_a; \neg \text{inArea}_a^*; \text{inArea}_a; \text{inArea}_a^*); \neg \text{inArea}_a^* \rangle \end
  \]

- “At every point, if it is hot then, if the air-conditioning system is off, turn it on, else don’t turn it off”:
  \[
  [\text{true}^*] (\text{if (hot) then}
  \text{if (\neg airOn) then turnOnAir}
  \text{else \neg turnOffAir}) \text{true}
  \]
**LTL$_f$/LDL$_f$ and automata**

**Key point**

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

\[ t \models \varphi \iff t \in L(A_{\varphi}) \]

where \( A_{\varphi} \) is the DFA \( \varphi \) is translated into.

**Example (Automata for some LTL$_f$/LDL$_f$ formulas)**

\[ \Diamond G \]

\[ \Box (A \supset \Diamond \Diamond B) \]

\[ \neg B \cup A \lor \Box \neg B \text{ "A before B"} \]

\[ \langle (\neg B^*; A; \neg B^*; B; B)^*; \neg B^* \rangle \text{ end} \]

"each time new A before B" (A and B not true simultaneously)
Reactive synthesis for $\text{LTL}_f/\text{LDL}_f$ specifications

**Framework:** We partition the set $P$ of fluents into two disjoint sets:
- $X$ controlled by environment
- $Y$ controlled by agent/system

*Can the agent/system set the values of $Y$ in such a way that for all possible values of $X$ a certain $\text{LTL}_f/\text{LDL}_f$ formula remains true?*

**Solution:** compute a strategy $f : (2^X)^* \rightarrow 2^Y$ such that for all generated traces satisfy the formula $\phi$.

**Algorithm for $\text{LDL}_f/\text{LTL}_f$ synthesis**

1. Given $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$
2. Compute corresponding NFA (exponential)
3. Determinize NFA to DFA (exponential)
4. Synthesize winning strategy for DFA game (linear)
5. Return strategy

**Thm:** $\text{LTL}_f/\text{LDL}_f$ synthesis is $2\text{EXPTIME}$-complete.

Same as for infinite traces, but:
- Good algorithms for determinization (one of the best is classical subset construction in textbooks)!
- Solving DFA game requires solving adversarial reachability (only one fixpoint!)
Handling domains

Handling domains in reactive synthesis (cf. Hadas Kress-Gazit’s IJCAI’19 talk)

Let us now consider
- \( Dom \), the (nondeterministic) domain, i.e., the model of the world (expressed as an \( \text{LTL}_f \) formula)
- \( Goal \), the temporally extended goal, i.e., the task specification (expressed again as an \( \text{LTL}_f \) formula)

Then we can do \( \text{LTL}_f \) synthesis for\(^{a}\)

\[ Dom \supset Goal \]

\(^{a}\)Actually there is a lot to be said about this implication, see [AminofDeGiacomoMuranoRubinICAPS2019]!

Can we do better? **Yes!**

Indeed **Reasoning about Actions** and **Planning** advocate to:

“Keep the domain (the model of the world) and goal (task specification) separated!”

The reason is that the nature and the size of the two specifications are very different.

*We should handle domains directly, i.e., we should do **planning for** \( \text{LTL}_f/\text{LDL}_f \) goals instead of synthesis!*
FOND: planning in nondeterministic domains

Nondeterministic domain (including initial state), e.g., [GeffnerBonet2013]

\[ \mathcal{D} = (2^\mathcal{F}, \mathcal{A}, s_0, \delta, \alpha) \] where:

- \( 2^\mathcal{F} \) fluents (atomic propositions)
- \( \mathcal{A} \) actions (atomic symbols)
- \( 2^\mathcal{F} \) set of states, \( s_0 \) initial state (initial assignment to fluents)
- \( \alpha(s) \subseteq \mathcal{A} \) represents action preconditions
- \( \delta(s, a, s') \) transition relation (i.e., nondeterministic) represents nondeterministic action effects.

Key Observation

The nondeterministic domain \( \mathcal{D} \) can be seen as a deterministic automaton \( A_D \).

