Knowledge Representation for Cognitive Robotic Systems

Emil Vassev
Lero–the Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
emil.vassev@lero.ie

Mike Hinchey
Lero–the Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
mike.hinchey@lero.ie

Abstract — **Cognitive robotics are autonomous systems capable of artificial reasoning. Such systems can be achieved with a logical approach, but still AI struggles to connect the abstract logic with real-world meanings. Knowledge representation and reasoning help to resolve this problem and to establish the vital connection between knowledge, perception, and action of a robot. Cognitive robots must use their knowledge against the perception of their world and generate appropriate actions in that world in compliance with some goals and beliefs. This paper presents an approach to multi-tier knowledge representation for cognitive robots, where ontologies are integrated with rules and Bayesian networks. The approach allows for efficient and comprehensive knowledge structuring and awareness based on logical and statistical reasoning.**

Keywords - knowledge representation; reasoning; robotics.

I. INTRODUCTION

Modern robotic systems boast intrinsic intelligence that helps them reason about situations where autonomous decision making is required. Robotic artificial intelligence (AI) mainly excels at formal logic, which allows it, for example, to find the right chess move from hundreds of previous games. The basic compound in the reasoning process is knowledge. Smart robots employ appropriately structured knowledge that is used by embedded inference engines. The knowledge is integrated in the robot system via knowledge representation techniques to build a computational model of the operational domain in which symbols serve as *knowledge surrogates* for real world artefacts, such as a robot's components and functions, task details, environment objects, etc. The domain of interest can cover any part of the real world or any hypothetical system about which one desires to represent knowledge for computational purposes.

In this paper, a framework called KnowLang developed for employing *knowledge representation and reasoning* (KR&R) within robotic systems is presented. The application of KR&R to robotic systems has been an increasingly interesting topic for cognitive robotics. Examples are found in semantic mapping [1], improving planning and control aspects [2], and most notably HRI systems [3, 4]. A key feature of the framework is a multi-tier specification model allowing for integration of ontologies together with rules and Bayesian networks [5]. The KnowLang framework aims at efficient and comprehensive knowledge structuring and awareness based on logical and statistical reasoning. It helps us tackle 1) explicit representation of domain concepts and relationships; 2) explicit representation of particular and general factual knowledge, in terms of predicates, names, connectives, quantifiers and identity; and 3) uncertain knowledge in which additive probabilities are used to represent degrees of belief. Other notable features are related to knowledge cleaning (allowing for efficient reasoning) and knowledge representation for autonomic robotic behavior.

The rest of this paper is organized as follows. Section II discusses the utility of KR&R. Section III presents some of the significant challenges we need to overcome for efficient KR&R. Section IV presents the KnowLang specification model. Section V discusses our KR strategy for autonomic behavior. Finally, Section VI provides brief concluding remarks and a summary of our future goals.

II. WHY KNOWLEDGE REPRESENTATION?

When it comes to AI, we think about the knowledge we must transfer to the computerized machines and make them use that knowledge, so they exhibit intelligence. In this regard, one of the first questions we need to answer is on the notion of knowledge. So, what is knowledge? To answer this question we should consider two facts: 1) it is known that knowledge is related to intelligence; and 2) the definition of knowledge should be given with terms from the computer domain. Scientists agree that the concept of intelligence is built upon four fundamental elements: data, information, knowledge, and wisdom. In general, data takes the form of measures and representations of the world—for example, raw facts and numbers. Information is obtained from data by assigning relevant meaning, usually by putting data in a specific context. Knowledge is a specific interpretation of information. And wisdom is the ability to apply relevant knowledge to a particular problem.

Intelligent system designers use knowledge representation to give computerized systems large amounts of knowledge that helps them understand the problem domain. Still computers "talk" in a "binary" language, which is simple, logical, and sound, and has no sense of ambiguity typical for a human language. Therefore, computers cannot be simply given textbooks, which they understand and use, just like humans do. Instead, the knowledge given to computers must be structured in well-founded computational structures that computer programs may translate to the binary language of a computer. Knowledge representation structures may be primitives such as rules, frames, semantic

networks and concept maps, ontologies, and logic expressions. These primitives may be combined into more complex knowledge elements. Whatever elements they use, designers must structure the knowledge so that the system can effectively process and it and humans can easily perceive the results.

