ERC Advanced Grant 2017
Research proposal [Part B2]

White-Box Self-Programming Mechanisms

WhiteMech

– Principal investigator (PI): **Giuseppe De Giacomo**
– Host institution: **Università degli Studi di Roma "La Sapienza"**
– Proposal duration: **60 months**

Contents

a State-of-the-art and objectives **2**
a.1 Generalized Planning . . . 3
a.2 Verification and Synthesis . . . 5
a.3 KR and First-Order State Representation . . . 7
a.4 Data-Awareness . . . 9
a.5 Componentization . . . 9
a.6 Integrating stochastic decision and reinforcement learning . . . 9
a.7 Driving Application Contexts . . . 10
a.7.1 Smart Manufacturing . . . 10
a.7.2 Internet of Things . . . 10
a.7.3 Business Processes . . . 10
a.8 High Risk, High Gain . . . 10

b Methodology **11**
b.1 Research Streams . . . 11
b.2 Workpackages . . . 11
b.3 Phases . . . 12

c Resources **12**
c.1 Budget . . . 12

A Ethics **19**

This part of the proposal should be read in conjunction with Part B1, which is to be considered an integral part of the project description.

a State-of-the-art and objectives

We are witnessing an increasing availability of **mechanisms** that offer some form of programmability. Obvious examples are software in our computers and mobile devices, as well as intelligent machines, such as cognitive robots, self-driving cars, flying drones, etc., which are becoming a reality. In particular, programmable mechanisms are increasingly becoming important in three contexts considered of pivotal importance in today's economy, namely **Manufacturing**, **Internet of Things**, and **Business Process Management**. These three areas will act as specific testbeds of the science and technology developed within WHITEMECH.

There are compelling reasons to introduce **self-programming abilities** in these systems, such as **self-adaptability** to changing users and environment conditions exploiting information gathered at runtime [134], and **automated exception handling** to suitably **recover from unexpected situations** [110].

The current technology, e.g., **autonomic computing**, which has long advocated self-configuration, self-healing, self-optimization, and self-protection, however, is essentially based on **preprogrammed solutions** by IT professionals [91]. That is, although sophisticated languages and methodologies for streamlining the development of adaptation and exception handling recovery procedures are available, IT professionals, software engineers and programmers, are still writing all code by hand in the end. As a result, in spite of the progresses in the organization of the software development process, this traditional way of tackling automated reactions in mechanisms is showing serious limitations. In many application areas, it is simply too **costly** and **error-prone** to delegate to software engineers to list and handle all possibly adaptation tasks that may arise in the mechanism execution. In fact, especially when applications have to handle widely unexpected circumstances, stemming from the interaction with the real world or with humans taking decisions based on unmodeled circumstances, as indeed in Smart Manufacturing, Internet of Things and Business Processes Management, it is considered simply **infeasible** to determine apriori all possible adaptations that may be needed at runtime [110].

In this state of affairs, the fantastic progress in Machine Learning that we have seen in recent years is attracting a lot of attention. **Machine Learning** is considered a powerful tool to avoid **preprogrammed solutions** in favor of **learned solutions**, and there are indeed great and fully justified expectations of what machine learning can bring about in relieving humans from having to preprogram mundane adaptation tasks.

However, in machine learning, we typically don't have an undestanding of how and why a certain solution has been chosen. This **lack of understandability of machine learning solutions** is increasingly becoming a concern in both the AI and CS scientific communities, [128, 1], and has been recently taken up by DARPA, through the DARPA-BAA-16-53 "Explainable Artificial Intelligence (XAI)" program.[1]. For example, the ACM Statement on Algorithmic Transparency and Accountability [1] says: "*There is also growing evidence that some algorithms and analytics can be opaque, making it impossible to determine when their outputs may be biased or erroneous.*"

Hence on the one hand there is a widespread recognition in the AI and CS communities that the objectives of WHITEMECH, that is, building self-programming mechanisms that are white-box, are very important. On the other hand, we are currently stuck between two extremes: either we provide preprogrammed solutions, but this is not cost-effective or even feasible for certain applications, or we use machine learning to produce solutions, but this often means that we give up transparency and accountability, and ultimately human understandability.

Beyond the opaqueness of machine-learning-produced solutions, there are also widespread concerns

[1] http://www.darpa.mil/program/explainable-artificial-intelligence

about the **safety** of using such solutions and whether they are thrustworthy.[2] For any system that operates autonomously, there must be guarantees that suitable safety constraints will always be respected. The relevant safety constraints must be specified in a language that human users understand. The system must be able to explain its decisions so that human users can understand the system's operation and confirm that it does follow the relevant rules of conduct. This is crucial for building users' **trust** in the system.

WHITEMECH aims at addressing the realization of white-box self-programming mechanisms on radically different bases. It will leverage on Reasoning about Action and Planning in AI which provide explicite representation, required for being white-box and process synthesis required for self-programming. However traditional Reasoning about Action and Planning are by far too simple to be used off-the-shelf for realizing white-box self-programming mechanisms. To do so WHITEMECH will take elements from Verification and Synthesis in Formal Methods and Data-aware processes in Databases, and combine the four areas in a novel way to get the right balance between power and effectiveness.

a.1 Generalized Planning

Planning is a branch of AI that addresses the problem of generating a course of action to achieve a specified objective, given an initial state of the world. It is an area that is central to the development of intelligent agents and autonomous robots. In the last decades, automated planning has seen a spectacular progress in terms of scalability, much of it achieved by the development of heuristic search approaches as well as symbolic ones [80, 83].

Models. In Planning, we describe the system by introducing a set of atomic facts, called fluents, whose truth value changes as the system evolves, and by specifying preconditions and the effects of actions on such a set of facts. In each state, corresponding to the truth value assignment to the fluents, the agent can select which action to execute among those whose preconditions is satisfied. This way of modeling comes directly form **Reasoning about Action** in KR [123] and is embedded in the de-facto standard Planning Domain Definition Language (PDDL) [114, 82] introduced for International Planning Competitions (IPCs).

Two crucial observations are in order. The first is that such modeling is by construction **human-comprehensible**, in the sense that fluents and actions describe the dynamics of the domain of interest in a high-level terminology ready accessible to humans. This is inherited by Reasoning about Action and more generally Knowledge Representation wich is based on an explicit representation of the agent knowledge of the world it operates in and how it changes through actions. **We will retain the human-comprehensibility of the model.**

The second observation is that Planning models make use of propositional logic (or equivalently first-order logic with a finite object domain). This makes such models implicite representations of finite transition systems. Hence these models are readily **verifiable**, e.g. by model checking [43, 11, 106], which operates on finite-state systems. **We will retain the verifiability of the model.** Hence we essentially adopt this modeling features (human-comprehensibility and verifiability) in modeling WHITEMECH mechanisms with their capabilities and the environment we operate.

