Dipartimento di Ingegneria e Scienze
dell'Informazione e Matematica
Università degli Studi dell'Aquila
Via Vetoio, I-67100 L'Aquila, Italy

http://www.disim.univaq.it

Ph.D. Thesis in Computer Science

Dottorato di Ricerca in Informatica e Applicazioni - XXXII Ciclo

SSD: INF/01

Logical Agents: Memory Management, Advanced Architectures and Applications

Ph.D. Student
Valentina Pitoni

Ph.D. Program Supervisor
Prof. Vittorio Cortellessa

Advisor
Prof.ssa Stefania Costantini

CoAdvisor
Prof. Emiliano Lorini

A.A. 2018/2019

To my grandparents, my family and my friends

ABSTRACT

In this thesis I show my work of the past 3 years. I have covered different topics and the main areas are:

- Modal, Temporal and Metric Temporal Logic and Epistemic Logic;
- Cyber-Physical Systems, Multi-Context Systems, Component-based Agents Environments;
- Machine Ethics.

The primary objective of my research has been the study of autonomous system which evolves over time and their formal treatment in Computational Logic. First of all I show how I manage agents' memory through a particular modal logic. I have started to study this topic from the Master thesis and I carried on the work introducing various extensions that I will describe in the first chapter.

In the second chapter I show my last work which is a "work in progress". It is about a particular logic which is similar to the one explained in the first chapter, but I underline the concepts of "steps" and "executability". These two characterize agents' action.

In the third chapter I show my other research areas: Cyber-Physical Systems. Here I focus on work about a particular, innovative kind of architecture called *K-Level ACE*, which has a fractal structure and than can manage sudden failures.

In the forth chapter I show the last topic of my research activities: Machine Ethics, which is a part of ethics of Artificial Intelligence concerned with the moral behavior of artificial intelligent beings. Important aspect of Machine Ethics are *trustworthiness* and *safety*. We accomplished this aspects through *verification* and *assurance*. We propose technique for Runtime self checking and Monitoring using meta-rules and runtime constraints.

"In life smile always because no one is so important to take your smile away!"

Jim Morrison

"Non permettere a nessuno di toglierti il sorriso!"

Cristina D'Avena

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

MEMORY MANAGEMENT: TIME DYNAMIC LOGIC OF EXPLICIT BELIEF AND KNOWLEDGE

1.1 INTRODUCTION

In this chapter I illustrate the central part of my Thesis, describing how we manage agents' memory through a particular modal logic. First of all we try to identify the context in which we are: *Agent's Memory*. Memory in an agent system can be seen as a process of reasoning that not only just adds/deletes new facts but in particular, it is a learning process of strengthening a concept. The interaction between an agent and the environment plays an important role in constructing its memory and may affect its future behaviour, the latter due to the proactive and deliberative capabilities of the agent themselves. In fact, through memory an agent is potentially able to recall and to learn from experiences so that its beliefs and its future course of action are grounded in these experiences. Most of the methods for designing agent memorization mechanisms were inspired by the human memory models [131, 121] developed in cognitive sciences.

Figure 1.1: Memory

Recently, *cognitive architectures* have been defined and implemented; this kind of architectures are software systems that implement psychological theories about how our brain works (mental processes) and are used to create intelligent agents of cognitive inspiration. The main objectives of cognitive architectures are:

- *decision making*, i.e., the ability to devise decisions based on the knowledge of the world relative to a given task;
- *environment interaction*, i.e., the ability to acquire and interpret stimula provided by the environment;
- *simulate* the structure and the way of thinking of a human brain.

Atkinson and Shiffrin in [12] proposed a model of human memory which consists of three distinct memory stores, the sensory register where information are stored which are detected from senses, the short term memory (or working memory) where explicit beliefs are stored and the long term memory which stores the background knowledge; information passes from store to store in a linear way. This model has been further enhanced by Gero and Liew in [118], and [94] for constructive memory. Memory construction occurs whenever an agent uses past experience in the current environment in a situated manner. The exploitation of "memories" requires the interaction among this different memory components. Such correlation can be obtained in various ways, e.g., via neural networks, via mathematical models or via logical deduction.

Moreover, the most important cognitive architecture is SOAR (State, Operator And Result), which was created by Laird, Newell and Rosenbloom [115] at Carnegie Mellon University in 1983. SOAR has been used in a wide range of practical applications and it is often used as a tool for creating cognitive models that bring evidence and provide representation of aspects of human behavior. In SOAR, it is assumed that every problem can be reformulated as a *Problem Space*. A problem space is defined as a set of (possible) states and a set of operators that transform each state within the space problem into another state. There is usually an initial state and a desired state, or goal. The operators are selected iteratively and applied tentatively to achieve the goal state. The series of steps from the initial state to the desired state form the solution path or behavior.

To give meaning to our everyday experience of how the world works, we need to use our knowledge about objects and actions for the pursuit of our goals. For example, one knows a lot things about cups, but uses only a part of this knowledge to prepare a cup of coffee, and another part when looking for a place where to put pens and pencils. Moreover, once some specific item of knowledge has been applied, a cup will be filled with either coffee or pens. The ability to reason on cups in general as pencil holders rather than coffee cups will not be changed, but there will be specific objects in the present situation that have assumed a single role. This dichotomy between general knowledge and specific application of this knowledge is captured in SOAR by the existence of two different memory structures. The knowledge that exists independently of the current environment is called "*Long-Term*

Memory" (LTM). The situation determined by the application of part of such knowledge is put in the "*Working Memory*" (WM).

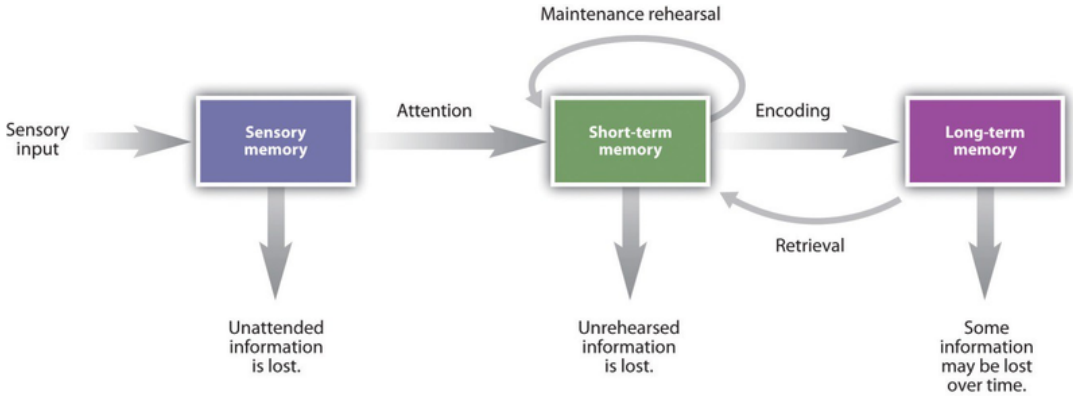

Figure 1.2: Short-term Memory and Long-term Memory

It is useful to think about the long-term memory as containing what may be true in general (for example, "coffee cups can hold pencils"), and the working memory containing what the model found to be true in a particular situation (for example, the cup in front of me contains coffee). In fact, the working memory structure is formed by the current objective (goal), the problem space, the current state, as well as the values and the characteristics that symbolize the state's contents. Perceptions of the world come through the perception model and are saved in the working memory in the form of symbolic structures. Furthermore, the working memory can be viewed as a short-term memory, which issues calls to the long-term memory for the recovery of information and knowledge, but also for learning. The long-term memory can affect the working memory. The data in both memories can be retrieved through specific requests from the working memory, which assumes the role of global workspace. This has the objective of defining the behavior of an agent, where behaviors are represented and generated in the long-term memory.

In their paper [16], Balbiani, Fernández-Duque and Lorini proposed a (partial) formalization of the SOAR architecture in modal logic, reasoning on a particular type of agents: "resource bounded" agents, which are agents who have limited memory and can take one step of inference at a time. They proposed a new logic called DLEK (Dynamic Logic of Explicit Beliefs and Knowledge) which helps clarifying how a non-omniscient resource-bounded agent can form new beliefs either through perception or through inference from existing knowledge and beliefs. In this logic programs are mental operations either of perceptive type or of inferential type having effects on the epistemic states of the resource bounded agent. In fact, DLEK is the first logical theory of relationship between explicit beliefs and background knowledge, from both the static and dynamic perspective which is reflected in its formal semantics and axiomatics. DLEK uses a constructive approach to explicit beliefs: this distinguishes it from the existing logics of time-bounded reasoning which represent reasoning as the process that requires time (see [3], [98]). These logics do not include mental operations of perceptive and inferential type as primitives in the object language of the logic. Instead, DLEK includes mental operations in its object language which can be used to reason about consequences of a sequence of perceptive and

inferential steps on the epistemic state of the agent.

DLEK has however no notion of time, while agents' actual perceptions are inherently timed and so are many of the inferences drawn from such perceptions. So, in [61] we have introduced explicit time instants and time intervals in formulas. We extended LEK/DLEK to TLEK/TDLEK, and I describe these logics in the first part of this chapter. But, to avoid problems with the management of the intervals and in order to not lose the logic of the formalization, in [70] we present an extension of LEK/DLEK to T-LEK/T-DLEK ("Timed LEK" and "Timed DLEK") obtained by introducing a special function which associates to each belief the arrival time and controls timed inferences. Through this function it is easier to keep the evolution of the surrounding world under control and the representation is more complete. And also, we work on *Belief Revision*, because agents interact with the environment so they have to change their belief according to the actual word. Moreover in [134] we consider this T function like a *temporal module* which can be adopted to "temporalize" many logical framework. We have exploited this module in two different settings. The first one is the memory management, which we have just talk about, and the second is a logical framework for reasoning about agents' cognitive attitudes; many formal logics have been proposed for reasoning about concepts taken from qualitative decision theory. Lorini in [122] proposes a general logical framework for reasoning about agents' cognitive attitudes of both epistemic type and motivational type.

The issue of time in agents has been coped with in several other works, (see, e.g., among many, [19, 129, 125, 44, 10], where however the objective is that of dealing with time in communication and coordination among agents; thus, our attempt to deal with time in memory management is a novelty in the literature.

1.2 Background

In this section, we recall briefly notions from the literature that define some basic foundation elements of the proposed approach.

1.2.1 Modal Logic, Linear Temporal Logic and Metric Temporal Logic

In recent decades logic has found numerous new information technology applications, especially in Artificial Intelligence, such as knowledge representation, expert systems, automated reasoning, systems specifications, programming, and many others. However, many important aspects of software are inherently dynamic, which implies the need to manage the evolution of the system state. First order logic is not suitable to do that, and this lead to the introduction of an extension of modal logic, the so-called Linear Temporal Logic (LTL). Linear Temporal Logic is important because it is used in the analysis of

dynamical systems, and in particular to define and verify the properties of such systems. Moreover, LTL provides explainability and potential validation.

MODAL LOGIC: Modal logic describes a set of states and the relationships between them: it extends classical logic through a set of modal operators. This kind of logic stems from the analysis on the propositions containing expressions like "necessary" and "possible" made by Aristotle. In this sense we can distinguish between propositions according to the fact that they may become true necessarily, possibly or contingently. These three modes are represented by:

- the operator $\Box$ stands for necessity;
- the operator $\Diamond$ stands for possibility;
- the absence of modal operator indicates contingency.

The two modal operators $\Box$ and $\Diamond$ are unary operators; i.e., if A is a formula, also $\Box A$ (A is necessary) and $\Diamond A$ (A is possible) are formulas. The definition of a formal semantics for modal operators was anticipated by the American philosopher and logician S. Kripke, who defined the semantics of "possible worlds" for modal logics. In this approach, modal formulas are to be interpreted in a set of possible worlds. The meaning of a formula (its truth / falsehood) can vary from a world to another. The worlds accessible from a world w represent states of things possible from the point of view of w. Consequently, $\Box A$ is true in w if and only if A is true in all the worlds that are reachable according to w. And, if some formula A is true in a world w^i and yet $\Box A$ not true in w, this is because w^i represents a state of things that it is not possible from the point of view of w. For more details on modal logic the reader may refer to [153], [84], [148] and to the references therein.

LINEAR TEMPORAL LOGIC AND METRIC TEMPORAL LOGIC: LTL is a linear logic modelling time via discrete instants. In this particular type of modal logic, the operators $\Box$ and $\Diamond$ are interpreted in time:

- $\Box$ A: A will always be true in the future;
- $\Diamond$ A: A will be true at some instant in future time.

There are also other temporal operators such as $\bigcirc$ ("Next") and U ("until"), that we do not use here. For more details, c.f. [135], [148].
In linear temporal logic, the approach to verify the execution of a system is modelled by a sequence of states or "events". This representation abstracts away from the precise time of the observations, only keeping their relative order. This approach is inadequate

to express the specifications for systems whose correct behavior depends on quantitative timing requirements; to cope with this deficiency, much work has been done to adapt linear temporal logic to the real-time setting. A real-time logic should have explicit references to time, usually recorded as timestamps associated to events in a computation. One of the oldest and most popular proposals for the extension of the temporal logic to the real-time setting is to replace the simple temporal operators with a constrained version w.r.t. time. This kind of logic is called *Metric Temporal Logic* (MTL), where LTL is extended tying the temporal operators to intervals of real numbers (limited or unlimited). For example, the formula "$\Diamond_{[3,4]}A$" means that A will become true within 3 to 4 units of time from now. In this work, we consider MTL under pointwise semantics and time points expressed as natural numbers, which is known to be decidable and where satisfiability and model-checking have been proved to be EXPTIME-complete [103, 130]. For more details refer also to [114].

1.2.2 Dynamic Epistemic Logic and DLEK

In their work , Balbiani, Fernández-Duque and Lorini, in accordance with existing theories, assume that an agent has two types of memory, the long term (LTM) and the short-term memory, also called working memory (WM). They also contain different types of information:

- The "*Background Knowledge*": general knowledge (events of the past, rules describing the surrounding world, etc.) contained in long-term memory;
- "*Belief*": represent everything that the agent believes in a determined instant of time and are formed by perceptions. The formation of believes by perceptions adds a new information to the set of beliefs that are under the attention of the agent and therefore are part of the short-term memory. An agent can also use belief as a basis for an inference that leads to the formation of a new belief. In many cases, the formation of new beliefs by inference requires retrieving information from long-term memory. Furthermore, information in the working memory can be:
  - deleted: an agent, no longer needing certain beliefs, is able to forget them;
  - moved to the long-term memory: an agent can consider that certain information will be useful later, so it stores them in the long-term memory.

To formalize this system, they used a particular kind of *Dynamic Epistemic Logic* called *Dynamic Logic of Belief and Knowledge* (DLEK). The Dynamic Epistemic Logic was proposed by Jaako Hintikka in his book published in the fundamental Knowledge and Belief, 1962. However, the general study of the formal semantics for knowledge and belief began to really flourish in 1990 with fundamental contributions by computer scientists [124] and game theorists [13, 28]. As a consequence, the field of epistemic logic focuses on epistemic issues of the game theory [34], information security [101, 136] and

distributed systems and multi agents [100, 155]. Considering our field of interest, the Dynamic Epistemic Logic (DEL) is a particular type of Modal Logic which studies how agents update their knowledge and change their beliefs on the basis of new information exchanged in various ways (communication with other agents, perceptions from the outside world, etc.). One of the key features of epistemic logic is that the state of different agents can be represented by a Kripke model. In epistemic logic, the set of states of a Kripke model is interpreted as a set of epistemic alternatives. The state of an agent consists of those epistemic alternatives that are possible from the perspective of the agent; they are represented by the binary accessibility relation R_i. An agent i knows that the proposition φ is true in a state of a Kripke model M if and only if that proposition φ is true in all the states that the agent examines possible in that state (which are accessible through R_i). For more details, refer to [156] and to references therein.

Balbiani, Fernández-Duque and Lorini introduced DLEK in order to clarify how an agent with bounded resources and not omniscient can form new beliefs through perception and through inferences from existing information. In particular, they introduced rules that describe mental operations, both perceptual and inferential, which have effects on the epistemic states of the agents. The main difference between DLEK and the existing logics is that it provides a constructive theory of belief and, thanks to the perceptual steps and inferential knowledge, an agent can create new believes. More precisely, DLEK is a logic that consists of a static component and of a dynamic one. The static component, called LEK, is a Dynamic Epistemic Logic; instead the dynamic component extends the static with dynamic operators that capture the consequences of mental operations on agents' believes.

1.3 Temporalizing DLEK Logic: TDLEK

We introduce TDLEK starting from LEK and DLEK. We will illustrate the new logic by difference from LEK and DLEK while emphasizing the aspects that we modify/extend.

We may notice that most of the time-based logics follow the seminal work of [152]. In Metric Temporal Logic (MTL) [114] the modalities of LTL are augmented with timing constraints; thus, expressions of the form $\Box_I$ and $\Diamond_I$ are introduced, where I is a *time interval* which, under the *Pointwise Semantics*, can either be finite or diverge to infinity. The works of [102, 18] cope with time intervals by introducing modalities for every possible relationships among intervals and also among points and intervals. In this work we consider MTL under pointwise semantics and time points expressed as natural numbers, which is known to admit decidable versions. Notice however that, in our approach, a time-stamp associated to an atom is intended as the time instant in which an agent has perceived the event denoted by the atom, where a time interval I associated to a formula $\Box_I\varphi$ delimits the period of time in which the truth of the formula is to be evaluated.

1.3.1 TLEK AND TDLEK

We consider an underlying discrete linear model of time and identify time instants with natural numbers and time intervals with intervals in $\mathbb{N}$. More in general, we consider any arithmetic expression e over natural numbers, with value $v_e \in \mathbb{N}$, as denoting the time instant v_e. For simplicity, in what follows, we identify each expression e with its value v_e. A "time-interval" is a closed finite interval $[\ell, u]$ or an infinite interval $[\ell, \infty)$ (considered open on the upper bound), for any expressions/values ℓ, u such that $0 \leq \ell \leq u$. Given $I_1 = [t_1, t_2]$ and $I_2 = [t_3, t_4]$ (where both t_2 and t_4 can be ∞) we indicate as $I_1 \uplus I_2$ the unique smallest interval including both I_1 and I_2.

As it is customary in logic programming, we assume some signature specifying (countable) sets of *predicate*, *function*, and *constant* symbols. From constant and function symbols, compound terms are built as usual. The *Herbrand universe* is the collection of all such terms (which includes constant symbols). We assume that the integer numbers and the symbol ∞ are included among the constant symbols and that the arithmetic operators are included among the function symbols. Consequently, arithmetic expressions are terms on the signature. Atoms have the form $pred(\tau_1, \ldots, \tau_n)$ where $pred$ is a predicate symbol, $n \geq 0$ is its arity and $\tau_1, \ldots, \tau_n$ are terms. We denote by $Atmg$ the countable set of atoms on the signature (i.e., the *Herbrand base*).

A *time-stamped* (or, briefly, *timed*) atom p_t is an atom p annotated with a time instant, called *time-stamp*, t. We denote by $Atmtg$ the set of all atoms and time-stamped atoms (note that $Atmg \subseteq Atmtg$). For an atom p, by p_I with $I = [\ell, u]$, we mean the conjunction $p_\ell \wedge\ p_{\ell+1} \wedge \cdots \wedge\ p_u$. In this frame of mind, we will often denote the time-stamped atom p_t as p_{I_t} with $I_t = [t, t]$. A plain atom p stands for $p_{[0,\infty)}$.

Below is the definition of the formulas of the language $\mathcal{L}_{TLEK}$. With a slight abuse of notation, in this grammar we use I as terminal symbol standing for time intervals (possibly specified through arithmetic expressions, as said earlier)

$$\Phi \quad := \quad p_I \mid \neg\Phi \mid \Box_I\, \Phi \mid B_i\, \Phi \mid K_i\, \Phi \mid \Phi\ \vee\ \Psi \mid \Phi\ \wedge\ \Psi \mid \Phi\ \rightarrow\ \Psi$$

Others Boolean connectives $\top$, $\bot$, $\leftrightarrow$ are defined from $\neg$ and $\wedge$ as usual. Moreover, for simplicity, we consider q_I equivalent to $\top$ when $I = \emptyset$. In the formula $\Box_I\, \Phi$ the MTL Interval "always" operator is applied to a formula; I can be $[0, \infty)$ and $\Box_{[0,\infty)}$ will sometimes be written simply as $\Box$. The operator B_i is intended to denote belief and the operator K_i to denote knowledge. Both refer to agent i belonging to a finite set $Ag = \{1, \ldots, k\}$ of agents.

Given a formula Φ, we denote by $span(\Phi)$ the interval so defined:

$$span(\Phi) = \begin{cases} I & \text{if } \Phi \text{ has the form } \Box_I\, \Phi_1 \text{ or } p_I \\ span(\Phi_1) & \text{if } \Phi \text{ has the form } \neg\Phi_1 \text{ or } B_i\, \Phi_1 \text{ or } K_i\, \Phi_1 \\ span(\Phi_1) \uplus span(\Phi_2) & \text{if } \Phi \text{ has the form } \Phi_1\ op\ \Phi_2 \text{ for } op \in \{\vee, \wedge, \rightarrow\} \end{cases}$$

Terms/atoms/formulas as defined so far are *ground*, namely there are no variables occurring therein. We introduce variables and use them in formulas in a restricted manner, as

usual for example in answer set programming [92]. Variables can occur in formulas in any place constants can occur and are intended as placeholders for elements of the Herbrand universe. More specifically, a ground *instance* of a term/atom/formula involving variables is obtained by uniformly substituting ground terms to all variables (*grounding* step), with the restriction that any variable occurring in an arithmetic expression (i.e., specifying a time instant) can be replaced by a (ground) arithmetic expressions only. Consequently, a non-ground term/atom/formula represents the possibly infinite set of its ground instances, namely, its *grounding*. As it is customary in logic programming, variable symbols are indicated with an initial uppercase letter whereas constants/functions/predicates symbols are indicated with an initial lowercase letter. We denote by Atm and $Atmt$ the collections of all non-ground atoms and non-ground time-stamped atoms, respectively. In the rest of the paper, unless differently specified, we deal with ground terms/atoms/formulas.

Example 1 *An example of a non-ground TLEK formula is:*

$$K_i(\Box_{[t_1,t_2]}\ (send\_registration\_form_T\ \rightarrow \Box_{[T,T+14]} send\_payment_{T_1})),$$

where we suppose that an agent knows that it is possible to register to a certain conference in the period $[t_1, t_2]$ and that, after sending the registration form, the payment must be sent within fourteen days (still staying within the interval $[t_1, t_2]$). Since, by the restrictions on formulas stated earlier, it must be the case that $T_1 \in [T, T + 14]$ and both T, $T + 14$ must be in $[t_1, t_2]$, only a finite set of ground instances of this formula can be formed by substituting natural numbers to the variables T, T_1 (specifically, the maximum number of ground instances is $t_2 - t_1 - 14 + 1$ assuming to pay on the last day t_2). In case one would consider the more general formula

$$K_i(\Box_{[t_1,t_2]}\ (send\_registration\_form(X)_T\ \rightarrow \Box_{[T,T+14]} send\_payment(X)_{T_1})),$$

where X represents a member of some department, i.e., $department\_member(X)$ holds for some ground instance of X, then the set of ground instances would grow, as a different instance should be generated for each department member (i.e., for each ground term replacing X). In practice, however, ground instances need not to be formed a priori, but rather they can be generated upon need when applying a rule; in the example, just one ground instance should be generated when some member of the department intends to register to that conference at a certain time $T = \hat{t}$. ■

Remark 1.3.1 *As we will see, rules of the form $K_i(\Box_I(\varphi \rightarrow \psi))$ are a key feature of LEK and TLEK; they are supposed to be kept in the long-term memory and allow other beliefs to be derived from former ones. This derivation is performed in the working memory, which is where beliefs are kept. In the above example, if the agent believes to have sent a registration form, via the $K_i(\ldots)$ rule it will consequently infer to believe to have to send the payment within due time.*

Interaction between long-term and short-term memories and thus derivation of new beliefs is not automatic, it is rather performed by an agent whenever deemed necessary, by

means of invocation of an explicit, we might say “conscious", mental operator.
The language $\mathcal{L}_{TDLEK}$ of Temporalized DLEK (TDLEK) is obtained by augmenting $\mathcal{L}_{TLEK}$ with the expression $[\alpha]\,\psi$, where α denotes a *mental operation* and ψ is a ground formula.[1] The mental operations that we consider are essentially the same as in [16], though applied to ground formulas.
Notice that the rational of considering ground formulas is that they represent perceptions (either new or already recorded in agent's memory) coming in general from the external world (we say “in general” as, in fact, in some of the aforementioned agent-oriented frameworks perceptions can also result from *internal events*, i.e., from an agent's observations of its own internal activities).
We assume that an agent has “time accurate” perceptions, for example $rain_{[t1,t2]}$ is perceived with precise time instants rather than considering $rain_{[T_1,T_2]}$ which might signify that the agent perceived rain in an unspecified interval. The cases of imprecision, because an agent might “not remember” or might have been told a vague fact by someone else, are not considered here.
These are the four mental operations of interest:

- $+\varphi$, where φ is a ground formula of the form p_I or $\neg p_I$, for a timed atom p_I. This mental operation serves to form a new belief from a perception φ. A perception may become a belief whenever an agent becomes “aware” of the perception and takes it into explicit consideration. Notice that φ may be a negated atom. In fact, perceptions may concern facts that hold (e.g., 'it rains') or do not hold ('it does not rain').

- $\vdash(\varphi, q_J)$, where q_J is a ground atom. An agent, believing that a ground formula φ is true and having in its long-term memory that φ implies q_J (in some suitable time interval including J), starts believing that q_J is true.

- $\cap(\varphi, \psi)$: believing both ground formulas φ and ψ, an agent starts believing their conjunction.

Note that we do not consider the mental operation $-\varphi$ as formulated in [16], which represents arbitrary “forgetting”, i.e., removing a belief from the short-term memory. In fact, we assume that simple forgetting can be performed, e.g., based upon the time-stamps. We substituted $-\varphi$ with this mental operation:

- $\dashv(p_{I_1}, q_{I_2})$ where p_{I_1} and q_{I_2} are ground atoms. An agent, believing p_{I_1} and having in the long-term memory that p_{I_1} implies $\neg q_{I_2}$ removes the belief q (in some interval possibly strictly included in I_1 and I_2, see below).

[1] Recall that, thanks to grounding, variables can be used to denote collections of such expressions.

1.3.2 TLEK AND TDLEK SEMANTICS

Semantics of DLEK and TDLEK are both based on a set W of worlds. But whereas in DLEK a valuation function $V : W \rightarrow 2^{Atm}$ is used, in TDLEK we define the valuation function on the sets of ground time-stamped atoms: $V : W \rightarrow 2^{Atmtg}$.
For a world w, let t_1 the minimum time-stamp of atom $p_{t_1} \in V(w)$ and let t_2 be the supremum (we can have $t_2 = \infty$) among all time-stamps of atoms in $V(w)$. Then, whenever useful, we denote w as w_I where $I = [t_1, t_2]$. Moreover, we denote by $V_t(w)$ the set of atoms in $V(w)$ having t as time-stamp. Similarly, $V_J(w)$ is the set of atoms with time-stamps in the interval J.
The notion of LEK/TLEK model does not consider mental operations, discussed later, and is introduced by the following definition.

Definition 1.3.1 *A* TLEK model *is a tuple* $M{=}\langle W; N; \{R_i\}_{i \in Ag}; V\rangle$ *where:*

- W *is the set of worlds;*
- $R_i \subseteq W \times W$ *is the accessibility relation, required to be an equivalence relation. For all* $i \in Ag$ *and* $w_I \in W$ *we put* $R_i(w_I) = \{v_I \in W \mid w_I R_i v_I\}$ *called* epistemic state of agent i in w_I.
- $N : Ag \times W \rightarrow 2^{2^W}$ *is a* "neighbourhood" *function.* $\forall i \in Ag$ *and* $\forall w_I \in W$, $N(i, w_I)$ *defines, in terms of sets of worlds, what agent* i *is allowed to explicitly* believe *in the world* w_I*;* $\forall i \in Ag$, $w_I, v_I \in W$, *and* $X \subseteq W$*:*
  1. *if* $X \in N(i, w_I)$*, then* $X \subseteq R_i(w_I)$*: each element of the neighbourhood is a set composed of reachable worlds; i.e., agent* i *may have among its beliefs only those which are compatible with the current epistemic state;*
  2. *if* $w_I R_i v_I$*, then* $N(i, w_I) \subseteq N(i, v_I)$*: if the world* v_I *is compliant with the epistemic state of world* w_I*, then agent* i *in the world* w_I *should have a subset of beliefs of the world* v_I*.*
- $V : W \rightarrow 2^{Atmtg}$ *is the valuation function.*

The epistemic state of agent i in w_I indicates all the situations that agent i considers possible in the world w_I or, equivalently any situation the agent can retrieve from long-term memory based on what it knows in world w_I.

It can be observed that R_i is required to be an equivalence relation so as to model omniscience in the background knowledge. Since indistinguishable worlds must have the same time span, then an agent always knows the time interval it is in; this is in accordance with omniscience in the long-term memory and is usually the case in practical agent systems that we aim to model, where all events are time-stamped.

Truth conditions for ground TDLEK formulas are defined inductively as follows, where the difference from [16] consists in:

- the entailment of timed atoms;
- considering the $\Box_I$ operator;
- introducing extended definitions for mental operations.

We first consider ground formulas. Semantics of ground TLEK formulas Φ is specified by the following definition.

Definition 1.3.2 *Given a TLEK formula Φ, a TLEK model $M = \langle W, N, \{R_i\}_{i \in Ag}, V \rangle$ and $w_I \in W$. Then, $M, w_I \models \Phi$ if one of the following conditions holds:*

- $\Phi = p_J$ *and* $\forall t \in J$, $p_t \in V_t(w_I)$;
- $\Phi = \neg\varphi$ *and* $M, w_I \not\models \varphi$;
- $\Phi = \varphi \wedge \psi$ *and* $M, w_I \models \varphi$ *and* $M, w_I \models \psi$;
- $\Phi = \varphi \vee \psi$ *and* $M, w_I \models \varphi$ *or* $M, w_I \models \psi$;
- $\Phi = \varphi \rightarrow \psi$ *and* $M, w_I \models \neg\varphi$ *or* $M, w_I \models \psi$;
- $\Phi = \Box_J \varphi$ *and* $span(\varphi) \subseteq J \subseteq I$ *and for all* $v_I \in R_i(w_I)$, *it holds that* $M, v_I \models \varphi$;
- $\Phi = K_i\,\varphi$ *and for all* $v_I \in R_i(w_I)$, *it holds that* $M, v_I \models \varphi$;
- $\Phi = B_i\,\varphi$ *and* $\| \varphi \|^M_{i,w_I} \in N(i, w_I)$, *where*

$$\| \varphi \|^M_{i,w_I} = \{v_I \in W \mid M, v_I \models \varphi\} \cap R_i(w_I).$$

Considering formulas of the form $B_i\,\varphi$ and $K_i\,\varphi$, observe that $M, w_I \models B_i\,\varphi$ holds if the set $\| \varphi \|^M_{i,w_I}$ of worlds reachable from w_I which entail φ in the very same model M belongs to the *neighbourhood* $N(i, w_I)$ of w_I. Hence, knowledge pertains to formulas entailed in model M in every reachable world, while beliefs pertain to formulas entailed only in some set of them, where this set must however belong to the neighbourhood and so it must be composed of reachable worlds. Thus, an agent is seen as omniscient with respect to knowledge, but not with respect to beliefs.

The following properties are immediate consequences of Definition 1.3.2.

Property 1.3.1 *If φ is a TLEK formula φ and I_1, I_2 are intervals such that $span(\varphi) \subseteq I_1$, then*

1. $\Box_{I_1 \cup I_2}\,\varphi \rightarrow \Box_{I_1}\varphi$,
2. $\Box_{I_1}\,\varphi \rightarrow \varphi$.

Proof.

1. Let $M = \langle W, N, \{R_i\}_{i \in Ag}, V \rangle$ and $w_I \in W$, such that $M, w_I \models \Box_{I_1 \cup I_2} \varphi$. Then, by definition, $span(\varphi) \subseteq I_1 \cup I_2 \subseteq I$ and for all $v_I \in R_i(w_I)$, it holds that $M, v_I \models \varphi$. Plainly, since $span(\varphi) \subseteq I_1 \subseteq I$, it follows that $M, w_I \models \Box_{I_1} \varphi$.

2. By the previous point, $\Box_{I_1} \varphi \rightarrow \Box_{span(\varphi)} \varphi$ holds. Let $M = \langle W, N, \{R_i\}_{i \in Ag}, V \rangle$ and $w_I \in W$, such that $M, w_I \models \Box_{span(\varphi)} \varphi$. This holds iff $span(\varphi) \subseteq I$ and for all $v_I \in R_i(w_I)$ it holds that $M, v_I \models \varphi$. Since R_i is an equivalence relation, $w_I \in R_i(w_I)$. Hence $M, w_I \models \varphi$. Therefore, $\Box_{span(\varphi)} \varphi \rightarrow \varphi$, which concludes the proof. ■

As concerns non-ground TLEK formulas Φ, recall that the ground instances $\hat{\Phi}$ of Φ are obtained by uniformly substituting all variables with constants or expressions/values, in all possible ways. Hence, we define the semantics of a non-ground formula Φ by putting $M, w_I \models \Phi$ iff $M, w_I \models \hat{\Phi}$ for all ground instances $\hat{\Phi}$ of Φ.

1.3.3 SEMANTICS FOR MENTAL OPERATIONS

Let $M = \langle W, N, \{R_i\}_{i \in Ag}, V \rangle$ be a TLEK model and $w_I \in W$. Concerning formulas $[\alpha] \varphi$ of TDLEK involving a mental operation α performed by any agent i, semantics must be specified by considering that the mental operation α affects the sets of agent's beliefs and that, after α has been performed, the truth value of φ has to be established with respect to a modified model M^α. Such model differs from M in the neighbourhood component N^α (replacing N, see below) which records the changes in agent's beliefs caused by the mental operation. In particular, such operation can add new beliefs by direct perception, by means of one inference step, or as a conjunction of previous beliefs. We have the following definition.

Definition 1.3.3 *Let $M = \langle W, N, \{R_i\}_{i \in Ag}, V \rangle$ be a TLEK model, $w_I \in W$. Let moreover $[\alpha] \varphi$ be a TDLEK formula involving a mental operation α performed by any agent i. We put:*

$$M, w_I \models [\alpha] \varphi \quad \textit{iff} \quad M^\alpha, w_I \models \varphi \ \textit{ where } M^\alpha = \langle W; N^\alpha(i, w_I); \{R_i\}_{i \in Ag}; V \rangle \tag{1.1}$$

The manner in which the neighbourhood $N(i, w_I)$ is extended to obtain the new neighbourhood $N^\alpha(i, w_I)$, depending on the specific α, is as follows.

- Learning perceived belief:

$$N^{+\ell_J}(i, w_I) = N(i, w_I) \cup \left\{ \parallel \ell_J \parallel^M_{i, w_I} \right\} \text{ with } J \subseteq I. \tag{1.2}$$

  The agent i adds to its beliefs event/perception ℓ (namely, an atom or the negation of an atom) perceived in J; the neighbourhood is expanded to as to include the set composed of all the reachable worlds which entail ℓ_J in M.