Many conventional developers doubt the utility of knowledge representation. The fact is that knowledge representation and accompanying reasoning can significantly slow a system down when it has to decide what actions to take, and it looks up facts in a knowledge base to reason with them at runtime. This is one of the main arguments against knowledge representation. Why not simply "compile out" the entire knowledge as "procedural knowledge", which makes the system relatively faster and more efficient. However, this strategy will work for a fixed set of tasks, i.e., procedural knowledge will give the system the entire knowledge the system needs to know. However, AI deals with an open set of tasks and those cannot be determined in advance (at least not all of them). This is the big advantage of using knowledge representation – AI needs it to solve complex problems where the operational environment is non-deterministic and a system needs to reason at runtime to find missing answers.

III. Challenges and Discussion

A. Completeness and Consistency

An essential assumption when working on KR&R is that represented knowledge cannot provide a complete picture of the domain of interest. The fundamental reasons are that domain objects often present real things that cannot be described by a finite set of symbolic structures. Moreover, such objects do not exist in isolation, but are included in unlimited sets of encompassing contexts. Therefore, KR&R should consider *incompleteness* as a drawback and the challenge is to find the level of *sufficient knowledge* so an AI system may rely on reasoning to infer missing knowledge.

Another aspect of the knowledge completeness is related to the way a system assumes its operational world. In this regard, we may have: 1) Closed World Assumption (CWA) - assumes a complete and closed model of the world; 2) Open World Assumption (OWA) - assumes an incomplete and open model of the world. Whereas the CWA strategy assumes that unless an atomic sentence is known to be true, it can be assumed to be false, the OWA strategy assumes that any information not explicitly specified (or such that cannot be derived from the known data) is considered unknown. Note that cognitive robots often operate in an open-ended environment, which requires OWA strategy. Moreover, although more restrictive, CWA provides for avoiding inconsistency in knowledge - if consistent, knowledge cannot become inconsistent, because CWA does not allow the addition of new facts, which may lead to inconsistency. Note that knowledge consistency is important for efficient reasoning but is not mandatory in knowledge representation. Here, a challenging task is how to preserve consistency in KR for cognitive robots employing the OWA strategy. A possible solution is to use as part of KR special constraints ensuring that the knowledge will be correctly processed by inference engines. For example, there could be constraints for knowledge acquisition, knowledge retrieval, knowledge update and knowledge inference.

B. Converting Sensor Data to KR Symbols

A cognitive robotic system has sensors that connect it to the world. These sensors generate raw data that represent the physical characteristics of the world. These low-level data streams must be: 1) converted to programming variables or more complex data structures that represent collections of sensory data; 2) those programing data structures must be labeled with KR symbols. Hence, it is required to relate encoded data structures with KR concepts and objects used for reasoning purposes.

C. Encoded vs. Represented Knowledge

Ontology [6], *concept maps* [7], *frames*, *rules* and *constraints* [8] are intended to present distinct pieces of knowledge that are worth representing differently. Note that an important distinction is between *ontological*, *factual* and *rule-based knowledge*. Whereas the first is related to the general categories (presented as concepts) and important objects in the domain of interest, the second makes assertions about some specific concepts and objects. The rules may imply special relations between the concepts and objects or impose a special semantics for such relations. Robotic platforms do not necessarily emphasize knowledge-centered systems. This means that developers may encode a large part of the "a priori" knowledge (knowledge given to the system before the latter actually runs) in the implemented classes and routines. In such a case, the knowledge-represented pieces of knowledge (e.g., concepts, relations, rules, etc.) may complement the knowledge codified into implemented program classes and routines. For example, rules could be based on classes and methods and a substantial concern about the rules organization is how to relate the knowledge expressed with rules to implemented methods and functions. A possible solution is to map concepts and objects to program classes and objects respectively and design rules working on the input (parameters, pre-conditions) and output (results, post-conditions) of implemented methods.