However for project **traditional models used in planning (e.g., PDDL) is too limited to represent mechanisms**. So, on the one hand, we will introduce **rich semantic descriptions** from Knowledge Representation to which the *PI has contributed significantly in the years [63, 52, 58, 131, 132, 57, 53, 55, 56, 12]*. In this way we will be able to incorporate data manipulation in mechanisms, which **lift models to a to a first-order state representation** from a a propositional one. See later. On the other hand, **we will incorporate componentization** typical of Service-Oriented Computing [24, 136, 18, 62, 49], so that we can consider the mechanisms formed by coordinated and orchestrated components each with its own features that together form the mechanism with its capabilities. *The*

[2] These concerns have been widely discussed, see for example https://futureoflife.org/background/benefits-risks-of-artificial-intelligence/.

PI has pioneered work on composition of stateful-services (services that react depending on the state there are in) [16, 15] and later extended these results to handle behavior compositions, e.g. of cognitive robotics systems in AI, [129, 130, 62, 49].

Goals. The goal in traditional planning is to find a plan (i.e., a program) that reach form the initial state a desired state of affairs specified in terms of a boolean combination of fluents. Notice that again the **goal is expressed declaratively at high-level, in a human comprehensible fashion**, as a formula that must hold in the state resulting from the execution of the plan. **We will retain this human comprehensibility of the goal**. Though for the mechanisms of interest in WHITEMECH, these **traditional form of goals is too limited**. Instead **we will handle to full-fledged temporal specification analogous to those used in model checking and in reactive synthesis** [43, 11, 119], though sidestepping the notorious difficulties algorithms required for reactive syntesis, by focusing on non-traditional kinds of specification formalisms proposed recently in, e.g., Reasoning about Action and Generalized Planning, such as LTL and LDL on finite traces, recently proposed by the PI together with Moshe Vardi (Rice U, Huston) [67, 64, 65] and adopted in generalized forms of Planning in AI [140, 38] and in declarative business processes in BPM [143, 45, 59], as well as safe/co-safe LTL/LDL formulas, which have been shown to be more expressive than expected while remaning **well-behaved** [75, 74, 102, 71]. Solvers for these are substantially simpler that for general reactive synthesis, being based on reachability and safety games, which are amenable to efficient implementations.

Algorithms. Planning in its classical version stems from the STRIPS planning model [73] that originated at the Shakey project [117, 97]. In classical planning actions are deterministic, that is, executing an action in a given state will result in a unique resulting state. Plans of this form are essentially sequence of actions, each executable when instructed, that lead from the initial state to a state where the goal is satisfied. In other words, planning in this context amounts into solving a reachability problem, in a transition system (finite state machine), whose states are represented logarithmically through fluents. Indeed the problem is PSPACE (-complete), being reachability NLOGSPACE and the FSM transition system exponentially larger then its specification, but constructible on-the-fly.

While classical planning assumes deterministic actions, many real-world planning problems are better suited to a model with non-deterministic (in the sense of adversarial) action outcomes. Fully observable non-deterministic (FOND) planning problems require the planner to consider every possible action outcome. Recent advances have greatly improved the efficiency of FOND planning [101, 111, 79, 96, 116], especially fo so called strong cyclic planning which guarantees achievement of the goal under an assumption of fairness.

Planning with sensing actions under partial observability in nondeterministic domains POND has also been studied, being considered fundamental to the realization of AI tasks in areas as diverse as robotics, game playing, and diagnostic problem solving. Indeed plan existence for conditional planning is 2EXPTIME-complete [125]. Some early POND planners were based on partial order planning (e.g., CNLP [118], Cassandra [122]) and Graphplan (e.g., SGP [7]). MBP [17] and BBSP [126] are more recent BDD-based model checking planners. Planners based on heuristic search include Contingent-FF [89], "POND" [27], and most recently CLG which has an offline variant [2, 3, 23]. Experimental results reported in the literature appear to indicate that CLG represents the state of the art in terms of scalability of offline conditional planning.

An agent planning to achieve a goal in a partially observable environment with sensing actions has the option of choosing how to act online by interleaving planning, sensing, and acting. Online planning is often used when the domain does not includes so call "deadends": i.e., actions whose effects cannot be undone. In this circumstances online plans are easier to compute since integrating online sensing with planning eliminates the need to plan for a potentially exponential (in the size of relevant unknown facts) number of contingencies. In the absence of deadends, we do have several fast and effective online planners. Recent advances include CLG and CLG+ [2, 3], K-Planner [22, 23], and SDR [25]. As deadends are allowed online planing becomes as difficult as offline planning, since even if we are

interested only in the first action at each step, to compute it we may need to compute the entire plan exactly to avoid deedends.

One specific observation we can draw is that like agents in Planning, we expect mechanisms to be able to handle quickly and efficiently most cases, i.e., those cases that do not require to solve difficult, "puzzle-like", situations. Indeed while the Planning community has concentrated on simpler forms of process synthesis, it has developed a sort of **science of search algorithms for Planning**, which has brought about improvements by orders of magnitude in the last decade [121, 137, 105, 59]. WHITEMECH will exploit this knowledge and extend algorithms and heuristics to handle generalized forms of Planning and to reactive Synthesis.

Specifically, recent work by the **PI has explored connections between generalized forms of planning and synthesis** when goals are temporally extended: goals are expressed as requirements on the entire execution instead of the final state of the execution [93, 8, 40, 66, 9, 37]. *Promising exploratory results by the PI and other [48, 133, 67, 64, 38]) are available*, but such results are far from clarifying the picture both from a theoretical perspective and from an algorithmic perspective.

Though it appears possible to **compile** the more general forms of synthesis problems which reduces to reachability and co-reachability/safety games **into synthetic planning domains so as to exploit the algorithmic insights from Planning**, Classical Planning, FOND and POND, to solve these games. In particular, it appears possible to exploit the correspondence between FOND and 2-player games, and the significant recent advances in the computational efficiency of FOND planning that have produced FOND planners that scale well in many domains (e.g., NDP [4], FIP [79, 78], MyND [111], Gamer [96] and PRP [116]). Moreover some of these solvers internals can be modified to handle directly reachability and co-reachability/safety games which are at the base of many of the synthesis problems of interest in this project. For example, focusing on FOND, most modern algorithms are for strong cyclic planning, instead for reachability games we need to solve strong planning. This requires some adjustment of the internals of the planner E.g., in MYND [111], we need to move from AO* to LAO* for searching the AND/OR graph, see e.g., ; in OBDD approaches [139], we need to change the algorithm for growing the "winning" set. The "state of the art" is PRP [116], which solves strong cyclic planning by iterated calls to LAMA [124] and hence uses classical planning algorithms, it's can be adjusted to by adding counters though this would not retain the maximal efficiency. Better techniques need to be developed in this case.