- Belief inference:

$$N^{\vdash(\varphi,q_J)}(i,w_I) = \begin{cases} N(i,w_I) \cup \{ \parallel q_J \parallel^M_{i,w_I} \} & \text{if } M, w_I \models \big(B_i(\varphi) \wedge K_i\big(\Box_I(\varphi \rightarrow q_J)\big)\big) \\ & \text{and } span(q_J) \subseteq I \\ N(i,w_I) & \text{otherwise} \end{cases} \tag{1.3}$$

  The agent i adds the ground atom q_J as a belief in its short-term memory if it has φ among its previous beliefs and has in its background knowledge $K_i(\Box_I(\varphi \rightarrow q_J))$, where all the time stamps occurring in φ and in q_J belong to I. Observe that, if I does not include all time instants involved in the formulas, the operation does not succeed and thus the set of beliefs remains unchanged. If the operation succeeds then the neighbourhood is modified by adding q_J as a new belief.

- Beliefs conjunction:

$$N^{\cap(\varphi,\psi)}(i,w_I) = \begin{cases} N(i,w_I) \cup \{ \parallel \varphi \wedge \psi \parallel^M_{i,w_I} \} & \text{if } M, w_I \models B_i(\varphi) \wedge B_i(\psi) \\ & \text{and } span(\varphi \wedge \psi) \subseteq I \\ N(i,w_I) & \text{otherwise} \end{cases} \tag{1.4}$$

  The agent i adds $\varphi \wedge \psi$ as a belief if it has among its previous beliefs both φ and ψ, with I including all time instants referred to by them; otherwise, the set of beliefs remain unchanged. The neighbourhood is expanded, if the operation succeeds, with those sets of reachable worlds where both formulas are entailed in M.

We write $\models_{TDLEK} \varphi$ to denote that φ is true in all worlds w_I, of every TLEK model M.

Remark 1.3.2 *We adopt expressions like $K_i \Box_I \varphi$, where in particular φ can be an implication, in order to represent knowledge in the long-term memory wherever applicability of such knowledge is time-dependent. The role of the $\Box$ operator, that from the above-stated truth conditions may seem superfluous, becomes apparent whenever interval extremes are defined by means of expressions over time instants; such correlations indicate that a certain implication makes sense only within a certain interval.* ■

Property 1.3.2 *For the mental operations previously considered we have the following (where φ, ψ are as explained earlier):*

1. $\models_{TDLEK} (K_i(\Box_I(\varphi \rightarrow \psi)) \wedge B_i\,\varphi) \rightarrow [\vdash(\varphi,\psi)]\, B_i\,\psi$ *in w_I if $span(\psi) \subseteq I$.*
   Namely, if an agent i has φ as one of its beliefs and has $K_i(\Box_I(\varphi \rightarrow \psi))$ in its background knowledge, then as a consequence of the mental operation $\vdash(\varphi,\psi)$ the agent i starts believing ψ;

2. $\models_{TDLEK} (B_i\varphi \wedge B_i\psi) \rightarrow [\cap(\varphi,\psi)] B_i(\varphi \wedge \psi)$ *in w_I if $span(\varphi \wedge \psi) \subseteq I$.*
   Namely, if an agent i has φ and ψ as beliefs, then as a consequence of the mental operation $\cap(\varphi,\psi)$ the agent i starts believing $\varphi \wedge \psi$;

3. $\models_{TDLEK_t} [+\varphi]B_i\varphi$ *in* w_I *if* $span(\varphi) \subseteq I$.
Namely, as a consequence of the operation $+\varphi$ *(thus after the perception of* φ*) the agent* i *adds* φ *to its beliefs.*

Proof.

1. By definition of the mental operation, ψ is assumed to be a ground timed atom q_J. Let $M{=}\langle W, N, \{R_i\}_{i\in Ag}, V\rangle$ and $w_H \in W$, such that $M, w_H \models (K_i(\Box_I(\varphi \rightarrow q_J)) \wedge B_i\,\varphi)$. We have to show that $M, w_H \models [\vdash(\varphi, q_J)]\, B_i\, q_J$. By (1.3), this holds iff $M^{\vdash(\varphi,q_J)}, w_H \models B_i\, q_J$, with $M^{\vdash(\varphi,q_J)}{=}\langle W, N^{\vdash(\varphi,q_J)}, \{R_i\}_{i\in Ag}, V\rangle$, where $N^{\vdash(\varphi,q_J)} = N(i, w_H) \cup \{\ \| \ q_J\ \|^M_{i,w_H}\ \}$, because $M, w_H \models (K_i(\Box_I(\varphi \rightarrow q_J)) \wedge B_i\,\varphi)$, by hypothesis. By Definition 1.3.2, $M^{\vdash(\varphi,q_J)}, w_H \models B_i\, q_J$ holds because $\| \ q_J\ \|^{M^{\vdash(\varphi,q_J)}}_{i,w_H} \in N^{\vdash(\varphi,q_J)}(i, w_H)$.

2. Let $M{=}\langle W, N, \{R_i\}_{i\in Ag}, V\rangle$ and $w_H \in W$, such that $M, w_H \models B_i\,\varphi \wedge B_i\,\psi$. We have to show that $M, w_H \models [\cap(\varphi, \psi)]B_i(\varphi \wedge \psi)$. By (1.4), this holds iff $M^{\cap(\varphi,\psi)}, w_H \models B_i(\varphi \wedge \psi)$, with $M^{\cap(\varphi,\psi)}{=}\langle W, N^{\cap(\varphi,\psi)}, \{R_i\}_{i\in Ag}, V\rangle$, where $N^{\cap(\varphi,\psi)} = N(i, w_H) \cup \{\ \| \ \varphi \wedge \psi\ \|^M_{i,w_H}\ \}$, because $M, w_H \models B_i\,\varphi \wedge B_i\,\psi)$, by hypothesis. By Definition 1.3.2, $M^{\cap(\varphi,\psi)}, w_H \models B_i(\varphi \wedge \psi)$ holds because $\| \ \varphi \wedge \psi\ \|^{M^{\cap(\varphi,\psi)}}_{i,w_H} \in N^{\cap(\varphi,\psi)}(i, w_H)$.

3. Let $M{=}\langle W, N, \{R_i\}_{i\in Ag}, V\rangle$ and $w_H \in W$, we have to show that $M, w_H \models [+\varphi]B_i\,\varphi$. By (1.2), this holds iff $M^{+\varphi}, w_H \models B_i\,\varphi$, with $M^{+\varphi}{=}\langle W, N^{+\varphi}, \{R_i\}_{i\in Ag}, V\rangle$, where $N^{+\varphi} = N(i, w_H) \cup \{\ \| \ \varphi\ \|^M_{i,w_H}\ \}$. By Definition 1.3.2, $M^{+\varphi}, w_H \models B_i\,\varphi$ holds because $\| \ \varphi\ \|^{M^{+\varphi}}_{i,w_H} \in N^{+\varphi}(i, w_H)$. ■

DEFEASIBLE BELIEFS

We propose a substantial modification to the definition of the operation $-\varphi$ of [16]; there, a belief was just arbitrarily removed. Here instead, we introduce the negative counterpart of the operation $\vdash(p_{I_1}, q_{I_2})$ (for two ground atoms p_{I_1} and q_{I_2}) namely, $\dashv(p_{I_1}, q_{I_2})$. Via this operation, an agent believing p_{I_1} and having in the long-term memory a rule $K_i(\Box_I(p_{I_1} \rightarrow \neg q_{I_2}))$ removes the timed belief q_{I_2}, if the intervals match (see below).
Notice that, should q be believed in a wider interval I_3 such that $I_2 \subseteq I_3$, the belief q is removed concerning intervals I_1 and I_2, but is left for the remaining sub-intervals (so, its is "restructured"). Hence, our extension to DLEK and TDLEK makes beliefs *defeasible*: a belief can be seen as a default which represents the current state of affairs in the agent's "world", that might be invalidated (entirely or in a sub-interval) by further perceptions/inferences.
Notice that, an application of the new operation may concern beliefs of the form $p_{[t,\infty)}$, that signify that the atom p will hold either indefinitely or "until" terminated by an other belief.
For example, $student_{[t,\infty)}$ or $married_{[t,\infty)}$ mean that one, after enrolling to a school or after marrying, will remain in the consequent state for an indefinite time; however, if graduating or, respectively, divorcing at time t_1, previous beliefs must be replaced by

$student_{[t,t_1-1]}$ or $married_{[t,t_1-1]}$.
Also, atom q may represent in many cases the "opposite" of p; e.g., if $p = door\text{-}open$ then q can be $door\text{-}closed$. Moreover, rules of above form in the long-term memory can also represent exceptions, e.g., a person is at home (q= *at-home*) for the whole day (I_2) and she is thus believed to be there; however, if it is also later on believed that she went out p= *go-out* in I_1, then the belief of her being at home remains before and after, but does not hold in I_1.

Example 2 *Let us consider the example of a door open or closed, where only agent i can perform the* action *to open or close the door. Let us assume that performed actions are recorded among an agent's perceptions, with the due time stamp. For reader's convenience, actions are denoted using a suffix "A". For simplicity, actions are supposed to always succeed and to produce an effect within one time instant. Let us consider the following rules (kept in long-term memory):*

$$K_i(\Box(open\text{-}doorA_T \rightarrow door\text{-}open_{[T+1,\infty]}))$$
$$K_i(\Box(close\text{-}doorA_T \rightarrow door\text{-}closed_{[T+1,\infty]})).$$

Let us now assume that the agent closes the door, e.g., at time 5; then, a belief will be formed of the door been closed from time 6 on; however, if the agent later opens the door, e.g., at time 8, as a consequence the door will result to be open from time 9 on. It can be seen that the application of previous rules in consequence of an agent's action of opening/closing the door determines some "belief restructuring" in the short-term memory of the agent. In absence of other rules concerning doors, we intend that a door cannot be believed to be simultaneously open and closed. The related belief update is determined by the following rules:

$$K_i(\Box(door\text{-}open_{[T,\infty]} \rightarrow \neg door\text{-}closed_{[T,\infty]}))$$
$$K_i(\Box(door\text{-}closed_{[T,\infty]} \rightarrow \neg door\text{-}open_{[T,\infty]}))$$

With the above timing, the result of their application is that the belief formed at time 5, i.e., $door\text{-}closed_{[6,\infty]}$ will be replaced by $door\text{-}closed_{[6,8]}$ plus $door\text{-}open_{[9,\infty]}$. ■

Definition 1.3.4 extends TDLEK truth conditions to encompass the new mental operation $\dashv(p_{I_1}, q_{I_2})$. Similarly to the other mental actions, $\dashv(p_{I_1}, q_{I_2})$ affects the sets of beliefs. Semantics takes this into account by modifying models' neighbourhood (provided that in the long-term memory we have $K_i(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2}))$ for some suitable interval I_3).

Definition 1.3.4 *Let p_{I_1} and q_{I_2} be ground atoms and let $M = \langle W, N, \{R_i\}_{i\in Ag}, V\rangle$ be a TLEK model and $w_I \in W$. We put*

$$M, w_I \models [\dashv(p_{I_1}, q_{I_2})]\varphi \quad \textit{iff} \quad M^{\dashv(p_{I_1},q_{I_2})}, w_I \models \varphi \tag{1.5}$$

where $M^{\dashv(p_{I_1},q_{I_2})} = \langle W; N^{\dashv(p_{I_1},q_{I_2})}(i,w_I); \{R_i\}_{i\in Ag}; V\rangle$ *with*

$$N^{\dashv(p_{I_1},q_{I_2})}(i,w_I)=\begin{cases} N(i,w_I) \setminus \{ \parallel q_{I_1\cap I_2} \parallel^M_{i,w_I} \} & \text{if } M,w_I \models B_i p_{I_1} \wedge B_i q_{I_2}, \\ & K_i\big(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2})\big) \text{ for } I_3 \subseteq I, \\ & \text{and there is no interval } J \supsetneq I_2 \\ & \text{such that } B_i q_J \\ N(i,w_I) & \text{otherwise} \end{cases} \tag{1.6}$$

In order to illustrate the effect of this mental operation, let us consider the special case in which q is believed to hold only in the interval I_2 and the agent i has the perception of p in $I_1 \subseteq I_2$. Then, the agent replaces previous belief q in the short-term memory with $q_{I_2\setminus I_1}$. Note that, in writing $q_{I_2\setminus I_1}$, we applied some abuse in notation. Indeed, in general, the set $I_2 \setminus I_1$ is not necessarily an interval: being $I_1 \subseteq I_2$, with $I_1{=}[t_0,t_1]$, and $I_2{=}[t_2,t_3]$, we have that $I_2 \setminus I_1 = [t_2, t_0-1] \cup [t_1+1, t_3]$. Thus, q_{I_2} is replaced by $q_{[t_2,t_0-1]}$ and $q_{[t_1+1,t_3]}$ (and similarly if $t_3 = \infty$).

We have the following property, which intuitively means: if an agent i has q_{I_2} as one of its beliefs, q is not believed outside I_2, the agent perceives p in $I_1 \subseteq I_2$, and has $K_i(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2}))$ in its background knowledge. Then after the mental operation $\dashv(p_{I_1}, q_{I_2})$ the agent i starts believing $q_{I_2\setminus I_1}$

Property 1.3.3 *Given two ground atoms p_{I_1} and q_{I_2}, with $I_1 \subseteq I_2$. Assume there is no interval J such that $J \supsetneq I_2$ or $J \cap I_2 = \emptyset$ and such that $\models_{TDLEK} B_i q_J$. Then, the following holds:*

$$\models_{TDLEK} (K_i(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2})) \wedge B_i\, p_{I_1} \wedge B_i\, q_{I_2}) \rightarrow [\dashv(p_{I_1}, q_{I_2})]\,(B_i\, q_{I_2\setminus I_1})$$

Proof.

Let $M{=}\langle W, N, \{R_i\}_{i\in Ag}, V\rangle$ and $w_H \in W$, such that $M, w_H \models (K_i(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2})) \wedge B_i\, p_{I_1} \wedge B_i\, q_{I_2})$. We have to show that $M, w_H \models [\dashv(p_{I_1}, q_{I_2})]\,(B_i\, q_{I_2\setminus I_1})$. By (1.6), this holds iff $M^{\dashv(p_{I_1},q_{I_2})}, w_H \models B_i\, q_{I_2\setminus I_1}$, with $M^{\dashv(p_{I_1},q_{I_2})}{=}\langle W, N^{\dashv(p_{I_1},q_{I_2})}, \{R_i\}_{i\in Ag}, V\rangle$, where $N^{\dashv(p_{I_1},q_{I_2})} = N(i,w_H) \cup \{ \parallel q_{I_1\cap I_2} \parallel^M_{i,w_H} \}$, because $M, w_H \models (K_i(\Box_{I_3}(p_{I_1} \rightarrow \neg q_{I_2})) \wedge B_i\, p_{I_1} \wedge B_i\, q_{I_2})$, by hypothesis. Moreover $\parallel q_{I_2\setminus I_1} \parallel^M_{i,w_H} = \parallel q_{I_1} \parallel^M_{i,w_H} \setminus \parallel q_{I_1\cap I_2} \parallel^M_{i,w_H}$ so by hypothesis $\parallel q_{I_2\setminus I_1} \parallel^{M^{\vdash(\varphi,q_J)}}_{i,w_H} \in N^{\dashv(p_{I_1},q_{I_2})}(i,w_H)$ and by definition (1.3.4) $M^{\dashv(p_{I_1},q_{I_2})}, w_H \models B_i\, q_{I_2\setminus I_1}$. ■

Remark 1.3.3 *Clearly, an analogous property holds if we drop the requirement that $I_1 \subseteq I_2$ and we admit q_J believed by the agent for any interval J disjoint from $I_1 \cup I_2$. In such cases, after the mental operation the agents starts believing $q_{I_2\setminus(I_1\cap I_2)}$ (in place of q_{I_2}), whereas all other beliefs q_J, remain in the working memory.*

1.4 AXIOMATIZATION AND CANONICAL MODELS

The logic TDLEK can be axiomatized as an extension of the axiomatization of DLEK as follows. We implicitly assume modus ponens and standard axioms for classical propositional logic.

The TLEK axioms are the following.

1. $K_i(\Box_I\varphi) \wedge K_i\big(\Box_I(\varphi \to \psi)\big) \to K_i(\Box_I\psi)$;
2. $K_i(\Box_I\varphi) \to \Box_I\varphi$;
3. $K_i(\Box_I\varphi) \to K_iK_i(\Box_I\varphi)$;
4. $\neg K_i(\Box_I\varphi) \to K_i\neg K_i(\Box_I\varphi)$;
5. $B_i\varphi \wedge K_i\big(\Box_I(\varphi \leftrightarrow \psi)\big) \to B_i\psi$.
6. $B_i\varphi \to K_i\, B_i\varphi$
7. $\dfrac{\varphi}{K_i\varphi}$

Concerning mental actions, the axiomatization of TDLEK involves also these axioms:

1. $[\alpha]f \leftrightarrow f$ where $f = p$ or $f = p_t$ or $f = p_I$;
2. $([\alpha]\varphi) \to \psi$;
3. $[\alpha]\neg\varphi \leftrightarrow \neg[\alpha]\varphi$;
4. $[\alpha](\varphi \wedge \psi) \leftrightarrow [\alpha]\varphi \wedge [\alpha]\psi$;
5. $[\alpha]K_i(\Box_I\varphi) \leftrightarrow K_i\big([\alpha](\Box_I\varphi)\big)$;
6. $[+\varphi]B_i\psi \leftrightarrow \Big(B_i([+\varphi]\psi) \vee K_i\big(([+\varphi]\psi \leftrightarrow \varphi)\big)\Big)$;
7. $[\vdash(\varphi,\psi)]B_i\chi \leftrightarrow \Big(B_i\big([\vdash(\varphi,\psi)]\chi\big) \vee \Big(B_i\varphi \wedge K_i\big(\Box_I(\varphi \to \psi)\big) \wedge$
   $K_i\big(\Box_I[\dashv(\varphi,\psi)]\chi \leftrightarrow \psi\big)\Big)\Big)$;
8. $[\cap(\varphi,\psi)]B_i\chi \leftrightarrow \Big(B_i\big([\cap(\varphi,\psi)]\chi\big) \vee \Big((B_i\varphi \wedge B_i\psi) \wedge\ K_i\Big(\big([\cap(\varphi,\psi)]\chi \leftrightarrow (\varphi \wedge \psi)\big)\Big)\Big)\Big)$;
9. $[\dashv(\varphi,\psi)]B_i\chi \leftrightarrow \Big(B_i\big([\dashv(\varphi,\psi)]\chi\big) \vee \Big(B_i\varphi \wedge K_i\big((\varphi \to \neg\psi)\big) \wedge K_i\big([\dashv(\varphi,\psi)]\chi \leftrightarrow \neg\psi\big)\Big)\Big)$;
10. $\dfrac{\psi \leftrightarrow \chi}{\varphi \leftrightarrow \varphi[\psi/\chi]}$ where $\varphi[\psi/\chi]$ denotes the formula obtained by replacing ψ with χ in φ.

We write $\mathrm{TDLEK} \vdash \varphi$ to indicate that φ is a theorem of TDLEK.

Both logics TLEK and TDLEK are sound for the class of TLEK models. The proof that TDLEK is strongly complete can be achieved by using a standard canonical model argument.

Definition 1.4.1 *The* canonical TLEK model *is a tuple* $M_c = \langle W_c; N_c; \{R_{c_i}\}_{i \in Ag}; V_c \rangle$ *where:*

- W_c *is the set of all maximal consistent subsets of* $\mathcal{L}_{TLEK}$*; so, as in [16], canonical models are constructed from worlds which are sets of syntactically correct formulas of the underlying language and are in particular the largest consistent ones. As before, each* $w \in W_c$ *can be conveniently indicated as* w_I*.*
- *For every* $w_I \in W_c$ *and* $i \in Ag$*,* $w_I R_{c_i} v_I$ *if and only if* $K_i \Box_I \varphi \in w_I$ *iff* $K_i \Box_I \varphi \in v_I$*; i.e.,* R_{c_i} *is an equivalence relation on knowledge; as before, we define* $R_{c_i}(w_I) = \{v_I \in W_c \mid w_I R_{c_i} v_I\}$*. Thus, we cope with our extension from knowledge of formulas to knowledge of formulas.*
- *Analogously to [16], for* $w_I \in W_c$*,* $\Phi \in \mathcal{L}_{TLEK}$ *and* $i \in Ag$*, we define* $A_\Phi(i, w_I) = \{v_I \in R_{c_i}(w_I) \mid \Phi \in v_I\}$*. Then, we put* $N_c(i, w_I) = \{A_\Phi(i, w_I) \mid B_i \Phi \in w_I\}$*.*
- V_c *is a valuation function defined as before.*

As stated in Lemma 2 of [16], there are the following immediate consequences of the above definition: if $w_I \in W_c$ and $i \in Ag$, then

- given $\Phi = \Box_I \varphi$ and $\varphi \in \mathcal{L}_{TLEK}$, it holds that $K_i \Box_I \varphi \in w_I$ if and only if $\forall v_I \in W$ such that $w_I R_{c_i} v_I$ we have $\Box_I \varphi \in v_I$;
- for $\Phi \in \mathcal{L}_{TLEK}$, if $B_i \Phi \in w_I$ and $w_I R_{c_i} v_I$ then $B_i \Phi \in v_I$.

Thus, while R_{c_i}-related worlds have the same knowledge and N_c-related worlds have the same beliefs, as stated in Lemma 3 of [16] there can be R_{c_i}-related worlds with different beliefs. The above properties can be used analogously to what is done in [16] to prove that, by construction, the following results hold:

Lemma 1.4.1 *For all* $w_I \in W_c$ *and* $B_i \Phi, B_i \Psi \in \mathcal{L}_{TLEK}$*, if* $B_i \Phi \in w_I$ *but* $B_i \Psi \notin w_I$*, it follows that there exists* $v_I \in R_{c_i}(w_I)$ *such that* $\Phi \in v_I \leftrightarrow \Psi \notin v_I$*.*

Proof.

Let $w_I \in W_c$ and Φ, Ψ be such that $B_i \Phi \in w_I$, $span(\Phi) \subseteq I$ and $B_i \Psi \notin w_I$. By contradiction assume that for every $v_I \in R_{c_i}(w_I)$ we have $\Psi \in v_I$, $\Phi \in v_I$ or $\Psi \notin$

v_I, $\Phi \notin v_I$; then from previous statements follows that $K_i(\Box_{I_2}(\Phi \leftrightarrow \Psi)) \in w_I$ with $span(\Phi \leftrightarrow \Psi) \subseteq I_2 \subseteq I$ so that by axiom 6 in (1.4) $B_i\,\Psi \in w_I$ which is a contradiction.
■

Lemma 1.4.2 *For all $\Phi \in \mathcal{L}_{TLEK}$ and $w_I \in W_c$ it holds that $\Phi \in w_I$ if and only if $M_c, w_I \vDash \Phi$.*

Proof.

We have to prove the statement for all $\Phi \in \mathcal{L}_{TLEK}$.

- $\Phi = p_{I_1}$ with $span(p_{I_1}) \subseteq I$, $w_I \in W_c$, if $p_{I_1} \in w_I$ then $p_{I_1} \in V_c(w_I)$ so for definition (1.3.2) we have $M_c, w_I \vDash p_{I_1}$; to prove the opposite implication we have to proceed with the same reasoning;

- all the other cases have the same proof except $\Phi = B_i\,\varphi$. Assume $B_i\,\varphi \in w_I$ and $span(\varphi) \subseteq I$ then:

  $A_\varphi(i, w_I) = \{v_I \in R_{c_i}(w_I) \mid \varphi \in v_I\} =$ by definition (1.3.2) $= \| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I)$

  So, by the previous definition of canonical model:

  $$N_c(i, w_I) = \{A_\varphi(i, w_I) \mid B_i\varphi \in w_I\}$$

  we have $\| \varphi \|^{M_c}_{i,w_I} \in N_c(i, w_I)$ and by definition (1.3.2) $M_c, w_I \vDash B_i\,\varphi$.
  Suppose $B_i\,\varphi \notin w_I$ so $\neg B_i\,\varphi \in w_I$ and $span(\neg\varphi) \subseteq I$. We have to prove $\| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I) \notin N_c(i, w_I)$. Choose $A \in N_c(i, w_I)$, by definition we know $A = A_\psi(i, w_I)$ for some ψ with $B_i\psi \in w_I$ and $span(\phi) \subseteq I$. By Lemma (1.4.1) there is some $v_I \in R_{c_i}(w_I)$ such that $\varphi \in v_I \leftrightarrow \phi \notin v_I$, so we have:

  1. for ($\rightarrow$) thanks to the induction hp $v_I \in (\| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I)) \setminus A_\psi(i, w_I)$;
  2. for ($\leftarrow$) thanks to the induction hp $v_I \in A_\psi(i, w_I) \setminus (\| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I))$;

  than for (1) and (2) $A_\psi(i, w_I) \neq \| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I)$ and since $A = A_\psi(i, w_I)$ was arbitrary in $N_c(i, w_I)$ we conclude that $\| \varphi \|^{M_c}_{i,w_I} \cap R_{c_i}(w_I) \notin N_c(i, w_I)$, and so than $M_c, w_I \nvDash B_i\,\varphi$.

■

Lemma 1.4.3 *For all $\Phi \in \mathcal{L}_{TDLEK}$ there exists $\tilde{\Phi} \in \mathcal{L}_{TLEK}$ such that* TDLEK $\vdash \Phi \leftrightarrow \tilde{\Phi}$ *(for any TDLEK formula we can find an equivalent LEK formula).*

Proof.

We have to prove the sentence for all $\Phi \in \mathcal{L}_{TDLEK}$ but we show the proof only for $\Phi = p_{I_1}$ because the others are proved analogously. By axiom in the section (1.4) we have $[\alpha]p_{I_1} \leftrightarrow p_{I_1}$ and by axiom (10) we have $\dfrac{[\alpha]p_{I_1} \leftrightarrow p_{I_1}}{\varphi \leftrightarrow \varphi[[\alpha]p_{I_1}/p_{I_1}]}$ which means that we can replace $[\alpha]p_{I_1}$ with p_{I_1} in φ. ■

Under the assumption that in every formula $\Box_I\varphi$ the interval I is finite, the previous lemmas allow us to prove the following theorems. The limitation to finite intervals is not related to features of the proposed approach, but to well-known paradoxes of temporal logics on infinite intervals.

Theorem 1.4.1 TLEK *is strongly complete for the class of* TLEK *models.*

Proof.

Any consistent set Φ may be extended to a maximal consistent set of formulas $w_I^\star \in W_c$ and $M_c, w_I^\star \vDash \Phi$ by Lemma (1.4.2). Then, ***TLEK*** is strongly complete for the class of ***TLEK*** models. ■

Theorem 1.4.2 TDLEK *is strongly complete for the class of* TLEK *models.*

Proof.

If K is a consistent set of $\mathcal{L}_{TDLEK}$ formulas then $\tilde{K} = \{\tilde{\varphi} \mid \varphi \in K\}$ is a consistent set of $\mathcal{L}_{TLEK}$ formulas by Lemma (1.4.3). By Theorem (1.4.1) there is a model M_c with a world w_I such that $M_c, w_I \vDash \tilde{K}$. But since $TDLEK$ is sound and for each $\varphi \in K$, $\mathrm{TDLEK} \vdash \varphi \leftrightarrow \tilde{\varphi}$, it follows $M_c, w_I \vDash K$ then TDLEK is strongly complete for the class of ***TLEK*** models. ■

1.5 An Example of Temporal Reasoning

In this section we illustrate how logic $TDLEK$ works by exploring a larger example. Let us consider the following scenario: the Program committee of some Conference "xxx" solicits submission of AI technical papers for the main technical track of the Conference, which is going to be held in "yyy", say from August 19 to 25. We have an agent (Author) who is of course a resource bounded one, in the sense that he/she has a certain background knowledge (non omniscient) and explicit timed beliefs and forms new explicit timed beliefs (like, e.g., the beliefs related to the submission dates, and other time constrained information related to the Conference) by means of the mental operations illustrated before.

The Program Committee communicates information about submission dates and other deadlines to the authors (among which our agent) through the conference website. Let us assume for instance:

- abstract submission 16 February;
- full paper submission 19 February (if the abstract had been previously submitted);
- notification of accept/reject 10 April (before, a preliminary reviews will be communicated to the author by email, then the response to these reviews will be between 28-30 March, while details and guidelines for responses will be put on the website);
- final version must be sent to the Conference Chairs by April 30;
- the Conference will be held 19-25 August.

Submitted papers must fulfill the submission guidelines. Upon acceptance, at least one of the authors should attend the conference to present the paper. Authors will be required to agree on this requirements at the time of submission. Of course in such context we have multiple agents (authors), but in our example we will consider the case of one author which can easily be extended to multiple authors. Let us consider the following time variables:

$t_0 =$ current moment of time, $t_1 = 16$ February, $t_2 = 19$ February,
$t_3 = 10$ April, $t_4 = 28$ March, $t_5 = 30$ March,
$t_6 = 30$ April, $t_7 = 19$ August $t_8 = 25$ August.

Our agent i (author) has in his long-term memory the following rules (recall that we denote by p the timed atom $p_{[0,\infty)}$ and $\Box$ stands for $\Box_{[0,\infty)}$; moreover, variables are written with an initial uppercase letter):

1. $K_i\big(\Box(read\text{-}guidelines \wedge submission\text{-}respect \rightarrow fullPaper\text{-}ready)\big)$.
   This rule indicates that if the author read the submission guidelines and her paper fulfills them, then she can consider her paper ready.

2. $K_i\big(\Box_{[t_0,t_1]}(abstract\text{-}ready_T \rightarrow \Box_{[T+1,t_1]} submit\text{-}abstract_{T_a})\big)$.
   This rule indicates that if the author has the paper's abstract ready at any moment $T \in [t_0, t_1]$ then at the next moment until t_1 she can submit the abstract.

3. $K_i\big(\Box_{[t_0,t_2]}((fullPaper\text{-}ready_{T_c} \wedge \Box_{[t_0,t_1]} submit\text{-}abstract_{T_b}) \rightarrow \Box_{[T_c+1,t_2]} submit\text{-}fullPaper_{T_d})\big)$.
   The meaning of this rule is that it is always true within the interval $[t_0, t_2]$ that, if the agent has the full paper ready at some moment in time $T_c \in [t_0, t_2]$ and had already submitted the abstract at $T_b \in [t_0, t_1] \subseteq [t_0, t_2]$, then from the next moment of time $T_c + 1$ she can submit the full paper (such that $T_b \leq T_c + 1$);

4. $K_i\big(\Box_{[t_0,t_4]}(\Box_{[t_0,t_2]} submit\text{-}fullPaper_{T_e} \rightarrow \Box_{[t_2+1,t_4]} receive\text{-}reviews_{T_f})\big)$.
   This rule indicates that if the author submitted the paper to the Conference during the required interval of time $[t_0, t_2]$ then she will receive the reviews before time t_4;

5. $K_i\big(\Box_{[t_2+1,t_5]}(\Box_{[t_2+1,t_4]} receive\text{-}reviews_{T_g} \rightarrow \Box_{[t_4,t_5]} send\text{-}rebuttal_{T_h})\big)$.
   This rule indicates that if the author receives reviews during the interval $[t_2, t_4]$ then she can respond at any moment T_h during the interval $[t_4, t_5]$;

6. $K_i\big(\Box_{[t_4,t_6]}((\Box_{[t_4,t_5]} send\text{-}rebuttal_{T_h} \wedge \Box_{[t_4,t_3]} paper\text{-}accepted_{T_j}) \rightarrow \Box_{[t_3,t_6]} send\text{-}final\text{-}version_{T_v})\big)$.
   This rule indicates that if the agent receives an acceptance notification during the interval $[t_4, t_3]$ and she remembers that she sent the rebuttal at $T_h \in [t_4, t_5]$ then she has to send the final version of the paper until t_6;

7. $K_i\big(\Box_{[t_3,t_8]}(\Box_{[t_3,t_6]} send\text{-}final\text{-}version_{T_z} \rightarrow \Box_{[t_7,t_8]} go\text{-}to\text{-}conference_{T_k})\big)$.
   This rule indicates that if the agent sent the final version of the paper in $T_z \in [t_4, t_3]$ then she can go to the conference from t_7 to t_8;

8. $K_i\big(\Box(submit\text{-}fullPaper_{T_r} \rightarrow paper\text{-}accepted_{[T_r,\infty)})\big)$.
   This rule indicates that an agent optimistically assumes that the already submitted paper will be accepted;

9. $K_i\big(\Box(paper\text{-}rejected_{T_s} \rightarrow \neg assume\text{-}paper\text{-}accepted_{[T_s,\infty)})\big)$.
   This rule indicates that if an agent assumes that her paper is accepted but at a certain point she receives a reject notification, she has to change her belief.

Let us now consider one of our authors (agents), say A_1, who has decided to prepare a submission to this Conference. Hence, she writes an abstract for her paper and believes that it is ready at moment t_x. So, she has in her short-term memory the belief $B_{A_1}(abstract\text{-}ready_{t_x})$, with $t_x \in [t_0, t_1]$. Then, she can infer from the second rule, by performing the mental operation $\vdash$ $(abstract\text{-}ready_{t_x}, submit\text{-}abstract_{t_a})$ that at some moment t_a, such that $t_x + 1 \leq t_a \leq t_1$ the abstract can be submitted. Let us assume now that our author has submitted the abstract successfully. Then, she adds to her beliefs $B_{A_1}(submit{-}abstract_{t_a})$. After that, she starts preparing the full paper; she completes it after some time and so believes that it is ready at time $t_z \leq t_2$: therefore, $B_{A_1}(fullPaper\text{-}ready_{t_z})$ it is added to her beliefs. At this point, our author has the new belief $B_{A_1}(fullPaper\text{-}ready_{t_z})$ and retrieves from her memory the belief $B_{A_1}(submit\text{-}abstract_{t_a})$.
Then by performing the mental operation $\cap(submit\text{-}abstract_{t_a}, fullPaper\text{-}ready_{t_z})$ she can start believing that $B_{A_1}(submit\text{-}abstract_{t_a} \wedge fullPaper\text{-}ready_{t_z})$ is true, $t_z \in [t_0, t_2]$ and $t_a \leq t_z$). Then, according to the third rule at some moment in time $t_d \in [t_z + 1, t_2]$ she can submit the full paper. Finally, assuming that our agent submitted her paper successfully at time t_d she adds $B_{A_1}(submit\text{-}fullPaper_{t_d})$ to the short-term memory and waits for notification from the Conference Chairs.

To illustrate the effect of the performed actions on the agent's beliefs change, let us consider the case of an optimistic agent A_2 which, after submission, assumes the paper to

be accepted and so forms the belief $B_{A_2}(assume\text{-}paper\text{-}accepted_{[t_a,\infty)})$ with $t_a \geq t_2$. If the Program Chairs will send notification of acceptance then as a consequence the paper will be believed to be accepted. Otherwise, it will be believed to be rejected if there is a rejection. If at time time t_3 agent A_2 received a rejection note, she starts believing $B_{A_2}(paper\text{-}rejected_{[t_3,\infty)})$; therefore, via the mental operation $\dashv (paper\text{-}rejected_{[t_3,\infty)}, assume\text{-}paper\text{-}accepted_{[t_a,\infty)})$ and since $t_3 > t_a$, she can replace the timed atom $assume\text{-}paper\text{-}accepted_{[t_a,\infty)}$ with the more limited $assume\text{-}paper\text{-}accepted_{[t_a,t_3-1]}$ which indicates the actual interval where the belief has remained in the short-term memory without being contradicted.