D. Explicit vs. Implicit Knowledge

Knowledge represented in an AI system may be considered as *explicit knowledge*. Explicit knowledge can be used by the system to infer *implicit knowledge*. The process is based on extracting the knowledge explicitly represented and acquiring the implicit knowledge through augmentation and inference techniques. One of the challenges we need to overcome is how the system shall differentiate between the knowledge that is inferred and such that is explicitly stored, e.g., inferred facts versus explicit facts. Here, one of the important questions is "Could knowledge that can be inferred also be available as explicit facts?". Due to its self-learning capabilities, a cognitive robotic system shall be able to register new concepts, objects, relations, facts, etc. Therefore, the question is not whether the system needs to store inferred knowledge, but rather how to decide on

thresholds determining what part of the inferred knowledge should be stored and when. Note that storing (and thus making explicit) all the inferred knowledge is not a solution, because this will introduce huge redundancy and eventually inconsistency, both having a great impact on reasoning. Our approach to discovering such inferred-explicit knowledge thresholds is to determine: 1) when an inferred piece of knowledge (a concept, an object, a relation, a fact, etc.) called knowledge k is used as a basis to infer a new piece of knowledge called knowledge k'; and 2) check whether knowledge k' is not further used to infer knowledge k (cyclical inference). Thus, the inferred knowledge should be only one level deep, i.e., a piece of inferred knowledge should be only based on explicit concepts, objects, etc.

E. Reasoning

When a cognitive robotic system needs to decide on a course of action and there is no explicit knowledge about this, the system must reason. Basically, reasoning is how a system uses its knowledge to figure out what it needs to know from what is already known. There are two main categories of reasoning: 1) *monotonic reasoning* - new facts can only produce additional beliefs; and 2) *non-monotonic reasoning* - new facts will sometimes invalidate previous beliefs. Computations over the represented knowledge are done by inference engines that act to produce new knowledge. Usually, the inference engines are based on First Order Logic (FOL) [9] or on Description Logics (DL) [10]. FOL-based inference engines (e.g., VAMPIRE, SPASS, E-Theorem Prover, etc.) use algorithms from automated deduction dedicated to FOL, such as theorem proving and model building. Theorem proving can help in finding contradictions or checking for new information. Finite model building can be seen as a complementary inference task to theorem proving, and it often makes sense to use both in parallel.

1) Decidability. The problem with FOL-based inference is that the logical entailment for FOL is *semi-decidable*, which means that if the desired conclusion follows from the premises then eventually resolution refutation will find a contradiction. As a result, queries often unavoidably do not terminate. Inference engines based on DL (e.g., Racer, DLDB, etc.) are extremely powerful when reasoning about *taxonomic knowledge*, since they can discover hidden subsumption relationships amongst classes. However, their expressive power is restricted in order to reduce the computational complexity and to guarantee the decidability (based on DL are decidable) of their deductive algorithms. Consequently, this restriction prevents taxonomic reasoning from being widely applicable to heterogeneous domains, e.g., navigating a crowded room. The problem is called the *symbol grounding problem* [11], i.e. how to make symbols used by an AI system refer to the "proper meaning". A cognitive robotic system deals with *infinitude* and *heterogeneous knowledge* to satisfy a multitude of conflicting and evolving demands. In order to process such knowledge properly, an AI system needs an internal mechanism for symbol grounding that will autonomously "make sense" of its heterogeneous world by cleaning and fixing its knowledge.

2) Hybrid Reasoning. For more powerful reasoning capabilities, different inferential engines can be used together and their results may eventually be "fused" together. The challenge is how to combine those results into one cognitive conclusion, which is related to resolving conflicting results coming out from the different inference engines.

3) Distributed Reasoning. Robot systems organized on the agent-based principle may comprise multiple autonomous cognitive robots that communicate and self-organize on common tasks. Such systems need to share knowledge and common interpretation of facts, so they can reason together about situations, etc.

4) Reasoning over "Clean" Knowledge. For efficient reasoning, it should be possible to reason by emphasizing the relevant knowledge and ignoring selected parts of the represented knowledge.