Again the crucial point is that these solvers based on Planning come with very effective domain independent heuristic, which are continuously improved by a scientific community focussing specifically on improving the search process in these solvers. Notice that to get this efficiency we do not need to relay on any specific modeling formalism [77]. Hence, in spite of the notable changes in the model and the goals we wil consider in WHITEMECH, **we are confident that the planning algorithms will scale up** .

a.2 Verification and Synthesis

WHITEMECH will draw on the rich modeling and algorithmic techniques from Formal Methods, particularly program verification and synthesis.

The main approach to program verification is *model-checking*. This is a set of algorithms and techniques, often based on logic and automata, for automatically determining whether a mathematical model of a system satisfies a specification formally expressed in mathematical logic [43, 11]. This field, for which the founders and proponents won two different Turing awards, has been used successfully for complex systems, and many hardware and software companies use these methods in practice, e.g., verification of VLSI circuits, communication protocols, software device drivers, real-time embedded systems, and security algorithms.

The same community has been developing *synthesis*, i.e., techniques to automatically construct a system that satisfies a given specification. Of particular relevance is synthesis applied to *reactive systems*: those maintain an ongoing interaction with an external environment.[3] Such *Reactive Syn-*

[3]This is in contrast to classical non-reactive programs that transform static inputs to static outputs.

thesis is best viewed as a game between the uncontrollable environment and the program to be synthesized. A correct program can then be viewed as a winning strategy in this game, and thus synthesis reduces to finding such a strategy [42, 119]. Over the years, the formal-methods community has exploited this game-theoretic viewpoint and developed a **comprehensive and mathematically elegant theory of reactive synthesis** connecting logics, automata theory and games [144, 99, 100, 98, 145, 86, 41, 20, 21, 75, 109, 76, 81, 72, 19, 6, 70, 26, 90].

In spite of the rich theory developed for program synthesis, little of this theory has been reduced to practice. Some researchers argue that this is because the realizability problem for linear-temporal logic (LTL) specifications is 2EXPTIME-complete [119, 127]. However, this argument is not compelling. First, experience with verification shows that even nonelementary algorithms can be practical, since the worst-case complexity does not arise often (cf., the model-checking tool MONA [69]). Furthermore, in some sense, synthesis is not harder than verification. This may seem to contradict the known fact that while verification is linear in the size of the model and at most exponential in the size of the specification [43], synthesis from LTL specifications is 2EXPTIME-complete. There is, however, something misleading in this fact: while the complexity of synthesis is given with respect to the specification only, the complexity of verification is given with respect to the specification and the program, which can be much larger than the specification. In particular, it is shown in [127] that there are temporal specifications for which every realizing program must be at least doubly exponentially larger than the specifications. Clearly, the verification of such programs is doubly exponential in the specification, just as the cost of synthesis.

We believe there are a number of reasons for the **lack of practical impact of the theory of reactive synthesis**. We list 4 of these, and include principles and directions for overcoming them.

(i) **The focus on expressing both the specification and the model in LTL (as in reactive synthesis) hides the relevant complexities**.

Exactly as in model checking, we need to distinguish the **complexity wrt the model of the mechanisms** and the **complexity wrt the specification of its tasks (goals)**. In this light, the complexity of solving games with LTL objectives is PTIME-complete in the size of the model, i.e., EXPTIME-complete in the compact representation of the model, and 2EXPTIME-complete in the size of the specification formula of the goal. Thus, even in the case that the model is given compactly (as in Planning), the complexity wrt the model is much lower than the complexity wrt the specification. This is a particularly important observation, since in general the model is going to be much larger than the goals.

(ii) **The specifications languages lead to constructions that are too complex**. The approach to LTL synthesis or solving LTL games involves two main steps. First, constructing parity tree-automata that realize all winning strategies, and second, testing such automata for emptiness. The first step can be done using determinization of Büchi automata. Unfortunately, determinization of these automata is not as simple as determinization of ordinary automata (on finite strings), and has been notoriously resistant to efficient implementations [142]. Alternative constructions that avoid determinization [100, 98] did not prove to be radically more efficient [74]. The second step involves solving parity games. This problem, solvable in quasi-polynomial time, is not known to be in PTIME [**?**] and has been attacked with many different algorithms.[4]

However, there are relevant classes of specifications, coming from generalized planning in AI and declarative business processes, **that sidestep these difficulties alltogether**. These advocate the use of linear-temporal logics over *finite traces*, i.e., LTL_f and its MSO-complete extension LDL_f. The principle technical advantage of using these is that they involve *ordinary* automata on finite words which are indeed amenable to quite good implementations [143, 67, 64, 65, 140, 38].

(iii) **Techniques are optimized for solving difficult synthesis problems, including puzzle-like ones**. Reactive synthesis is typically applied to low-level problems in which there is no reason to think that solutions are going to be easy to find — the planning literature calls these puzzle-like problems. On the other hand, it is unrealistic to require WHITEMECH to handle such artificial problems. Instead, just as in Planning, WHITEMECH will aim at solving large problems that are

[4] See https://github.com/tcsprojects/pgsolver.

not puzzle-like. Following this assumption, Planning has obtained a spectacular improvement in the last two decades, relying on heuristics which ultimately can be seen as abstraction that relax detail of the problem at hand. In WHITEMECH we will apply such heuristics to synthesis. We will also explore promising techniques from the formal-methods community, i.e., bounded-synthesis where small controllers are explored before larger ones [75, 74, 71] (currently, even this construction is too complicated to be applied in real cases [92]). Are you sure?

(iv) **Most synthesis techniques are not compositional**. This is a major methodological issue. Most current theory of program synthesis assumes that one gets a comprehensive set of temporal assertions as a starting point. In the context of WHITEMECH and the applications we have in mind, this is not realistic. A more realistic approach would be to assume an evolving formal specification: temporal assertions can be added, deleted, or modified. Since it is rare to have a complete set of assertions at the very start of the design process, there is a need to develop compositional synthesis algorithms. Such algorithms can, for example, refine designs when provided with additional temporal properties [98, 74, 6]. Are these the right citations? The idea of composing solutions has been pioneered in service-oriented computing, and indeed the work by the PI pioneered this approach [16, 48, 62, 33].

a.3 KR and First-Order State Representation

The third ingredient of WHITEMECH approach is **Knowledge Representation** and **Reasoning about Action**.

The key property of white-box self-programming mechanisms is the ability of describing their specification, the programs they generate and the relationship between the two in human terms.

To do so the domain where the mechanism operates, the mechanism itself, its capabilities and limitations, as well as its specifications and goals need to be formally described in terms of concepts that can be shared with humans.