Nondeterministic domain \( \mathcal{D} \) as DFA \( A_D \)

\[ A_D = (2^\mathcal{F} \cup \mathcal{A}, (2^\mathcal{F} \cup \{s_{init}\}), s_{init}, \varrho, F) \] where:

- \( 2^\mathcal{F} \cup \mathcal{A} \) alphabet (actions \( \mathcal{A} \) include dummy start action)
- \( 2^\mathcal{F} \cup \{s_{init}\} \) set of states, \( s_{init} \) dummy initial state
- \( F = 2^\mathcal{F} \) (all states of the domain are final)
- \( \varrho(s, [a, s']) = s' \) with \( a \in \alpha(s) \), and \( \delta(s, a, s') \)

\( \varrho(s_{init}, [\text{start, s_0}]) = s_0 \)

(abbreviation: \( [a, s'] \) stands for \( \{a\} \cup s' \))
FOND for $\text{LTL}_f/\text{LDL}_f$ goals

The approach extends immediately to any $\text{LTL}_f/\text{LDL}_f$ goal!

Example (Simplified Yale shooting domain + $\text{LTL}_f/\text{LDL}_f$ goal “$\Diamond \Box \neg a$”)

$$A_D \cap A_{\Diamond \Box \neg a} :$$
Algorithm: FOND for LTL\(_f\)/LDL\(_f\) goals

1: Given LTL\(_f\)/LDL\(_f\) domain \(\mathcal{D}\) and goal \(\varphi\)
2: Compute NFA for \(\varphi\) (exponential)
3: Determinize NFA to DFA (exponential)
4: Compute intersection with DFA of \(\mathcal{D}\) (polynomial)
5: Synthesize winning strategy for DFA game (linear)
6: Return strategy

Theorem

\textbf{FOND for LTL\(_f\)/LDL\(_f\) goals is:}

- \textit{EXPTIME-complete in the domain} (assuming a logarithmic representation as in PDDL);
- \textit{2EXPTIME-complete in the goal}.

Note we have \textit{separated cost} in the \textit{model} (the domain) from that in the \textit{task} (the goal)!

(cf. data vs query complexity [ChandraHarel1980], [Vardi1982], [AbiteboulHullVianu1995])
Working with finite traces is being very fruitful!

The possibility of **expressing properties declaratively in logic** ($\text{LTL}_f$/$\text{LDL}_f$), and obtain an equivalent $\text{DFA}$ to be manipulated with graph operations such as (i.e., Cartesian product) is a very powerful tool to get sound, complete, computationally optimal and practically effective algorithms for

- Non-Markovian domains [BrafmanDeGiacomoIJCAI2019a], [BrafmanDeGiacomoIJCAI2019b]

- Adding temporal logic environment assumption to domains [BonetDeGiacomoGeffnerRubin2017], [CamachoBienveniMcIlraith2018], [AminofDeGiacomoMuranoRubin2019]

- MDPs with non-Markovian rewards [BachousBoutilierGrove1996], [ThiebauxGrettonSlaneyPriceKabanza2006], [CamachoChenSannerMcIlraith2017], [BrafmanDeGiacomoPatrizi2018]

- Reinforcement Learning with temporal task specifications [IcarteKlassenValenzanoMcIlraith2018], [CamachoIcarteKlassenValenzanoMcIlraithIJCAI2019], [DeGiacomoFavoritoLocchiPatrizi2019].

*The latter point is related to integrating model-free and model-based approaches!*
Outline

1. White-box Self Programming Mechanisms
2. Need a New Generation of AI-Based Dynamic Systems
3. Starting Point: Reasoning about Actions and Planning
4. Reasoning about Actions and Planning + Reactive Synthesis
5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts
6. Lifting to First Order State Representations
7. Conclusions
A Way to Mix Model-Based and Model-Free AI

Inspired by Michael Littman’s talk at IJCAI 2015

Learning Agent
(model-free)

Environment/World
features action

Features Extractor

Rewards Extractor

rewards from environment

KR-based Monitor
(model-based)

fluents

rewards from monitor

Learning Agent
(model-free)

Environment/World

Features Extractor

Rewards Extractor

Fluents Extractor

Action Actuator

White-box Self Programming Mechanisms
Restraining Bolts

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.

https://www.starwars.com/databank/restraining-bolt
Restraining Bolts

- **Restraining bolts** are used to control/restrict agents’ behavior.
- **Restraining bolts** are placed on different kinds of agents.
- **Restraining bolts** cannot rely on the internals of the agents they control.
- **Agents** are not specifically built to be controlled by the restraining bolt.