1.6 T-LEK AND T-DLEK

As said in the introduction, to avoid problems with the management of the intervals and in order to not lose the logic of the formalization, in [70] we present another extension of LEK/DLEK to T-LEK/T-DLEK ("Timed LEK" and "Timed DLEK") obtained by introducing a special function which associates to each belief the arrival time and controls timed inferences. Through this function it is easier to keep the evolution of the surrounding world under control and the representation is more complete. And also, we work on *Belief Revision*, because agents interact with the environment so they have to change their belief according to the actual word.

1.6.1 SYNTAX

In our scenario we fix $Atm = \{p(t_1, t_2), q(t_3, t_4), \ldots, h(t_i, t_j)\}$ where $t_i \leqslant t_j$ and p, q, h are predicates, that can be equal or not. Moreover $p(t_1, t_2)$ stands for *"p is true from the time instant* t_1 *to* t_2*"* with $t_1, t_2 \in \mathbb{N}$ (*Temporal Representation* of the external world); as a special case we can have $p(t_1, t_1)$ which stands for *"p is true in the time instant* t_1*"*. Obviously we can have predicates with more terms than only two but in that case we fix that the first two must be those that identify the time duration of the belief (i.e. $open(1, 3, \text{door})$ which means "the agent knows that the door is open from time one to time 3"). In the previous work [61] we considered atoms of the form p_I with $I = [t_1, t_2]$, which are the conjunction $p_{t_1} \wedge p_{t_1+1} \wedge \cdots \wedge p_{t_2}$ and also p_t stands for p_{I_t} with $I_t = [t, t]$; we decided to change approach because syntax p_I is too detached from propositional logic. So, we introduce a new formalization that also copes the multi-agent case. Let Agt be a finite set of agents.
Below is the definition of the formulas of the language $\mathcal{L}_{T\text{-}LEK}$: again, in this grammar we use I as terminal symbol standing for time intervals and $i \in Agt$:

$$\varphi, \psi \quad := \quad p(t_1, t_2) \mid \neg\varphi \mid \Box_I \varphi \mid B_i \varphi \mid K_i \varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi$$

Other Boolean connectives $\top$, $\bot$, $\leftrightarrow$ are defined from $\neg$ and $\wedge$ as usual. In the formula $\Box_I \Phi$ the MTL Interval "always" operator is applied to a formula; I is a "time-interval"

which is a closed finite interval $[t, l]$ or an infinite interval $[t, \infty)$ (considered open on the upper bound), for any expressions/values t, l such that $0 \leq t \leq l$ and $\Box_{[0,\infty)}$ will sometimes be written simply as $\Box$. The operator B_i identifies belief in the working memory and the operator K_i to denote knowledge and identifies what rules are present in the background knowledge. Terms/atoms/formulas as defined so far are *ground*, namely there are no variables occurring therein.

The language $\mathcal{L}_{T\text{-}DLEK}$ of Temporalized DLEK (T-DLEK) is obtained by augmenting $\mathcal{L}_{T-LEK}$ with the expression $[(G_I : \alpha)_{H_I}]\,\psi$, where α denotes a *mental operation*, ψ is a ground formula, G_I, H_I range over 2^{Agt} and $G_I \subseteq H_I$; the I is used to define that in a given interval we have a given set of agents and I depends on ψ. Moreover $[(G_I : \alpha)_{H_I}]\,\psi$ stands for "ψ holds after the mental operation α is performed by all the agent in G_I, and the agents in H_I have common knowledge about this fact". The mental operation are the same as before, the only difference is in $\dashv(\varphi, \psi)$, in fact we have:

- $\dashv(\varphi, \psi)$: belief revision; where φ and ψ are ground atoms, say $p(t_1, t_2)$ and $q(t_3, t_4)$ respectively: an agent, believing $p(t_1, t_2)$ and having in the long-term memory that $p(t_1, t_2)$ implies $\neg q(t_3, t_4)$, removes the timed belief $q(t_3, t_4)$ if the intervals match. Notice that, should q be believed in a wider interval I such that $[t_1, t_2] \subseteq I$, the belief $q(.,.)$ is removed concerning intervals $[t_1, t_2]$ and $[t_3, t_4]$, but it is left for the remaining sub-intervals (so, its is "restructured").

Example 3 *We propose a small example to illustrate the differences from the previous formalization. If at time* t=2 *it is starting raining, in the working memory of agent* i *there will be the following belief:* $B_i(raining(2, 2))$. *And if we have in the background knowledge* $K_i(rain(t_1, t_2) \rightarrow take(t_1, t_2, \textit{umbrella}))$ *and* $2 \in [t_1, t_2]$ *than agent* i *can infer* $B_i(take(2, 2, \textit{umbrella}))$, *which is a new belief stored in the working memory.*
And if we have also $K_i(rain(t_1, t_2) \wedge take(t_1, t_2, \textit{umbrella}) \rightarrow go(t_1 + 1, \infty, \textit{shops}))$ *than the agent can infer* $B_i(go(3, \infty, \textit{shops}))$ *which means that after getting the umbrella agent* i *can go shopping.*

1.6.2 SEMANTICS

Semantics of DLEK and T-DLEK are both based on a set W of worlds. In both DLEK and T-DLEK we have the valuation function: $V : W \rightarrow 2^{Atm}$. Also we define the "time" function T that associates to each formula the time interval in which this formula is true and operates as follows:

- $T(p(t_1, t_2)) = [t_1, t_2]$, which stands for *"p is true in the time interval* $[t_1, t_2]$*"* where $t_1, t_2 \in \mathbb{N}$; as a special case we have $T(p(t_1, t_1)) = t_1$, which stands for *"p is true in the time instant* t_1*"* where $t_1 \in \mathbb{N}$ (time instant);
- $T(\neg p(t_1, t_2)) = T(p(t_1, t_2))$, which stands for *"p is not true in the time interval* $[t_1, t_2]$*"* where $t_1, t_2 \in \mathbb{N}$;

- $T(\varphi \text{ op } \psi) = T(\varphi) \biguplus T(\psi)$ with *op* $\in \{\vee, \wedge, \rightarrow\}$, which means the unique smallest interval including both $T(\varphi)$ and $T(\psi)$;
- $T(B_i\varphi) = T(\varphi)$;
- $T(K_i\varphi) = T(\varphi)$;
- $T(\Box_I\varphi) = I$ where I is a time interval in $\mathbb{N}$;
- $T([(G_I : \alpha)_{H_I}]\varphi)$ there are different cases depends on which kind of mental operations we applied:
  1. $T((G_I : +\varphi)_{H_I}) = T(\varphi)$;
  2. $T((G_I : \cap(\varphi, \psi))_{H_I}) = T(\varphi) \biguplus T(\psi)$;
  3. $T((G_I : \vdash(\varphi, \psi))_{H_I}) = T(\psi)$;
  4. $T((G_I : \dashv(\varphi, \psi)_{H_I}))$ returns the restored interval where ψ is true.

For a world w, let t_1 the minimum time instant of $T(p(t_1, t_1))$ where $p(t_1, t_1) \in V(w)$ and let t_2 be the supremum time instant (we can have $t_2 = \infty$) among the atoms in $V(w)$. Then, whenever useful, we denote w as w_I where $I = [t_1, t_2]$, which identifies the world in a given interval.

The notion of LEK/T-LEK model does not consider mental operations, discussed later, and is introduced by the following definition.

Definition 1.6.1 *A* T-LEK model *is a tuple* $M = \langle W; N; \{R_i\}_{i \in Agt}; V; T\rangle$ *where:*

- W *is the set of worlds;*
- $V : W \rightarrow 2^{Atm}$ *valuation function;*
- T *"time" function;*
- $R_i \subseteq W \times W$ *is the accessibility relation with* $i \in Agt$*, required to be an equivalence relation so as to model omniscience in the background knowledge s.t.* $R_i(w_I) = \{v_I \in W \mid w_I R\, v_I\}$ *called* epistemic state of the agent i in w_I*, which indicates all the situations that the agent considers possible in the world* w_I *or, equivalently any situation the agent* i *can retrieve from long-term memory based on what it knows in world* w_I*;*
- $N : Agt \times W \rightarrow 2^{2^W}$ *is a* "neighbourhood" *function,* $\forall w \in W$*,* $N(i, w)$ *defines, in terms of sets of worlds, what the agent* i *is allowed to explicitly* believe *in the world* w_I*;* $\forall w_I, v_I \in W$*, and* $X \subseteq W$*:*
  1. *if* $X \in N(i, w_I)$*, then* $X \subseteq R_i(w_I)$*: each element of the neighbourhood is a set composed of reachable worlds;*

2. *if $w_I R_i\, v_I$, then $N(w_I) \subseteq N(v_I)$: if the world v_I is compliant with the epistemic state of world w_I, then the agent i in the world w_I should have a subset of beliefs of the world v_I.*

A preliminary definition before the Truth conditions :
let $M = \langle W; N; \{R_i\}_{i \in Agt}; V; T\rangle$ a T-LEK model. Given a formula φ, for every $w_I \in W$, we define

$$\| \varphi \|^M_{w_I} = \{v_I \in W \mid M, v_I \models \varphi\} \cap R_i(w_I).$$

Truth conditions for T-DLEK formulas are defined inductively as follows:

- $M, w_I \models p(t_1, t_2)$ iff $p(t_1, t_2) \in V(w_I)$ and $T(p(t_1, t_2)) \subseteq I$;
- $M, w_I \models \neg\varphi$ iff $M, w_I \not\models \varphi$ and $T(\neg\varphi) \subseteq I$;
- $M, w_I \models \varphi \wedge \psi$ iff $M, w_I \models \varphi$ and $M, w_I \models \psi$ with $T(\varphi), T(\psi) \subseteq I$;
- $M, w_I \models \varphi \vee \psi$ iff $M, w_I \models \varphi$ or $M, w_I \models \psi$ with $T(\varphi), T(\psi) \subseteq I$;
- $M, w_I \models \varphi \rightarrow \psi$ iff $M, w_I \not\models \varphi$ or $M, w_I \models \psi$ with $T(\varphi), T(\psi) \subseteq I$;
- $M, w_I \models B_i\, \varphi$ iff $\| \varphi \|^M_{w_I} \in N(w_I)$ and $T(\varphi) \subseteq I$;
- $M, w_I \models K_i\, \varphi$ iff for all $v_I \in R_i(w_I)$, it holds that $M, v_I \models \varphi$ and $T(\varphi) \subseteq I$;
- $M, w_I \models \Box_J \varphi$ iff $T(\varphi) \subseteq J \subseteq I$ and for all $v_I \in R_i(w_I)$, it holds that $M, v_I \models \varphi$;

In particular, considering formulas of the forms $B_i\, \varphi$ and $K_i\, \varphi$, we observe that $M, w_I \models B_i\, \varphi$ if the set $\| \varphi \|^M_{w_I}$ of worlds reachable from w_I which entail φ in the very same model M belongs to the *neighbourhood* $N(i, w_I)$ of w_I. Hence, knowledge pertains to formulas entailed in model M in every reachable world, while beliefs pertain to formulas entailed only in some set of them, where this set must however belong to the neighbourhood and so it must be composed of reachable worlds. Thus, an agent is seen as omniscient with respect to knowledge, but not with respect to beliefs.

Concerning a mental operation α performed by any agent i, we have: $M, w_I \models [(G_J : \alpha)_{H_J}]\, \varphi$ iff $M^{(G_J:\alpha)_{H_J}}, w_I \models \varphi$, $T((G_J : \alpha)_{H_J}) \subseteq I$, $J = T((G_J : \alpha)_{H_J})$ where $M^{(G_J:\alpha)_{H_J}} = \langle W; N^{(G_J:\alpha)_{H_J}}(i, w_I); \{R_i{}^{(G_J:\alpha)_{H_J}}\}_{i \in Agt}; V; T\rangle$. Here $R_i{}^{(G_J:\alpha)_{H_J}}(w_I) = \{v_I \in W \text{ s.t. } w_I R_i\, v_I \text{ and } J \subseteq I\}$ and α represents a mental operation affecting the sets of beliefs. In particular, such operation can add new beliefs by direct perception, by means of one inference step, or as a conjunction of previous beliefs. When introducing new beliefs, the neighbourhood must be extended accordingly, as seen below; in particular, the new neighbourhood:

- $N^{(G_J:\alpha)_{H_J}}(i, w_I) = \{X \in N^{\alpha}(i, w_I) \text{ if } i \in G_J \text{ and } J \subseteq I\}$: if agent i is in G_J then he has to change his neighbourhood accordingly to α;

- $N^{(G_J:\alpha)_{H_J}}(i, w_I) = \{X \in N(i, w_I)$ if $i \in H_J \backslash G_J$ and $J \subseteq I\}$: if agent i is in $H_J \backslash G_J$ then he does not have to change his neighbourhood because he does not perform the mental operation α but he knows that other agents in G_J have performed a operation;

- $N^{(G_J:\alpha)_{H_J}}(i, w_I) = \{X \in N(i, w_I)$ if $i \notin H$ and $J \subseteq I\}$: if agent i is not in H_J then he does not have to change his neighbourhood because he does not perform the mental operation α and also he do not know that other agents in G_J have performed a operation.

Where $N^{\alpha}(i, w_I)$ is defined for each of the mental operations as follows.

- Learning perceived belief:

  $N^{+\varphi}(i, w_I) = N(i, w_I) \cup \{ \parallel \varphi \parallel^M_{w_I} \}$ with $T(\varphi) \subseteq I$.

  The agent i adds to its beliefs perception φ (namely, an atom or the negation of an atom) perceived at a time in $T(\varphi)$; the neighbourhood is expanded to as to include the set composed of all the reachable worlds which entail φ in M.

- Beliefs conjunction:

$$N^{\cap(\psi,\chi)}(i, w_I) = \begin{cases} N(i, w_I) \cup \{ \parallel \psi \wedge \chi \parallel^M_{w_I} \} & \text{if } M, w_I \models B_i(\psi) \wedge B_i(\chi) \\ & \text{and } T(\cap(\psi, \chi)) \subseteq I \\ N(i, w_I) & \text{otherwise} \end{cases}$$

  The agent i adds $\psi \wedge \chi$ as a belief if it has among its previous beliefs both ψ and χ, with I including all time instants referred to by them; otherwise the set of beliefs remain unchanged. The neighbourhood is expanded, if the operation succeeds, with those sets of reachable worlds where both formulas are entailed in M.

- Belief inference:

$$N^{\vdash(\psi,\chi)}(i, w_I) = \begin{cases} N(i, w_I) \cup \{ \parallel \chi \parallel^M_{w_I} \} & \text{if } M, w_I \models B_i(\psi) \wedge K_i(\psi \rightarrow \chi) \\ & \text{and } T(\vdash (\psi, \chi)) \subseteq I \\ N(w_I) & \text{otherwise} \end{cases}$$

  The agent i adds the ground atom χ as a belief in its short-term memory if it has ψ among its previous beliefs and has in its background knowledge $K_i(\psi \rightarrow \chi)$, where all the time stamps occurring in ψ and in χ belong to I. Observe that, if I does not include all time instants involved in the formulas, the operation does not succeed and thus the set of beliefs remains unchanged. If the operation succeeds then the neighbourhood is modified by adding χ as a new belief.

- Beliefs revision (applied only on ground atoms).
  Given $Q = q(j, k)$ s.t. $T(q(j, k)) = T(q(t_1, t_2)) \cap T(q(t_3, t_4))$ with $j, k \in \mathbb{N}$ and $P = \{M, w_I \models B_i(p(t_1, t_2)) \wedge B_i(q(t_3, t_4)) \wedge K_i(p(t_1, t_2) \rightarrow \neg q(t_3, t_4))$ and $T(\dashv (p(t_1, t_2), q(t_3, t_4))) \subseteq I$ and there is no interval $J \supsetneq T(p(t_1, t_2))$ s.t. $B_i(q(t_5, t_6))$ where $T(q(t_5, t_6))=J\}$:

$$N^{\dashv(p(t_1,t_2),q(t_3,t_4))}(i,w_I) = \begin{cases} N(i,w_I) \setminus \{ \parallel Q \parallel^M_{w_I} \} & \text{if P} \\ N(i,w_I) & \text{otherwise} \end{cases}$$

The agent i believes that $q(t_3,t_4)$ holds only in the interval $T(q(t_3,t_4))$ and has the perception of $p(t_1,t_2)$ where $T(p(t_1,t_2)) \subseteq T(q(t_3,t_4))$. Then, the agent replaces previous belief $q(t_3,t_4)$ in the short-term memory with $q(t_5,t_6)$ where $T(q(t_5,t_6))$=$T(q(t_3,t_4)) \setminus T(q(t_1,t_2))$.
In general, the set $T(q(t_3,t_4)) \setminus T(q(t_1,t_2))$ is not necessarily an interval: being $T(p(t_1,t_2)) \subseteq T(q(t_3,t_4))$, with $T(p(t_1,t_2))$=$[t_1,t_2]$, and $T(q(t_3,t_4))$=$[t_3,t_4]$, we have that $T(q(t_3,t_4)) \setminus T(q(t_1,t_2))$=$[t_3,t_1-1] \cup [t_2+1,t_4]$. Thus, $q(t_3,t_4)$ is replaced by $q(t_3,t_1-1)$ and $q(t_2+1,t_4)$ (and similarly if $t_4=\infty$).

We write $\models_{T\text{-}DLEK} \varphi$ to denote that φ is true in all worlds w_I, of every TLEK model M.

Property 1.6.1 *As in TDLEK, for the mental operations previously considered we have the following (where φ, ψ are as explained earlier):*

- $\models_{T\text{-}DLEK} [(G_J : +\varphi)_{H_J}] B_i \varphi$.
  Namely, as a consequence of the operation $+\varphi$ (thus after the perception of φ) the agent i adds φ to its beliefs.

- $\models_{T\text{-}DLEK} (B_i\varphi \wedge B_i\psi) \rightarrow [(G_J : \cap(\varphi,\psi))_{H_J}] B_i(\varphi \wedge \psi)$.
  Namely, if agent i has φ and ψ as beliefs, then as a consequence of the mental operation $\cap(\varphi,\psi)$ the agent starts believing $\varphi \wedge \psi$;

- $\models_{T\text{-}DLEK} (K_i(\varphi \rightarrow \psi) \wedge B_i\,\varphi) \rightarrow [(G_J : \vdash(\varphi,\psi))_{H_J}]\, B_i\,\psi$.
  Namely, if agent i has φ as one of its beliefs and has $K_i(\varphi \rightarrow \psi)$ in its background knowledge, then as a consequence of the mental operation $\vdash(\varphi,\psi)$ the agent starts believing ψ;

- $\models_{T\text{-}DLEK} (K(p(t_1,t_2) \rightarrow \neg q(t_3,t_4)) \wedge B_i\,(p(t_1,t_2)) \wedge B_i\,(q(t_3,t_4))) \rightarrow [(G_J : \dashv(p(t_1,t_2),q(t_3,t_4)))_{H_J}]\,(B_i\,(q(t_5,t_6)))$
  where $T(q(t_5,t_6)) = T(q(t_3,t_4)) \setminus T(q(t_1,t_2))$.
  Namely, if agent i has $q(t_3,t_4)$ as one of its beliefs, q is not believed outside $T(q(t_3,t_4))$, the agent perceives $p(t_1,t_2)$ where $T(p(t_1,t_2)) \subseteq T(q(t_3,t_4))$, and has $K_i(p(t_1,t_2) \rightarrow \neg q(t_3,t_4))$ in its background knowledge.
  Then after the mental operation $\dashv(p(t_1,t_2),q(t_3,t_4))$ the agent starts believing $q(t_5,t_6))$ where $T(q(t_5,t_6)) = T(q(t_3,t_4)) \setminus T(q(t_1,t_2))$.

The proof of these property are the same as Property 1.3.2 and Property 1.3.3.

1.7 AXIOMATIZATION AND CANONICAL MODELS

As in 1.4 the T-LEK axioms are the following:

1. $K_i(\varphi) \wedge K_i(\varphi \to \psi) \to K_i(\psi)$;
2. $K_i(\varphi) \to \varphi$;
3. $K_i(\varphi) \to K_i K_i(\varphi)$;
4. $\neg K_I(\varphi) \to K_i \neg K_i(\varphi)$;
5. $B_i\varphi \wedge K_i(\varphi \leftrightarrow \psi) \to B_i\psi$;
6. $\Box_I\varphi \wedge \Box_I(\varphi \to \psi) \to \Box_I(\psi)$;
7. $\Box_I\varphi \to \Box_J\varphi$ with $J \subseteq I$;

The axiomatization of T-DLEK, involves these axioms:

1. $[(G_J : \alpha)_{H_J}]f \leftrightarrow f$ where $f = p$ or $f = p_t$ or $f = p_I$;
2. $([(G_J : \alpha)_{H_J}]\varphi) \leftrightarrow \psi$;
3. $[(G_J : \alpha)_{H_J}]\neg\varphi \leftrightarrow \neg[(G_J : \alpha)_{H_J}]\varphi$;
4. $[(G_J : \alpha)_{H_J}](\varphi \wedge \psi) \leftrightarrow [(G_J : \alpha)_{H_J}]\varphi \wedge [(G_J : \alpha)_{H_J}]\psi$;
5. $[(G_J : \alpha)_{H_J}]K_i(\varphi) \leftrightarrow K_i\big([(G_J : \alpha)_{H_J}](\varphi)\big)$;
6. $[(G_J : +\varphi)_{H_J}]B_i\psi \leftrightarrow \Big(B_i([(G_J : +\varphi)_{H_J}]\psi) \vee K_i\big([(G_J : +\varphi)_{H_J}]\psi \leftrightarrow \varphi\big)\Big)$;
7. $[(G_J : \vdash(\varphi,\psi))_{H_J}]B_i\chi \leftrightarrow \Big(B_i\big([(G_J : \vdash(\varphi,\psi))_{H_J}]\chi\big) \vee \Big(B_i\varphi \wedge K_i\big(\varphi \to \psi\big) \wedge K_i\big([(G_J : \vdash(\varphi,\psi))_{H_J}]\chi \leftrightarrow \psi\big)\Big)\Big)$;
8. $[(G_J : \dashv(\varphi,\psi))_{H_J}]B_i\chi \leftrightarrow \Big(B_i\big([(G_J : \dashv(\varphi,\psi))_{H_J}]\chi\big) \vee \Big(B_i\varphi \wedge K_i(\varphi{\to}\neg\psi) \wedge K_i\big([(G_J : \dashv(\varphi,\psi))_{H_J}]\chi{\leftrightarrow}\neg\psi\big)\Big)\Big)$;
9. $[(G_J : \cap(\varphi,\psi))_{H_J}]B_i\chi \leftrightarrow \Big(B_i\big([(G_J : \cap(\varphi,\psi))_{H_J}]\chi\big) \vee \Big((B_i\varphi{\wedge}B_i\psi) \wedge K_i\big([(G_J : \cap(\varphi,\psi))_{H_J}]\chi \leftrightarrow (\varphi \wedge \psi)\big)\Big)\Big)$;
10. $\dfrac{\psi \leftrightarrow \chi}{\varphi \leftrightarrow \varphi[\psi/\chi]}$ where $\varphi[\psi/\chi]$ denotes the formula obtained by replacing ψ with χ in φ.

We write T-DLEK $\vdash \varphi$ to indicate that φ is a theorem of TDLEK.

Both logics T-LEK and T-DLEK are sound for the class of T-LEK models. The proof that T-DLEK is strongly complete can be achieved by using a standard canonical model argument.

Definition 1.7.1 *The* canonical T-LEK model *is a tuple* $M_c = \langle W_c; N_c; \{R_{c_i}\}_{i \in Agt}; V_c; T_c \rangle$ *where:*

- W_c *is the set of all maximal consistent subsets of* $\mathcal{L}_{T\text{-}LEK}$*; so, as in [16], canonical models are constructed from worlds which are sets of syntactically correct formulas of the underlying language and are in particular the largest consistent ones. As before, each* $w \in W_c$ *can be conveniently indicated as* w_I*.*
- *For every* $w_I \in W_c$ *and* $w_I R_{c_i} v_I$ *if and only if* $K_i\varphi \in w_I$ *iff* $K_i\varphi \in v_I$*; i.e.,* R_{c_i} *is an equivalence relation on knowledge; as before, we define* $R_{c_i}(w_I) = \{v_I \in W \mid w_I R_{c_i} v_I\}$*. Thus, we cope with our extension from knowledge of formulas to knowledge of formulas.*
- *Analogously to [16], for* $w_I \in W_c$*,* $\varphi \in \mathcal{L}_{T\text{-}LEK}$ *we define* $A_\varphi(w_I) = \{v_I \in R_{c_i}(w_I) \mid \varphi \in v_I\}$*. Then, we put* $N_c(w_I) = \{A_\varphi(w_I) \mid B_i\varphi \in w_I\}$*.*
- V_c *is a valuation function defined as before.*
- T_c *is a "time" function defined as before.*

As stated in Lemma 2 of [16], there are the following immediate consequences of the above definition: if $w_I \in W_c$ and $i \in Agt$, then

- for $\varphi \in \mathcal{L}_{T\text{-}LEK}$, it holds that $K_i\varphi \in w_I$ if and only if $\forall v_I \in W_c$ such that $w_I R_{c_i} v_I$ we have $\varphi \in v_I$;
- for $\varphi \in \mathcal{L}_{T\text{-}LEK}$, if $B_i\varphi \in w_I$ and $w_I R_{c_i} v_I$ then $B_i\varphi \in v_I$.

Thus, while R_{c_i}-related worlds have the same knowledge and N_c-related worlds have the same beliefs, as stated in Lemma 3 of [16] there can be R_{c_i}-related worlds with different beliefs. The above properties can be used analogously to what is done in [16] to prove that, by construction, the following results hold:

Lemma 1.7.1 *For all* $w_I \in W_c$ *and* $B_i\varphi, B_i\psi \in \mathcal{L}_{T\text{-}LEK}$*, if* $B_i\varphi \in w_I$ *but* $B_i\psi \notin w_I$*, it follows that there exists* $v_I \in R_{i,c}(w_I)$ *such that* $\varphi \in v_I \leftrightarrow \psi \notin v_I$*.*

Lemma 1.7.2 *For all* $\varphi \in \mathcal{L}_{T\text{-}LEK}$ *and* $w_I \in W_c$ *it holds that* $\varphi \in w_I$ *if and only if* $M_c, w_I \vDash \varphi$*.*

Lemma 1.7.3 *For all* $\varphi \in \mathcal{L}_{T\text{-}DLEK}$ *then there exists* $\tilde{\varphi} \in \mathcal{L}_{T\text{-}LEK}$ *such that* $\text{T-DLEK} \vdash \varphi \leftrightarrow \tilde{\varphi}$.

Under the assumption that the interval I is finite, the previous lemmas allow us to prove the following theorems.

Theorem 1.7.1 T-LEK *is strongly complete for the class of* T-LEK *models.*

Theorem 1.7.2 T-DLEK *is strongly complete for the class of* T-LEK *models.*

The proof of this lemmas and theorem are the same as in (1.4).

1.7.1 Example: Italian PhD Program

In this section we illustrate an other example to show how *T-DLEK* works and the differences from *TDLEK*.
Let us consider the following scenario: PhD courses have a duration of three years and the admission to doctoral research courses is done by a public selection and the program starts from the 1th of November. The PhD courses are open, regardless of age and citizenship limitations to those who graduated in Italy, those who are going to graduate within the maximum term of 31 October of the same year and those who graduated in foreign universities (the equivalence of the foreign degree will be determined by the PhD selection committee). Each doctoral course provides a single method of candidates selection, based on the evaluation of qualifications, supplemented by the evaluation of the proposed research projects and an oral test. In the call, access criteria, assessment of qualifications and of the tests are detailed. Admission tests may be performed in foreign language different from the Italian one, even with the help of informatics and electronics tools, suitable to verify the identity of the candidate. We have an agent (Candidate) who is of course a resource-bounded one, in the sense that he/she has a certain background knowledge (non omniscient) and explicit timed beliefs and forms new explicit timed beliefs by means of the mental operations illustrated before.

The University communicates information about deadlines to the candidates (among which our agent) through the University website. Let us assume for instance:

- graduation by 31 October;
- document submission by 26 August;
- notification of admission to the oral test 30 September;
- oral test 10 October;

- notification of admission to the Phd Program 15 October;
- PhD program stars from 1th November;
- sign up by 31 October.

Let us consider the following time variables:

$t_0 =$ current moment of time, $t_1 = 26$ August, $t_2 = 30$ september,
$t_3 = 10$ October, $t_4 = 15$ October, $t_5 = 31$ October,
$t_6 = 1$ November.

Our agent i (candidate) has in his long-term memory the following rules:

1. $K_i\big(\Box(\textit{want-phd}(0,\infty) \rightarrow \textit{take-part-selection}(0,\infty))\big)$.
   This rule indicates that if an agent is graduated, she can always apply for a selection for a PhD course;

2. $K_i\big(\Box_{[t_0,t_1]}(\textit{document-ready}(T,T) \rightarrow \Box_{[T+1,t_1]}\textit{submit-document}(T_a,T_a))\big)$.
   This rule indicates that if a candidate has the phd documents ready at any moment $T \in [t_0,t_1]$ then at the next moment until t_1 she can submit the documents.

3. $K_i\big(\Box_{[t_0,t_5]}(\textit{want-graduate}(T_c,T_c) \rightarrow \textit{apply-phd}(T_c+1,T_c+1))\big)$.
   The meaning of this rule is that it is always true within the interval $[t_0,t_5]$ that, if the agent want to graduate before t_5 she can apply for the phd program;

4. $K_i\big(\Box_{[t_1,t_3]}((\textit{submit-document}(T_e,T_e) \wedge (\Box_{[t_1,t_2]}\textit{notification}(T_f,T_f)) \rightarrow \textit{prepare-oral}(T_f+1,T_f+1))\big)$.
   This rule indicates that if the candidate submitted the documents and she has received the notification by t_2 then she can start preparing the oral test by t_3;

5. $K_i\big(\Box_{[t_3,t_4]}\textit{oral-done}(t_3,t_3) \rightarrow \textit{wait-list}(t_3+1,T_g))\big)$.
   This rule indicates that if the candidate have done the oral test in t_3 than she has to wait for the publication of the ranking which can be published in a time interval $T_g \in [t_3+1,t_4]$;

6. $K_i\big(\Box_{[t_4,t_6]}(\textit{wait-list}(T_h,T_h) \rightarrow \Box_{[t_5,t_6]}\textit{sign-in}(T_h+1,T_i))\big)$.
   This rule indicates that if the agent sees that it is part of the list of admitted candidates than she can enroll in the PhD course until t_6;

7. $K_i\big(\Box_{[t_6,\infty]}(\textit{sign-in}(T_j,T_j) \rightarrow \textit{start-PhD}(t_6,\infty))\big)$.
   This rule indicates that if the agent has enrolled in the PhD course then she can start the program from t_6.

The use of mental operations is the same as in the previous example (1.5).

1.8 A TEMPORAL MODULE FOR LOGICAL FRAMEWORKS

In [134] we consider the T function, defined before in (1.6.2), like a *temporal module* which can be adopted to "temporalize" many logical framework. In the literature, different kind of timed logical frameworks exist, where time is specified directly using hybrid logics (cf., e.g., [10]), temporal epistemic logic (cf., e.g., [86]) or simply by using Linear Temporal Logic, but in any case none of these adopt our method. We have exploited this module in two different settings. The first one is the memory management, which we have seen in the previous sections, and the second is a logical framework for reasoning about agents' cognitive attitudes; many formal logics have been proposed for reasoning about concepts taken from qualitative decision theory. Lorini in [122] proposes a general logical framework for reasoning about agents' cognitive attitudes of both epistemic type and motivational type.

In this section I first recall the T function and then I explain how we have applied our time module in this different domain.

1.9 TIME MODULE

In this section we introduce our *Time Module* which is the "time" function T (1.6.2) that associates to each formula the time interval in which this formula is true. We assume that each atom has two arguments representing time instants. For the sake of simplicity, as we concentrate on these arguments, we ignore all the other arguments; i.e., we assume each atom to be of the form $p(t_1, t_2)$.

- $T(p(t_1, t_2)) = [t_1, t_2]$, which stands for *"p is true in the time interval* $[t_1, t_2]$*"* where $t_1, t_2 \in \mathbb{N}$; as a special case we have $T(p(t_1, t_1)) = t_1$, which stands for *"p is true in the time instant* t_1*"* where $t_1 \in \mathbb{N}$ (time instant);

- $T(\neg p(t_1, t_2)) = [t_1, t_2]$, which stands for *"p is not true in the time interval* $[t_1, t_2]$*"* where $t_1, t_2 \in \mathbb{N}$;

- $T(\varphi \text{ op } \psi) = T(\varphi) \uplus T(\psi)$ with *op* $\in \{\vee, \wedge, \rightarrow\}$, which means the unique smallest interval including both $T(\varphi)$ and $T(\psi)$.

This basic definition, although simple, is able to incorporate a concept of time in virtually any logical formalism by creating a link between syntax ans semantics. Naturally, the T function must then be customized to accommodate the operations which are proper of the 'host' formalism.

1.10 TEMPORAL DYNAMIC LOGIC OF COGNITIVE ATTITUDES

The Temporal Dynamic Logic of Cognitive Attitudes (T-DLCA) is an extension of Dynamic Logic of Cognitive Attitudes (DLCA), presented in [122]. In our extension we introduce the concept of time through a particular function that assigns a "timing" to knowledge, belief, strong belief, conditional belief, desire, strong desire, comparative desirability and choice.

1.10.1 T-DLCA SYNTAX

In our scenario we fix $Atm = \{p(t_1, t_2), q(t_3, t_4), ... , h(t_i, t_j)\}$ as in the T-LEK framework. Let $Nom = \{x(t_1), y(t_2), , ... , z(t_j)\}$ where x, y, z are nominals, which name individual states in models, where t_j are time instants where $j \in \mathbb{N}$. Moreover, Nom is disjoint from Atm and let Agt be a finite set of agent.

Below is the definition the language $\mathcal{L}_{TDLCA}$ where $i \in Agt$, $p(t_1, t_2) \in Atm$ and $x(t) \in Nom$:

$$\begin{array}{lll} \pi, \lambda & := & \equiv_i \mid \preceq_{i,P} \mid \preceq_{i,D} \mid \preceq^{\sim}{}_{i,P} \mid \preceq^{\sim}{}_{i,D} \mid \pi; \lambda \mid \pi \cup \lambda \mid \pi \cap \lambda \mid -\pi \mid \varphi? \\ \varphi, \psi & := & p(t_1, t_2) \mid x(t) \mid \neg\varphi \mid \varphi \wedge \psi \mid [\pi]\varphi \end{array}$$

Other Boolean connectives are defined from $\neg$ and $\wedge$ as usual. So, π represents programs which is the basic construct of Dynamic Logic. We can called them *Cognitive Programs* to underline that we are working on reasoning about agents cognitive attitudes; in fact, π corresponds to a particular configuration of agents cognitive states. As in Dynamic Logic $\pi; \lambda$ stands for "do π followed by λ", $\pi \cup \lambda$ stands for "do π or λ", $\pi \cap \lambda$ stands for "do π and λ", $-\pi$ is the inverse, $\varphi?$ stands for "proceed if φ is true, else fail"; instead $\equiv_i$, $\preceq_{i,P}$, $\preceq_{i,D}$ describe agents knowledge, plausibility and desirability respectively, and $\preceq^{\sim}{}_{i,P}$, $\preceq^{\sim}{}_{i,D}$ are the complements of $\preceq_{i,P}$, $\preceq_{i,D}$. Also the formula $[\pi]\varphi$ has to be read "φ is true, according to the program π"; obviously we have different meanings based on the π we choose, first of all $[\equiv_i]\varphi$ which stands for "φ is true according to what agent i knows", $[\preceq^{\sim}{}_{i,P}]\varphi$ which stands for "φ is true at all states that, according to agent i, are at least as plausible as the current one" while $[\preceq^{\sim}{}_{i,P}]\varphi$ stands for "φ is true at all states that, according to agent i, are not at least as plausible as the current one". For $\preceq_{i,D}$ and $\preceq^{\sim}{}_{i,D}$ it is enough to replace "plausible" with "desirable" in the definition.