WHITEMECH will consider this issue upfront, by founding white-box self-programming mechanisms on the area of AI called **Knowledge Representation**.

The PI is a prominent member of the scientific community working on Knowledge Representation. He is indeed the single researcher with most papers appeared in the Flagship Conference of the community: the International Conference on Principles of Knowledge Representation and Reasoning (KR) ranked A according to CORE.*

Knowledge Representation stems from a deep tradition in logic. In particular, it aims at building systems that know about their world they operate in and are able to act in an informed way in it, as humans do. A crucial part of these systems is that knowledge is represented "*symbolically*", and that "*reasoning procedures*" are able to extract consequences of such knowledge as new symbolic representations. Such an ability is used to deliberate in an informed fashion the course of actions to take, understanding the consequences of the action performed.

In [103, 104], it is argued that Knowledge Representation is radically new idea in human history. It comes about after a long gestation, stemming from Aristotle, who developed the initial notion of logic though unrelated to notion of computation; continued by Leibniz, who brought forward a notion of "thinking as computation", though not yet symbolic; and later by Frege, who developed the notion of symbolic logic, though unrelated to computation; and finally by the breakthrough in human thinking of the early part of last century with Church, Godel, and Turing, who set the bases for symbolic logic bound together with computation and ultimately for Computer Science, though even them did not think about logic as a way of representing knowledge. The use of a symbolic systems to represent knowledge expressed in terms of human concepts can only be traced back to McCarthy's 1959 seminal work on Artificial Intelligence [112].

Knowledge Representation has developed enormously since then in diverse directions and subareas. Here we focus mainly on the area of reasoning about action.

Reasoning about Action takes a first-person view of an agent. [5] The agent as a representation of the domain in which it operates; a representation of the possible actions in terms of preconditions and effects formally expressed over the representation of domain; finally it has a representation of complex agent behaviors and capabilities typically described through high-level programs, whose atomic instruction and tests corresponds, respectively, to actions and queries over the representation of the domain. Through reasoning on these representations the agent undestand what doing an action or enacting a certain behavior will bring about, enabling it to make informed decisions on which action to choose or to exclude in relation to its current tasks/goals/specifications. Reasoning about action has been studied in depth through comprehensive frameworks, such as that of Situation Calculus [113, 123]. As mentioned, it has deeply influenced the models used in Planning, including PDDL, which, though drastically simplified, come directly from these studies.

The PI has deeply contributed to the development of Reasoning about Action. In particular, within the framework of Sitation Calculus. Specifically, the PI introduced of the high-level programming languages ConGolog and IndiGolog, the distinction between off-line executions and online executions, the notion of execution monitoring and recovery, interleaving execution and planning, the notion of ability [63, 52, 131, 132].

It is important ot notice that the work Reasoning about Action in Situation Calculus as well as in most other frameworks, is based on a **First-Order Representation of the State**. In other words the current state of the Agent is represented in terms of predicates/relations. This give rise to infinite transition systems which are typically impossible to verify directly.

Recently the PI has devised a set of novel results that shows the effective computability of expressive variants of the original full-fledged (predicate based) Situation Calculus [55, 56, 54, 12, 35]. Such results are being complemented by the possibility of combining action theories with ontological representations in description logics [32, 30, 87, 36]. Moreover, the techniques for applying belief revision to transition systems based proposed recently in [88, 39], open up the possibility of grounding computationally the notion of "model revision" for mechanisms.

We stress that in WHITEMECH we intend to represents that state of the mechanism as well as the state of the enviroment in with it operates, in a semantically rich fashion. To do so we relay on Description Logic and Ontology-Based Data Access.

Description Logics are the formalism of election in for representing the information on the domain of interest in terms of objects grouped in classes or concepts and relationships among such classes. Moreover **description logics** are nowadays considered the standard formalism for ontologies and deeply shaped semantic technologies including the current W3C standard OWL 2.

The PI has deeply contributed in the development of description logics. First he worked on the correspondence between expressive description logics and modal logics of programs such as Dynamic Logic [51, 31]. Then more recently, together with his group in Rome he developed one of the best know light weight description logics, DL-lite [29], which is essentially able to formalize UML class diagrams and Entity-Relationship diagrams, while keeping inference and conjunctive query answering tractable (the latter in AC0, the same cost as standard SQL). DL-lite in turn made it possible to develop ***Ontology Based Data Access****, which can be considered the most successful use of semantic technologie for data integration [120, 135, 95].*

The role of these technologies in WHITEMECH is to represent in terms of an ontology chosen so has to capture the doman on interest in which the mechanism is operating as well as the (high-level) data structures of the mechanism itself in terms that are sharable by humans as e.g., in [138]. In particular WHITEMECH intends to exploiting the techniques for efficient query-aswering provided by DL-lite and Ontology Based Data Access, especially in the write-also variants, explored recently [61, 50].

Notice that on the other hand WHITEMECH does not aim at defining new concrete representation languages. Instead it intends to use well-known formalisms such as BPMN, UML, OWL, etc. as concrete

[5] This contrast with the work in Multi Agents Systems, where typically a third person view (a "designer" view) is adopted.

languages, though with a precise formal semantics to allow for automated reasoning, verification and synthesis, see e.g., [14, 60].

a.4 Data-Awareness

Crucially, differently from current work in Planning and in Verification and Synthesis, which operate with a propositional representation of the state, WHITEMECH does not want to discard data, and hence it will need to consider a first-order or relational representation of the state. This gives rise to infinite transition systems which are generally problematic to analyze. *However the PI has already shown within the EU FP7-ICT-257593 ACSI: Artifact-Centric Service Interoperation, that such difficulties can be overcome in notable cases [15, 28, 10, 34] in the context of* ***Data-Aware Processes*** *in Databases.* A key decovery is that under natural assumptions these infinite-state transitions systems admit faithful **abstractions** to finite-state ones, hence enabling the possibility of using the large body of techniques developed within Verification and Synthesis in Formal Methods. Moreover important advancements in undestanding how to deal with such complexity have been established as well as relationships with Reasoning about Action in KR [87, 13, 36, 35, 12]. We will leverage on these ideas to lift our results so as to handle data.

Excellent results, but no synthesis.

a.5 Componentization

WHITEMECH intends to **support component-based approaches.**

Interfaces capture essential properties of components while hiding irrelevant details. Interface theories [44] provide composition operators, compatibility notions (is the composition of a set of components legal?), and conditions for substitutability (when can one component be replaced by another without compromising the properties of the overall system?) that enable component-based design and alleviate state explosion. Compositionality and incrementality raise a number of questions in the context of synthesis: How to separately synthesize controllers for individual components, or for the same component but with respect to different properties, and then combine them into a single controller? How to derive the component interfaces? Because components take different forms depending on the application domain (they can be pieces of hardware, of software, or of models written in a high-level language such as Simulink or UML), there is no unique, one-size-fits-all interface model. For instance, interfaces carry different information in synchronous vs. dataflow components, or when reasoning about I/O dependencies vs. correctness vs. timing and performance properties [108, 107, 141]. Domain knowledge is key in identifying the right level of abstraction and information content of the interfaces. In conjunction with this, we will develop methods to derive composite interfaces from basic interfaces automatically, thus maximizing the synergy of human and synthesizer.

a.6 Integrating stochastic decision and reinforcement learning

WHITEMECH aims at **allowing learning and stochastic decisions, while remaining within safe bounds**.