*R2-D2 and C-3PO both with restraining bolts*
Restraining Bolts

- **Two distinct representations of the world:**
  - one for the **agent**, by the **designer of the agent**
  - one for the **restraining bolt**, by the **authority imposing the bolt**

- Are these two representations related to each other?
  - **NO:** the **agent designer** and the **authority imposing the bolt** are not aligned
  - **YES:** the agent and the bolt act in the **real world**. *(why should they!)*

- But can restraining bolt exist at all?
  - **YES:** for example based on **Reinforcement Learning**!
Given a learning agent $M = \langle S, A, Tr_{ag}, R_{ag} \rangle$ with features determining the state space $S$, actions $A$, and $Tr_{ag}$ and $R_{ag}$ unknown and

- a restraining bolt $RB = \langle L, \{ (\phi_i, r_i) \}_{i=1}^m \rangle$ formed by $m$ LTLf/LDLf formulas $\phi_i$ over the fluents $L$ with associated rewards $r_i$.

- Learn a non-Markovian policy $\rho : S^* \rightarrow A$ that maximizes the expected cumulative reward.
Example: **BREAKOUT + remove column left to right**

- **Learning Agent**
  - **LA features**: paddle position, ball speed/position
  - **LA actions**: move the paddle
  - **LA rewards**: reward when a brick is hit

- **Restraining Bolt**
  - **RB fluents**: bricks/columns status (broken/not broken)
  - **RB LTLf/LDLf restraining specification**: all the bricks in column \( i \) must be removed before completing any other column \( j > i \) \((l_i\) means: the \( i_{th} \) column of bricks has been removed): 

\[
\langle (\neg l_0 \land \neg l_1 \land \ldots \land \neg l_n)^*; (l_0 \land \neg l_1 \land \ldots \land \neg l_n); (l_0 \land \neg l_1 \land \ldots \land \neg l_n)^*; \ldots; (l_0 \land l_1 \land \ldots \land l_n) \rangle_{tt}
\]
Example: **CocktailParty** Robot + no alcohol to minors

- **Learning Agent**
  - **LA features:** robot’s pose, location of objects (drinks and snacks), and location of people
  - **LA actions:** move in the environment, can grasp and deliver items to people
  - **LA reward:** rewards when a deliver task is completed.

- **Restraining Bolt**
  - **RB fluents:** identity and age of people
    
    *(in practice, tools like Microsoft Cognitive Services Face API can be integrated into the bolt to provide this information.)*
  - **RB LTL$_f$/LDL$_f$ restraining specification:** do not serve alcoholic drinks to minors
Solution [DeGiacomoFavoritilocchiPatrizi2019]

Building blocks:

- **Classic Reinforcement Learning:**
  - An agent interacts with an environment by taking actions so to maximize rewards;
  - No knowledge about the transition model, but assume Markov property (history does not matter): Markov Decision Process (MDP)
  - Solution: **Markovian policy** $\rho : S \rightarrow Act$

- **RL for non-Markovian reward decision process with LTL_f/LDL_f rewards** [BrafmanDeGiacomoPatrizi2018]:
  - Rewards depend from history, not just the last transition;
  - Specify proper behaviours by using LTL_f/LDL_f formulas;
  - Solution: **Non-Markovian policy** $\rho : S^* \rightarrow Act$
  - Reduce the problem to MDP (with extended state space)

---

Transform each $\varphi_i$ into DFA $A_{\varphi_i}$

Do RL over an MDP $MDP'$ with extended state space to have memory needed for non-Markovian rewards ($Q_i$ are the states of $A_{\varphi_i}$):

$$S' = Q_1 \times \cdots \times Q_m \times S$$

Rely on off-the-shelf RL algorithms (Q-Learning, Sarsa, ...)
Connections between KR components and RL components can be tighter!