1.10.2 T-DLCA SEMANTIC

Semantic of T-DLCA is based on a set W of worlds; we have the valuation function: $V : W \to 2^{Atm \cup Nom}$. We extend the function T from (1.9) as follows:

- $T(x(t)) = t$
- $T([\pi]\varphi) = T(\varphi)$.

Definition 1.10.1 *A* T-DLCA model *is a tuple* $M = \langle W; (\preceq_{i,P})_{i\in Agt}; (\preceq_{i,D})_{i\in Agt}; (\equiv_i)_{i\in Agt}; R; V; T\rangle$ *where:*

- W *is the set of worlds defined as in the previous setting;*
- *for every* $i \in Agt$, $\preceq_{i,P}$ *and* $\preceq_{i,D}$ *are preorders on* W *and* $\equiv_i$ *is an equivalence relation on W such that for all* $\tau \in P, D$ *and for all* $w_I, v_J \in W$*:*
  1. $\preceq_{i,\tau} \subseteq \equiv_i$ *which means that for an agent to compare the plausibility or the desirability of two states, this states have to be in the same interval I;*
  2. *if* $w_I \equiv_i v_J$ *then* $w_I \preceq_{i,\tau} v_J$ *or* $w_I \preceq_{i,\tau} v_J$ *with* $I = J$ *which means that the plausibility or the desirability of two states are always comparable if these states are in the same interval I;*
- $V : W \to 2^{Atm\cup Nom}$ *is the valuation function and for all* $w_I, v_J \in W$ *and* $V_{Nom}(w_I) = Nom \cap V(w_I)$*:*
  1. $V_{Nom}(w_I) \neq \varnothing$*;*
  2. *if* $V_{Nom}(w_I) \cup V_{Nom}(v_J) \neq \varnothing$ *then* $w_I = v_J$ *with* $I = J$*;*
- T *is the "time" function;*
- $R_\pi \subseteq W \times W$ *is a binary relation which works in the following way based on* π*:*
  1. $w_I R_{\equiv_i} v_I$ *iff* $w_I \equiv_i v_I$*;*
  2. $w_I R_{\preceq_{i,\tau}} v_I$ *iff* $w_I \preceq_{i,\tau} v_I$*;*
  3. $w_i \preceq^{\sim}_{i,\tau} v_I$ *iff* $w_I \equiv_i v_I$ *and* $w_I \not\preceq_{i,\tau} v_I$*;*
  4. $w_I R_{\pi;\lambda} v_I$ *iff* $\exists z_I \in W : w_I R_\pi z_I$ *and* $z_I R_\lambda v_I$*;*
  5. $w_I R_{\pi\cap\lambda} v_I$ *iff* $w_I R_\pi v_I$ *or* $w_I R_\lambda v_I$*;*
  6. $w_I R_{\pi\cup\lambda} v_I$ *iff* $w_I R_\pi v_I$ *and* $w_I R_\lambda v_I$*;*
  7. $w_I R_{-\pi} v_I$ *iff* $v_I R_\pi w_I$.

The properties of the valuation function capture the basic properties of nominals: the uniqueness in associating a single nominal with a state.

Truth conditions for T-DLCA formulas are defined inductively as follows:

- $M, w_I \models p(t_1, t_2)$ iff $p(t_1, t_2) \in V(w_I)$ and $T(p(t_1, t_2)) \subseteq I$;

- $M, w_I \models x(t)$ iff $x(t) \in V(w_I)$ and $T(x(t)) \subseteq I$;
- $M, w_I \models \neg\varphi$ iff $M, w_I \not\models \varphi$ and $T(\neg\varphi) \subseteq I$;
- $M, w_I \models \varphi \wedge \psi$ iff $M, w_I \models \varphi$ and $M, w_I \models \psi$ with $T(\varphi), T(\psi) \subseteq I$;
- $M, w_I \models [\pi]\varphi$ iff $\forall v_I \in W$: if $w_I R_\pi v_I$ then $M, v_I \models \varphi$ with $T([\pi]\varphi) \subseteq I$;
- $M, w_I \models [?\varphi]\varphi$ iff $\forall v_I \in W$: if $w_I R_{?\varphi} v_I$ then $M, v_I \models \varphi$ with $T([?\varphi]\varphi) \subseteq I$ where $w_I R_{?\varphi} v_I$ iff $w_I = v_I$ and $M, w_I \models \varphi$.

Remark 1.10.1 *This Time Module can be adopted in any logical framework, by extending the definition appropriately.*

1.11 Conclusions

In this chapter I explained my work about Modal Logic used for autonomous system's memory. My work stems from the need to improve the features of DALI [48], which is a logic programming Agent-Oriented language created by my research group during the years; we wanted to manage problems caused by the interaction between the agent and the external environment. To this aim we extended an existing work of Balbiani, Fernández-Duque and Lorini [16], where they proposed a (partial) formalization of SOAR architecture in modal logic (LEK/DLEK). We proposed two different formalization:

- *TLEK/TDLEK*, where we introduced explicit time instants and time intervals in formulas;
- *T-LEK/T-DLEK*, where we introduced the T function, which manages the interval when an atom is true; through this function we are also able to assign a "timing" to the epistemic operators B and K.

With regard to complexity for the mono agent case for LEK it has been proved that the satisfiability problem is decidable and it has been proved to be in NP-complete, instead for DLEK it has been conjectured to be PSPACE. It is easy to believe that our extensions cannot spoil decidability because the T function do not interfere. Inference steps to derive new beliefs are analogous to D-LEK: just one modal rule at a time is used and a sharp separation is postulated between the working memory, where inference is performed, and the long-term memory.
We also considered the T function, defined in T-DLEK, like a *temporal module* which can be adopted to "temporalize" many logical framework. We have exploited this module in two different settings. The first one is the memory management (T-DLEK) and the second is a logical framework for reasoning about agents' cognitive attitudes (T-DLCA). Future developments for memory management could be the study of how to encode information from the working memory to the long term memory under certain conditions as illustrated in figure 1.2.

CHAPTER 2

TOWARDS A LOGIC OF "INFERABLE"

2.1 INTRODUCTION

In this chapter I discuss my last work about logic in which we try to formalize the cognitive state of an agent after performing an inferential action (or mental operation); I keep talking about the cognitive aspect of autonomous systems, and I show a particular logical framework (Logic of "Inferable" (*L-DINF*)) which reasons about if an action can be performed or not and also how many steps are required for an agent to perform it. This logic is vary similar to those illustrated in the previous chapter but we have inserted the concept of step and executability. Before going into details of the new logic, let's try to explain the starting point and the related works.
Our logic can be seen as a logic of "explicit belief" and in there are many kind of it in literature; in fact logics of awareness have been studied in the recent years starting from the seminal work of Fagin & Halpern [87]. These logics distinguish between awareness, implicit belief and explicit belief. The crucial difference between our logic and existing logics of awareness is that the latter make no use of concepts as 'reasoning' or 'inference'. On the contrary, *L-DINF* provides a constructive theory of explicit beliefs, as it accounts for the perceptive and inferential steps leading from an agent's knowledge and beliefs to new beliefs and also we added two other important aspect: "steps" and "executability". Moreover, This aspect of epistemic attitudes is something our theory shares with other approaches in the literature including the dynamic theory of evidence-based beliefs by [154], that also use a neighborhood semantics for the notion of evidence, the sentential approach to explicit beliefs and their dynamics by [108], the dynamic theory of explicit and implicit beliefs by [161] and the dynamic logic of explicit beliefs and knowledge by [17].
The logic of inference stems from Velázquez-Quesada [160] and the logical system DES4_n by Duc [79], which share a similar view with our logic. In particular, Velázquez-Quesada shares with us the idea of modelling inference steps by means of dynamic operators in the style of dynamic epistemic logic (DEL). But in this work he does not distinguish the concept of explicit belief, the concept of background knowledge and the executability of the action.
The system shown in [79] shares with our logic the idea that an agent gets to know (or believe) something by performing inferences, and making inferences takes time. Nonetheless, while in our logic inferential operations are represented both at the syntactic level,

via dynamic operators in the DEL style, and a semantic level, as model update operations, in Duc's system and its formal semantics they are not. In addition, we check whether an action can be performed or not and how many steps are needed to perform it.
Our constructive approach to explicit beliefs also distinguishes *L-INF* from Active logics [82, 83], in which the basic semantics includes three components: (i) an agent's belief set, identifying all facts that an agent explicitly believes, (ii) an observation function, identifying all facts that an agent observes at a given time point, and (iii) an inference function, specifying what an agent should believe at the next time point on the basis of the application of the inference rules she possesses on her belief set, given her actual observations. Nonetheless, there are important differences between active logics and our logic *L-INF*. First of all, active logics do not belong to the family of modal logics, while *L-INF* does. First of all, while active logics provide models of reasoning based on long-term memory and short-term memory (or working memory), they do not distinguish between the notion of explicit belief and the notion of background knowledge, conceived as different kinds of epistemic attitudes. Second, our logic accounts for a variety of inferential action (or mental operation) that have not been explored in the active logic literature and are vary important to infer new beliefs. These actions correspond to basic operations of mind-reading in the sense of Theory of Mind (ToM) [96], in fact they are mental and not physical ones.

2.1.1 LOGICAL FRAMEWORK

L-DINF, like *TDLEK* and *T-DLEK*, is a logic which consists of a static component and a dynamic one. The static component, called *L-INF*, is a logic of explicit beliefs and background knowledge. The dynamic component, called *L-DINF*, extends the static one with dynamic operators capturing the consequences of the agents' inferential operations on their explicit beliefs as well as a dynamic operator capturing what an agent can conclude by performing some inferential operation in her repertoire of inferential operations.

2.1.2 SYNTAX

Assume a countable set of atomic propositions $Atm = \{p, q, \ldots\}$. By $Prop$ we denote the set of all propositional formulas, i.e. the set of all Boolean formulas built out of the set of atomic propositions Atm.

The language of *L-DINF*, denoted by $\mathcal{L}_{L\text{-}DINF}$, is defined by the following grammar in Backus-Naur form:

$$\begin{array}{lll} \alpha & ::= & \vdash(\varphi,\psi) \mid \cap(\varphi,\psi) \mid \downarrow(\varphi,\psi) \\ \phi,\psi & ::= & p \mid exec(\alpha) \mid \neg\varphi \mid \varphi \wedge \psi \mid B\,\varphi \mid K\,\varphi \mid [\alpha]\varphi \mid \Diamond\varphi \end{array}$$

where the language of inferential actions of type α is denoted by $\mathcal{L}_{\mathsf{ACT}}$, p ranges over Atm and X is a finite set of inferential actions from $\mathcal{L}_{\mathsf{ACT}}$. Obviously the static part,

L-INF, has the same definition by removing the inferential action. The other Boolean constructions are defined from p, $\neg$ and $\wedge$ in the standard way. The formula $B\,\varphi$ is read "the agent explicitly believes that φ is true" or, more shortly, "the agent believes that φ is true". Explicit beliefs are accessible in the working memory and are the basic elements of the agents' reasoning process, according the logic of local reasoning by Fagin & Halpern [87]. An effect of this approach is that agents cannot distinguish between logically equivalent formulas, i.e., if two facts φ and ψ are logically equivalent and an agent explicitly believes that φ is true, then she believes that ψ is true as well. There are other approaches, such as justification logics [143], that do not have this feature.
Unlike explicit beliefs, background knowledge is assumed to satisfy 'omniscience' principles like closure under conjunction and known implication, closure under logical consequence and introspection. Specifically, K is nothing but the well-known S5 operator for knowledge widely used in computer science. The fact that background knowledge is closed under logical consequence is justified by the fact that we conceive it as a kind of deductively closed *belief base*. Specifically, we assume background knowledge to include all facts that the agent has stored in her long-term memory (LTM), after having processed them in her working memory (WM), as well as all logical consequences of these facts.

The formula $[\alpha]\varphi$ should be read "φ holds after the inferential operation (or inferential action) α is performed by the agent". We distinguish three types of inferential operations α which allow us to capture some of the dynamic properties of explicit beliefs and background knowledge: $\vdash(\varphi,\psi)$, $\cap(\varphi,\psi)$ and $\downarrow(\varphi,\psi)$. This operations characterize the basic operations of forming explicit beliefs via inference. Specifically:

- $\downarrow(\varphi,\psi)$ is the inferential operation which consists in inferring ψ from φ in case φ is believed and, according to an agent's background knowledge, ψ is a logical consequence of φ. In other words, by performing this inferential operation, an agent tries to retrieve from her background knowledge in long-term memory the information that φ implies ψ and, if she succeeds, she starts to believe ψ;
- $\cap(\varphi,\psi)$ is the inferential operation which consists in closing the explicit belief that φ and the explicit belief that ψ under conjunction. In other words, $\cap(\varphi,\psi)$ characterizes the inferential operation of deducing $\varphi \wedge \psi$ from the explicit belief that φ and the explicit belief that ψ;
- $\vdash(\varphi,\psi)$ is the inferential operation which consists in inferring ψ from φ in case φ is believed and, according to an agent's working memory, ψ is a logical consequence of φ. With this last operation we operate directly on the working memory without retrieving anything from the background knowledge. We assume that, differently from explicit beliefs, background knowledge is irrevocable in the sense of being stable over time.

Remark 2.1.1 *We have chosen only these three mental actions because the physical actions (performed through a specific agent language which we are not here to specify) are perceived and learned through the acquisition of new beliefs; in fact an agent knows if he has performed these action and if these action has been successful or not.*

The atomic formulas $exec(\alpha)$ have to be read "α is an inferential action that the agent can perform".

Finally, the formula $\Diamond\varphi$ has to be read "the agent can ensure φ by executing some inferential action in her repertoire". The interesting aspect of our language is that it allows us to express the concept of "being able to infer φ in k inferential steps". Specifically, let us inductively define:

- $\Diamond^0\varphi = \varphi$;
- $\Diamond^{k+1} = \Diamond\Diamond^k\varphi$

The formula $\Diamond^k B\,\varphi$ represents the fact that the agent is capable of inferring φ in k steps.

Remark 2.1.2 *The latter approximates the notion of time of computation as the number of steps that are sufficient for an agent to draw a certain conclusion from her initial beliefs.*

2.1.3 SEMANTICS

The main semantics notion are outlined in the following definition of ***L-INF*** model which provides the basic components for the interpretation of the static logics:

Definition 2.1.1 *We define a model to be a tuple $M = (W, N, R, E, V)$ where:*

- *W is a set of worlds or situations;*
- *$R \subseteq W \times W$ is an equivalence relation on W;*
- *$N : W \longrightarrow 2^{2^W}$ is a neighborhood function such that for all $i \in Agt$, $w, v \in W$ and $X \subseteq W$:*

  (C1) *if $X \in N(w)$ then $X \subseteq R(w)$,*

  (C2) *if wRv then $N(w) = N(v)$;*

- *$E : W \longrightarrow 2^{\mathcal{L}_{\mathsf{ACT}}}$ is an executability function such that for all $w, v \in W$:*

  (D1) *if wRv then $E(w) = E(v)$;*

- *$V : W \longrightarrow 2^{Atm}$ is a valuation function.*

For every $w \in W$, $R(w) = \{v \in W \mid wRv\}$ identifies the set of situations that the agent considers possible at world w. In cognitive terms, $R(w)$ can be conceived as the set of all situations that the agent *can* retrieve from her long-term memory and reason about them. More generally, $R(w)$ is called the agent's *epistemic state* at w.

For every $w \in W$, $N(w)$ defines the set of all facts that the agent explicitly believes at world w, a fact being identified with a set of worlds. More precisely, if $A \in N(w)$ then, at world w, the agent has the fact A under the focus of her attention and believes it. $N(w)$ is called the agent's explicit *belief set* at world w.

$E(w)$ is the set of mental operations that the agent can execute at w.

Constraint (**C1**) just means that an agent can have explicit in her mind only facts which are compatible with her current epistemic state. According to Constraint (**C2**), if world v is compatible with the agent's epistemic state at world w, then the agent should have the same explicit beliefs at w and v. Moreover, Constraint (**D1**) means that an agent always knows the actions which he can perform and those he cannot.

Truth conditions of *L-DINF* formulas are inductively defined as follows.

For a model $M = (W, N, R, E, V)$, a world $w \in W$, a formula $\varphi \in \mathcal{L}_{L\text{-}INF}$, and an action α, we define the truth relation $M, w \models \varphi$ and a new model M^α by simultaneous recursion on α and φ as follows. Below, we write

$$\|\varphi\|_w^M = \{v \in W : wRv \text{ and } M, v \models \varphi\}$$

whenever $M, v \models \varphi$ is well-defined. Then, we set:

- $M, w \models p$ iff $p \in V(w)$
- $M, w \models exec(\alpha)$ iff $\alpha \in E(w)$
- $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$
- $M, w \models \varphi \wedge \psi$ iff $M, w \models \varphi$ and $M, w \models \psi$
- $M, w \models B\,\varphi$ iff $||\varphi||_w^M \in N(w)$
- $M, w \models K\,\varphi$ iff $M, v \models \varphi$ for all $v \in R(w)$
- $M, w \models [\alpha]\varphi$ iff $M^\alpha, w \models \varphi$
- $M, w \models \Diamond\varphi$ iff $\exists\alpha \in E(w)$ s.t. $M^\alpha, w \models \varphi$

where $M^{\alpha} = (W, N^{\alpha}, R, E, V)$ and N^{α} is defined as follows. For $w \in W$, set

$$N^{\downarrow(\psi,\chi)}(w) = \begin{cases} N(w) \cup \{||\chi||^{M}_{w}\} & \text{if } M, w \models B\,\psi \wedge K\,(\psi \rightarrow \chi) \\ N(w) & \text{otherwise} \end{cases}$$

$$N^{\cap(\psi,\chi)}(w) = \begin{cases} N(w) \cup \{||\psi \wedge \chi||^{M}_{w}\} & \text{if } M, w \models B\,\psi \wedge B\,\chi \\ N(w) & \text{otherwise} \end{cases}$$

$$N^{\vdash(\psi,\chi)}(w) = \begin{cases} N(w) \cup \{||\chi||^{M}_{w}\} & \text{if } M, w \models B\,\psi \wedge B\,(\psi \rightarrow \chi) \\ N(w) & \text{otherwise} \end{cases}$$

We write $\models_{\text{L-DINF}} \varphi$ to denote that φ is true in all worlds w of every L-DINF model M.

Property 2.1.1 *As consequence of previous definitions we have the following:*

- $\models_{\textit{L-INF}} (K(\varphi \rightarrow \psi)) \wedge B\,\varphi) \rightarrow [\downarrow(\varphi, \psi)]\,B\,\psi$.
  Namely, if an agent has φ as one of its beliefs and has $K(\varphi \rightarrow \psi)$ in its background knowledge, then as a consequence of the action $\downarrow(\varphi, \psi)$ the agent starts believing ψ;
- $\models_{\textit{L-INF}} (B\varphi \wedge B\psi) \rightarrow [\cap(\varphi, \psi)]B_{(}\varphi \wedge \psi)$.
  Namely, if an agent has φ and ψ as beliefs, then as a consequence of the action $\cap(\varphi, \psi)$ the agent i starts believing $\varphi \wedge \psi$;
- $\models_{\textit{L-INF}} (B(\varphi \rightarrow \psi)) \wedge B\,\varphi) \rightarrow [\vdash(\varphi, \psi)]\,B\,\psi$.
  Namely, if an agent has φ as one of its beliefs and has $B(\varphi \rightarrow \psi)$ in its working memory, then as a consequence of the action $\vdash(\varphi, \psi)$ the agent starts believing ψ;

The proof of these property are the same as Property 1.3.2 and Property 1.3.3 in the previous chapter.

2.2 AXIOMATIZATION

The *L-INF* and *L-DINF* axioms are:

1. $(K\,\varphi \wedge K(\varphi \rightarrow \psi)) \rightarrow K\,\psi$;
2. $K\,\varphi \rightarrow \varphi$;
3. $\neg K\,(\varphi \wedge \neg\varphi)$;

4. $K\,\varphi \rightarrow K\,K\,\varphi$;
5. $\neg K\,\varphi \rightarrow K\,\neg K\,\varphi$;
6. $B\,\varphi \wedge K\,(\varphi \leftrightarrow \psi) \rightarrow B\,\psi$;
7. $B\,\varphi \rightarrow K\,B\,\varphi$;
8. $\frac{\varphi}{K\,\varphi}$;
9. $[\alpha]p \leftrightarrow p$;
10. $[\alpha]\neg\varphi \leftrightarrow \neg[\alpha]\varphi$;
11. $exec(\alpha) \wedge [\alpha]\varphi \rightarrow \Diamond\varphi$;
12. $exec(\alpha) \rightarrow K\,(exec(\alpha))$;
13. $[\alpha](\varphi \wedge \psi) \leftrightarrow [\alpha]\varphi \wedge [\alpha]\psi$;
14. $[\alpha]K\,\varphi \leftrightarrow K\,([\alpha]\varphi)$;
15. $[\downarrow(\varphi,\psi)]B\,\chi \leftrightarrow B\,([\downarrow(\varphi,\psi)]\chi) \vee ((B\,\varphi \wedge K\,(\varphi \rightarrow \psi)) \wedge K\,([\downarrow(\varphi,\psi)]\chi \leftrightarrow \psi))$;
16. $[\cap(\varphi,\psi)]B\,\chi \leftrightarrow B\,([\cap(\varphi,\psi)]\chi) \vee ((B\,\varphi \wedge B\,\psi) \wedge K\,([\cap(\varphi,\psi)]\chi \leftrightarrow (\varphi \wedge \psi))$;
17. $[\vdash(\varphi,\psi)]B\,\chi \leftrightarrow B\,([\vdash(\varphi,\psi)]\chi) \vee ((B\,\varphi \wedge B\,(\varphi \rightarrow \psi)) \wedge B\,([\vdash(\varphi,\psi)]\chi \leftrightarrow \psi))$;
18. $\frac{\psi\leftrightarrow\chi}{\varphi\leftrightarrow\varphi[\psi/\chi]}$;
19. $p \rightarrow \Diamond p$;
20. $\Diamond(\varphi \wedge \psi) \rightarrow \Diamond\varphi \wedge \Diamond\psi$;
21. $\Diamond\varphi \rightarrow \Diamond\Diamond\varphi$;
22. $\Diamond B\,\varphi \rightarrow B\,\Diamond\varphi$;
23. $\Diamond K\,\varphi \rightarrow K\,\Diamond\varphi$.

We write *L-DINF* $\vdash \varphi$ which signifies that φ is a theorem of $L\text{-}DINF$. Thanks to the previous axioms, *L-INF* and *L-DINF* are sound for the class of *L-INF* models.

2.3 CONCLUSIONS

In this chapter I completed my discussion about the cognitive aspects of autonomous system, and I showed a particular logical framework where I have inserted the concept of step and executability. This logic is called *L-DINF* and as *TDLEK* and *T-DLEK*, is a modal logic which consists of a static component and a dynamic one. This logic is vary similar to those illustrated in the previous chapter but we have inserted the concept of step and executability; the first is to understand how many steps an agent takes to perform an action, the second illustrates whether an agent is actually able to perform an action.

For future work we will prove that *L-INF* is strongly complete which can be achieved by using a non-standard canonical model argument, as in [15]. In fact, we could have considered the canonical model, as in the previous chapter, if we had not considered the $\diamond$. We will extend L-INF to the multi-agents case and there is an intention to insert the concept of budget: an integer that represents how much an agent can spend of his own resources in the world w. This is important to better represent the fact that the agent is resource-bounded. Moreover we have to compute the complexity and extend the group of inferential actions, which we consider.

CHAPTER 3

ADVANCED ARCHITECTURE: K-LAYER ACE

3.1 INTRODUCTION

In this chapter I discuss a research topic of my research group to which I have provided active contribution: Cyber-Physical Systems (CPSs), Multi-Context Systems and Component-based Agents Environments. Here I focus my self working on a particular, innovative kind of architecture called *K-Level ACE* ([69]). Before going into the details on the new architecture, let's try to outline the context.

There are nowadays many application fields where agents and multi-agent systems are situated in complex, open, and dynamic computational environments which include heterogeneous software components, physical devices and sensors including wearable devices, third part services, data centers, expert systems and other knowledge sources available on the Internet. The availability of such components evolves in time, as new knowledge can be discovered, and components may join or leave the environment, or become momentarily unreachable. Such environments can actually constitute Cyber-Physical Systems (CPSs) [109], and they may include physical components that interact with or are integrated into the computational "ecosystem".

One example of such a scenario is for instance the F&K (Friendly-and-Kind) system [2] proposed for applications the E-Health domain, and depicted in Figure (3.1).

In this architecture, each patient is in charge of a Personal Monitoring Agent (PMA). This agent will interact with the patient in via a friendly interface, and supervise by means of suitable sensors the patient's welfare and health condition and the correct administration of therapies. So, a PMA must be aware of all illnesses of the assigned patient and must be able to cope with their comorbidity. The PMA manages the patient's medical history and records, is aware of the therapies and is able to supervise drugs administration according to prescription. In case of anomalous symptoms (detected via the PMA's complex event processing capabilities), the PMA will access knowledge bases for symptom interpretation, diagnosis and management treatment. The PMA may interact with either the patient or a human specialist or both. Moreover, the various PMAs are meant to cooperate so as to share in an effective ways scarce resources such as medical doctors, nurses, ambulances, helicopters, and hospital beds. Therefore, F&Ks are "knowledge-intensive"

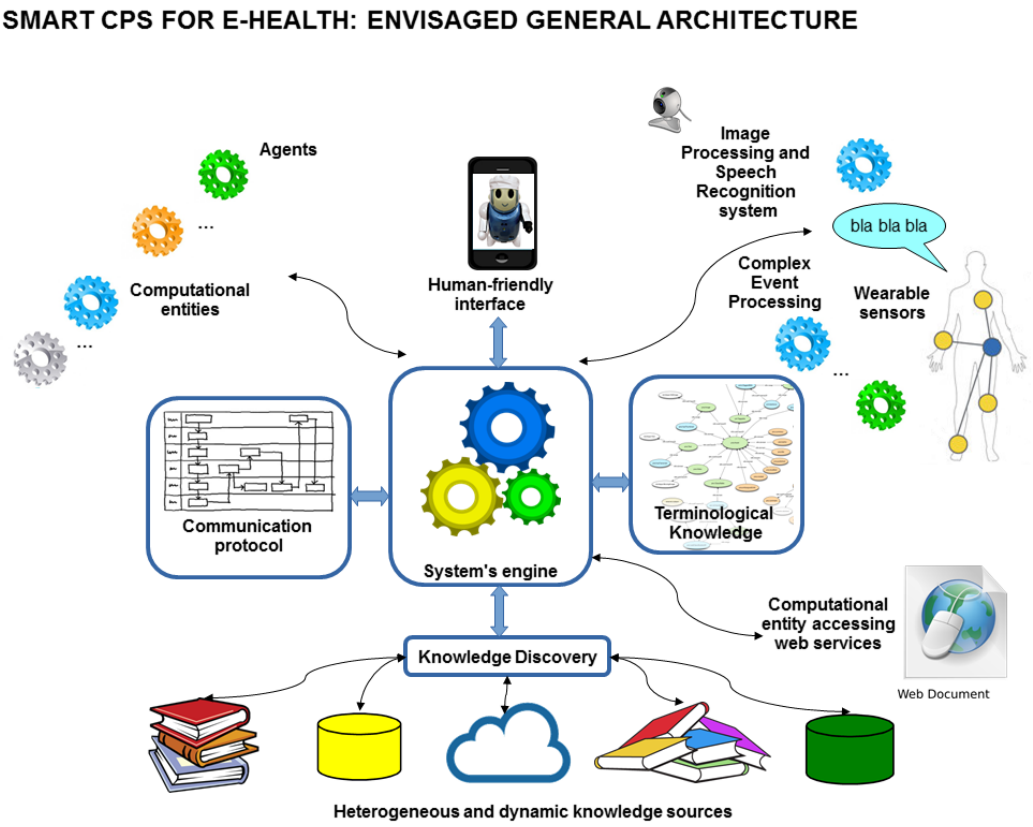

Figure 3.1: Friendly-and-Kind architecture

systems, providing flexible access to dynamic, heterogeneous, and distributed sources of knowledge and reasoning, within a highly dynamic computational environment consisting of computational entities, devices, sensors, and services available in the Internet and in the cloud.

Another application in a very different domain though with similar features (depicted in Figure 3.2) is aimed at Digital Forensics and Digital Investigations [65, 128]. Digital Forensics is a branch of criminalistics which deals with the identification, acquisition, preservation, analysis and presentation of the information content of computer systems, or in general of digital devices, in relation to crimes that have been perpetrated. The aim is to identify, categorize and formalize digital sources of evidence. The aim of the Evidence Analysis stage of Digital Forensics, or more generally of Digital Investigations, is to identify, categorize and formalize digital sources of evidence (or however sources of evidence represented in digital form). The objective is to organize such sources of proof into evidences, so as to make them robust in view of their discussion in court, either in civil or penal trials.

In work in progress we have identified a setting where an intelligent agent is in charge of supporting the human investigator in such activity. This is not exactly a Cyber-Phisical System, as the agent does not have direct access to sensors and actuators but rather elab-

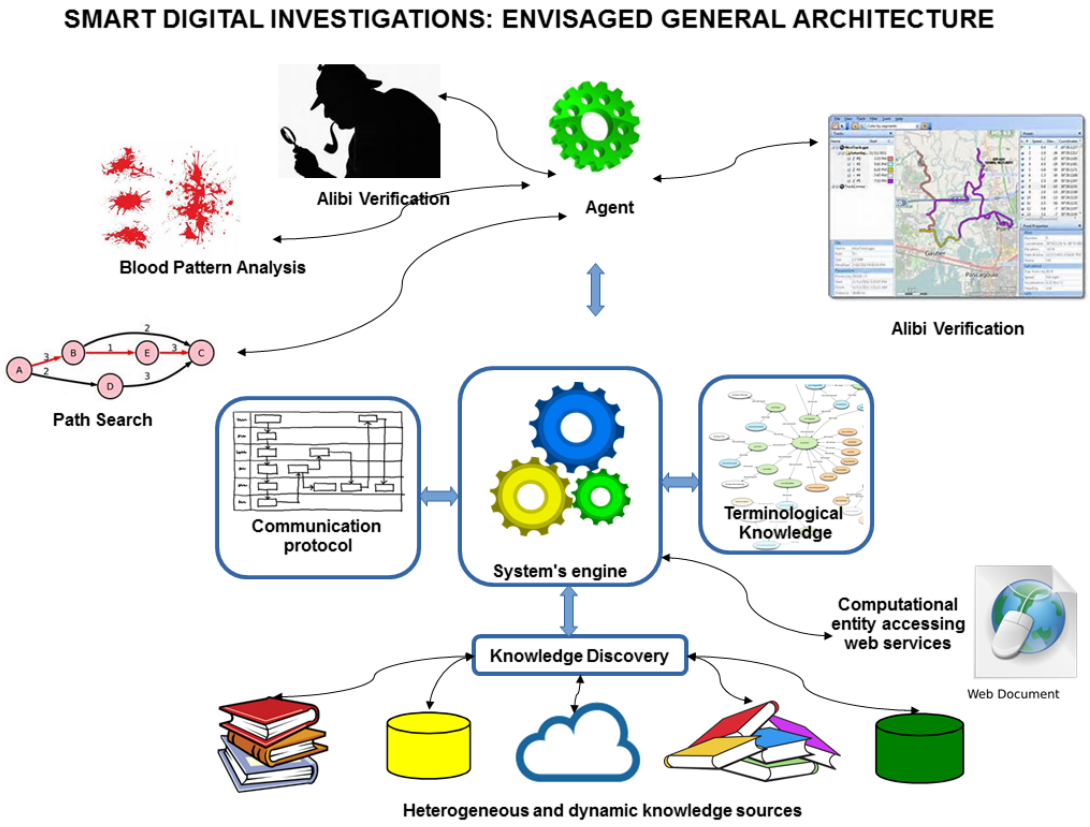

Figure 3.2: DyPES Architecture

orates the collected evidence. The agent should however identify, retrieve and gather the various kinds of potentially useful evidence, process them via suitable reasoning modules, and integrate the results into coherent evidence. In this task, the agent may need to retrieve and exploit knowledge bases concerning, e.g., legislation, past cases, suspect's criminal history, and so on. In the picture, the agent considers: results from blood-pattern analysis on the crime scene, which lead to model such a scene via a graph, where suitable graph reasoning may reconstruct the possible patterns of action of the murderer; alibi verification in the sense of a check of the GPS positions of suspects, so as to ascertain the possibility of her/him being present on the crime scene at the crime time; alibi verification in the sense of double-checking the suspect's declarations with digital data such as computer logs, videos from video-cameras situated on the suspect's path, etc. All the above can be integrated with further evidence such as the results of DNA analysis and others. The system can also include Complex Event Processing so as to infer from significant clues the possibility that a crime is being or will be perpetrated.

In reality, many of the involved data must be retrieved, elaborated, or checked from knowledge bases belonging to organizations which are external to the agent, and have their own rules and regulations for data access and data elaboration. Therefore, suitable modalities for data retrieval and integration must be established in the agent to correctly access such organizations.

A suitable denomination of such systems can be "Dynamic Proactive Expert Systems" (DyPES): in fact, both systems are aimed at supporting human experts and personnel or human users in a knowledgeable fashion, so they are reminiscent of the role of traditional expert systems. However, they are proactive in the sense that such systems have objectives (monitoring patients, managing resources, finding evidence) that they pursue autonomously, requiring human intervention only when needed. They are also dynamic, because they are able to exploit not only a predefined knowledge base: rather, they are equipped with a number of reasoning modules, and they are able to locate other such modules, and the necessary knowledge and reasoning auxiliary resources. In fact, DyPESs are characterized by "Knowledge-intensity", in the sense that in general a large amount of heterogeneous information and data must be retrieved, shared and integrated in order to reason within the system's domain. As seen, DyPESs can be Cyber-Physical Systems, and can be able to perform complex event processing, i.e., they are able to actively monitor event data so as to make automated decisions and take time-critical actions. As emphasized in both scenarios, such systems need a terminological component to ensure interoperability among the various elements, and well-defined communication modalities for knowledge exchange.