While classical formal verification and synthesis relies on deductive reasoning and decision procedures for constraint-satisfaction problems, our view of synthesis also involves inference of program components from example behaviors, both positive and negative, using computational learning. While there is a wealth of literature on learning theory (see, for instance, [94]), we plan to explore how it can be adopted and advanced in the context of program synthesis. As an illustrative example, consider the recent interactive add-in for Microsoft Excel spreadsheet software based on the concept of "programming by examples" [84, 85]. In this system, the user demonstrates the desired transformation task (for example, rewriting all names to a uniform format such as first-name followed by a single space, followed by last-name) using a few examples, and the synthesis tool automatically generates

the best program (in the form of an Excel Macro) that is consistent with all the examples. The synthesized program is then used to transform all the entries in the spreadsheet, and if this synthesized transformation does not match the users intent, the user can guide the synthesizer by highlighting the incorrect updates, thus, providing negative examples for the subsequent iteration. While this tool has the potential of enormous impact by facilitating intuitive programming by millions of end users, the theoretical foundations of learning the desired program from examples are not yet understood. A suitable theoretical abstraction for the desired program is a "string-to-string" transducer. While there is a well developed theory of minimization and learning for the class of sequential transducers [115], sequential transducers are not expressive enough to capture the typical transformations that involve swapping of substrings. The recently proposed model of streaming string transducers [5] has appealing theoretical properties, such as well characterized expressiveness. Since this model can capture the desired Excel transformations, algorithms for minimization and learninga topic for proposed research, can lead to efficient synthesis procedure with guarantees of convergence.

a.7 Driving Application Contexts

WHITEMECH will ground its scientific results in three diverse real **application** to demonstrate the actual utilization of the scientific achievements within the project: Smart manufacturing (Industry 4.0), Smart spaces (IoT), and Business Processes Management Systems (BPM). *The PI and his group at Sapienza have contributed to all these fields, see e.g., [46, 47, 68]. Moreover the PI has applied advanced science to real-cases in the area Semantic Data Integration where he and his group have invented the Ontology-Based Data Access paradigm, possibly the most successful approach for Semantic Data Integration [120, 135, 95]. Such an approach has matured to the point that the PI founded a Sapienza Start-Up* ***OBDA Systems*** *(http://www.obdasystems.com) to commercially exploit it in real data integration scenarios.*

a.7.1 Smart Manufacturing

a.7.2 Internet of Things

a.7.3 Business Processes

a.8 High Risk, High Gain

Recent foundational results by the PI chart a novel path that within WHITEMECH will revolutionize **Reasoning about Action** in KR and **Generalized Planning** in AI by introducing rich objectives, data, and componentization in order to produce a **breakthrough in engineering self-programming mechanisms that are human-comprehensible and safe by design.** Moreover WHITEMECH aims at being the spark that will bring together and cross-fertilize four distinct research areas, namely **Reasoning about Action** in *Knowledge Representation*, **Data-aware Processes** in *Databases*, **Verification and Synthesis** in *Formal Methods*, and **Generalized Planning** in *AI*. The PI has profoundly contributed to all these areas, and he is one of the most prominent AI scientist leading this cross-fertilization.

WHITEMECH is a **high risk, high gain project**. It is **high gain** since, if successful, it will result in a radically more useful automated mechanisms than what we have today, namely **white-box self-programming mechanisms**, unleashing full potential of **self-programmability** and removing the main barriers to the uptake of automated mechanisms in real business context, namely *predefined forms of automation*, and difficulties in *formally analyzing their automated behavior in human terms*. Given the crucial role that automated mechanisms plays in our modern economy and society and that **white-box self-programming mechanisms** have the potential to resolve a number of central hurdles, this implies a potentially very high impact on computer science as well as in crucial business contexts, facilitating the already ongoing uptake of automated mechanisms in industry eventually also on economy and society. As a result, WHITEMECH is **very timely** and of **greatest significance for European science**.

The WHITEMECH involves **high risk** because the ultimately we need to merge explicit representation, with advanced form of process synthesis/coordination and refinement and identify novel and useful islands of effective feasibility that WHITEMECH aims at is technically extremely challenging, much more so than the related Verification and Planning both of which have made tremendous progresses in the last decade, touching upon several notorious open problems in computer science. On top of that, the path from theoretical analysis to practically efficient algorithm, which we plan to fully explore within WHITEMECH, is a huge challenge given that we aim realizing real automated mechanisms to be deployed in real business scenarios.

Despite the challenges that WHITEMECH will face, as discussed above, the general **feasibility** of WHITEMECH is demonstrated by the PI original, exploratory publications [67, 64, 65, 45, 62, 49], and follow ups [140, 38].

b Methodology

b.1 Research Streams

The scientific work in WHITEMECH will be methodologically structured into 3 broad research streams:

- **Principles and Foundations** that will deal with the scientific foundations of white-box self-programming mechanisms.
- **Algorithms and Tools** that will deal with the development of practical algorithms, optimizations and tools for realizing such mechanisms.
- **Applications and Evaluation** that will evaluate white-box self programming mechanisms in the three business critical driving applications mentioned above.

The **Principles and Foundations** and **Algorithms and Tools** streams will cut across 5 workpackages (WPs) roughly corresponding to the 5 objectives above. The **Applications and Evaluation** stream will be further refined into 3 separate WPs for the 3 driving applications.

b.2 Workpackages

Research will be organized in five scientific WPs, and three applicative WPs. The five scientific WPs will handle all objectives by looking first to the representation; then to self-programming in the core case in which mechanisms are finite state; then the advanced case in which mechanisms have a FOL state and hence need to be abstracted to finite state before self-programming can be performed; then will focus specifically at mechanisms that are component-based; and finally we will look at the integration of ML and stochastic decision making techniques into the framework.

WP1: Representing mechanisms and their operation domains

This WP addresses the objectives of **making white-box self-programming mechanisms formally verifiable and comprehensible to humans**.

WP2: Finite-state white-box self-programming mechanisms

This WP addresses the objective of **equipping mechanisms with general self-programming abilities**

WP3: FOL-state white-box self-programming mechanisms

This WP addresses the objective of **making self-programming mechanisms data-aware**

WP4: Component-based white-box self-programming mechanisms

This WP addresses the objective of **supporting component-based approaches.**

WP5: Integrating stochastic decision and reinforcement learning

This WP addresses the objective of **allowing learning and stochastic decisions, while remaining within safe bounds**.