![Diagram showing connections between KR components and RL components.](image-url)
Connections between KR components and RL components for safety!

The idea of restraining bolt can be subscribed to that part of research generated by the urgency of providing safety guarantees to AI techniques based on learning.


However, the Restraining Bolt must impose its requirements without knowing the internals of controlled agent, which remain a black-box.
Outline

1. White-box Self Programming Mechanisms
2. Need a New Generation of AI-Based Dynamic Systems
3. Starting Point: Reasoning about Actions and Planning
4. Reasoning about Actions and Planning + Reactive Synthesis
5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts
6. Lifting to First Order State Representations
7. Conclusions
Until now we have considered states that are essentially representable propositionally (there are only a finite number of them).

**Practical dynamic systems need a first-order representation of the state**

Actual applications, such as in smart manufacturing, IoT, robotic-process automation, process aware information systems need a rich representation of both

- **data**, which determine the information of interest
- **processes**, which determine how data changes and evolves over time.

Both these aspects are modelled **conceptually** in Software Engineering, as we would do in **Knowledge Representation** in AI.
Example [CAiSE17]

UML class diagram

**Employee**
- name: Sting
- costPerHour: Real

**TemporayEmployee**
- firstContractDate: Date

**Expense**
- acronym: Sting
- url: String
- costPerHour: Real
- date: Date
- description: Sting

- requires 1..1, 0..*
- #hours: Int
- worksOn 0..* 1..*

BPMN diagram

**SelectProject**

**AcquireDates**

**CalcCost**

**CalcRatio**

**ProdReport**

**Continue?**

**User**

Artifact (glue) – see “Artifact-Centric Approach”

[NigamCaswell03], [EU project ACSI, 2010–2013]
Transition systems with first-order-state

These kind of system need a first-order representation of the state and generate their dynamics can be seen as an transition systems with infinite many states.

First-order-state transition systems

A TS over finite fluents \( \mathcal{F} \) and constants \( \mathcal{C} \) is a tuple \( T = (\Delta, Q, q_0, \rightarrow, \mathcal{I}) \):

- \( \Delta \) is the \textbf{(infinite) object domain};
- \( Q \) is the \textbf{(infinite) set of states};
- \( q_0 \in Q \) is the \textbf{initial state};
- \( \rightarrow \subseteq Q \times Q \) is the \textbf{transition relation}; and
- \( \mathcal{I} : Q \mapsto \text{Int}_{\Delta}^{\mathcal{F},\mathcal{C}} \) is the \textbf{state labeling function} associating to each state \( q \) an interpretation \( \mathcal{I}(q) = (\Delta, \cdot \mathcal{I}(q)) \) such that the constants in \( \mathcal{C} \) are interpreted in the same way in all the states over which \( \mathcal{I} \) is defined.

These kind of transitions systems are \textbf{notoriously difficult to analyze}

- \textbf{Model checking} does not work directly
- \textbf{Approximations via abstraction} to finite states transition systems are developed in Formal Methods
- However, first-order representation of the states have been well-studied in \textbf{Knowledge Representation} in Reasoning about Actions, e.g., with the Situation Calculus

A key recent result is the decidability of \textbf{model checking of “bounded-state” dynamic systems}
Bounded First-Order State Action Theories

**Key Idea:** The state can change arbitrarily over time, however you can store information about at most \( n \) (for a fixed \( n \)) object in it [KR12, JAIR12, BelardinelliLomuscioPatriziKR12, PODS13, BelardinelliLomuscioPatriziJAIR14, KR16, AIJ16, IC18].

### Example (The Bookshelf: Prototypical Example of Boundedness)

- Have an agent that is an avid reader. He has a bookshelf of a given size. He acquires books that put them in his bookshelf, reads them, write reviews on them, and the put them back on the bookshelf or gives them away. We assume that the available space in the bookshelf given in units and that each book consumes a certain number of unit. The avid reader cannot acquire a book if there is not enough space in the bookshelf.