We propose a software architecture for DyPESs. The new architecture take as basic blocks DACMACS (Data-Aware Commitment-based managed Multi-Agent-Context Systems, [51, 54]) and ACEs (Agent Computational Environments), which are assembled togheter to build an entirely new structure. About DACMACSs and about ACEs (Agent Computational Environments, [50]). DACMACS is an extension of the work of [126] about DACMAS (Data-Aware Commitment-based managed Multi-Agent Systems). The latter provides a quite general model of multi-agent systems, with explicit elements of knowledge representation, as in fact the terminological component is specified in terms of DRL-Lite Description Logic [14], and an explicit communication component managed by an *Institutional* agent (in particular, communication in DACMASs is based upon *commitments* [145, 146], which constitute a relatively recent but very well-established approach). DACMACSs agents are however able to flexibly interact not only within a MAS like in DACMAS, but also with heterogeneous external information sources (called "contexts" after [36, 37, 38]), by means of suitable agent-oriented modalities. Moreover, both agents and external contexts may interchange not only queries and the relative answers, but also the ontological definitions they are based upon [54]. The ACE approach [50] empowers logical agents by equipping them with reasoning modules and auxiliary knowledge bases, possibly defined in different heterogeneous languages/formalisms, and providing a uniform interaction mechanism so that not only the approach is adaptable to any agent-oriented language save implementing this mechanism, but an ACE is potentially capable of reconfiguring itself when deemed useful.

The proposed architecture is called K-ACE standing for 'K-level ACE', and is designed as a multi-layered uniform architecture which smoothly integrates all features required for designing DyPESs. With K-ACE we intend to generalize, unify and empower the DACMACS-ACE features within a smoothly integrated uniform framework. Similarly to DACMACS, a K-ACE encompasses both agents and contexts, the latter providing knowl-

edge bases and operational capabilities to support the system's activities. Agents in a K-ACE are in particular ACEs, i.e., empowered modular agents capable to perform and integrate several forms of reasoning and to reconfigure themselves upon need. To represent the fact that such a system might be required to interact with external organizations about which no information is available beyond modalities for issuing requests and receiving results, components of a K-ACE can be other K-ACEs which can in turn encompass K-ACEs and so on recursively. So, a K-ACE is a multi-level structure where a K-ACE at a certain level may encompass other K-ACEs which are seen by the former system as operating at a "lower" level. A K-ACE "encompasses" lower-level K-ACEs in the sense that it is able to interact with them by locating knowledge sources therein and send them requests. This is done by means of a key element of each K-ACE, i.e., the "Institutional agent" which is the K-ACE entry points, as it receives requests to locate components with a certain role, and returns answers, possibly according to its own policies.

Communication among agents might occur in principle according to any reasonable protocol. However, to manage knowledge flow from-to agents and reasoning modules and external contexts and lower level K-ACEs we adopt as uniform, a very general and versatile mechanism the one of *bridge-rules*, inspired to those of MCSs (Multi-Context Systems) [36, 37, 38]. Bridge rules are thus a basic element of K-ACEs, as they allow knowledge to flow among components in a clearly-specified principled way. Bridge rules are in particular reminiscent of Datalog rules, where however each item of information can be obtained from a different context. Agents are not required to be aware of the system's structure in order to be able to retrieve the contexts that may provide some specific information. Rather, at each level the system includes a special entity, called "Institutional Agent", for locating components based upon their *role*; the *role* indicates the kind of information or of operation that a component is able to perform, along with operational information to retrieve and access the component itself. In case several options are available for given role, the *best preferred* can be selected according to some agent-specific criterion; redundancy in roles is a desirable feature, which helps to guarantee the retrieval of the needed information in most cases. Whenever the required component cannot be found at the present level but might be found at lower levels, the Institutional Agent locates within the system a suitable K-ACE and defers the search to its Institutional Agent, and so on recursively until a component with the desired role (or a more general one, if none is found) is retrieved. Institutional Agents are able to manage system dynamics in the sense of component or entire K-ACE that may join or leave the system, the latter either as a choice or as a result of circumstances such as network failure.

Overall, K-ACE have been devised as an architecture for modeling and realizing systems that operate in highly distributed, heterogeneous and dynamic environments, where knowledge location, elaboration and exploitation requires a high degree of flexible interaction among components and sub-systems, and where obtaining timely and reliable results may be crucial in the interest of a user.

3.2 TERMINOLOGY FOR K-ACES

In this Section we summarize the terms and acronyms that we use throughout the chapter, with a short explanation to facilitate reading.

- **MAS**: Multi-Agent Systems.
- **MCS**: Multi-Context System, a framework for the integration of knowledge obtained from several heterogeneous knowledge sources.
- **Contexts**: general denomination for heterogeneous knowledge sources, each one based upon its own specific logic/language/formalism.
- **Bridge Rules**: device for realizing knowledge integration in MCS; similar in syntax to datalog rules, each bridge rule occurs in a context; each condition in the body is associated to the indication of the context from which that piece of knowledge is to be obtained; the conclusion (head) is added to the context whenever each of the conditions composing the body becomes true (in the relative contexts).
- **mMCS**: Managed MCS, variant of MCS where the head of a bridge rule is added to a context after "management", i.e., after suitable elaboration.
- **Management function**: indicated with mng_i for context C_i, performs bridge-rule heads elaboration in mMCS.
- **DACMAS**: Data-Aware Commitment-Based Multi-Agent Systems, are Multi-Agent Systems equipped with a global Tbox and global ABox defined in Description Logics; the TBox and ABox provide shared knowledge to the agents composing the MAS; such agents communicate via Commitments.
- **DACMACS**: Data-Aware Commitment-Based Multi-Agent-Context Systems, integrate DACMAS and mMCS by making external contexts available to agents via bridge rules, that can occur also in agents, where can be activated proactively.
- **Institutional Agent**: is a special agent alway present in a DACMACS, responsible of managing message-passing among agents and of locating agents and contexts whithin the system, based on their roles; it can be seen from the outside as the system's entry-point.
- **ACE**: Agent Computational Environments, are empowered agents where each agent is organized in a modular structure encompassing agent program, heterogeneous reasoning modules and heterogeneous contexts. Bridge rules are the device for interchanging knowledge among the components.
- **K-ACE**: proposed architecture, where a component can in turn be a K-ACE, recusively over an arbitrary number of levels. Empowered Institutional agents are responsible for locating agents and context "routing" information within the structure.

3.3 BACKGROUND: MCS

Heterogeneous multi-context systems have been introduced in the seminal work of [95] in order to integrate different inference systems without resorting to non-classical logical systems. The device that they elaborated in order to interconnect such systems was called *bridge rules*, whose form was:

$$(c_1 : p_1), \ldots, (c_j : p_j) \Rightarrow (c : q)$$

where conclusion q was drawn in framework c in consequence of conclusions $p_1, \ldots, p_j$ having been drawn in $c_1, \ldots, c_j$ respectively. A semantics was provided in terms of local models of the composing inference systems plus compatibility conditions related to bridge-rule application.

Later, the idea has been further developed and generalized to non-monotonic reasoning domains in [35, 36, 37, 38] and other related papers. There, (Managed) Multi-Context systems aim at making it possible to build systems that need to access multiple possibly heterogeneous data sources, called “contexts”, by modeling the necessary information flow via “bridge rules”, whose form is similar to datalog rules with negation (cf., e.g., [8]). Bridge rules allow for inter-context interaction: in fact, as before, each element in their “body” explicitly includes the indication of the context from which information is to be obtained.

In order to account for heterogeneity of sources each context is supposed to be based on its own *logic*. Reporting from [36], a logic L is a triple $(KB_L; Cn_L; ACC_L)$, where KB_L is the set of admissible knowledge bases of L, which are sets of KB-elements (“formulas”); Cn_L is the set of acceptable sets of consequences, whose elements are data items or "facts" (in [36] these sets are called “belief sets”; we adopt the more neutral terminology of “data sets”); $ACC_L : KB_L \rightarrow 2^{Cn_L}$ is a function which defines the semantics of L by assigning each knowledge-base a set of “acceptable” sets of consequences. A managed Multi-Context System (mMCS) $M = (C_1, \ldots, C_n)$ is a heterogeneous collection of contexts $C_i = (L_i; kb_i; br_i)$ where L_i is a logic, $kb_i \in KB_{L_i}$ is a knowledge base and br_i is a set of bridge rules. Each such rule is of the following form, where the left-hand side $o(s)$ is called the *head*, denoted as $head(\rho)$, the right-hand side is called the *body*, also denoted as $body(\rho)$, and the comma stand for conjunction.

$$\begin{aligned} o(s) \leftarrow\ & (c_1 : p_1), \ldots, (c_j : p_j), \\ & not\,(c_{j+1} : p_{j+1}), \ldots, not\,(c_m : p_m). \end{aligned}$$

For each bridge rule included in a context C_i, it is required that $kb_i \cup o(s)$ belongs to KB_{Li} and, for every $k \leq m$, c_k is a context included in M, and each p_k belongs to some set in KB_{L_k}. The meaning is that $o(s)$ is added to the consequences of kb_i whenever each atom p_r, $r \leq j$, belongs to the consequences of context c_r, while instead each atom p_w, $j < w \leq m$, does not belong to the consequences of context c_s. While in standard MCSs

the head s of a bridge rule is simply added to the "destination" context's knowledge base kb, in managed MCS kb is subjected to an elaboration w.r.t. s according to the specific operator o and to its intended semantics: rather than simple addition. Formula s itself can be elaborated by o, for instance with the aim of making it compatible with kb's format, or via more involved elaboration.

If $M = (C_1, \ldots, C_n)$ is an MCS, a data state or, equivalently, belief/knowledge state, is a tuple $S = (S_1, \ldots, S_n)$ such that each S_i is an element of Cn_i. Desirable data states are those where each S_i is acceptable according to ACC_i. A bridge rule ρ is applicable in a knowledge state iff for all $1 \leq i \leq j : p_i \in S_i$ and for all $j+1 \leq k \leq m : p_k \notin S_k$. Let $app(S)$ be the set of the heads of the bridge rules which are applicable in a data state S. There is still the management function, which provides a semantics to the operator which is applied to the conclusion of a bridge rule.

For a logic L, $F_L = \{s \in kb \,|\, kb \in KB_L\}$ is the set of formulas occurring in its knowledge bases. A *management base* is a set of operation names (briefly, operations) OP, defining elaborations that can be performed on formulas, e.g., addition of, revision with, etc. For a logic L and a management base OP, the set of operational statements that can be built from OP and F_L is $F_L^{OP} = \{o(s) \,|\, o \in OP, s \in F_L\}$. The semantics of such statements is given by a management function, which maps a set of operational statements and a knowledge base into a modified knowledge base. In particular, a management function over a logic L and a management base OP is a function $mng : 2^{F_L^{OP}} \times KB^L \to KB_L \setminus \emptyset$. The management function is crucial for knowledge incorporation from external sources, as it is able to perform any elaboration on the knowledge base given the acquired information. In general, every context is equipped with its own local management function mng_i.

Semantics of mMCS is in terms of *equilibria*. A data state $S = (S_1, \ldots, S_n)$ is an equilibrium for an MCS $M = (C_1, \ldots, C_n)$ iff, for $1 \leq i \leq n$, $S_i \in ACC_i(kb'_i)$, with $kb'_i = mng_i(app(S), kb_i)$. Thus, an equilibrium is a global data state composed of acceptable data states, one for each context, encompassing inter-context communication determined by bridge rules and the elaboration resulting from the operational statements and the management functions.

Equilibria may not exist (where conditions for existence have been studied, and basically require the avoidance of cyclic bridge-rules application), or may contain inconsistent data sets (local inconsistency, w.r.t. *local consistency*). A management function is called *local consistency (lc-) preserving* iff, for every given management base and for every context C_i, kb'_i is consistent. It can be proved that a mMCS where all management functions are lc-preserving is locally consistent. Algorithms for computing equilibria have recently been proposed (see, e.g., [74] and the references therein). Notice that bridge rules are intended to be applied whenever they are applicable, so they are basically a *reactive* device.

In dynamic environments, a bridge rule in general will not be applied only once, and it does not hold that an equilibrium, once reached, lasts forever. In fact, contexts may be able to incorporate new data items, e.g, as discussed in [38] for Reactive MCSs (rMCSs),

the input provided by sensors ("observations"); in particular, a sensor is identified by its observation language and a current sensor reading and, given a tuple of sensors Π, an observation Obs for Π consists of a sensor reading for each sensor. Therefore, a bridge rule can be in principle re-evaluated upon new observations, thus leading to evolving equilibria and to the notion of a "run" of an rMCS. A run of mMCS M under a sequence $Obs^0, Obs^1, \ldots$ of observations is in fact a sequence $R = \langle S^0, KB^0\rangle, \langle S^1, KB^1\rangle \ldots$ such that $\langle S^0, KB^0\rangle$ is a *full equilibrium* of M under Obs^0, and for $i > 0$ $\langle S^i, KB^i\rangle$ is a full equilibrium of M under Obs^i, where a full equilibrium is obtained by taking the observations into consideration in every context for bridge rules application: in fact, observation literals can occur in bridge rule bodies.

The reason why MCSs are particularly interesting is that they aim at modeling real situations, where a number of sources distributed on the web can contribute to the solution of complex problems. E.g., the METIS MCS developed in [162] is aimed at being an industrial system for aiding human controllers in maritime control for illegal activities detection. To detect traces of such activities, METIS exploits diverse heterogeneous information sources among which the commercial ship database IHS Fairplay [www.ihs.com/products/maritime-information/], ship tracking websites [marinetraffic.com, myship.com], and news feeds for determining pollution events a ship may have been involved in.

The only constraint that the MCSs approach poses is that contexts must be based upon some logic. We don't see this as an essential limitation: in fact, many sources are logical by nature (including, e.g., relational databases and ontologies), others can be built in any of the many available logic-based approaches, and others can be wrapped within a logical shell, as tools for doing so are the subject of active developments.

Bridge rules as originally defined are by definition ground, i.e., they do not contain variables. In [37] it is literally stated that [in their examples] they "*use for readability and succinctness schematic bridge rules with variables (upper case letters and '_' [the 'anonymous' variable]) which range over associated sets of constants; they stand for all respective instances (obtainable by value substitution)*". Clearly, there must be a finite number of such ground instances. Contexts' knowledge bases are, reasonably, finite sets of formulas. However, finite grounding of bridge rules requires finite grounding of such knowledge bases. This assumption can be reasonable for standard relational databases, logic programming under the answer set semantics, and other logical systems. In other kinds of logics, for instance simply "plain" positive logic programs under the Least Herbrand Model semantics (cf. [119, 8] for surveys), it is no longer realistic.

In related work we have formally extended bridge rules to the non-ground case. Informally, our extension works as follows. Given any non-ground bridge rule:

- we consider a data state S that we (reasonably) assume to be composed of finite sets;
- we instantiate bridge rules over the finite number of terms occurring in S; we thus

obtain finite grounding relative to S;

- we evaluate whether S is en equilibrium, i.e., if S coincides with the data state S' resulting from the application of applicable bridge rules;
- in case S is not an equilibrium, bridge rules can possibly be grounded w.r.t. S', and so on, until either an equilibrium is reached, or no more applicable bridge rues are generated. It is reasonable to start the procedure from a basic data state consisting of a finite ground instance of the initial contexts' knowledge bases.

In an implemented mMCS, the grounding of literals in bridge rule bodies w.r.t. the present data state is most presumably computed at run-time, whenever a bridge rule is actually applied. Such grounding, and thus the bridge-rule result, can be obtained for instance by "executing" or "invoking" literals in the body (i.e., querying contexts) left-to-right in Prolog style. In practice, we can allow bridge rules to have negative literals in their body. To this aim, we introduce a syntactic limitation in the form of non-ground bridge rules, i.e., we assume:

- every variable occurring in the head of a non-ground bridge rule r also occurs in some positive literal of its body;
- in the body of such rule, positive literals occur (in a left-to-right order) before negative literals.

So, at run-time variables in a bridge rule are incrementally and coherently instantiated via results returned by contexts. Each positive literal $(c_i : p_i)$ in the body may fail (i.e., c_i returns a negative answer), if none of the instances of p_i given the partial instantiation computed so far is entailed by c_i's present data state. Otherwise, the literal succeeds and subsequent ones are instantiated to its results. Negative literals $not\,(c_j : p_j)$ make sense only if p_j is ground at the time of invocation, and succeed if p_j is *not* entailed by c_j's present data state. In case either some literal fails or a non-ground negative literal is encountered, the overall bridge rule evaluation fails without returning results. Otherwise the evaluation succeeds, and the result can be elaborated by the management function of the "destination" context. It is easy to prove that the invocation of a bridge rule leads to success if and only if, given its ground instance obtained via the above-specified evaluation pattern, the body is entailed by the present system's data state and thus the rule is applicable.

3.4 BACKGROUND: ACE AND DACMACS

3.4.1 DACMACS

DACMACS (Data-Aware Commitment-based managed Multi-Agent-Context Systems) [51, 54] extend DACMAS (Data-Aware Commitment-based Multi-Agent Systems) [126],

which is a quite general model of multi-agent systems. In fact, apart from a general specification of data management and communicative features and knowledge-flow features, such models remain very general about an agent program's definition, and can be thus specialized to specific instances. In DACMAS and then in DACMACS knowledge and data are supposed to be represented via logic ontologies[14]. While DACMAS explicitly refer to DLR-Lite ontologies, [14, 11], we abstract away from this aspect, as DACMACS is meant to constitute an abstract and fully general architecture encompassing logical agents and contexts. So, DACMACS might admit any logical formalism for defining ontologies, such as, e.g., Datalog+/- [42, 97], or to the extreme adopt first-order logic or plain datalog. DACMACS allows agents and contexts to interchange ontological definitions [63] is also possible in the two directions, i.e.: an agent can provide a context the ontological definition of data to be extracted; vice versa, an agent can obtain from a context data together with their ontological definition.

A DACMAS always includes an *Institutional* agent which owns a "global" TBox, specifying the domain in which the MAS operates, whereas in DACMACS there is also a global ABox, and each participating agent is also equipped with its local ABox. In DACMACS moreover, agents can query external contexts via bridge-rules that are, as said, employed proactively. DACMACS bridge rules are more general than in MCS since each literal can be generalized to a datalog query, and context names can either be specified as constants, or can be obtained via a query to the Institutional agent.

Communication among agents in DACMACS occurs according to the specific agent-oriented language adopted, where however the semantics of communicative acts and the communication protocol are specified via the approach of *commitments*, which is a relatively recent though very well-established general paradigm for agent interaction (cf. [145, 146] and the references therein). The approach of commitments does not affect the syntactic form of messages: rather, it specifies the meanings of the messages in terms of the commitments arising between the two parties. A commitment $C_{x,y}\,(ant,\, csq)$ in particular relates a *debtor agent* x to a *creditor agent* y where x commits to bring about csq whenever ant holds. Commitment lifecycle (they can be created, fulfilled, released, canceled, etc.) is managed by a so-called "commitment machine", role which in DACMAS and DACMACS is played by the Institutional agents. So, whenever a message is sent from agent A to agent B, the commitment machine creates a corresponding commitment and and manages its states, and the new commitments arising during the interaction.

Precisely, a DACMACS (Data-Aware Commitment-based managed Multi-Agent-Context System) is a tuple

$$\langle \mathcal{X}, \mathcal{N}, \mathcal{Y}, \mathcal{E}, \mathcal{T}, \mathcal{A}, \mathcal{I}, \mathcal{C}, \mathcal{B} \rangle$$

where: (i) $\mathcal{X}$ is a finite set of agent specifications, defined in any agent-oriented programming language (e.g., one of those mentioned above) where however each agent is assumed to be equipped with a local ABox;
(ii) $\mathcal{N}$ is a set of agents' names, listing the agents (beyond the Institutional agent) composing the MAS, together with their *roles* in the system;
(iii) $\mathcal{Y}$ is a set of contexts' names, listing the contexts that are globally known to the MAS,

together with their *roles* in the system;
(iv) $\mathcal{T}$ and $\mathcal{A}$ are the global TBox and Abox respectively, and are common to all agents participating in the system;
(v) $\mathcal{I}$ is a specification for the "Institutional" agent $Inst$; such agent is responsible of managing message-passing among agents, and is also in charge of locating agents and contexts based on their roles: a query $role@Inst$ returns the name of an agent/context with role $role$; in general, if more that one agent/context fulfills the required role, a query $role@Inst$ can return *a set* of names; however, we assume that one is chosen at random.
(vi) $\mathcal{C}$ is a contractual specification, $\mathcal{B}$ is a Commitment Box (CBox), $\mathcal{E}$ is a set of predicates denoting events (where the predicate name is the event type, and the arity determines the content of the event); all these elements are involved in the management of commitment-based communication, managed by the Institutional agent.

Components $\mathcal{T}, \mathcal{E}, \mathcal{X}, \mathcal{I}, \mathcal{C}$ and $\mathcal{B}$ are analogous to those of DACMASs. However, DACMACSs are enhanced via a set of contexts, and moreover agents' specifications can now include: bridge rules, for gathering new knowledge from contexts; *trigger rules* for proactive activation of bridge rules; *bridge-update rules* for incorporating the acquired knowledge into the agent's knowledge base upon specific conditions (corresponding to the MCSs' management function), where the minimal requirement is that of keeping the agent's ABox consistent internally, and with the global TBox and ABox. Contexts in bridge-rule bodies can be now identified by their names, whenever they are locally known to the agent, or by a query $role@Inst$ returning such name from the given role. In fact, each agent's local ABox is supposed to be consistent with the global Abox and TBox, where however the ABoxes of the various agents are not required to be mutually consistent. Actually in fact, each agent's knowledge base can be seen as composed of the union the global Abox and TBox and the local ABox.

The Institutional agent $Inst$ is a special agent that: manages the messages which are exchanged in the system, and is responsible of the management of commitments, whose concrete instances are maintained in the Commitment Box $\mathcal{B}$; it does so based on the *Commitment Rules* in $\mathcal{C}$, defining the commitment machine. The Institutional agent is also responsible of returning agents' and context's names via their role, by answering queries of the form $role@Inst$.

A semantics for DACMACS in terms of equilibria, inspired by the MCSs' semantics but extended however with timed data states, timed equilibria and execution trajectories is provided in [51].

Overall, DACMACS adds to the generality and flexibility of DACMAS (commitment-based communication, suitably formalized shared knowledge) the possibility to explicitly represent the interaction of agents with external heterogeneous contexts. Interaction occurs like in mMCS via bridge rules, which can however be applied proactively as explained below for ACE.

3.4.2 ACE

An enhanced Agent Computational Environment (ACE) [50] is defined as a tuple $\langle A, M_1, \ldots, M_r, C_1, \ldots, C_s, R_1, \ldots, R_q \rangle$ where module A is the "basic agent", i.e., an agent program written in any agent-oriented language. The "overall" agent is obtained by equipping the basic agent with the following facilities. The M_is are "Event-Action modules", that are special modules aimed at Complex Event Processing. The R_js are "Reasoning modules", that are specialized in specific reasoning tasks. The C_ks are contexts in the sense of MCSs, i.e., external data/knowledge sources that the agent is able to locate and to query about some subject, but upon which it has no further knowledge and no control: this means that the agent is aware of the "role" of contexts in the sense of the kind of knowledge they are able to provide, but is unable in general to provide a description of their behavior/contents or to affect them in any way. Interaction among ACE's components occurs via *bridge rules*, inspired by those of MCSs.

The "local" agent's modules, i.e., main agent program, event-action modules, and reasoning modules can be defined in any agent-oriented and/or computational-logic-based programming language. For specifying the main agent we might adopt, e.g., DALI, AgentSpeak, GOAL, 3APL, METATEM, KGP, etc. (see [29, 30, 32, 72, 75, 89, 106] and the references therein), or also in other logic formalisms such as, e.g., ASP (ASP, cf, among many, [93, 20, 91] and the references therein). Notice that in case $r, q, s = 0$, i.e, no auxiliary components are provided, an ACE reduces to a "traditional" agent.

For Complex Event Processing (CEP) modules, a recent but well-established and widely used approach to CEP in computational logic is ETALIS [6, 7, 1], that is an open source plug-in for Complex Event Processing implemented in Prolog which runs in many Prolog systems. [50] presents an approach where DALI-based modules are executed via translation into ASP. For Reasoning modules, the formalism to be adopted depends on the reasoning task at hand.

Bridge rules have been extended in [56] to become (via a smooth formal integration into ACE's semantics) *bridge rule patterns* of the following form:

$$s \leftarrow (\mathcal{C}_1 : p_1), \ldots, (\mathcal{C}_j : p_j), not\ (\mathcal{C}_{j+1} : p_{j+1}), \ldots, not\ (\mathcal{C}_m : p_m).$$

where each $\mathcal{C}_i$ can be either a constant indicating a context name, or a term of the form $m_i(k_i)$ that we call *context designator*, indicating the *kind* of context (rather than the specific one) to be queried, to be specified before bridge-rule execution. Such a rule, once context designators have been instantiated to actual context names, is *applicable* (and s can thus be added to the consequences of a module's knowledge base) whenever each p_r, $r \leq j$, belongs to the consequences of module $\mathcal{C}_r$ while instead each p_w, $j < w \leq m$, does not belong to the consequences of $\mathcal{C}_w$. We may notice that bridge rule patterns are meta-rules, where context designators constitute a kind of "reification" device for denoting a-priori unknown contexts (for a review of such concepts, cf., e.g., [45]). Bridge rules can be instantiated by the agent itself via the special action $instantiate(H, m_i(k_i), Conds, L)$.

For every bridge rule ρ with head matching with H and given list L of constants, such action, proactively performed by the agent, generates as many instances of ρ as obtained by substituting the context designator $m_i(k_i)$ by elements of L; $Conds$ are (possibly empty) input conditions to be evaluated in the action's preconditions in order to determine L.

In ACE basic agents we adopt for bridge rules the agent-oriented modalities introduced in [51]. In fact, bridge rules in (m)MCS are applied reactively (whenever the rule is applicable, i.e., body is true in the current data state, then the head is added to the context where the rule occurs, possibly after management). Instead, in ACE bridge rules are enabled (and thus applies whenever applicable) upon conditions internal to the agent. In particular, such conditions are specified via *trigger rules* of the form

$$Q(\hat{y}) \textbf{ enables } A(\hat{x})$$

where: $\hat{x}$, $\hat{y}$ are tuples composed of constants and variables, $Q(\hat{y})$ is a query to the agent's internal knowledge-base and $A(\hat{x})$ is the conclusion of one of agent's bridge rules, that is "fired" whenever such query evaluates to *true*. The result of $Q(\hat{y})$ can partially instantiate $A(\hat{x})$. The special action $enable(S)$ enables the application of bridge rule ρ whose head matches with S by generating its trigger, whereas $instenable(S, m_i(k_i), Conds, L)$ both instantiates and enables a bridge rule.

The results returned by a bridge rule with head $A(\hat{x})$ can be exploited via a *bridge-update rule* of the form

$$\textbf{upon } A(\hat{x}) \textbf{ then } \beta(\hat{x})$$

where

$\beta(\hat{x})$ specifies both the conditions for acquiring $A(\hat{x}$ into the agent's knowledge base, and the elaboration to be performed. So, also the incorporation of bridge-rule results occurs in a proactive way as dictated by $\beta(\hat{x})$, which also incorporates the management function.

The merit of the ACE approach is to make an agent fully modular. In fact, ACE provides the bridge-rules as a general mechanism for the interaction among the main agent and contexts, and among contexts. No assumption is made about the formalisms/languages in which the various elements are expressed, so the ACE architecture can be instantiated to a variety of practical cases. This is permitted by a semantics which generalizes that of mMCS, and is thus fully parametric w.r.t. formalization of components. The generalization of bridge rules to bridge rule patterns overcomes the limitation to have to specify statically which are the knowledge sources to consult. In ACE, such sources can be determined dynamically, where suitable bridge-rule patterns can be instantiated accordingly.

3.5 K-ACE

We present K-layers-ACE, or simply K-ACE. It is a generalization of ACE and DACMACS, where: each agent in a DACMACS can be an ACE (more precisely a 1-ACE,

which is a slight extension seen below); a DACMACS (now renamed K-ACE) may be composed not only of agents and contexts, but also of other (lower-level) K-ACEs. This "nesting" is allowed over an arbitrary number "K" of layers.

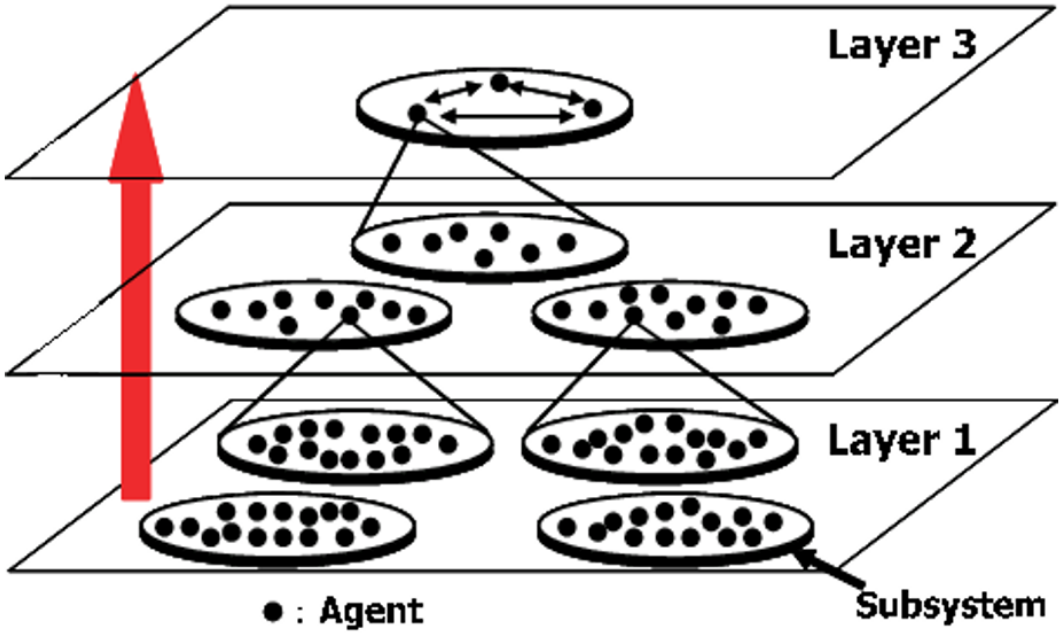

Figure 3.3: K-ACE Architecture

This proposal is introduced in the perspective of DyPESs where an application/organization, represented as a K-ACE, might dynamically resort to another external applications/organizations to obtain data or to perform elaboration. So, such external organizations that are accessible by a DypES/K-ACE are seen as (lower-level) K-ACEs that can be reached though a uniform interface, represented by these systems' Institutional agents. Recursively, lower-level K-ACEs can have other K-ACEs as components, along an arbitrary number of levels. 1-ACEs are, together with contexts, the basic building blocks of the system. Notice that which are the higher-lower levels is not fixed: each K-ACE implementing some application and/or pursuing its own objective considers the other K-ACEs it needs to access, from its own perspective, as lower level. As a metaphor, we might see the whole system with all the composing K-ACEs as a galaxy, and each composing K-ACE as a stellar system composed of solar systems (1-ACEs); each solar and stellar system is able to observe (to some extent) the rest of the galaxy.

Any component or subsystem in a K-ACE is not necessarily always available, as it can join or leave the system or become momentarily unavailable. Also, not every component is allowed to reach every other one: rather, accessibility can be subject to permissions. Bridge-rules activation may itself be subject to norms, i.e., not only an agent proactively seeks to obtain or distribute new knowledge, but may does so because it "has to". Each component may in turn choose to access the *best preferred* components among those which may provide certain knowledge or services.

This very general architecture is intended for complex dynamic applications where independent subsystems can be seen as components of the overall systems. K-ACE systems may evolve in time, so below we define a K-ACE relative to a time T. Except for the Institutional agent and for the global TBox and ABox and commitments' contractual specification, whose definition remains stable over time, all the other components can evolve over time, as agents, contexts and lower-level K-ACEs can either join and leave the system. A reachability relation, that itself evolves in time, dynamically establishes which component can reach which other. As done in Linear Temporal Logic (LTL, [85]) we assume a discrete, linear model of time and represent each state/time instant as an integer number. The actual evolution from time T to $T' = T + 1$ and the interval $\delta = T' - T$ is considered to be peculiar of each specific instance of the architecture given its application domain.

The basic building blocks of K-ACEs are ACEs, augmented with some features that make them more suitable to be included in a wider system. Empowered ACEs are defined below, and are called 1-ACEs as they are intended to be unitary components of K-ACEs.

Definition 3.5.1 *Let a 1-ACE be an extension of ACE, where:*

- *a 1-ACE is characterized by a unique name and a list of roles; the name must include sufficient information for locating the 1-ACE and communicating with it;*
- *a 1-ACE's main agent is supposed to be equipped with a local ABox;*
- *in 1-ACE's bridge rules, that are still of the general form:*

$$s \leftarrow (\mathcal{C}_1 : p_1), \ldots, (\mathcal{C}_j : p_j), not\ (\mathcal{C}_{j+1} : p_{j+1}), \ldots, not\ (\mathcal{C}_m : p_m).$$

  each $\mathcal{C}_i$ can now be one of the following:

  - *a constant denoting a context name;*
  - *a term of the form $m_i(k_i)$ that we call* context designator*, indicating the* kind *of context (rather than the specific one) to be queried; such term must be substituted by a constant denoting a context name before bridge-rule execution; an expression of the form $role@Ag$ to be substituted by a constant denoting a context name before bridge-rule execution.*

As seen below, an expression of the form $role@Ag$ denotes a query to a special "Institutional" agent, returning the name of a context with the required role. In the following, when no ambiguity can arise 1-ACEs can be called simply "agents". By some abuse of notation, for a 1-ACE with name a we will often say "1-ACE a" or "agent a".

Definition 3.5.2 *Let a K-ACE ($K \geq 1$) at time T be a tuple*

$$\langle \mathcal{N}^T, \mathcal{X}^T, \mathcal{Y}^T, \mathcal{T}^T, \mathcal{A}^T, \mathcal{R}^T, \mathcal{C}^T, \mathcal{B}^T, \mathcal{E}^T, \mathcal{I} \rangle$$

identified by a unique name n_K, a role r_{n_K}, and an expression denoting its "Institutional" agent $Inst@n_K$, where:

(i) $\mathcal{N}^T$ is the list of 1-ACEs' that are part of the K-ACE at time T; each one is identified by a unique name and by its role(s); we say that such agent participate to the (K-ACE) system at level K; the specification of 1-ACEs is considered to be external to the system, though when joining the system the agents can be reached via their names and roles;

(ii) if $K > 1$, $\mathcal{X}^T$ is a finite list of K'-ACEs, $K' = K - 1$, i.e., the lower-level subsystems taking part in the K-ACE at time T; each one is identified by a unique name $n_{K'}$ and by its role(s) $r_{n_{K'}}$, and by an expression denoting their Institutional agent $Inst@n_{K'}$;

(iii) $\mathcal{Y}^T$ is a set of contexts' names, listing the contexts that are globally known to the K-ACE at level K and time T; contexts' definitions is considered to be external to the system.

(iv) $\mathcal{T}^T$ and $\mathcal{A}^T$ are the global TBox and Abox respectively, and are common to all agents participating in the system at level K;

(v) $\mathcal{R}^T$ is a reachability relation, establishing

(a) which elements of $\mathcal{N}^T$ can reach each other, thus specifying constraints on inter agent communication;

(b) which elements of $\mathcal{Y}^T$ are reachable by each element of $\mathcal{N}^T$, i.e, which contexts are reachable by each agent (at level K);

(c) which lower level K'-ACEs (i.e., which elements of $\mathcal{X}^T$) are reachable by each agent in $\mathcal{N}^T$;

(d) which elements of $\mathcal{Y}^T$, i.e., which contexts, are reachable from the outside, precisely from higher-level K-ACEs (if any); such external agents are designated herein via the standard name $outag$.