WP6: Application and Evaluation

b.3 Phases

– **Core Science**
– **Refinement and Engineering**
– **Towards Practical Adoption**

c Resources

c.1 Budget

Cost Category			**Total in Euro**
Direct Costs	**Personnel**	PI	0
		Senior Staff (Development engineer)	0
		Postdocs	0
		Students	0
		Other	0
	i. Total Direct Costs for Personnel (in Euro)		0
	Travel		0
	Equipment		0
	Other goods and services	Consumables	0
		Publications (including Open Access fees), etc.	0
		Other (Collaboration)	0
	ii. Total Other Direct Costs (in Euro)		0
A - Total Direct Costs (i + ii) (in Euro)			0
B - Indirect Costs (overheads) 25% of Direct Costs (in Euro)			0
C1 - Subcontracting Costs (no overheads) (in Euro)			0
C2 - Other Direct Costs with no overheads (in Euro)			0
Total Estimated Eligible Costs (A + B + C) (in Euro)			0
Total Requested EU Contribution (in Euro)			0

Please indicate the duration of project in months:	0
Please indicate the % of working time the PI dedicates to the project over the period of the grant:	0%
Please indicate the % of working time the PI spends in an EU Member State or Associated Country over the period of the grant:	100%

References

[1] ACM U.S. Public Policy Council and ACM Europe Policy Committee. Statement on algorithmic transparency and accountability. ACM, 2017.

[2] A. Albore, H. Palacios, and H. Geffner. A translation-based approach to contingent planning. In *IJCAI*, pages 1623–1628, 2009.

[3] A. Albore, M. Ramírez, and H. Geffner. Effective heuristics and belief tracking for planning with incomplete information. In *ICAPS*, 2011.

[4] R. Alford, U. Kuter, D. S. Nau, and R. P. Goldman. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artif. Intell.*, 216:206–232, 2014.

[5] R. Alur and P. Cerný. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *POPL*, pages 599–610, 2011.

[6] R. Alur, S. Moarref, and U. Topcu. Compositional synthesis of reactive controllers for multi-agent systems. In *CAV*, pages 251–269, 2016.

[7] C. Anderson, D. Smith, and D. Weld. Conditional effects in graphplan. In *AIPS*, pages 44–53. AAAI Press, 1998.

[8] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. Math. Artif. Intell.*, 22(1-2):5–27, 1998.

[9] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2):123–191, 2000.

[10] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. In *PODS*, pages 163–174, 2013.

[11] C. Baier, J.-P. Katoen, and K. Guldstrand Larsen. *Principles of Model Checking*. MIT Press, 2008.

[12] B. Banihashemi, G. De Giacomo, and Y. Lespérance. Abstraction in situation calculus action theories. In *AAAI*, pages 1048–1055, 2017.

[13] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of agent-based artifact systems. *J. Artif. Intell. Res. (JAIR)*, 51:333–376, 2014.

[14] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artif. Intell.*, 168(1-2):70–118, 2005.

[15] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *VLDB*, pages 613–624, 2005.

[16] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *ICSOC*, pages 43–58, 2003. 10 year most influential paper award at ICSOC'13.

[17] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *IJCAI*, 2001.

[18] P. Bertoli, M. Pistore, and P. Traverso. Automated composition of web services via planning in asynchronous domains. *Artif. Intell.*, 174(3-4):316–361, 2010.

[19] R. Bloem, K. Chatterjee, S. Jacobs, and R. Könighofer. Assume-guarantee synthesis for concurrent reactive programs with partial information. In *TACAS*, pages 517–532, 2015.

[20] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3), 2012.

[21] A. Bohy, V. Bruyère, E. Filiot, N. Jin, and J. Raskin. Acacia+, a tool for LTL synthesis. In *CAV*, pages 652–657, 2012.

[22] B. Bonet and H. Geffner. Planning under partial observability by classical replanning: Theory and experiments. In *IJCAI*, pages 1936–1941, 2011.

[23] B. Bonet and H. Geffner. Flexible and scalable partially observable planning with linear translations. In *AAAI*, pages 2235–2241, 2014.

[24] A. Bouguettaya, Q. Z. Sheng, and F. Daniel, editors. *Web Services Foundations*. Springer, 2014.

[25] R. I. Brafman and G. Shani. Online belief tracking using regression for contingent planning. *Artif. Intell.*, 241:131–152, 2016.

[26] R. Brenguier, J. Raskin, and O. Sankur. Assume-admissible synthesis. *Acta Inf.*, 54(1):41–83, 2017. Work partially supported by the ERC inVEST (279499) project.

[27] D. Bryce, S. Kambhampati, and D. E. Smith. Planning graph heuristics for belief space search. *J. Artif. Intell. Res.*, 26:35–99, 2006.

[28] D. Calvanese, G. De Giacomo, R. Hull, and J. Su. Artifact-centric workflow dominance. In *ICSOC*, pages 130–143, 2009.

[29] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[30] D. Calvanese, G. De Giacomo, D. Lembo, M. Montali, and A. Santoso. Ontology-based governance of data-aware processes. In *RR*, pages 25–41, 2012.

[31] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS*, pages 149–158, 1998.

[32] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Actions and programs over description logic ontologies. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, 2007.

[33] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Regular open APIs. In *KR*, pages 329–338, 2016.

[34] D. Calvanese, G. De Giacomo, M. Montali, and F. Patrizi. Verification and synthesis in description logic based dynamic systems. In *RR*, pages 50–64, 2013. Best paper award.

[35] D. Calvanese, G. De Giacomo, M. Montali, and F. Patrizi. First-order μ-calculus over generic transition systems and applications to the situation calculus. *Information & Computation*, 2017. To appear.

[36] D. Calvanese, G. De Giacomo, and M. Soutchanski. On the undecidability of the situation calculus extended with description logic ontologies. In *IJCAI*, pages 2840–2846, 2015.

[37] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Reasoning about actions and planning in LTL action theories. In *KR*, pages 593–602, 2002.

[38] A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724, 2017.

[39] M. Carrillo and D. A. Rosenblueth. CTL update of kripke models through protections. *Artif. Intell.*, 211:51–74, 2014.

[40] S. Cerrito and M. C. Mayer. Bounded model search in linear temporal logic and its application to planning. In *TABLEAUX*, pages 124–140, 1998.

[41] K. Chatterjee and T. A. Henzinger. Assume-guarantee synthesis. In *TACAS*, pages 261–275, 2007.

[42] A. Church. Logic, arithmetics, and automata. In *Proc. International Congress of Mathematicians, 1962*. institut Mittag-Leffler, 1963.