- Let assume that the bookshelf has \( n \) units of space and that each books consumes 1 unit. Possible actions:
  - \textit{acquire(book)}. PRE: \textit{book} not in the bookshelf already, space available in the bookshelf. POST: \textit{book} in the bookshelf and one less unit available in the bookshelf.
  - \textit{store(book)}. PRE: \textit{book} in the hand of the avid reader, space available in the bookshelf. POST: \textit{book} in the bookshelf and one less unit available in the bookshelf.

*Originally devised in the context of the Situation Calculus, but in fact completely general.*
Results on State-Boundedness

State-Boundless Key Result

Under the very general condition* state-boundedness leads to the existence of a finite abstract transition system that is bisimilar to the original infinite one.

*(namely “genericity”: if two states are isomorphic they induce the “same” transitions, modulo isomorphism, [AbiteboulHullVianu95].)

- Results:
  - Decidability of model checking for full first-order $\mu$-calculus with unrestricted quantification across
  - Decidability of LTL model checking for formulas without quantification across
  - LTL model checking for formulas with quantification across is undecidable instead: that is in the first-order case, LTL cannot be captured by $\mu$-calculus!

- These results apply to:
  - Standard SitCalc!
  - Data-aware business processes
  - Dynamic systems built over databases
  - Dynamic systems built over write-also ontology-based data access systems

- Automated synthesis also possible, being ultimately based on fixpoints over a game. [DeGiacomoFelliLoganPatriziSardina2019sub].
Outline

1. White-box Self Programming Mechanisms

2. Need a New Generation of AI-Based Dynamic Systems

3. Starting Point: Reasoning about Actions and Planning

4. Reasoning about Actions and Planning + Reactive Synthesis

5. Reasoning about Actions and Reinforcement Learning: Restraining Bolts

6. Lifting to First Order State Representations

7. Conclusions
Conclusions

- **White-box Self Programming Mechanisms** are increasingly advocated by several application areas of pivotal importance in the current socio-economic context, such has Smart Manufacturing, Internet of Things, Business Process Management, etc.

- **Queryability**, i.e., the ability of querying the model and the synthesized behaviors is a crucial property (at the base of trustworthiness, explainability, explicability, transparency, accountability, etc.)

- Important advancements are coming from synergies among different areas of CS:
  - AI (KR, Planning, Semantic Technologies)
  - Formal Methods (Verification, Synthesis, Temporal Logics, Automata Theory)
  - Databases (Conceptual Models, Data Integration)
  - Processes (BPM, Service Composition)

- Several issues remain open, but with good suggestions on how to proceed to tackle them, e.g., efficient handling of \( \text{LTL} \) (on infinite traces) environment assumptions, alternative way of specifying environment assumptions, aspects of partial observability, general form of synthesis in the first order setting, etc.

- Merging model-based and model free (dynamic) systems is one of the most challenging and most important current issue in AI. Some encouraging results are starting to be available.

- One increasingly important issue, not touched in this talk, is **Goal Formation** (where do the goal come from?). It is related to Goal Reasoning [Aha2018] and the notion of obedient vs. rebellious agent [AhaComan2017]. Still wide open.

*Psychology/Behavioral Science can give inspiration!*
ECAI 2020 – June 8-12, 2020
Santiago de Compostela, Spain

• ECAI is the flagship conference in AI in Europe
• Organized by EurAI (the European Association for AI)
• It is biennial and every other edition is together with IJCAI (last IJCAI-ECAI 2018, next IJCAI-ECAI 2022)
• ECAI is not for Europe only! Researchers from all over the world are welcome!
• An extensive program of workshops, tutorials, and invited talks
• Prestigious Applications of Intelligent Systems conference (PAIS)
• Lab To Market
• Job Fair
• Starting AI Researcher Symposium (STAIRS)
• Doctoral Consortium
• Three Minutes Thesis Competition
• Women in AI Breakfast

http://ecai2020.eu
http://ecai2020.eu/sponsorships

Local Chair:
• Senen Barro
• Alberto Bugarin

Program Chair:
• Giuseppe De Giacomo

General Chair:
• Jerome Lang

Area Chairs among the very best AI scientists in the world, including:
• James Kwok
• Fanghzen Lin
• Yang Yu
• Min-Ling Zhang
• Zhi-Hua Zhou