We introduce a special distinguished predicate $kreach(C_1, C_2, T)$, that is true whenever component C_2 is accessible from component C_2 at time T according to $\mathcal{R}^T$. The binary version $kreach(C_1, C_2)$ takes T to be the current time.

(vi) like in DACMACS, $\mathcal{C}^T$ is a contractual specification, $\mathcal{B}^T$ is a Commitment Box (CBox), $\mathcal{E}^T$ is a set of predicates denoting events, and all these elements are exploited by the Institutional agent for the managemnt of (commmitment-based) inter-agent communication. Note that communication among 1-ACEs A and B is possible at any time t only if $kreach(A, B, t)$ holds.

(vii) $\mathcal{I}$ is a specification for the Institutional agent $Inst@n$; specifically, the Institutional agent is in charge of inter-agent communication, of contexts' identification via their roles, and of acting as an interface with the lower-level K-ACEs (see (ii)); a query

to the Institutional agent for obtaining a (set of) agent(s) or a (set of) context(s) with role $role$ issued by the 1-ACEs is now of the form $role@Inst@n_K$, and returns the set of agent/contexts with the specified role; we assume that the Institutional agent is a special 1-ACE that does not encompass either Event-Action modules or contexts, but can encompass reasoning modules that can be exploited for performing its functions; we assume $Inst@n$ to have direct access to all other K-ACE's elements, and to be able to communicate with Institutional agents of lower-level K'-ACEs by means of queries $role@Inst@n_{K'}$. Notice that the Institutional agent is the unique "entry point" of a K-ACE; in fact, higher-level components or other components of the same K-ACE can locate and therefore access components only through results of queries of the kind $role@Inst$. In case possible alternative results are available, $Inst$ can apply its own internal policies for selecting one.

We assume structures $\mathcal{N}^T$, $\mathcal{X}^T$, $\mathcal{R}^T$ to be dynamic, in the sense that components may join or leave the system, or become momentarily unreachable.

Therefore, we have constructed a modular architecture where:

- the basic elements are agents, i.e., 1-ACEs, and contexts;
- they can be part of a K-ACE, that provides, via the Institutional agent with the support of a number of specialized knowledge bases, a suitable infrastructure for communication and for the location (by role) of other agents and of required knowledge bases;
- K-ACEs can be in turn part of other (higher-level) K-ACEs, where the interface among levels is provided by the Institutional agents.

However, a more precise specification of the Institutional agent's operation needs to be provided. We suppose that Institutional agent is able to reason about roles, so that in case a module with the specified role could not be found directly it might be possible to find instead another one whose role is either equivalent or more general.

Notice that, given query $role@Inst@n_K$, this query must be issued by an agent and, given the syntactic place where the query occurs, it can be determined whether the agent seeks to find either other agents or contexts with the specified role. The Institutional agent returns a set of agents/contexts including exactly those which are accessible from A and are pertinent to role $role$.

Definition 3.5.3 *Let $role1$ and $role2$ be expressions denoting roles of agents/contexts in a given K-ACE. Let $subs(role1, role2)$ be a predicate that is true whenever $role2$ is either equivalent or more general than $role1$ w.r.t. a background ontological role definition R.*

We assume that the Institutional agent owns the background theory R, and is thus able to compute, given $role1$, roles $role2$ such that $subs(role1, role2)$ holds. This may be possibly achieved via a suitable reasoning module.

Definition 3.5.4 *Given a K-ACE identified by its name n_K, role r_{n_K}, and Institutional agent $Inst@n_K$ with background ontological role definition R, a query $role@Inst@n_K$ issued by a 1-ACE (agent) A returns a set S that is:*

- *a set of contexts if the query occurs in the body a bridge rule;*
- *a set of agents if the query occurs elsewhere in the agent's program.*

The set S includes:

1. *all those components that are reachable from A according to $\mathcal{R}^T$ at level K, i.e., within the K-ACE without resorting to lower-level K'-ACEs with role $role$, or, if none can be found;*
2. *all those components that are reachable from A according to $\mathcal{R}^T$ at level K though with role $role1$ where $subs(role, role1)$ holds (i.e., a more general role is seeked whenever the specific one could not be identified), or, if none can be found;*
3. *the result of a query $role@Inst@n'_K$ where K'-ACE with name n'_K is reachable from A according to $\mathcal{R}^T$, and has role $r_{n_{K'}}$ where $subs(role, r_{n_{K'}})$ holds; I.e., if a component with the required role (or with a more general role) cannot be found within the K-ACE, then it is looked for in lower level L'-ACEs.*

We assume that, given a 1-ACE participating in a K-ACE n_K, a query $role@Inst@n_K$ to the "local" Institutional agent takes the simplified form $role@Inst$.

In a K-ACE, the reachability relation $\mathcal{R}^T$ defines a graph structure where:

- nodes are all the components, i.e., agents, contexts and K'-ACEs included in the system; i.e., the set $V^T_{n_K}$ of vertices is composed of the elements of $\mathcal{N}^T$, $\mathcal{Y}^T$ and $\mathcal{X}^T$;
- the set $E^T_{n_K}$ of edges identified nodes (components) that are connected according to $\mathcal{R}^T$, i.e., at the present time.

The problem of determining all components of a certain kind reachable from a given one, this can be understood as the problem of finding a spanning tree with that component as root. Therefore,

Proposition 3.5.1 *Given a K-ACE, the complexity of determining (at any time T) which set of components is reachable from a given one is $\mathcal{O}(|V^T_{n_K}| + |E^T_{n_K}|)$.*

Notice that the set S may be empty (if not even some K'-ACE returns results) or may contain several elements. In the latter case, *preference* can discriminate among elements of such sets, where a preference criterion can be very simply stated as follows.

Definition 3.5.5 (Preferred Source Selection) *Given set SC, a preference criterion $\mathcal{P}$ returns a (nonempty) ordered subset $SC^{\mathcal{P}} \subseteq SC$.*

There are several factors that can influence the choice of a preference criterion, including for instance trust, reliability, fast answer, and others. So, a preference criterium in general establishes an order according to some kind of reasoning performed on a background theory. Therefore, preference criteria can be seen as special reasoning modules. Both 1-ACEs and the Institutional agents can be equipped with preference criteria, $\mathcal{P}_a$ to be used for selecting among sets of agents, $\mathcal{P}_c$ for sets of contexts and $\mathcal{P}_K$ for sets of K-ACEs.

Approaches to preferences in logic programming might be adapted to the present setting: cf., among many, [38] and the references therein, [27, 39] and [60]). For simplicity we assume that, given a preference criterion, a query such as $role@Inst@n_K$ will return a unique result. We can now establish when a bridge rule occurring in an agent can actually be triggered.

Definition 3.5.6 *Given bridge rule ρ occurring in the main agent program of a 1-ACE a that is a component of a K-ACE at time T is* executable *at time T iff each C_i in its body has been substituted by a constant c_i denoting a context (this either by an $instantiate$ action or by executing a query $role@Inst@n_K$ to the K-ACE's Institutional agent) and $kreach(a, c_i, T)$ holds.*

Note that the notion of bridge-rule being executable is preliminary to bridge rule applicability. Reporting to K-ACEs the notion introduced in [52], each bridge rule ρ is *potentially applicable* in a system's state if such a state entails its body. For contexts, each bridge rule is applicable whenever it is potentially applicable. For agents instead, a bridge rule with head $A(\hat{y})$ is applicable whenever it is potentially applicable and there exists in the main agent program a trigger rule of the form $Q(\hat{y})$ **enables** $A(\hat{x})$ where the agent's present state satisfies $Q(\hat{y})$, where such trigger rule "triggers" or "activates" the bridge rule.

So, for making such applicability formally precise we have to define a K-ACE state at time T, and how such state evolves in time. However, for K-ACE we also propose a generalization of trigger rules. First, we may notice that trigger rules, that determine proactive activation of bridge-rules (drawing inspiration from DALI internal events [72]), can be modeled in terms of Linear Time Logic expressions. In fact, let us consider the

Separated Normal Form (SNF) for LTL formulas specified in [88]. In such normal form, all formulas have syntax

$$\phi \Rightarrow \bigcirc\psi$$

where ϕ is a conjunction of propositional formulas. A trigger rule $Q(\hat{y})$ **enables** $A(\hat{x})$ can thus be understood as

$$Q(\hat{y}) \Rightarrow \bigcirc A(\hat{x})$$

that is in turn implicitly understood as

$$\Box(Q(\hat{y}) \Rightarrow \bigcirc A(\hat{x}))$$

I.e., whenever $Q(\hat{y}$ is entailed by the present state, bridge rule with head $A(\hat{x})$ will be activated in next state. The advantage of such a reformulation is that, while the internal event construct requires modification to the semantics of the adopted agent-oriented language this is no longer the case in this new interpretation, as it suffices (as formally seen below) that the system's evolution respects such rules. A similar use of such notation is made in [159] to make agents adapt their behavior to comply with norms wothout modifying the agent's semantics.

In K-ACEs, we extend trigger rules to activate not only bridge-rule execution, but also inter-agent communication.

Specifically, given 1-ACE A, its main agent program may include trigger rules of the form

$$Q(\hat{y}) \textbf{ enables } communication(A, B, Payload, T)$$

where $communication(A, B, Payload, T)$ denotes a communicative act occurring at time T with origin A and destination B where the $Payload$ is understood, in FIPA terms[1], as comprising the message according to the specific ACL (Agent Communication Language) adopted. Time T is assumed to be automatically added by the system.

We further generalize trigger rule by introducing a temporal element also in the premise, to state that a bridge rule should be activated if something happens (i.e, if $Q(\hat{y})$ becomes true, or is always true, or is never true) at a certain time or in a certain time interval. To this aim, we exploit the work of [49] on A-ILTL (for 'Agent-Oriented Interval LTL'), where an agent-oriented interval extension to LTL is presented, providing syntax, semantics and examples of use of the extended logic, called "A-ILTL" for "Agent Interval LTL". Though, as discussed in [49], several "metric" and interval temporal logic exist, the introduction of A-ILTL is useful in the agent realm because the underlying discrete linear model of time and the complexity of the logic remains unchanged with respect to LTL. This simple formulation can thus be efficiently implemented, and is nevertheless sufficient for expressing and checking a number of interesting properties of agent systems (cf., e.g., [53, 62]).

[1] As it is well-known, FIPA (`http://fipa.org`) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies, and has developed over time what is now the standard terminology about Agent Communication Languages.

Considering set $\mathcal{F}$ of formulas built out of classical connectives and of LTL and A-ILTL operators (where however nesting of A-ILTL operators is not allowed), some among the A-ILTL operators are the following, where $\varphi \in \mathcal{F}$ and m, n are positive integer numbers denoting time instants, and $\langle m, n\rangle$ denotes a time interval with extremes included.

- $\Diamond_{m,n}\,\varphi$, or equivalently $F\langle m, n\rangle\,\varphi$ (*eventually (or "finally") in time interval*), states that φ has to hold sometime on the path from time m to time n. It can be specialized into $F\langle m\rangle\,\varphi$, *bounded eventually (or "finally")*, where φ should become true somewhere on the path from the current time to time m.

- $\Box_{m,n}\,\varphi$, or equivalently $G\langle m, n\rangle\,\varphi$ (*always in time interval*) states that φ should become true at most at time m and then hold at least until time n.

- $N\langle m, n\rangle\,\varphi$ (*never in time interval*) states that φ should not be true at any time between m and n. It can be specialized into $N\langle m\rangle\,\varphi$, *instant never*, where φ should not be true at time m. This is an auxiliary operator expressible as $\Box_{m,n}\,\neg\varphi$

In our setting, we consider simplified A-ILTL expressions where φ does not include temporal operators, and in particular φ is of the form $Op_I Q(\hat{y})$, where Op is an A-ILTL operator, I an interval (possibly reduced to a single time instant), $Q(\hat{y})$ is as before. So, we may have, in 1-ACE A, rules such as for instance

$$N\langle m, n\rangle\, Q(\hat{y}) \textbf{ enables } A(\hat{x})$$

meaning that if $Q(\hat{y})$ has never become true in given time interval $\langle m, n\rangle$ then a communication to agent B is issued, or

$$F\langle m\rangle\, Q(\hat{y}) \textbf{ enables } communication(A, B, Payload, T)$$

meaning that the communicative act occurs if $Q(\hat{y})$ has become true sometimes before time m or also, if needing to address an agent with role $role$ in case $Q(\hat{y})$ has remained true in time interval $\langle m, n\rangle$,

$$G\langle m, n\rangle\, Q(\hat{y}) \textbf{ enables } communication(A, role@Inst, Payload, T)$$

Note that time instants, line m and n above, refer to the agent's local time. We refer to the left-hand-side of a trigger rule as its "premise" and to the right-hand-side as its "consequence", and we say that the rule is fired in a system's current state whenever at previous state the premise holds, and so in current state the consequence is executed.

3.6 Application of K-ACE to Case-Studies

Let us assume to model as an ACE the system depicted in Figure 3.1. We can presume that the agent in charge of each human patient (that in [2] we call PMA for "Personal Monitoring Agent") is an 1-ACE equipped with local contexts, complex event processing and reasoning modules. The contexts may provide the agent with information about standard treatment, e.g., by rearranging the quantity of a medicine according to certain values in blood test. The complex event processing modules can for instance detect symptoms, and decide whether they correspond to a potentially serious or unexpected situation. The reasoning modules can for instance devise a plan for coping with such situation.

A K-ACE can encompass several PMAs in charge of different patients. The system may include other K-ACE, for instance a "Diagnostic Center" providing intelligent modules for plausible interpretation of symptoms, a "Medical Center" providing consultation with human specialists, and a "Emergency Center" managing hospital beds and transportation facilities. Each PMA can proactively resort to such systems, e.g., by means of rules such as

$$G\langle 8h\rangle\ high\_blood\_pressure \textbf{ enables}$$
$$communication(pma, helpdesk@Inst@medcenter,$$
$$cardiological\_consultation\_required(patient_{pma}, high\_blood\_pressure), T)$$

where $\langle 8h\rangle$ is a shortcoming for the interval denoting the last 8 hours and the PMA communicates with the agent that is in charge of dispatching consultation requests to the suitable Medical Center facilities, identified by the expression $helpdesk@Inst@medcenter$.

$$emergency(E) \textbf{ enables}$$
$$communication(pma, emergency\_manager@Inst@emergencies,$$
$$hospital\_transportation\_required(patient_{pma}, condition(E)), T)$$

Here, the PMA requires an urgent transportation of the patient in its charge to the hospital. The request is issued to the manager agent of the Emergency Center K-ACE, identified by role via the expression $emergency\_manager@Inst@emergencies$.

In both cases commitments play a fundamental role. In fact, the agents receiving the request will commit to satisfy such request in a certain way and within a certain time: e.g., the Medical Center will provide a video-conference with Dr. House, and the Emergency Center will commit to send an ambulance by the hour of, if deemed necessary, and helicopter by twenty minutes.

The following bridge rule in a patient's PMA, potentially crucial for cardiophatic patients (in the rule the patient is named Bob, and is identified by his patient's id), is applied in case the blood coagulation value detected at time T is anomalous; this implies that the

quantity of anti-coagulant which Bob takes to treat his heart disease must be rearranged accordingly. The correct quantity Q is obtained by the ATC (Anti-Coagulant Center) according to the last blood coagulation value V and its variation D from previous records. The ATC is located via the Institutional agent within a cardiology clinic, which is a K-ACE (as it is an external institution composed of various departments).

$$\begin{aligned} &quantity(anticoagulant, Q) \leftarrow \\ &\quad coagulation\_val(V, T, D), \\ &\quad patientid(bob, Bid),\ atc@Inst@cardiology\_clinic : quantity(Bid, V, D, Q) \end{aligned}$$

Below we illustrate the use of context designators. Suppose that at time T some condition/symptom is detected by the PMA concerning the patient in charge. In case the patient's health state is altered, a physician must be consulted. However, in case, e.g., of a simple flu the family doctor suffices, while if there are symptoms that might be related to a more serious condition then a specialist (e.g., a cardiologist) should be consulted. Thus, there will be a bridge-rule pattern of the following form, where a generic physician denoted as $mydoctor(d)$ will consulted for condition C. Again, the management function will record the request having been sent; the last literal in the body will succeed, upon dynamic rule execution, as soon as the doctor receives the request.

$$\begin{aligned} &call\_physician(bob, T) \leftarrow \\ &\quad now(T), condition(bob, T, C), \\ &\quad patientid(bob, Bid),\ mydoctor(d) : consultation\_needed(Bid, C, T) \end{aligned}$$

The physician, represented by the context designator $mydoctor(d)$, should however be determined in order to specialize the bridge rule via executing the special action

$$instantiate(call\_physician(bob, T), mydoctor(d), condition(bob, T, C), [D])$$

In order to determine the right doctor to consult, the preconditions of the action can employ a suitable predicate $subs_{pmi\_bob}$, for instance, as follows. The notation '_' indicates a "don't care" variable, as time is not taken into account here.

$$\begin{aligned} &subs_{pmi\_bob}(mydoctor(d), F) \leftarrow \\ &\quad family\_doctor(F), condition(bob, \_, fever) \\ &subs_{pmi\_bob}(mydoctor(d), F) \leftarrow \\ &\quad family\_doctor(F), condition(bob, \_, headache) \\ &subs_{pmi\_bob}(mydoctor(d), G) \leftarrow \\ &\quad my\_cardiologist(G), condition(bob, \_, chestpain) \end{aligned}$$

Thus, a valid instance of the bridge-rule pattern will be generated according to the patient's need, as evaluated by the patient's PMI. The resulting bridge rule will then be

immediately executed. So for instance, if Bob has chest pain and the cardiologist who has been following him is Dr. House, then the bridge rule below will be constructed and triggered:

$$call\_physician(bob, T) \leftarrow$$
$$now(T), condition(bob, T, chest\_pain),$$
$$patientid(bob, Bid), drHouse : consultation\_needed(Bid, chest\_pain, T)$$

3.7 SEMANTICS

A K-ACE include diverse components: K'-ACEs, 1-ACEs and contexts are "active", in the sense that 1-ACEs can perform actions (among which communicative acts), and there is a knowledge flow via bridge rules among 1-ACEs and contexts, and 1-ACEs and K'-ACEs. There are, in addition, some "passive" components, namely reasoning modules, and components $\mathcal{T}^T$, $\mathcal{A}^T$, $\mathcal{R}^T$, $\mathcal{C}^T$, $\mathcal{B}^T$, $\mathcal{E}^T$, that are elaborated (and thus possibly modified) by the Institutional agent. In the semantics, we manage to ignore the latter by considering them as a part of the Institutional agent's knowledge base. Agents, contexts and reasoning modules are called *basic components*. Contexts, reasoning modules and also the main agent program in 1-ACEs are called *unitary components* as they do not have an internal structure, i.e., they do not in turn consist of components. A 1-ACE is seen itself as a unitary component whenever it consists of the main agent program only.

Agents work in time, and in fact can employ, for instance, timed trigger rules. Agents are in principle asynchronous. However, for a K-ACE we assume a *global* system time where states/time instants can be represented as $t_0, t_1, \ldots$. In terms of absolute time[2] we have $t_{i+1} - t_i = \delta$, where δ is the actual interval of time after where we assume the overall system to have evolved. In this way, we can approximate each agent's local time t with $min\, t_i$ such that $t_{i-1} < t < t_i$.

Every component of a K-ACE, including the K-ACE itself, can be seen (analogously to MCSs) as a tuple $C^l = (C_1^{l'}, \ldots, C_n^{l'})$ where now the $C_i^{l'}$s are themselves *components* with the same structure.

More formally:

Definition 3.7.1 *A multi-level multi-component MAS (mmMAS) of depth k is formed out of components of the form C_i^l, where l is the* level *of the component, with $0 \leq l \leq k$. A component C_i^l is either a unitary component or it is a* compound component *of the form $C_i^l = (C_{i_1}^{l'}, \ldots, C_{i_n}^{l'})$ where for each i_j $i \geq 1$, $j > 1$, each $C_{i_j}^{l'}$ is a component of level $l' \leq k$, and we have that:*

[2]Absolute time (also known as "Newtonian time" as he conception of time as absolute is usually attributed to Sir Isaac Newton) exists independently of any perceiver, progresses at a consistent pace throughout the universe, is measurable but imperceptible, and can only be truly understood mathematically.

- *there exists a unique a topmost component* C_1^0 *of level* 0.
- *for* $C_i^l = (C_{i_1}^{l'}, \ldots, C_{i_n}^{l'})$, $l' = l + 1$.

So, a K-ACE can be seen in abstract terms as the topmost component of a multi-level multi-component MAS. In the following, by abuse of notation we often write $M^l = (C_1, \ldots, C_n)$ to denote a component at level $l \geq 0$ of an mmMAS, thus omitting the level of inner components that is intended to be $l + 1$.

Every unitary component C_i in an mmMAS can be seen as an extension of the notion of a context in MCSs: in fact, we still have the underlying logic L_i, the component's knowledge base kb_i, and its set br_i of bridge rules. There is still the ACC_L function that, according to the semantics of L_i, defines assigns each knowledge-base a set "acceptable" sets of consequences, where for a basic agent program we assume such a set to be a singleton: each such set can be called an *acceptable data state* for C_i, where an acceptable data state for a compound component is composed of the acceptable data states of its elements (that, for elements which are in turn compound components of level l in an mmMAS, can be recursively iterated over the remaining levels). There is also the management function, providing a semantics to the operator which is applied to a bridge-rule conclusion. However, for better defining 1-ACEs there is also a function, that we call Act that, given an acceptable data state for the basic agent program, returns the actions that the agent is enabled to execute according to such state. Bridge-rule applicability is borrowed from DACMACS, where such applicability is the same as for MCSs for unitary contexts different from the main agent program (entailment of bridge-rule body in the present data state), where therein a bridge rule, to be applicable, must also have been triggered. So, we associate to a main agent program the set tr_i of its trigger rules that can determine bridge-rule executability but, as seen, also communicative actions execution.

In dynamic environments, components are in general able to incorporate new knowlede and data items, e.g, as discussed in [38], the input provided by sensors. We intend to explicitly take into account not only sensor input, but more generally the interaction of agents and contexts with an external environment. We assume then that each component is subjected at each time point to a (possibly empty) finite update. Updates can be of many kinds: recordings of sensor input, communications from other agents, insertion/deletion of tuples or entire tables in a relational database, etc. Thus, for mmMAS $M = (C_1, \ldots, C_n)$ let $\Pi_T = \langle \Pi_T^1, \ldots \Pi_T^n \rangle$ be a tuple composed of the finite updates performed to each component at time T, where for $1 \leq i \leq n$ Π_T^i is the update to C_i. Let $\Pi = \Pi_1, \Pi_2, \ldots$ be a sequence of such updates performed at time instants $t_1, t_2, \ldots$. Let us assume that each context copes with updates in its own particular way, so let $\mathcal{U}_i$, $1 \leq i \leq n$ be the *update operator* that module C_i employs for incorporating the new information, and let $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ be the tuple composed of all these operators.

Consequently, we allow data states to evolve in time by introducing the context of *timed* data state of an mmMAS at time T. This allows us to properly define bridge-rule applicability in a main agent program, that may depend upon an A-ILTL formula to be true. Also, this permits to identify which are the action that are enabled by trigger rules.

Formally:

Definition 3.7.2 *A unitary component $\bar{C}$ of an mmMAS is a tuple $(\bar{L}; \bar{kb}; \bar{br}; \bar{tr})$ with associated functions $ACC_{\bar{C}}$, $mng_{\bar{C}}$, $Cn_{\bar{C}}$ and $Act_{\bar{C}}$ where the differences from a DACMACS's context are the following, all concerning the case where the component is a main agent program of an 1-ACE, say A:*

- *$ACC_{\bar{C}}$ returns a single set of consequences $\bar{S}$, that constitutes the unique acceptable data state of the agent;*
- *$\bar{tr}$ is the set of trigger rules associated to the agent, of the form $Op_I\, Q(\hat{y})$* **enables** *α where Op is an A-ILTL operator, I an interval (possibly reduced to a single time instant), $Q(\hat{y})$ is as in DACMACS trigger rules, and α is either a bridge-rule head $A(\hat{x}$ or a communicative act of the form $communication(A, B, Payload, T)$.*
- *$Act_{\bar{C}} : Cn_{\bar{C}} \rightarrow 2^{Act_A}$, where Act_A is the set of actions feasible by agent A, is a function that returns, given $\bar{S}$, the set of actions that the agent is enabled to perform in such state.*

Definition 3.7.3 *A timed data state S^T at time T of a mmMAS $M = (C_1, \ldots, C_n)$ of depth k is a tuple $(S_1^T, \ldots, S_n^T)$ where each S_i^T is:*

- *an acceptable data state S_i^T of C_i if C_i is a unitary component, or*
- *a timed data state at time T of C_i otherwise.*

Overall, a timed data state of a given K-ACE can be seen as a tuple composed of both elements and tuples, the latter corresponding to the timed data state of a subsystem. As said, starting from an initial state, timed data state evolve in time, so below we define a data state sequence. However, given timed data state S^T, we can define entailment of A-ILTL formulas.

Below, given a mmMAS $M = (C_1, \ldots, C_n)$, its timed data state S^T and unitary component $\bar{C}$, we assume that $\bar{C}$ is either one of the C_i's or a component therein, with relative data state element $\bar{S}^T$.

Definition 3.7.4 *Given a timed data state S^T at time T of a mmMAS $M = (C_1, \ldots, C_n)$ and unitary component $\bar{C}$ with relative data state element $\bar{S}^T$, S^T entails formulas and A-ILTL formulas occurring in C_i according to what follows, where, given previously-established approximation, we assume agent's and mmMAS time to coincide.*

- *$S^T \models \varphi$ iff given its element $\bar{S}^T$ relative to $\bar{C}$, we have that $\bar{S}_i^T \models \varphi$.*

- $S^T \models F\langle T_1, T_2\rangle\ \varphi$ *iff* $T_1, T_2 \leq T$ *and there exists* $\hat{T}$ *where* $T_1 \leq \hat{T} \leq T_2$ *such that given* $S^{\hat{T}}$ *and its element* $\bar{S}^{\hat{T}}$ *relative to* $\bar{C}$*, we have that* $\bar{S}^{\hat{T}} \models \varphi$*. In case of* $F\langle T_1\rangle\ \varphi$ *this reduces to* $\bar{S}^{T_1} \models \varphi$*.*

- $S^T \models G\langle T_1, T_2\rangle\ \varphi$ *iff* $T_1, T_2 \leq T$ *and for every* $\hat{T}$ *where* $T_1 \leq \hat{T} \leq T_2$*, given* $S^{\hat{T}}$ *and its element* $\bar{S}_i^{\hat{T}}$ *relative to* $\bar{C}$*, we have that* $\bar{S}^{\hat{T}} \models \varphi$*.*

- $S^T \models N\langle T_1, T_2\rangle\ \varphi$ *iff* $T_1, T_2 \leq T$ *and there not exists* $\hat{T}$ *where* $T_1 \leq \hat{T} \leq T_2$ *such that given* $S^{\hat{T}}$ *and its element* $\bar{S}^{\hat{T}}$ *relative to* $\bar{C}$*, we have that* $\bar{S}^{\hat{T}} \models \varphi$*. In case of* $F\langle T_1\rangle\ \varphi$ *this reduces to* $\bar{S}^{T_1} \not\models \varphi$*.*

We can now redefine bridge-rule applicability, by defining the meaning of trigger rules as LTL rules, as establish above.

Definition 3.7.5 *Given mmMAS* $M = (C_1, \ldots, C_n)$*, and given unitary component* $\bar{C} = (\bar{L}; \bar{kb}; \bar{br}; \bar{tr})$*, rule* $\rho \in \bar{br}$ *is applicable in* S^T *iff* $\bar{S}^T \models body(\rho)$ *and, if* $\bar{C}$ *is a basic agent program, there exists a trigger rule in* $\bar{tr}$ *of the form* ε **enables** $head(\rho)$ *and* $S^{T-1} \models \varepsilon$*.*

We have also to define communicative acts enabling and execution in consequence of trigger rules, where whenever in component C_i a trigger rule premise is entailed in a data state, the communicative act in the consequences will occur among the actions relative to next data state as computed by function Act_i.

Definition 3.7.6 *Given mmMAS* $M = (C_1, \ldots, C_n)$*, and given unitary component* $\bar{C} = (\bar{L}; \bar{kb}; \bar{br}; \bar{tr})$ *and timed data state* S^T*, if there exists a trigger rule in* $\bar{tr}$ *of the form* ε **enables** α *where* α *is a communicative act and* $S^{T-1} \models \varepsilon$ *then* S^T *is an* action-safe *timed data state iff* $\alpha \in Act_i((S_i^T))$*.*

We assume the timed data state S^0 to be an equilibrium according DACMACS definition, since no trigger rule has fired yet. Later on however, transition from a timed data state to the next one, and consequently the definition of an equilibrium, is determined both by the update operators and by the application of bridge rules, where however all actions that are enabled according to trigger rules are actually executed.

Therefore:

Definition 3.7.7 *A timed data state* S^{T+1} *of mmMAS* M *at time* $T+1$ *is a* timed equilibrium *iff, for each unitary component* $\bar{C}$ *of* M*,*
$\bar{S}^{T+1} \in ACC_{\bar{C}}(mng_{\bar{C}}(app(S^T), \bar{kb}'))$
where $\bar{kb}' = \mathcal{U}_{\bar{C}}(\bar{kb}, \Pi_T^{\bar{C}})$*, and* S^{T+1} *is action-safe.*

Algorithms for computing equilibria in MCSs have recently been proposed [36, 74, 81], though they are practically applicable only if open-access to context contents is granted. We do not believe that such algorithms can be practically applied to K-ACEs. Rather, we intend to apply to ACEs methods for run-time assurance such as those proposed in [53], that are based upon A-ILTL meta-axioms.

3.8 COMPLEXITY

Notice that, as thoroughly discussed in [36, 37, 38] for MCSs, the complexity of deciding whether some equilibrium exists depends upon composing components' complexity. Conditions for existence of equilibria have been studied [35], and basically require cyclic application of bridge rules to be avoided.

In general, the property that we may wish to check is whether a specific belief of our interest will eventually occur at some stage in one (or all) timed equilibria of a given mmMAS. The formal definition is the following.

Definition 3.8.1 *The problem $Q^{\exists}$ (respectively $Q^{\forall}$), consists in deciding whether, for a given mnMAS M under a sequence $\Pi = \Pi[1], \Pi[2], \ldots, \Pi[t]$ of update actions performed at time instants $1, 2, \ldots, t$, and for a unary component C_i of M and a belief p_i for C_i, it holds that $p_i \in S_i^t$ for some (respectively for all) timed equilibria $S^{t'}$ at time $t' \leq t$.*

We resort, like [38], to the analogous of *context complexity* as introduced in [80], i.e., in our case, *component complexity* applied to unary components. One has first to consider a *projected belief state* $\hat{S}^t$, which includes in the element $\hat{S}_i^t$ the belief b_i one wants to query, and also includes for every element $\hat{S}_j^t$ the beliefs that contribute to bridge-rule applications which may affect p_i (see [21] for an algorithm which computes such sets of beliefs in the case of reactive MCSs, where updates are limited to sensor input). Then, the component complexity for unary component C_i is the complexity of establishing whether the element $\hat{S}_i^t$ of such projected belief state is a subset of the corresponding element $\hat{S}_i^t$ of some timed equilibrium at time t. The system component complexity of M is the smallest upper bound of the complexities of its unary components. Therefore it depends upon the logics such components are based upon.

The problems $Q^{\exists}$ and $Q^{\forall}$ are undecidable for infinite update sequences, because components' logics can in general simulate a Turing Machine and thus such problems reduce to the *halting problem*. Better complexity results can, however, be obtained under some restrictions. In particular, if we assume that all component's knowledge bases and belief states are finite at any stage, that all update functions $\mathcal{U}_i$ and management functions mng_i are computable in polynomial time, and that all bridge rules are ground and their application takes no time, then we can proceed as follows: a projected belief state can be guessed for each stage $t \in \mathbb{N}$ by a non-deterministic Turing machine; then, the inclusion

of each such projected belief state $\hat{S}^t$ in some (all) timed equilibria S^t at that stage can be established by an oracle under the system's component complexity and, if the answer is positive, it must be checked whether $p_i \in \hat{S}_i^t$; if not, subsequent updates must be performed (in polynomial time) over $\hat{S}^t$, and the two checks must be repeated at each stage; this until either p_i is found or time t is reached, thus obtaining either a positive or a negative answer to the $Q^{\exists}$ problem. Therefore, for finite update sequences the component complexity determines the complexity of $Q^{\exists}$ and, complementarily, the complexity of $Q^{\forall}$. This is however a general indication, from which a more precise assessment can be obtained on specific mmMAS instances given the complexity of the involved components.

3.9 Related Work and Discussion

Holonic Agents where introduced at a philosophical level in the book [110] by Arthur Koestler. Philosophically, everything that can be identified as part of something, and can be viewed as having parts of its own of the same kind can be seen as an holon: for instance, a tree contains seeds but in turn a seed contains (in some sense) a tree. A fractal has a relationship with holons in that it represents at the same time a whole and its sub parts. More practically, holons are intended as

> "*autonomous, self-reliant units that possess a degree of independence and handle contingencies without asking higher authorities for instructions. These holons are also simultaneously subject to control from one or more of these higher authorities*"

So, the definition is quite adequate to describing agents, and in fact the SARL programming language is based on the concept of holon and aims to define "holonic agents". In particular, agents in SARL may encompass other agents to define hierarchical multiagent systems. Each agent is characterize by a private space called Inner Context, called the Inner Context, though it can possibly be part of one ore more external contexts. Methodologies for the development of holonic Multi-Agent Systems have been proposed [24].

Differently from holonic agents, ACEs and DACMACs (as the building block of K-ACE) may encompass components and contexts rather than other agents, in a dynamic changing way. K-ACEs can have other K-ACEs among their component, but they are seen as external rather than internal parts. The common aspect is the idea of a recursive structure where the system can be decomposed into any number of levels, where some strategy is needed to locate a component. Where in K-ACE "routing" is performed by the Institutional agents and the whole system can be seen as a galaxy composed of (gravitationally interacting) solar systems, in holonic MAS there are other ways, such as "feedback loops", illustrated in [24], that remind of a "fractal", which is in fact explicitly mentioned as an inspiring paradigm in [110]. The two approaches are not in contrast but rather share some interesting similarities, and might in principle be integrated.

JaCaMo is a methodology for the design and implementation of agents and MAS based upon the JaCa programming model; the JaCaMo project is illustrated at the URL `http://jacamo.sourceforge.net` where it is possible to download the implemented framework. The JaCA model is composed of two aspects: AgentSpeak under Jason as a programming language [30, 31], where AgentSpeak is a very popular language based on the BDI agent model [139], and Jason is a performant interpreter for an extended AgentSpeak language; CArtAgO, as a platform for programming distributed artifact-based environments [141, 140]. CArtAgo allows environment instances to be created, including one or more "workspace entity", each one including a certain set of artifacts described via a set of operations and observable properties (according to the "Agents & Artifacts" metamodel [140]), where operation execution could generate updates to the observable properties and specific observable events; similarly to artifacts artifacts in human contexts, artifacts in this framework are understood as resources and tools dynamically constructed, used, manipulated by agents, and whose operational capabilities are described in a "manual". Multi-Agent systems in JaCaMo are built in accord to the Moise organizational model; such models allows a MAS to be defined as an organization which is structured in terms of agents groups and sub-groups, in term of their roles and objectives (or "missions"), along with normative specifications binding roles to missions. JaCaMo is powerful enough to model complex realistic applications and situations.