[43] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[44] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT*, pages 148–165, 2001.

[45] G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, and M. Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *BPM*, pages 1–17, 2014.

[46] G. De Giacomo, C. Di Ciccio, P. Felli, Y. Hu, and M. Mecella. Goal-based composition of stateful services for smart homes. In *OTM*, pages 194–211, 2012.

[47] G. De Giacomo, M. Dumas, F. M. Maggi, and M. Montali. Declarative process modeling in BPMN. In *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, pages 84–100, 2015.

[48] G. De Giacomo, P. Felli, F. Patrizi, and S. Sardiña. Two-player game structures for generalized planning and agent composition. In *AAAI*, 2010.

[49] G. De Giacomo, A. E. Gerevini, F. Patrizi, A. Saetti, and S. Sardiña. Agent planning programs. *Artif. Intell.*, 231:64–106, 2016.

[50] G. De Giacomo, D. Lembo, X. Oriol, D. F. Savo, and E. Teniente. Practical update management in ontology-based data access. In *ISWC*, 2017.

[51] G. De Giacomo and M. Lenzerini. Tbox and abox reasoning in expressive description logics. In *KR*, pages 316–327, 1996.

[52] G. De Giacomo, Y. Lespérance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[53] G. De Giacomo, Y. Lespérance, and C. J. Muise. On supervising agents in situation-determined congolog. In *AAMAS*, pages 1031–1038, 2012.

[54] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Sardiña. Verifying ConGolog programs on bounded situation calculus theories. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 950–956, 2016.

[55] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Vassos. LTL verification of online executions with sensing in bounded situation calculus. In *ECAI*, pages 369–374, 2014.

[56] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Vassos. Progression and verification of situation calculus agents with bounded beliefs. *Studia Logica*, 104(4):705–739, 2016.

[57] G. De Giacomo, Y. Lespérance, and A. R. Pearce. Situation calculus based programs for representing and reasoning about game structures. In *KR*, 2010.

[58] G. De Giacomo, H. J. Levesque, and S. Sardiña. Incremental execution of guarded theories. *ACM Trans. Comput. Log.*, 2(4):495–525, 2001.

[59] G. De Giacomo, F. M. Maggi, A. Marrella, and F. Patrizi. On the disruptive effectiveness of automated planning for LTL_f-based trace alignment. In *AAAI*, pages 3555–3561, 2017.

[60] G. De Giacomo, X. Oriol, M. Estañol, and E. Teniente. Linking data and BPMN processes to achieve executable models. In *CAISE*, pages 612–628, 2017.

[61] G. De Giacomo, X. Oriol, R. Rosati, and D. F. Savo. Updating dl-lite ontologies through first-order queries. In *ISWC*, pages 167–183, 2016.

[62] G. De Giacomo, F. Patrizi, and S. Sardiña. Automatic behavior composition synthesis. *Artif. Intell.*, 196:106–142, 2013.

[63] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *KR*, pages 453–465, 1998.

[64] G. De Giacomo and M. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564, 2015.

[65] G. De Giacomo and M. Vardi. LTL_f and LDL_f synthesis under partial observability. In *IJCAI*, pages 1044–1050, 2016.

[66] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *ECP*, pages 226–238, 1999.

[67] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[68] L. de Silva, P. Felli, J. C. Chaplin, B. Logan, D. Sanderson, and S. M. Ratchev. Synthesising industry-standard manufacturing process controllers. In *AAMAS*, pages 1811–1813, 2017.

[69] J. Elgaard, N. Klarlund, and A. Møller. MONA 1.x: New techniques for WS1S and WS2S. In *CAV*, pages 516–520, 1998.

[70] J. Esparza, J. Kretínský, J. Raskin, and S. Sickert. From LTL and limit-deterministic büchi automata to deterministic parity automata. In *TACAS*, pages 426–442, 2017. Work partially supported by the ERC inVEST (279499) project.

[71] P. Faymonville, B. Finkbeiner, M. N. Rabe, and L. Tentrup. Encodings of bounded synthesis. In *TACAS*, pages 354–370, 2017. Supported by the European Research Council (ERC) Grant OSARES (No. 683300).

[72] J. Fearnley, D. A. Peled, and S. Schewe. Synthesis of succinct systems. *J. Comput. Syst. Sci.*, 81(7):1171–1193, 2015.

[73] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.

[74] E. Filiot, N. Jin, and J. Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.

[75] B. Finkbeiner and S. Schewe. Bounded synthesis. *STTT*, 15(5-6):519–539, 2013.

[76] S. Fogarty, O. Kupferman, M. Y. Vardi, and T. Wilke. Profile trees for büchi word automata, with application to determinization. *Inf. Comput.*, 245:136–151, 2015.

[77] G. Frances, M. Ramirez, N. Lipovetzky, and H. Geffner. Purely declarative action representations are overrated: Classical planning with simulators. In *IJCAI*, 2017.

[78] J. Fu, A. C. Jaramillo, V. Ng, F. B. Bastani, and I. Yen. Fast strong planning for fully observable nondeterministic planning problems. *Ann. Math. Artif. Intell.*, 78(2):131–155, 2016.

[79] J. Fu, V. Ng, F. B. Bastani, and I. Yen. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *IJCAI*, pages 1949–1954, 2011.

[80] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[81] B. Genest, D. A. Peled, and S. Schewe. Knowledge = observation + memory + computation. In *FoSSaCS*, pages 215–229, 2015.

[82] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619 – 668, 2009.

[83] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[84] S. Gulwani. Automating string processing in spreadsheets using input-output examples. In *POPL*, pages 317–330, 2011.

[85] S. Gulwani. Programming by examples: Applications, algorithms, and ambiguity resolution. In *IJCAR*, pages 9–14, 2016.

[86] A. Harding, M. Ryan, and P.-Y. Schobbens. A new algorithm for strategy synthesis in ltl games. In *TACAS*, pages 477–492, 2005.

[87] B. B. Hariri, D. Calvanese, M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. Description logic knowledge and action bases. *J. Artif. Intell. Res. (JAIR)*, 46:651–686, 2013.

[88] A. Herzig, M. V. de Menezes, L. N. de Barros, and R. Wassermann. On the revision of planning tasks. In *ECAI*, pages 435–440, 2014.

[89] J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.*, 170(6-7):507–541, 2006.

[90] P. Hunter, G. A. Pérez, and J. Raskin. Reactive synthesis without regret. *Acta Inf.*, 54(1):3–39, 2017. Work partially supported by the ERC inVEST (279499) project.

[91] IBM. An architectural blueprint for autonomic computing. IBM White Paper, 2005.

[92] S. Jacobs, R. Bloem, R. Brenguier, A. Khalimov, F. Klein, R. Könighofer, J. Kreber, A. Legg, N. Narodytska, G. A. Pérez, J. Raskin, L. Ryzhyk, O. Sankur, M. Seidl, L. Tentrup, and A. Walker. The 3rd reactive synthesis competition (SYNTCOMP 2016): Benchmarks, participants & results. In *SYNT@CAV*, pages 149–177, 2016.

[93] F. Kabanza, M. Barbeau, and R. St.-Denis. Planning control rules for reactive agents. *Artif. Intell.*, 95(1):67–11, 1997.

[94] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, 1994.

[95] E. Kharlamov, D. Bilidas, D. Hovland, E. Jimenez-Ruiz, D. Lanti, H. Lie, M. Rezk, M. Skjaeveland, A. Soylu, G. Xiao, D. Zheleznyakov, M. Giese, Y. Ioannidis, Y. Kotidis, M. Koubarakis, and A. Waaler. Ontology based data access in statoil. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2017. In print.

[96] P. Kissmann and S. Edelkamp. Gamer, a general game playing agent. *KI*, 25(1):49–52, 2011.

[97] B. Kuipers, E. A. Feigenbaum, P. E. Hart, and N. J. Nilsson. Shakey: From conception to history. *AI Magazine*, 38(1):88–103, 2017.

[98] O. Kupferman, N. Piterman, and M. Y. Vardi. Safraless compositional synthesis. In *CAV*, pages 31–44, 2006.

[99] O. Kupferman and M. Vardi. Synthesis with Incomplete Informatio. *ICTL*, 1997.

[100] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542, 2005.

[101] U. Kuter, D. S. Nau, E. Reisner, and R. P. Goldman. Using classical planners to solve nondeterministic planning problems. In *ICAPS*, pages 190–197, 2008.

[102] B. Lacerda, D. Parker, and N. Hawes. Optimal policy generation for partially satisfiable co-safe LTL specifications. In *IJCAI*, pages 1587–1593, 2015.

[103] H. J. Levesque. On our best behaviour. *Artif. Intell.*, 212:27–35, 2014.

[104] H. J. Levesque. *Common Sense, the Turing Test, and the Quest for Real AI.* MIT Press, 2017.

[105] N. Lipovetzky and H. Geffner. Best-first width search: Exploration and exploitation in classical planning. In *AAAI*, pages 3590–3596, 2017.

[106] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.

[107] R. Lublinerman, C. Szegedy, and S. Tripakis. Modular code generation from synchronous block diagrams: modularity vs. code size. In *POPL*, pages 78–89, 2009.

[108] R. Lublinerman and S. Tripakis. Modular code generation from triggered and timed block diagrams. In *RTAS*, pages 147–158, 2008.

[109] Y. Lustig and M. Y. Vardi. Synthesis from component libraries. *STTT*, 15(5-6):603–618, 2013.

[110] A. Marrella, M. Mecella, and S. Sardiña. Intelligent process adaptation in the smartpm system. *ACM TIST*, 8(2):25:1–25:43, 2017.

[111] R. Mattmüller, M. Ortlieb, M. Helmert, and P. Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, pages 105–112, 2010.

[112] J. McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 756–791, 1957.

[113] J. McCarthy and P. J. Hayes. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

[114] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CT, 1998.

[115] M. Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234(1-2):177–201, 2000.

[116] C. J. Muise, S. A. McIlraith, and J. C. Beck. Improved non-deterministic planning by exploiting state relevance. In *ICAPS*, 2012.

[117] N. J. Nilsson. Shakey the robot. Technical Report 323, SRI Artificial Intelligence Center, 1984.

[118] M. Peot and D. E. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Proc. 1st Int. Conf. on AI Planning Systems*, pages 189–197, 1992.

[119] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[120] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[121] F. Pommerening, M. Helmert, and B. Bonet. Higher-dimensional potential heuristics for optimal classical planning. In *AAAI*, pages 3636–3643, 2017.

[122] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Journal of AI Research*, 4:287–339, 1996.

[123] R. Reiter. *Knowledge in Action.* MIT Press, 2001.

[124] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177, 2010.

[125] J. Rintanen. Complexity of planning with partial observability. In *ICAPS*, pages 345–354, 2004.

[126] J. Rintanen. Distance estimates for planning in the discrete belief space. In *AAAI*, pages 525–530, 2004.

[127] R. Rosner. *Modular Synthesis of Reactive Systems.* PhD thesis, Weizmann Institute of Science, 1992.

[128] S. Russell, D. Dewey, and M. Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4), 2015.

[129] S. Sardiña and G. De Giacomo. Realizing multiple autonomous agents through scheduling of shared devices. In *ICAPS*, pages 304–312, 2008.

[130] S. Sardiña and G. De Giacomo. Composition of congolog programs. In *IJCAI*, pages 904–910, 2009.

[131] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the semantics of deliberation in Indigolog - from theory to implementation. *Ann. Math. Artif. Intell.*, 41(2-4):259–299, 2004.

[132] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the limits of planning over belief states under strict uncertainty. In *KR*, pages 463–471, 2006.

[133] S. Sardiña and N. D'Ippolito. Towards fully observable non-deterministic planning as assumption-based automatic synthesis. In *IJCAI*, pages 3200–3206, 2015.

[134] R. Seiger, S. Huber, and T. Schlegel. Toward an execution system for self-healing workflows in cyber-physical systems. *Software & Systems Modeling*, pages 1–22, 2016.

[135] J. F. Sequeda and D. P. Miranker. A pay-as-you-go methodology for ontology-based data access. *IEEE Internet Computing*, 21(2):92–96, 2017.

[136] S. Sohrabi, N. Prokoshyna, and S. McIlraith. Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, pages 319–334, 2009.

[137] M. Steinmetz and J. Hoffmann. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *Artif. Intell.*, 245:1–37, 2017.

[138] M. Tenorth and M. Beetz. Representations for robot knowledge in the knowrob framework. *Artif. Intell.*, 247:151–169, 2017.

[139] Á. Torralba, V. Alcázar, P. Kissmann, and S. Edelkamp. Efficient symbolic search for cost-optimal planning. *Artif. Intell.*, 242:52–79, 2017.

[140] J. Torres and J. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, pages 1696–1703, 2015.

[141] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee. A theory of synchronous relational interfaces. *ACM Trans. Program. Lang. Syst.*, 33(4):14:1–14:41, 2011.

[142] M. Tsai, S. Fogarty, M. Vardi, and Y. Tsay. State of büchi complementation. *Logical Methods in Computer Science*, 10(4):1–27, 2014.

[143] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.

[144] M. Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In *CAV*, 1995.

[145] M. Y. Vardi. From church and prior to PSL. In *25 Years of Model Checking - History, Achievements, Perspectives*, pages 150–171, 2008.

A Ethics