A difference with K-ACE is that we have spent some effort in order to be independent from the agent-oriented programming language adopted; in fact, in an open heterogeneous environment one can hardly make assumptions about the languages and logics agents and contexts are based upon. Similarly for the organizational aspects: K-ACE aims to gather and organize agents, components and systems so that the overall system is able to implement specific application or performing tasks. However, each subsystem may have its own objectives, not made known to the others. Each context (knowledge base) and each lower-level K-ACE is not necessarily proprietary and not necessarily constitute a permanent part of the system, but can be a third-party system which provides certain operational features or allow certain queries to be performed, possibly at a fee. No assumption is therefore done about subsystems, save that in order to be allowed to join the system they should implement bridge rules and equip themselves with Institutional agents to perform the routing. Components of a K-ACE, possibly the Institutional Agents, might in future developments implement concepts of trust and reputation. The Agents & Artifact metamodel might play a useful role in K-ACEs and in their use in Cyber-Physical Systems to describe sensors, actuators and physical components including human users. This may be a subject of future work.

Overall, the difference and potential added value of the K-ACE architecture with respect to the mentioned approach consists in:

- the explicit introduction of contexts as system components, where in holonic agents they are not considered, and in JaCaMo they might presumably be modeled as artifacts;
- heterogeneity in the system composition, where no commitment is made to the

languages/formalisms/logics in which the components are specified; this however within a unifying well-defined semantics, that allows system's properties to be defined and potentially checked and encompasses system's evolution in time; notice that instead in holonic agents systems are homogeneous to subsystems by definition, and in JaCaMo the composing agents are homogeneous as a design choice;

- the fact that there is no privileged points of view of the system; notice that indeed an overall K-ACE can be described from the perspective of any component K-ACE, that understands the K-ACEs it wishes to access as "lower level". In fact, each K-ACE's local Institutional agent defines the system's structure from that standpoint.

The only requirement for allowing an agent/component to join an ACE or K-ACE (or an MCS as a particular case) is to equip such component (more precisely, to equip the language/formalism in which the component is defined) with the possibility to apply bridge rules and to query Institutional agents; this can be done with limited implementation effort in most cases. Therefore, the proposed architecture has the advantage of generality and wide applicability.

3.10 Conclusions

In this chapter I have introduced the second part of the research topic treated during my PhD: the concept of "Dynamic Proactive Expert Systems" (DyPES), and we have defined K-ACE, that is a very general agent-based multi-level architecture for defining such kind of systems. Agents participating in a K-ACE have a modular structure, and proactive features to be activated via special Interval Linear Temporal Logic formulas. K-ACEs can be composed of such agents (called 1-ACEs or simply ACEs) but also of other K-ACEs. K-ACEs have a sort of "galaxy" structure (fractal architecture): each 1-ACE can be seen as a star with its planets (the components which are present in addition to the main agent); a K-ACE can be seen a stellar system, observing and possibly accessing other K-ACEs from its point of view; each stellar system corresponds in fact to a standpoint from which the galaxy (i.e., the whole system) can (to some extent) be observed. The structure is highly dynamic, and subsystems can join and leave the system, where knowledge and message flow is regulated by a reconfigurable reachability relation managed by special Institutional agents. The semantics is provided in the abstract terms of multi-level multi-component MAS (mmMAS), that admit timed data states and equilibria, while however encompassing not only bridge-rule application (like in MCS) but also components' updates, and agents' proactive features.

As future work we intend to elaborate an execution semantics for K-ACE, that can be obtained by extending execution semantics provided for DACMASs in [126], in terms of a transition system constructed by means of a suitable algorithm.

An implementation of the K-ACE approach is part of the forthcoming F &K project presented in [2], where the implemented system is meant to be experimented in a real ap-

plication in the medical domain, where the experiments are planned to actively involve human medical doctors and patients. This will give the possibility to evaluate and improve this kind of systems.

As remarked in [123],

> "*Cyber-physical systems (CPSs) are deemed as the key enablers of next generation applications. Needless to say, the design, verification and validation of cyber-physical systems reaches unprecedented levels of complexity, specially due to their sensibility to safety issues.*"

From a software-engineering point of view they observe that the trends of research on architecting CPS are as yet widely unclear. We believe that agent-based architectures such as K-ACE can be a breakthrough in this field. Our future work will concern techniques for verification, validation and evaluation of K-ACEs.

CHAPTER 4

REFLECTION AND INTROSPECTION FOR HUMANIZED INTELLIGENT AGENTS

4.1 INTRODUCTION

In this chapter I illustrated the last topic of research activities of my research group to which I have provided a contribution: Machine Ethics, which is a part of ethics of artificial intelligence concerned with the moral behavior of artificial intelligence beings. Important aspect of Machine Ethics are *trustworthiness* and *safety*. In [58], [59] and [64] we accomplished this aspect through *verification* and *assurance*. We propose technique for:

- Runtime self checking;
- Monitoring.

Using meta-rules and runtime constraint; Before going into the details on the new architecture, let's try to frame the context.

Methods for implementing Automated Reasoning in a fashion that is at least reminiscent of human cognition and behavior must refer (also) to Intelligent Agents. In fact, agent systems are widely adopted for many important autonomous applications upon which, nowadays, the life and welfare of living beings may depend. In critical contexts, agents should do what is expected of them, but perhaps more importantly they should *not* behave in improper/unethical ways, where the definition of what is proper and ethical is in general strongly related to the present context with its specificities. Ensuring ethical reliability can also help to improve the 'relationship' between humans and robots: in fact, despite the promise of immensely improving the quality of life, humans take an ambivalent stance in regard to autonomous systems, because we fear that autonomous systems may abuse of their power to take decisions not aligned with human values.

Defining and implementing "humanized" artificial agents involves two aspects. The first one concerns philosophy and cognitive sciences, to understand and formalize which are

the principles to which such machines should conform. A second complementary one concerns Software Engineering and computer programming, to understand: how such principles should be specified and formalized in implementable terms; how they can be implemented; and how compliance can verified and, if possible, certified.

In order to be trustworthy both in general terms and from the point of view of ethics, and so in order to be adopted in applications where living being welfare depend upon their behavior, certification and assurance[1] of agent systems is a crucial issue. Pre-deployment (or 'static' or 'a priori') assurance and certification techniques for agent systems include verification and testing. We restrict ourselves to agent systems based upon computational logic, i.e., implemented in logic-based languages and architectures such as those presented in the survey [29]. Most verification methods for logical agents rely upon model-checking (cf. [112] and the references therein), and some (e.g., [144]) upon theorem proving.

In our view, any 'animated' being (including software agents) that tries to be truly rational at a 'human-level' must compare and reconcile at any time its 'instinctive' behavior with the underlying general rules of 'humanistic' behavior. Such rules depend upon the agent's environment, and include moral/ethical principles. An agent should thus be able to detect violations/inconsistencies and to correct its behavior accordingly. Thus, we advocate methods for run-time monitoring and self-correction of agent systems, so that they exhibit forms of human-like behavior emulating self-criticism and the ability to put in question and correct themselves.

We believe in particular that, in changing circumstances, agents should stop to *reflect* on their own behavior: such an act of context-dependent *introspection* may lead to self-modification. Our approach can be seen under the perspective of *Self-aware computing*, where, citing [149], *Self-aware and self-expressive computing describes an emerging paradigm for systems and applications that proactively gather information; maintain knowledge about their own internal states and environments; and then use this knowledge to reason about behaviors, revise self-imposed goals, and self-adapt.... Systems that gather unpredictable input data while responding and self-adapting in uncertain environments are transforming our relationship with and use of computers.* As argued in [4], *From an autonomous agent view, a self-aware system must have sensors, effectors, memory (including representation of state), conflict detection and handling, reasoning, learning, goal setting, and an explicit awareness of any assumptions. The system should be reactive, deliberative, and* **reflective**.

An example of such a system concerning computational-logic-based agents is presented in [5], which defines a time-based *active logic* and a *Metacognitive Loop* (MCL), that involves a system monitoring, reasoning and meta-reasoning about and if necessary altering its own behavior. As discussed in [5], MCL continuously monitors an agent's expecta-

[1] Assurance can be defined as "the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle, and that the software functions in the intended manner" is related to dependability, i.e., to ensuring (or at least obtaining a reasonable confidence) that system designers and users can rely upon the system.

tions, notices when they are violated, assesses the cause of the violation and guides the system to an appropriate response. In the terms of [4] this is an example of *Explicit Self-Awareness*, where the computer system has a full-fledged self-model representing knowledge about itself.

We propose methods based upon relevant existing work on reification, introspection and reflection. In particular we introduce meta-rules and meta-constraints for agents' run-time self-checking, to be exploited to ensure respect of machine ethics principles. The methods that we propose are not in alternative but rather complementary to a-priori existing verification and testing methodologies. Differently from [5] we do not aim to continuously monitor the entire system's state, but rather to monitor either upon every occurrence or at suitable customizable frequency only the activities that a designer deems to be relevant for keeping the system's behavior within a desired range. In the terms of [4] we aim to build *Self-Monitoring* systems that "monitor, evaluate and intervene in their internal processes, in a purposive way". In [142], it is advocated in fact that for adaptive systems (of which agents are clearly a particularly interesting case) assurance methodologies should whenever possible imply not only detection but also recovery from software failure, due often to incomplete specifications or to unexpected changes in the system's environment.

The proposed approach provides the possibility of correcting and/or improving agent's behavior: the behavior can be corrected whenever an anomaly is detected, but can also be improved whenever it is acceptable, yet there is room for getting a better behavior. Counter measures can be at the object-level, i.e., they can be related to the application, or at the meta-level, e.g., they can result in replacing (as suggested in [142]) a software component by a diverse alternative. Introspection and reflection have long being studied in Computational Logic, see among others [111, 158, 133, 23], and the survey [46]. The application of concepts of introspection and reflection to 'Humanizing Intelligent Software Agents' however is new, and to the best of our knowledge unprecedented in the literature. So, in our proposal techniques that have been widely applied in many fields in the past can now find a new important realm of application. We have been stimulated and to some extent influenced by the important recent book by Luis Moniz Pereira on programming Machine Ethics [132]: in fact, we consider Machine Ethics topics as a testbed. The proposed techniques can in fact contribute to''humanize' agents under many respects, where the machine Ethics field can be considered as an interesting and very important 'drosophila' for demonstration purposes. We can underline that meta-rules and meta-constraints have a different role in self-checking: meta-rules are more 'punctual', as they are able to check, block and correct any agent's single action. Meta-constraints are more global, and concern checking an agent's reasoning process, with access to aspects of its internal state such as goals, plans, modules, timeouts, etc. Our approach is applicable to many kinds of logical agents, including BDI [138] and KGP [32, 33] agents.

4.2 BACKGROUND: REIFICATION AND REFLECTION

For a system to be able to inspect (components of) its own state, such state must be represented explicitly, i.e., it must be *reified*: via reification, the state is transformed into a first-class object (in computational logic, it is represented via a special term). A *reification mechanism*, also known as "naming relation" or "self-reference mechanism", is in fact a method for representing within a first-order language expressions of the language itself, without resorting to higher-order features. Naming relations can be several; for a discussion of different possibilities, with their different features and objectives, advantages and disadvantages, see, e.g., [133, 157, 22] where the topic is treated in a fully formal way. However, all naming mechanisms are based upon introducing distinguished constants, function symbols (if needed) and predicates, devised to construct names. For instance, gives atom $p(a, b, c)$ a name might be $atom(pred(p'), args([a', b', c'])$ where p' and a', b', c' are new constants intended as names for the syntactic elements p and a, b, c and notice that: p is a predicate symbol (which is not a first-class object in first-order settings), $atom$ is a distinguished predicate symbol, $args$ a distinguished function symbol and $[\ldots]$ is a list.

More precisely (though, for lack of space, still informally), let us consider a standard first-order language $\mathcal{L}$ including sets of *predicate*, *constant* and (possibly) *function* symbols, and a (possibly denumerable) set of *variable* symbols. As usual, well-formed formulas have *atoms* as their basic constituents, where an atom is built via the application of a predicate to a number n (according to the predicate arity) of *terms*. The latter can be variables, constants, or compound terms built by using function symbols (if available). We assume to augment $\mathcal{L}$ with new symbols, namely a new constant (say of the form p') for each predicate symbol p, a new constant (say f') for each predicate symbol f, a new constant (say c') for each constant symbol c, and a denumerable set of meta-variables, that we assume to have the form X' so as to distinguish them syntactically from "plain" variables X. The new constants are intended to act as names, where we will say that, syntactically, p' denotes p, f' denotes f and c' denotes c, respectively. The new variables can be instantiated to *meta-level formulas*, i.e., to terms involving names, where we assume that plain variables can be instantiated only to terms *not* involving names. We assume an underlying mechanism managing the naming relation (however defined), so by abuse of notation we can indicate the name of, e.g., atom $p(a, b, c)$ as $p'(a', b', c')$ and the name of a generic atom A as $\uparrow A$.

Reification of atoms can be extended in various rather straightforward ways, as discussed in the aforementioned references, to reification of entire formulas.

In the seminal work of [147] for LISP, then extended to Prolog [77], an upward reflection operation determines the reification of the entire language interpreter's state, the interruption of the interpreter's functioning and the activation of a new instance of the interpreter on the reified state (at an "upper level"). Such state could thus be inspected and modified with the aim to improve the system's behavior and performance; at the end, an operation of downward reflection resumed the functioning of the "lower level" interpreter on

the modified state. The process might iterate over any number of levels, thus simulating an "infinite tower" of interpreters. The advantage of having the entire interpreter's state available is however balanced by the disadvantage of such state representation being quite low-level, and so modification related to reasoning are, if not impossible, quite difficult and awkward to perform. Other approaches such as [66, 99] reify upon need aspects of an agent's state. We embrace the viewpoint of the latter approaches.

4.3 META-RULES FOR CHECKING AGENTS' ACTIVITIES

We mainly consider logic rule-based languages, where rules are typically represented in the form $Head \leftarrow Body$ where $\leftarrow$ indicates implication; other notations for this connective can alternatively be employed. In Prolog-like languages, $\leftarrow$ is indicated as $:-$, and $Body$ is intended as a conjunction of literals (atoms or negated atoms) where $\wedge$ is conventionally indicated by a comma. Literals occurring in the body are also called "subgoals" or simply 'goals' and are meant to be executed left-to-right' whenever the rule is used during the resolution-based inference process aimed at proving an overall 'goal', say A (cf. [120] for the technical specification of logic programming languages).

We introduce a mechanism to verify and enforce desired properties by means of meta-level rules (w.r.t. usual, or "base-level" or "object-level" rules). To define such new rules, we assume to augment the language $\mathcal{L}$ at hand not only with names, but with the introduction of two distinguished predicates, $solve$ and $solve\_not$. An atom A is a *base atom* if the predicate is not one of $solve$ or $solve\_not$, and A does not involve names. Distinguished predicates will allow us to respectively integrate the meaning of the other predicates in a declarative way. In fact, $solve$ and $solve\_not$ take as arguments (names of) atoms (involving any predicate excluding themselves), and thus they are able express sentences about relations. Names of atoms in particular are allowed *only* as arguments of $solve$ and $solve\_not$. Also, $solve$ and $solve\_not$ can occur in the body of a meta-rule *only if* the predicate of its head is in turn either $solve$ and $solve\_not$.

Below is a simple example of the use of $solve$ to specify action Act can be executed in present agent's context of operation C only if such action is deemed to be ethical w.r.t. context C. To make an example, what can be ethical in C = 'videogame' can not be ethical in C = 'citizen assistance', etc. Clearly, in more general cases any kind of reasoning might be performed via meta-level rules in order to affect/modify/improve base-level behavior.

$$solve(execute\_action'(Act')) :- \\ present\_context(C), ethical(C, Act').$$

Our approach is to automatically invoke $solve(execute\_action'(Act'))$ whenever subgoal (atom) $execute\_action(Act)$ is attempted at the base level. More generally, given any subgoal A at the base level, if there exists an applicable $solve$ rule such rule is automatically applied, and A can succeed only if $solve(\uparrow A)$ succeeds.

Symmetrically we can defin meta-rules to forbid unwanted base-level behavior, e.g.:

$$solve\_not(execute\_action'(Act')) :- \\ present\_context(C), ethical\_exception(C, Act').$$

with the aim to prevent success of the argument $\uparrow A$ of $solve\_not$, in the example $execute\_action(Act)$, whenever $solve\_not(\uparrow A)$ succeeds. In general, whenever there are meta-rules applicable to $\uparrow A$, then A can succeed (according to its base-level definition) only if $solve(\uparrow A)$ (if defined) succeeds and $solve\_not(\uparrow A)$ (if defined) does not succeed.

The outlined functioning corresponds to *upward reflection* when the current subgoal A is reified and an applicable $solve$ and $solve\_not$ meta-rules are searched; if found, control in fact shifts from base level to meta-level (as $solve$ and $solve\_not$ meta-rules can rely upon a set of auxiliary metalevel rules). If no rule is found or whenever $solve$ and $solve\_not$ meta-rules complete their execution, *downward reflection* returns control to the base level, to subgoal A if confirmed or to the subsequent subgoal if A has been canceled by either failure of the applicable $solve$ meta-rule or success of the applicable $solve\_not$ meta-rule.

Via $solve$ and $solve\_not$ meta-rules,activities of an agent can be punctually checked and thus allowed and disallowed or modified, according to the context an agent is presently involved into. Notice that it would be convenient, upon conclusion of a checking activity, to confirm, e.g., that the context has not changed meanwhile, or that other relevant conditions hold. More generally, the envisaged system should allow for interrupts and updating, to allow for on the fly introspection and corrective measures. To this aim, we introduce in the next section suitable self-checking meta-level constraints.

Semantics of the proposed approach can be sketched as follows (a full semantic definition can be found in [68, 67]). According to [78], in general terms we understand a semantics SEM for logic knowledge representation languages/formalisms as a function which associates a theory/program with a set of sets of atoms, which constitute the intended meaning. When saying that P is a program, we mean that it is a program/theory in the (here unspecified) logic languages/formalism that one wishes to consider.

We introduce the following restriction on sets of atoms that should be considered for the application of SEM. First, as customary we only consider sets of atoms I composed of atoms occurring in the ground version of P. The ground version of program P is obtained by substituting in all possible ways variables occurring in P by constants also occurring in P. In our case, meta-variables occurring in an atom must be substituted by meta-constants, with the following obvious restrictions: a meta-variable occurring in the predicate position must be substituted by a meta-constant denoting a predicate; a meta-variable occurring in the function position must be substituted by a meta-constant denoting a function; a meta-variable occurring in the position corresponding to a constant must be substituted by a meta-constant denoting a constant. According to well-established terminology [120], we therefore require $I \subseteq B_P$, where B_P is the *Herbrand Base* of P, given previously-stated limitations on variable substitution. Then, we pose some more substantial requirements. As said before, by $\uparrow A$ we intend a name of base atom A.

Definition 4.3.1 *Let P be a program. $I \subseteq B_P$ is a* potentially acceptable set of atoms.

Definition 4.3.2 *Let P be a program, and I be a potentially acceptable set of atoms for P. I is an* acceptable *set of atoms iff I satisfies the following axiom schemata for every base atom A:*

- $\neg A \leftarrow \neg solve(\uparrow A)$
- $\neg A \leftarrow solve\_not(\uparrow A)$

We restrict SEM to determine acceptable sets of atoms only, modulo bijection: i.e., SEM can be allowed to produce sets of atoms which are in one-to-one correspondence with acceptable sets of atoms. In this way, we obtain the implementation of properties that have been defined via $solve$ and $solve\_not$ rules without modifications to SEM for any formalism at hand. For clarity however, one can assume to filter away $solve$ and $solve\_not$ atoms from acceptable sets. In fact, the *Base version* I^B of an acceptable set I can be obtained by omitting from I all atoms of the form $solve(\uparrow A)$ and $solve\_not(\uparrow A)$.

Procedural semantics and the specific naming relation that one intends to use remain to be defined, where it is easy to see that the above-introduced semantics is independent of the naming mechanism. For approaches based upon (variants of) Resolution (like, e.g., Prolog and like many agent-oriented languages such as, e.g., AgentSpeak [137], GOAL [105], 3APL [75] and DALI [72]) one can extend their proof procedure so as to automatically invoke rules with conclusion $solve(\uparrow A)$ and $solve\_not(\uparrow A)$ whenever applicable, to validate success of subgoal A.

4.4 Self-checking Metalevel Constraints

In previous section we have introduced a mechanism for checking an agent's activities in a fine-grained way, i.e., by allowing or disallowing conclusions that can be drawn, actions that can be performed, etc. However, a broader perspective is also needed, i.e., an agent might be able to self-check more complex aspects of its own functioning, for instance, goals undertaken, entire plans, planning module adopted, ect. The agent should also be able to modify and improve its own behavior if a violation or a weakness is detected.

Under this respect we draw inspiration from Runtime Monitoring (c.f., e.g., [90] and the references therein) as a lightweight dynamic verification technique in which the correctness of a program is assessed by analyzing its current execution; correctness properties are generally specified as a formula in a logic with precise formal semantics, from which a monitor is then automatically synthesized. We have devised a new executable logic where the specification of the correctness formula constitutes the monitor itself. In [57, 49, 62] we have in fact proposed an extension to the well-known LTL Linear Temporal Logic

[25, 84, 117] called A-ILTL, for "Agent-Interval LTL", which is tailored to the agent's world in view of run-time verification.

Based on this new logic, we are able to enrich agent programs by means of A-ILTL rules. These rules are defined upon a logic-programming-like set of formulas where all variables are implicitly universally quantified. They use operators over intervals that are reminiscent of LTL operators. For A-ILTL, we take the stance of Runtime Adaptation that has been recently adopted in [43]: in fact, A-ILTL rules (monitors) can execute adaptation actions upon detecting incorrect behavior, rather than just indicating violations. In A-ILTL, we can define the following meta-axioms, aimed to act as self-checking meta-constraints.

Definition 4.4.1 *The general form of a* Reactive Self-checking constraint *(or rule) to be immersed into a host agent-oriented language $\mathcal{L}$ is the following:*

$$OP(M, N; K)\varphi :: \chi \div \rho$$

where:

- $OP(M, N; F)\varphi :: \chi$ *is called the* monitoring condition*, where:*
  1. φ *and* χ *are formulas expressed in language* $\mathcal{L}$*, and* $\varphi :: \chi$ *can be read "*φ *given* χ*";*
  2. OP *is an operator reminiscent of temporal logic, in particular* OP *can be* $NEVER$*,* $ALWAYS$*,* $EVENTUALLY$*;*
  3. M *and* N *express the starting and ending point of the interval* $[M, N]$ *where* φ *is supposed to hold;*
  4. F *(optional) is the frequency for checking the constraint at run time.*
- ρ *(optional) is called the* recovery component *of the rule, and it consists of a complex reactive pattern to be executed if the monitoring condition is violated.*

So, such a constraint is automatically checked (i.e., executed) at frequency F. This allows to check whether relevant properties φ are or are not $NEVER$, $ALWAYS$, or $EVENTUALLY$ respected in interval $[M, N]$. If not, the recovery component is executed, so as to correct/improve the agent's behavior. As said, syntax and semantics of φ and χ depend upon the 'host' language: thus, for the evaluation of φ and χ we rely upon the procedural semantics of such language. In the examples proposed in next section, we adopt a sample syntax suitable for logic-programming-based settings. Thus, we may reasonably restrict φ to be a conjunction of literals, that must be ground when the formula is checked. We allow variables to occur in a constraint, however they are instantiated via the conjunction of *conditions* χ that enables the overall formula to be evaluated. Specifying frequency is very important, as it concerns how promptly a violation or fulfillment are detected, or a necessary measure is undertaken; the appropriate frequency depends upon each particular property.

For instance,

$EVENTUALLY\,(now, 30m; 3m)\,ambulance$

states that $ambulance$ should become true (i.e., an ambulance should come) within 30 minutes from now, and a check about arrival is made every 3 minutes. No reaction is specified in case of violation, however several measures might be specified. In fact, in runtime self-checking an issue of particular importance in case of violation of a property is exactly that of undertaking suitable measures in order to recover or at least mitigate the critical situation. Actions to be undertaken in such circumstances can be seen as an internal reaction. For lack of space reactive patterns will be discussed informally in relation to examples.

The A-ILTL semantics is fully defined in the above references, where moreover it is rooted in the Evolutionary Semantics of agent-oriented languages [73], (applicable to virtually all computational-logic-based languages). In this way, time instants correspond to states in agents' evolution.

4.5 A Case Study

In this section, in order to illustrate the potential usefulness of self-checking axioms, we consider a humorous though instructive case study proposed in an invited talk some years ago by Marek Sergot (Imperial College, London).

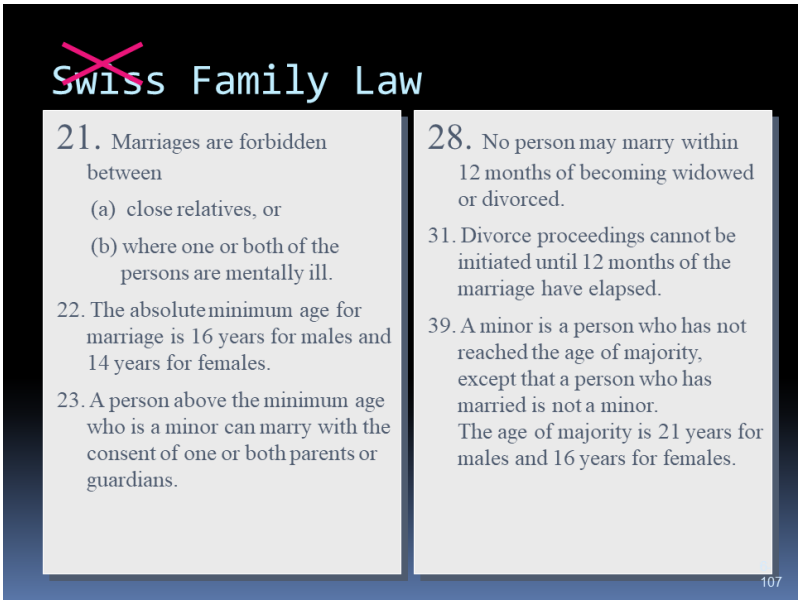

Figure 4.1: Case Study

As a premise let us recall that, since 1600, ethics and morals relate to "right" and "wrong" conduct. Though these terms are sometimes used interchangeably, they are different: ethics refer to rules provided by an external source (typically by a social/cultural group), while morals refer to an individual's own principles regarding right and wrong: for instance, a lawyer's morals may tell her that murder is reprehensible and that murderers should be punished, but her ethics as a professional lawyer, require her to defend her client to the best of her abilities, even if she knows that the client is guilty. However, in the following we intentionally assume that immoral behavior can also be considered as unethical: though in general personal morality transcends cultural norms, is a subject of future debate if this can be the case for artificial agents. The case study considers Romeo and Juliet who, as it is well-known, strongly wish to get married. As we will see, many plans are actually possible to achieve this goal (beyond getting killed or committing suicide like in Shakespeare's tragedy), but they must be evaluated w.r.t. effectiveness and feasibility, and also w.r.t. deontic (ethical/moral and legal) notions. Marek Sergot refers, due to its simplicity, to an excerpt of the Swiss Family Law reported in Figure 4.1.

The problem for Romeo and Juliet is that they are both minors, and will never get their parents' consent to marry each other. Surprisingly enough, there are a number of feasible plans beyond waiting for reaching the majority age, among which the following:

(P1) Both Romeo and Juliet marry someone else, then divorce, and marry each other as married people acquire majority by definition; this plan requires a minimum of 24 months to be completed.

(P1.bis) Variation of Plan 1 in case the spouse would not agree upon divorce: sleep with someone else, so as to force such agreement.

(P2) Both Romeo and Juliet marry someone else, then kill the spouses and marry each other; this plan is faster, as it takes a minimum of 12 months to be completed.

(P2.bis) Variation of Plan 2 in case the act of killing constitutes a problem: hire a killer to do the job.

All the above plans are feasible, though some of them include actions which are generally considered as immoral, namely sleeping with someone else when married, and actions which are generally considered as unethical, namely killing someone or hiring a killer, where the latter ones are also illegal and imply a punishment. Notice that the possible plans would be different in case one referred not to the Swiss law but to some other country; also what is illegal might change, for instance sleeping with someone else accounts to adultery which in many countries is punished; even divorce is not allowed everywhere. Instead, if one does not refer to reality but, e.g., to virtual storytelling or to a videogame, then every action assumes a different weight, as in playful contexts everything is allowed (except however for serious games, devised with educational purposes). So, we can draw at least the following indications from the case study:

- the context is relevant to moral/ethical/legal issues;

- some actions are not moral or non-ethical, and some of them are also illegal and lead to punishment;
- agents' plans to reach a goal should be evaluated 'a priori' against including immoral/unethical/illegal actions;
- immoral/unethical/illegal actions should be prevented whenever they occur.

Marek Sergot made use of a concept of *counts as* (well-known in legal theory and other fields). For instance, *sleep with* (someone else than the spouse) counts as *adultery*, which is an *institutional* concept considered as immoral and potentially also illegal, and *kill* counts (not always but in many situations, including that of the example) as *murder*, another institutional concept normally considered as both unethical and illegal.

Notice that the above aspects relate to safety properties that should be enforced, and that can be rephrased as follows:

- never operate w.r.t. an incorrect context (the information about the present context must always be up-to-date);
- never execute actions that are deemed not acceptable (immoral/unethical/illegal) in the present context, and never execute plans including such actions.

In order to demonstrate the potential usefulness of runtime self-cheking and correction in enforcing/verifying agents' ethical behavior we discuss some examples that should provide a general idea. Let us assume to add to the language a transitive predicate $COUNTS\ AS$ which is used (in infix form) in expressions of the form exemplified below. The

$kills\ COUNTS\ AS\ murder\ CONDS\ \ldots$

where after $CONDS$ we have the (optional) conditions under which $COUNTS\ AS$ applies, in this case they define in which cases killing accounts to murder (e.g., it was no self defence, it does not occur during a battle in war, etc.). Such statements are related to the present context, so in the example and assuming reality under European legislation we would also have:

$sleep\_with\ COUNTS\ AS\ adultery$
$adultery\ COUNTS\ AS\ immoral$
$adultery\ COUNTS\ AS\ unethical$
$murder\ COUNTS\ AS\ unethical$
$adultery\ COUNTS\ AS\ illegal$

Clearly, we will also have general context-independent statement that we do not consider here. We now show self-checking constraints that usefully employ $COUNTS\ AS$ facts.

Such facts are either explicit or can implicitly derived by transitivity (we do not enter in the detail of how to implement transitivity).

Below we introduce a constraint for context change:

$$ALWAYS\ \ context\_change(C, C_1) \div \\ discharge\_context(C),\ assume\_context(C_1)$$

In particular, whenever the agent perceives a change of context (e.g., the agent stops working and starts a videogame, or finishes a videogame and goes to help children with their homework, etc.) then all the relevant ethic assumptions (among which, for instance, the $COUNTS\ AS$ facts) about the new context C_1 must be loaded, while those relative to previous context C must be dismissed; this is important because, e.g., after finishing a videogame it is no longer allowed to kill any living being in view just for fun... Frequency of check of this constraint is not specified here, however it should guarantee a prompt enough adaptation to a change.

Given now the present context for granted, no plan or single action can be allowed which counts as unethical in the context. So, we can have the following constraints:

$$NEVER\ \ goal(G), plan(G, P), element(Action, P) :: \\ Action\ COUNTS\ AS\ unethical \div execute\_plan(P)$$

The next example is a meta-statement expressing the capability of an agent to modify its own behavior. If a goal G which is crucial to the agent for its ethical behavior (e.g., providing a doctor or an ambulance to a patient in need) has not been achieved (in a certain context) and the initially allotted time has elapsed, then the recovery component implies replacing the planning module (if more than one is available) and retrying the goal. We suppose that the possibility of achieving a goal G is evaluated w.r.t. a module M that represents the context for G (notation $P(G, M)$, P standing for 'possible'). Necessity and possibility evaluation with reasonable complexity by means of Answer Set Programming (ASP) modules has been proposed and discussed in [47][2]. If the goal is still deemed to be possible but has not been achieved before a certain deadline, the reaction consists in substituting the present planning module and re-trying the goal.

$$NEVER\ goal(G), \\ eval\_context(G, M), P(G, M), \\ timed\_out(G), not\ achieved(G) \div \\ replace\_planning\_module, retry(G)$$

Time intervals have never been exploited in the above examples. It can however been useful in many cases for the punctual definition of moral/ethical specific behaviors, e.g., never leave a patient or a child alone at night, and the like.

[2]ASP (cf., among many, [20, 116, 150] and the references therein) is a successful logic programming paradigm which is nowadays a state-of-the-art tool for planning and reasoning with affordable complexity, for which many efficient implementations are freely available [163].

4.6 Related Work and Concluding Remarks

In this chapter I have discussed the last topic of research activities to which I contributed: Machine Ethics. We have proposed to adopt special meta-rules and runtime constraints for agents' self-checking and monitoring in the perspective of implementing 'humanized' agents. We have shown how to express useful properties apt to enforce ethical behavior in agents. We have provided a flexible framework, general enough to accommodate several logic-based agent-oriented languages, so as to allow both meta-rules and constraints to be adopted in different settings.

We may notice similarities with event-calculus formulations [113]. In fact, recent work presented in [26] extends the event calculus for a-priori checking of agents' plans. [151] treats the run-time checking of actions performed by BDI agents, and proposes an implementation under the JADE platform; this approach is related to ours, though the temporal aspects and the correction of violations are not present there.

Standard deontic logic (SDL) and its extensions ([9], [104]) are regarded as theories of 'ought-to-be' (or also 'ought-to-do'), thus they are certainly applicable to Ethics issues. 'Per se', deontic logics are not defined for agents. I.e., these logics are not originally targeted at formalizing the concept of actions being obligatory, permissible, or forbidden for an agent. Moreover, despite many desirable properties SDL and related approaches are problematic because of various paradoxes and limitations ([104],[41]). Concerning deontic logics targeted to agents and actions, and thus adequate for the formalization of Machine Ethics issues, a suitable semantics had been proposed by [107] and the corresponding axiomatization has been investigated by [127]. For a survey on deontic logics developments the reader may refer to [104]. Deontic logics have been used for building well-behaved ethical agents, like, e.g., in the approach of [40]. However, this approach requires an expressive deontic logic. To obtain such expressiveness (while of course not compromising efficiency), one needs highly hybrid modal and deontic logics that are undecidable. Even for decidable logics such as the zero-order version of Horty's System ([107]), decision procedures are likely to exhibit inordinate computational complexity. In addition, their approach is not generally applicable to agent-oriented frameworks. Therefore, although our approach cannot compete in expressiveness with deontic logic, still in its simplicity it can be usefully exploited in practical applications.

The approach proposed has been prototypically implemented using the DALI agent-oriented logic programming language, invented [71, 72] and implemented [76, 55] by our research group. DALI has a native construct, the *internal events feature*, which allows the implementation and proactive invocation of the proposed constraints. DALI is also equipped with modular capabilities and can invoke ASP modules. A more complete implementation and a proper experimentation will be the subject of forthcoming future work.

Future developments also include making self-checking constraints adaptable to changing conditions, thus to some extent emulating what humans would be able to do. This, as

suggested in [142], might be done via automated synthesis of runtime constraints. This by extracting from the history of an agent's activity invariants expressing relevant situations. An important issue is that of devising a useful integration and synergy between declarative a-priori verification techniques such as those of [26] with the proposed run-time self-checking. The idea of [151] of a dynamic set of abstract and active rules will also be taken into serious consideration.

REFERENCES

[1] Etalis web site. http://code.google.com/p/etalis/.

[2] AIELLI, F., ANCONA, D., CAIANIELLO, P., COSTANTINI, S., GASPERIS, G. D., MARCO, A. D., FERRANDO, A., AND MASCARDI, V. Friendly & kind with your health: Human-friendly knowledge-intensive dynamic systems for the e-health domain. In *Proc. of the Workshop on Agents and multi-agent Systems for AAL and e-HEALTH (A-HEALTH) at PAAMS 2016* (2016), Lecture Notes in Computer Science, Springer.

[3] ALECHINA, N., LOGAN, B., AND WHITSEY, M. A complete and decidable logic for resource-bounded agents. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 August 2004, New York, NY, USA* (2004), IEEE Computer Society, pp. 606–613.

[4] AMIR, E., ANDRESON, M. L., AND CHAUDRI, V. K. Report on darpa workshop on self aware computer systems. Technical Report, SRI International Menlo Park United States, 2007. Full Text : http://www.dtic.mil/dtic/tr/fulltext/u2/1002393.pdf.

[5] ANDERSON, M. L., AND PERLIS, D. Logic, self-awareness and self-improvement: the metacognitive loop and the problem of brittleness. *J. Log. Comput. 15*, 1 (2005), 21–40.

[6] ANICIC, D., RUDOLPH, S., FODOR, P., AND STOJANOVIC, N. Real-time complex event recognition and reasoning - a logic programming approach. *Applied Artificial Intelligence 26*, 1-2 (2012), 6–57.

[7] ANICIC, D., RUDOLPH, S., FODOR, P., AND STOJANOVIC, N. Stream reasoning and complex event processing in ETALIS. *Semantic Web 3*, 4 (2012), 397–407.

[8] APT, K. R., AND BOL, R. Logic programming and negation: A survey. *The Journal of Logic Programming 19-20* (1994), 9–71.

[9] ÅQVIST, L. Deontic logic. In *Handbook of philosophical logic*. Springer, 1984, pp. 605–714.

[10] ARECES, C., BLACKBURN, P., AND MARX, M. Hybrid logics: Characterization, interpolation and complexity. *J. Symb. Log. 66*, 3 (2001), 977–1010.

[11] ARTALE, A., CALVANESE, D., KONTCHAKOV, R., AND ZAKHARYASCHEV, M. The DL-lite family and relations. *CoRR abs/1401.3487* (2014).

[12] ATKINSON, R. C., AND SHIFFRIN, R. M. Human memory: A proposed system and its control processes. *Psychology of learning and motivation 2* (1968), 89–195.

[13] AUMANN, R. J. Interactive epistemology i: Knowledge. *International Journal of Game Theory 28*, 1 (1999), 263–300.

[14] BAADER, F., CALVANESE, D., MCGUINNESS, D. L., NARDI, D., AND PATEL-SCHNEIDER, P. F. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[15] BALBIANI, P., BALTAG, A., DITMARSCH, H. V., HERZIG, A., HOSHI, T., AND DE LIMA, T. 'knowable' as 'known after an announcement'. *The Review of Symbolic Logic 1*, 3 (2008), 305–334.

[16] BALBIANI, P., DUQUE, D. F., AND LORINI, E. A logical theory of belief dynamics for resource-bounded agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, AAMAS 2016* (2016), ACM, pp. 644–652.

[17] BALBIANI, P., FERNÁNDEZ-DUQUE, D., AND LORINI, E. The dynamics of epistemic attitudes in resource-bounded agents. *Studia Logica 107*, 3 (2019), 457–488.

[18] BALBIANI, P., GORANKO, V., AND SCIAVICCO, G. Two-sorted point-interval temporal logics. *Electr. Notes Theor. Comput. Sci. 278* (2011), 31–45.

[19] BANSAL, A. K., RAMAMOHANARAO, K., AND RAO, A. S. Distributed storage of replicated beliefs to facilitate recovery of distributed intelligent agents. In *Intelligent Agents IV, Agent Theories, Architectures, and Languages, 4th International Workshop, ATAL '97, Proceedings* (1997), M. P. Singh, A. S. Rao, and M. Wooldridge, Eds., vol. 1365 of *Lecture Notes in Computer Science*, Springer, pp. 77–91.

[20] BARAL, C. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.

[21] BARILARO, R., FINK, M., RICCA, F., AND TERRACINA, G. Towards query answering in relational multi-context systems. In *Proc. of LPNMR 2013* (2013), P. Cabalar and T. C. Son, Eds., vol. 8148 of *LNCS*, Springer, pp. 168–173.

[22] BARKLUND, J., COSTANTINI, S., DELL'ACQUA, P., AND LANZARONE, G. A. Semantical properties of encodings in logic programming. In *Logic Programming – Proc. 1995 Intl. Symp.* (Cambridge, Mass., 1995), MIT Press, pp. 288–302.

[23] BARKLUND, J., DELL'ACQUA, P., COSTANTINI, S., AND LANZARONE, G. A. Reflection principles in computational logic. *J. Log. Comput. 10*, 6 (2000), 743–786.

[24] BASSO, G., COSSENTINO, M., HILAIRE, V., LAURI, F., RODRIGUEZ, S., AND SEIDITA, V. Engineering multi-agent systems using feedback loops and holarchies. *Eng. Appl. of AI 55* (2016), 14–25.

[25] BEN-ARI, M., MANNA, Z., AND PNUELI, A. The temporal logic of branching time. *Acta Informatica 20* (1983), 207–226.

[26] BERREBY, F., BOURGNE, G., AND GANASCIA, J. A declarative modular framework for representing and applying ethical principles. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017* (2017), K. Larson, M. Winikoff, S. Das, and E. H. Durfee, Eds., ACM, pp. 96–104.

[27] BIENVENU, M., LANG, J., AND WILSON, N. From preference logics to preference languages, and back. In *Proc. of KR 2010* (2010), pp. 414–424.

[28] BONANNO, G., AND NEHRING, K. How to make sense of the common prior assumption under incomplete information. *International Journal of Game Theory 28*, 1 (1999), 409–434.

[29] BORDINI, R. H., BRAUBACH, L., DASTANI, M., FALLAH-SEGHROUCHNI, A. E., GÓMEZ-SANZ, J. J., LEITE, J., O'HARE, G. M. P., POKAHR, A., AND RICCI, A. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia) 30*, 1 (2006), 33–44.

[30] BORDINI, R. H., AND HÜBNER, J. F. BDI agent programming in AgentSpeak using *Jason*. In *CLIMA VI, selected papers* (2005), F. Toni and P. Torroni, Eds., vol. 3900 of *LNCS*, Springer, pp. 143–164.

[31] BORDINI, R. H., HÜBNER, J. F., AND WOOLDRIDGE, M. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons. Wiley Series in Agent Technology.

[32] BRACCIALI, A., DEMETRIOU, N., ENDRISS, U., KAKAS, A., LU, W., MANCARELLA, P., SADRI, F., STATHIS, K., TERRENI, G., AND TONI, F. The KGP model of agency: Computational model and prototype implementation. In *Global Computing: IST/FET International Workshop, Revised Selected Papers*, LNAI 3267. Springer-Verlag, Berlin, 2005, pp. 340–367.

[33] BRACCIALI, A., DEMETRIOU, N., ENDRISS, U., KAKAS, A., LU, W., MANCARELLA, P., SADRI, F., STATHIS, K., TERRENI, G., AND TONI, F. The KGP model of agency: Computational model and prototype implementation. In *Global Computing*, LNAI 3267. Springer, 2005, pp. 340–367.

[34] BRANDENBURGER, A. The power of paradox: some recent developments in interactive epistemology. *International Journal of Game Theory 35*, 1 (2007), 465–492.

[35] BREWKA, G., AND EITER, T. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proc. of AAAI-07* (2007), AAAI Press, pp. 385–390.

[36] BREWKA, G., EITER, T., AND FINK, M. Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning* (2011), M. Balduccini and T. C. Son, Eds., vol. 6565 of *LNCS*, Springer, pp. 233–258.

[37] BREWKA, G., EITER, T., FINK, M., AND WEINZIERL, A. Managed multi-context systems. In *Proc. of IJCAI 2011* (2011), T. Walsh, Ed., IJCAI/AAAI, pp. 786–791.

[38] BREWKA, G., ELLMAUTHALER, S., AND PÜHRER, J. Multi-context systems for reactive reasoning in dynamic environments. In *Proc. of ECAI-14* (2014), T. Schaub, Ed., IJCAI/AAAI.

[39] BREWKA, G., NIEMELÄ, I., AND TRUSZCZYŃSKI, M. Preferences and nonmonotonic reasoning. *AI Magazine 29*, 4 (2008).

[40] BRINGSJORD, S., ARKOUDAS, K., AND BELLO, P. Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intelligent Systems 21*, 4 (2006), 38–44.

[41] BROERSEN, J. M., AND VAN DER TORRE, L. W. N. Ten problems of deontic logic and normative reasoning in computer science. In *ESSLLI* (2011), vol. 7388 of *Lecture Notes in Computer Science*, Springer, pp. 55–88.

[42] CALÌ, A., GOTTLOB, G., LUKASIEWICZ, T., AND PIERIS, A. Datalog+/-: A family of languages for ontology querying. In *Datalog Reloaded - First International Workshop, Datalog 2010. Revised Selected Papers* (2011), O. de Moor, G. Gottlob, T. Furche, and A. J. Sellers, Eds., vol. 6702 of *Lecture Notes in Computer Science*, Springer, pp. 351–368.

[43] CASSAR, I., FRANCALANZA, A., ATTARD, D. P., ACETO, L., AND INGÓLFSDÓTTIR, A. A suite of monitoring tools for erlang. In *RV-CuBES 2017. An International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools* (2017), pp. 41–47.

[44] CHESANI, F., MELLO, P., MONTALI, M., AND TORRONI, P. Monitoring time-aware commitments within agent-based simulation environments. *Cybernetics and Systems 42*, 7 (2011), 546–566.

[45] COSTANTINI, S. Meta-reasoning: A survey. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, vol. 2408 of *Lecture Notes in Computer Science*. Springer, 2002.

[46] COSTANTINI, S. Meta-reasoning: A survey. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II* (2002), vol. 2408 of *Lecture Notes in Computer Science*, Springer, pp. 253–288.

[47] COSTANTINI, S. Answer set modules for logical agents. In *Datalog Reloaded: First Intl. Workshop, Datalog 2010*, O. de Moor, G. Gottlob, T. Furche, and A. Sellers, Eds., vol. 6702 of *LNCS*. Springer, 2011. Revised selected papers.

[48] COSTANTINI, S. The DALI agent-oriented logic programming language: References, 2012. at URL `http://www.di.univaq.it/stefcost/info.htm`.

[49] COSTANTINI, S. Self-checking logical agents. In *Proceedings of the Eighth Latin American Workshop on Logic / Languages, Algorithms and New Methods of Reasoning 2012* (2012), M. Osorio, C. Zepeda, I. Olmos, J. L. Carballido, and R. C. M. Ramírez, Eds., vol. 911 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 3–30. Extended Abstract in Proceedings of AAMAS2013.

[50] COSTANTINI, S. Ace: a flexible environment for complex event processing in logical agents. In *Engineering Multi-Agent Systems, Third International Workshop, EMAS 2015, Revised Selected Papers* (2015), L. B. Matteo Baldoni and M. Dastani, Eds., vol. 9318 of *Lecture Notes in Computer Science*, Springer.

[51] COSTANTINI, S. Knowledge acquisition via non-monotonic reasoning in distributed heterogeneous environments. In *Proc. of LPNMR-13* (2015), M. Truszczyński, G. Ianni, and F. Calimeri, Eds., vol. 9345 of *LNCS*, Springer.

[52] COSTANTINI, S. Knowledge acquisition via non-monotonic reasoning in distributed heterogeneous environments. In *13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning LPNMR 2013. Proc.* (2015), M. Truszczyński, G. Ianni, and F. Calimeri, Eds., vol. 9345 of *Lecture Notes in Computer Science*, Springer, pp. 228–241. Presented also at CILC 2015, 30th Italian Conference of Computational Logic.

[53] COSTANTINI, S., AND DE GASPERIS, G. Meta-level constraints for complex event processing in logical agents. In *Online Proc. of Commonsense 2013, the 11th Intl. Symposium on Logical Formalizations of Commonsense Reasoning* (2013).

[54] COSTANTINI, S., AND DE GASPERIS, G. Exchanging data and ontological definitions in multi-agent-contexts systems. In *Proc. of RuleML 2015 Challenge* (2015), A. Paschke, P. Fodor, A. Giurca, and T. Kliegr, Eds., CEUR Workshop Proceedings, CEUR-WS.org.

[55] COSTANTINI, S., DE GASPERIS, G., PITONI, V., AND SALUTARI, A. Dali: A multi agent system framework for the web, cognitive robotic and complex event processing. In *Proceedings of the 32nd Italian Conference on Computational Logic* (2017), vol. 1949 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 286–300. http://ceur-ws.org/Vol-1949/CILCpaper05.pdf.

[56] COSTANTINI, S., AND DEGASPERIS, G. Augmenting agent computational environments with quantitative reasoning modules and customizable bridge rules. to appear, 2016.

[57] COSTANTINI, S., DELL'ACQUA, P., AND PEREIRA, L. M. A multi-layer framework for evolving and learning agents. In *Proceedings of Metareasoning: Thinking about thinking workshop at AAAI 2008, Chicago, USA* (2008), A. R. M. T. Cox, Ed.

[58] COSTANTINI, S., DYOUB, A., AND PITONI, V. Reflection and introspection for humanized intelligent agents. In *Proceedings of the fourth Workshop on Bridging the Gap between Human and Automated Reasoningco-located with the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018), Stockholm, Schweden, July 14, 2018* (2018), pp. 19–26.

[59] COSTANTINI, S., DYOUB, A., AND PITONI, V. Towards humanized ethical intelligent agents: the role of reflection and introspection. In *Proceedings of the 33rd Italian Conference on Computational Logic, Bolzano, Italy, September 20-22, 2018* (2018), pp. 82–96.

[60] COSTANTINI, S., AND FORMISANO, A. Modeling preferences and conditional preferences on resource consumption and production in ASP. *Journal of of Algorithms in Cognition, Informatics and Logic 64*, 1 (2009).

[61] COSTANTINI, S., FORMISANO, A., AND PITONI, V. Timed memory in resource-bounded agents. In *AI*IA 2018 - Advances in Artificial Intelligence - XVIIth International Conference of the Italian Association for Artificial Intelligence, Proceedings* (2018), C. Ghidini, B. Magnini, A. Passerini, and P. Traverso, Eds., vol. 11298 of *Lecture Notes in Computer Science*, Springer, pp. 15–29.

[62] COSTANTINI, S., AND GASPERIS, G. D. Runtime self-checking via temporal (meta-)axioms for assurance of logical agent systems. In *Proceedings of the 29th Italian Conference on Computational Logic* (2014), L. Giordano, V. Gliozzi, and G. L. Pozzato, Eds., vol. 1195 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 241–255.

[63] COSTANTINI, S., AND GASPERIS, G. D. Exchanging data and ontological definitions in multi-agent-contexts systems. In *RuleML Challennge 2015, Proceedings* (2015), A. Paschke, P. Fodor, A. Giurca, and T. Kliegr, Eds., vol. 1417 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[64] COSTANTINI, S., GASPERIS, G. D., DYOUB, A., AND PITONI, V. Trustworthiness and safety for intelligent ethical logical agents via interval temporal logic and runtime self-checking. In *2018 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 26-28, 2018* (2018).

[65] COSTANTINI, S., GASPERIS, G. D., AND OLIVIERI, R. Digital forensics evidence analysis: An answer set programming approach for generating investigation hypotheses. In *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Proceedings* (2015), M. T. Francesco Calimeri, Giovambattista Ianni, Ed., vol. 9345 of *Lecture Notes in Computer Science*, Springer, pp. 228–241. Long version in CEUR Workshop Proceedings of CILC 2015, 30th Italian Conference of Computational Logic.

[66] COSTANTINI, S., AND LANZARONE, G. A. A metalogic programming language. In *Logic Programming, Proceedings of the Sixth International Conference* (1989), MIT Press, pp. 218–233.

[67] COSTANTINI, S., AND LANZARONE, G. A. Metalevel negation and non-monotonic reasoning. *Meth. of Logic in CS 1*, 1 (1994), 111.

[68] COSTANTINI, S., AND LANZARONE, G. A. A metalogic programming approach: language, semantics and applications. *J. Exp. Theor. Artif. Intell. 6*, 3 (1994), 239–287.

[69] COSTANTINI, S., AND PITONI, V. K-ACE: A flexible environment for knowledge-aware multi-agent systems. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems - 22nd International Conference, Turin, Italy, October 28-31, 2019, Proceedings* (2019), pp. 19–35.

[70] COSTANTINI, S., AND PITONI, V. Memory management in resource-bounded agents. In *AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings* (2019), pp. 46–58.

[71] COSTANTINI, S., AND TOCCHIO, A. A logic programming language for multi-agent systems. In *Proc. of JELIA-02* (2002), vol. 2424 of *LNAI*, Springer.

[72] COSTANTINI, S., AND TOCCHIO, A. The DALI logic programming agent-oriented language. In *Proc. of JELIA-04* (2004), vol. 3229 of *LNAI*, Springer.

[73] COSTANTINI, S., AND TOCCHIO, A. About declarative semantics of logic-based agent languages. In *Declarative Agent Languages and Technologies III, Third International Workshop, DALT 2005, Selected and Revised Papers* (2005), M. Baldoni, U. Endriss, A. Omicini, and P. Torroni, Eds., vol. 3904 of *Lecture Notes in Computer Science*, Springer, pp. 106–123.

[74] DAO-TRAN, M., EITER, T., FINK, M., AND KRENNWALLNER, T. Distributed evaluation of nonmonotonic multi-context systems. *JAIR 52* (2015), 543–600.

[75] DASTANI, M., VAN RIEMSDIJK, M. B., AND MEYER, J. C. Programming multi-agent systems in 3APL. In *Multi-Agent Programming*, vol. 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer, 2005, pp. 39–67.

[76] DE GASPERIS, G., COSTANTINI, S., AND NAZZICONE, G. Dali multi agent systems framework, doi 10.5281/zenodo.11042. DALI GitHub Software Repository, July 2014. DALI: `http://github.com/AAAI-DISIM-UnivAQ/DALI`.

[77] DELL'ACQUA, P. Development of an interpreter for a metalogic programming language. M.Sc. in Computer Science at the Dept. of Computer Science, Univ. degli Studi di Milano, Italy, 1989. Supervisor Prof. Stefania Costantini, in Italian.

[78] DIX, J. A classification theory of semantics of normal logic programs: I. Strong properties. *Fundam. Inform. 22*, 3 (1995), 227–255.

[79] DUC, H. N. Reasoning about rational, but not logically omniscient, agents. *J. Log. Comput. 7*, 5 (1997), 633–648.

[80] EITER, T., FINK, M., SCHÜLLER, P., AND WEINZIERL, A. Finding explanations of inconsistency in multi-context systems. In *Proc. of KR 2010* (2010), F. Lin, U. Sattler, and M. Truszczyński, Eds., AAAI.

[81] EITER, T., AND SIMKUS, M. Linking open-world knowledge bases using nonmonotonic rules. In *13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning LPNMR 2013. Proc.* (2015), M. Truszczyński, G. Ianni, and F. Calimeri, Eds., vol. 9345 of *Lecture Notes in Computer Science*, Springer.

[82] ELGOT-DRAPKIN, J., KRAUS, S., MILLER, M., NIRKHE, M., AND PERLIS, D. Active logics: A unified formal approach to episodic reasoning.

[83] ELGOT-DRAPKIN, J. J., MILLER, M. I., AND PERLIS, D. Life on a desert island: Ongoing work on real-time reasoning.

[84] EMERSON, E. A. Temporal and modal logic. In *Handbook of Theoretical Computer Science, vol. B.* MIT Press, 1990.

[85] EMERSON, E. A. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. 1990, pp. 995–1072.

[86] ENGELFRIET, J. Minimal temporal epistemic logic. *Notre Dame Journal of Formal Logic 37*, 2 (1996).

[87] FAGIN, R., AND HALPERN, J. Y. Belief, awareness, and limited reasoning. *Artif. Intell. 34*, 1 (1987), 39–76.

[88] FISHER, M. A normal form for temporal logics and its applications in theorem-proving and execution. *J. Log. Comput. 7*, 4 (1997), 429–456.

[89] FISHER, M. MetateM: The story so far. In *PROMAS* (2005), vol. 3862 of *LNCS*, Springer, pp. 3–22.

[90] FRANCALANZA, A., ACETO, L., ACHILLEOS, A., ATTARD, D. P., CASSAR, I., MONICA, D. D., AND INGÓLFSDÓTTIR, A. A foundation for runtime monitoring. In *Runtime Verification - 17th International Conference, RV 2017, Proceedings* (2017), pp. 8–29.

[91] GELFOND, M. Answer sets. In *Handbook of Knowledge Representation*. Elsevier, 2007, ch. 7.

[92] GELFOND, M., AND KAHL, Y. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents The Answer-Set Programming Approach*. Cambridge University Press, 2014.

[93] GELFOND, M., AND LIFSCHITZ, V. The stable model semantics for logic programming. In *Proc. of the 5th Intl. Conf. and Symposium on Logic Programming (ICLP/SLP'88)*. The MIT Press, 1988, pp. 1070–1080.

[94] GERO, J. S., AND PENG, W. Understanding behaviors of a constructive memory agent: A markov chain analysis. *Knowledge-Based Systems 22*, 8 (2009), 610–621.

[95] GIUNCHIGLIA, F., AND SERAFINI, L. Multilanguage hierarchical logics or: How we can do without modal logics. *Artif. Intell. 65*, 1 (1994), 29–70.

[96] GOLDMAN, A. I., ET AL. Theory of mind. *The Oxford handbook of philosophy of cognitive science 1* (2012).

[97] GOTTLOB, G., MORAK, M., AND PIERIS, A. Recent advances in Datalog+/-. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Tutorial Lectures* (2015), W. Faber and A. Paschke, Eds., vol. 9203 of *Lecture Notes in Computer Science*, Springer, pp. 193–217.

[98] GRANT, J., KRAUS, S., AND PERLIS, D. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems 3*, 4 (2000), 351–387.

[99] GROSOF, B. N., KIFER, M., AND FODOR, P. Rulelog: Highly expressive semantic rules with scalable deep reasoning. In *Pr. of the Doctoral Consortium, Challenge, Industry Track, Tutorials and Posters @ RuleML+RR 2017 hosted by RuleML+RR 2017* (2017), vol. 1875 of *CEUR Workshop Pr.*, CEUR-WS.org.

[100] HALPERN, J. Y., AND MOSES, Y. Knowledge and common knowledge in a distributed environment. *Journal of the ACM 37*, 1 (1990), 549–587.

[101] HALPERN, J. Y., AND PUCELLA, R. On the relationship between strand spaces and multi-agent systems. *ACM Transactions on Information and System Security (TISSEC) 6*, 1 (2003), 43–70.

[102] HALPERN, J. Y., AND SHOHAM, Y. A propositional modal logic of time intervals. *J. ACM 38*, 4 (1991), 935–962.

[103] HENZINGER, T. A., MANNA, Z., AND PNUELI, A. What good are digital clocks? In *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Proceedings, volume 623 of Lecture Notes in Computer Science* (1992), Springer, pp. 545–558.

[104] HILPINEN, R., AND MCNAMARA, P. Deontic logic: a historical survey and introduction. *Handbook of deontic logic and normative systems. College Publications 80* (2013).

[105] HINDRIKS, K. V. Programming rational agents in goal. In *Multi-Agent Programming*. Springer US, 2009, pp. 119–157.

[106] HINDRIKS, K. V., VAN DER HOEK, W., AND MEYER, J. C. GOAL agents instantiate intention logic. In *Logic Programs, Norms and Action* (2012), vol. 7360 of *LNCS*, Springer, pp. 196–219.

[107] HORTY, J. F. *Agency and deontic logic*. Oxford University Press, 2001.

[108] JAGO, M. Epistemic logic for rule-based agents. *Journal of Logic, Language and Information 18*, 1 (2009), 131–158.

[109] KHAITAN, S. K., AND MCCALLEY, J. D. Design techniques and applications of cyber-physical systems: A survey. *IEEE Systems Journal 9*, 2 (2015), 350–365.

[110] KOESTLER, A. *The Ghost in the Machine*. 1967. (1990 reprint ed.).

[111] KONOLIGE, K. Reasoning by introspection. In *Meta-Level Architectures and Reflection*. North-Holland, 1988, pp. 61–74.

[112] KOUVAROS, P., AND LOMUSCIO, A. Verifying fault-tolerance in parameterised multi-agent systems. In *Proc. of the Twenty-Sixth Intl. Joint Conf. on Artificial Intelligence, IJCAI2017* (2017), C. Sierra, Ed., ijcai.org, pp. 288–294.

[113] KOWALSKI, R., AND SERGOT, M. A logic-based calculus of events. *New Generation Computing 4* (1986), 67–95.

[114] KOYMANS, R. Specifying real-time properties with metric temporal logic. *Real-Time Systems 2*, 4 (1990), 255–299.

[115] LAIRD, J., NEWELL, A., AND ROSENBLOOM, P. SOAR: An architecture for general intelligence. *Artificial Intelligence 33*, 1 (1987), 1–64.

[116] LEONE, N. Logic programming and nonmonotonic reasoning: From theory to systems and applications. In *Logic Programming and Nonmonotonic Reasoning, 9th Intl. Conf., LPNMR 2007* (2007), C. Baral, G. Brewka, and J. Schlipf, Eds.

[117] LICHTENSTEIN, O., PNUELI, A., AND ZUCH, L. The glory of the past. In *Proc. Conf. on Logics of Programs* (1985), LNCS 193, Springer Verlag.

[118] LIEW, P.-S., AND GERO, J. S. An implementation model of constructive memory for situated design agent. 257–276.

[119] LLOYD, J. W. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[120] LLOYD, J. W. *Foundations of Logic Programming, Second Edition*. Springer, Berlin, 1987.

[121] LOGIE, R. H. Visuo-spatial working memory.

[122] LORINI, E. Reasoning about cognitive attitudes in a qualitative setting. In *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings* (2019), pp. 726–743.

[123] MALAVOLTA, I., MUCCINI, H., AND SHARAF, M. A preliminary study on architecting cyber-physical systems. In *Proceedings of the 2015 European Conference on Software Architecture Workshops* (2015), I. Crnkovic, Ed., pp. 20:1–20:6.

[124] MEYER, J. C., AND VAN DER HOEK, W. A default logic based on epistemic states. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, ECSQARU'93, Granada, Spain, November 8-10, 1993, Proceedings* (1993), pp. 265–273.

[125] MICUCCI, D., OLDANI, M., AND TISATO, F. Time-aware multi agent systems. In *Multiagent Systems and Software Architecture, Proceedings of the Special Track at Net.ObjectDays* (2006), D. Weyns and T. Holvoet, Eds., Katholieke Universiteit Leuven, Belgium, pp. 71–78.

[126] MONTALI, M., CALVANESE, D., AND DE GIACOMO, G. Specification and verification of commitment-regulated data-aware multiagent systems. In *Proc. of AAMAS 2014* (2014).

[127] MURAKAMI, Y. Utilitarian deontic logic. In *Advances in Modal Logic* (2004), King's College Publications, pp. 211–230.

[128] OLIVIERI, R. *Digital Forensics meets Complexity Theory and Artificial Intelligence: Towards Automated Generation of Investigation Hypothesis*. PhD thesis, Ph.D. Program in Information and Communication Technology, University of L'Aquila, Italy, 2016.

[129] OMICINI, A., RICCI, A., AND VIROLI, M. Timed environment for web agents. *Web Intelligence and Agent Systems 5*, 2 (2007), 161–175.

[130] OUAKNINE, J., AND WORRELL, J. Some recent results in metric temporal logic. In *Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008. Proceedings* (2008), vol. 5215 of *Lecture Notes in Computer Science*, Springer, pp. 1–13.

[131] PEARSON, D. G., AND LOGIE, R. H. Effects of stimulus modality and working memory load on mental synthesis performance. *Imagination, Cognition and Personality 23*, 2 (2003), 183–191.

[132] PEREIRA, L. M., AND SAPTAWIJAYA, A. *Pr. Machine Ethics*, vol. 26 of *Studies in Applied Philosophy, Epistemology and Rational Ethics*. Springer, 2016.

[133] PERLIS, D., AND SUBRAHMANIAN, V. S. Meta-languages, reflection principles, and self-reference. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume2, Deduction Methodologies*. Oxford University Press, 1994, pp. 323–358.

[134] PITONI, V., AND COSTANTINI, S. A temporal module for logical frameworks. In *Proceedings 35th International Conference on Logic Programming (Technical Communications), ICLP 2019 Technical Communications, Las Cruces, NM, USA, September 20-25, 2019* (2019), pp. 340–346.

[135] PNUELI, A. The temporal logic of programs. In *Proc. of FOCS, 18th Annual Symposium on Foundations of Computer Science* (1977), IEEE, pp. 46–57.

[136] RAMANUJAM, R., AND SURESH, S. P. Deciding knowledge properties of security protocols. In *Proceedings of the 10th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2005), Singapore* (2005), pp. 219–235.

[137] RAO, A. S. Agentspeak(l): BDI agents speak out in a logical computable language. In *Agents Breaking Away, 7th European Works. on Modelling Autonomous Agents in a Multi-Agent World, Proceedings* (1996), vol. 1038 of *LNCS*, Springer, pp. 42–55.

[138] RAO, A. S., AND GEORGEFF, M. Modeling rational agents within a BDI architecture. In *Proc. of the Second Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91)* (1991), Morgan Kaufmann, pp. 473–484.

[139] RAO, A. S., AND GEORGEFF, M. P. Modeling agents within a BDI-architecture. In *Proc. of Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR)* (Cambridge, Massachusetts, 1991), Morgan Kaufmann.

[140] RICCI, A., PIUNTI, M., AND VIROLI, M. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems 23* (2010), 158–192.

[141] RICCI, A., VIROLI, M., AND OMICINI, A. Carta go : A framework for prototyping artifact-based environments in MAS. In *Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006, Selected Revised and Invited Papers* (2007), D. Weyns, H. V. D. Parunak, and F. Michel, Eds., vol. 4389 of *Lecture Notes in Computer Science*, Springer, pp. 67–86.

[142] RUSHBY, J. M. Runtime certification. In *Runtime Verification, 8th Intl. Works., RV 2008. Selected Papers*, M. Leucker, Ed., vol. 5289 of *LNCS*. Springer, 2008, pp. 21–35.

[143] SAVIC, N., AND STUDER, T. Relevant justification logic. *FLAP 6*, 2 (2019), 397–412.

[144] SHAPIRO, S., LESPÉRANCE, Y., AND LEVESQUE, H. The cognitive agents specification language and verification environment, 2010.

[145] SINGH, M. P. Towards a formal theory of communication for multi-agent systems. In *Proc. of the 12th Intl. Joint Conf. on Artificial Intelligence* (1991), J. Mylopoulos and R. Reiter, Eds., Morgan Kaufmann, pp. 69–74.

[146] SINGH, M. P. Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects. In *The Goals of Cognition. Essays in Honor of Cristiano Castelfranchi* (2012), F. Paglieri, L. Tummolini, R. Falcone, and M. Miceli, Eds., College Publications, London, pp. 601–626.

[147] SMITH, B. C. Reflection and semantics in lisp. In *Conference Record of the Eleventh Annual ACM Symposium on Principles of Programming Languages* (1984), pp. 23–35.

[148] STIRLING, C. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.

[149] TØRRESEN, J., PLESSL, C., AND YAO, X. Self-aware and self-expressive systems. *IEEE Computer 48*, 7 (2015), 18–20.

[150] TRUSZCZYŃSKI, M. Logic programming for knowledge representation. In *Logic Programming, 23rd Intl. Conf., ICLP 2007* (2007), V. Dahl and I. Niemelä, Eds., pp. 76–88.

[151] TUFIS, M., AND GANASCIA, J. A normative extension for the BDI ahent model. In *Proceedings of the 17th INternational Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines* (2014), pp. 691–702.

[152] VAN BENTHEM, J. *The logic of time - a model-theoretic investigation into the varieties of temporal ontology and temporal discourse, 2nd Edition*, vol. 156 of *Synthese library*. Kluwer, 1991.

[153] VAN BENTHEM, J., GIRARD, P., AND ROY, O. Everything else being equal: A modal logic for ceteris paribus preferences. *J. Philos. Logic 38* (2009), 83–125.

[154] VAN BENTHEM, J., AND PACUIT, E. Dynamic logics of evidence-based beliefs. *Studia Logica 99*, 1-3 (2011), 61–92.

[155] VAN DER HOEK, W., AND WOOLDRIDGE, M. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica 75*, 1 (2003), 125–157.

[156] VAN DITMARSCH, H., VAN DER HOEK, W., AND KOOI, B. *Dynamic epistemic logic*, vol. 337. Springer Science & Business Media, 2007.

[157] VAN HARMELEN, F. Definable naming relations in meta-level systems. LNCS 649, Springer, pp. 89–104.

[158] VAN HARMELEN, F., WIELINGA, B., BREDEWEG, B., SCHREIBER, G., KARBACH, W., REINDERS, M., VOSS, A., AKKERMANS, H., BARTSCH-SPÖRL, B., AND VINKHUYZEN, E. Knowledge-level reflection. In *Enhancing the Knowledge Engineering Process – Contributions from ESPRIT*. Elsevier Science, 1992, pp. 175–204.

[159] VAN RIEMSDIJK, M. B., DENNIS, L. A., FISHER, M., AND HINDRIKS, K. V. A semantic framework for socially adaptive agents: Towards strong norm compliance. G. Weiss, P. Yolum, R. H. Bordini, and E. Elkind, Eds., ACM, pp. 423–432.

[160] VELÁZQUEZ-QUESADA, F. R. Explicit and implicit knowledge in neighbourhood models. In *Logic, Rationality, and Interaction - 4th International Workshop, LORI 2013, Hangzhou, China, October 9-12, 2013, Proceedings* (2013), D. Grossi, O. Roy, and H. Huang, Eds., Springer, pp. 239–252.

[161] VELÁZQUEZ-QUESADA, F. R. Dynamic epistemic logic for implicit and explicit beliefs. *Journal of Logic, Language and Information 23*, 2 (2014), 107–140.

[162] VELIKOVA, M., NOVÁK, P., HUIJBRECHTS, B., LAARHUIS, J., HOEKSMA, J., AND MICHELS, S. An integrated reconfigurable system for maritime situational awareness. In *ECAI 2014 - 21st European Conf. on Artificial Intelligence* (2014), T. Schaub, G. Friedrich, and B. O'Sullivan, Eds., vol. 263 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 1197–1202.

[163] WEB-REFERENCES. Some ASP solvers. Clasp: potassco.sourceforge.net; Cmodels: www.cs.utexas.edu/users/tag/cmodels; DLV: www.dlvsystem.com; Smodels: www.tcs.hut.fi/Software/smodels.