5) Heuristic Reasoning about Potential Correlations. As part of the symbol grounding problem, groups of concepts or attributes whose names (structures) have terms (components) in common should be suggested as candidates for explicit relationships between them.

F. Probability and Statistics, Experience

Decision-making is a complex process that is often based on more than *logical conclusions*. Probability and statistics may provide for the so-called *probabilistic* and *statistical reasoning* intended to capture uncertain knowledge in which additive probabilities are used to represent degrees of belief of rational agents in the truth of statements. For example, the purpose of a statistical inference might be to draw conclusions about a population based on data obtained from a sample of that population. Probability theory and Baye's theorem [12] lay the basis for such reasoning where Bayesian networks [5] are used to represent belief probability distributions, which actually summarize a potentially infinite set of possible circumstances. The key point is that nodes in a Bayesian network have direct influence on other nodes and given values for some nodes, it is possible to infer the probability distribution for values of other nodes. How a node influences another node is defined by the conditional probability for the nodes usually based on past experience. The experience can be associated with the success of the actions generated in the physical environment by the cognitive robot. Maintaining an execution history of the actions shall help the robot eventually compute the success probability for those actions. In that way, the robot may learn (infer new knowledge) not to execute actions that traditionally have a low success rate. Experience plays a central role in the learning process. For example, experience can be gained while a robot is acting and then eventually abstracted and stored in the knowledge base. The problem is how to abstract this experience with KR symbols.

IV. KNOWLANG

KnowLang is a formal specification language providing a comprehensive specification model aiming at addressing the knowledge representation problem for intelligent systems (robotic systems in particular). The complexity of the problem necessitates the use of a specification model where knowledge can be presented at different levels of abstraction. KnowLang imposes a multi-tier specification model (see Figure 1), where we specify *knowledge corpuses*, *KB (knowledge base) operators* and *inference primitives* at different hierarchically organized tiers.

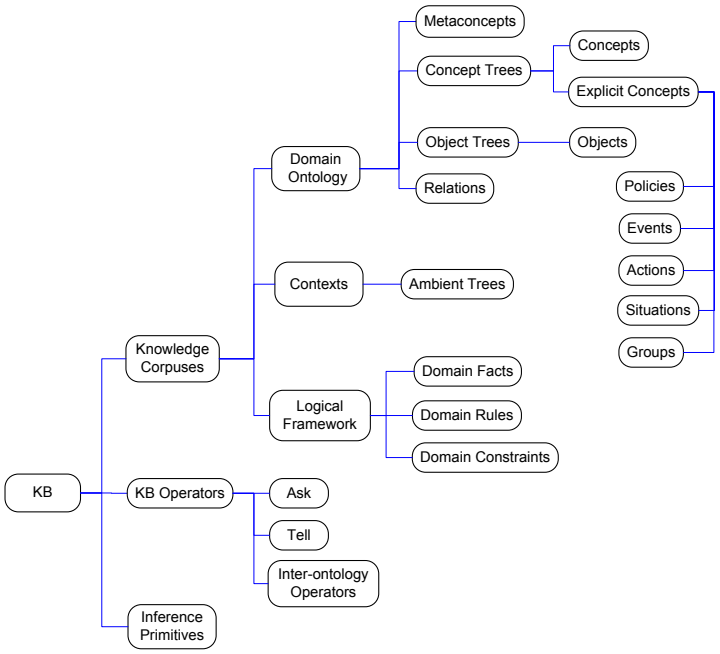

Figure 1. KnowLang Multi-tier Specification Model

Definitions 1 through 49 outline a formal representation of the KnowLang specification model. As shown in Definition 1, a Knowledge Base is a tuple of three main knowledge components - *knowledge corpus* (Kc), *KB operators* (Op) and *inference primitives* (Ip). A Kc is a tuple of three knowledge components – *ontologies* (O), contexts (Cx) and *logical framework* (Lf) (see Definition 2). Further, a domain ontology is composed of hierarchically organized sets of *meta-concepts* (Cm), *concept trees* (Ct), *object trees* (Ot) and *relations* (R) (see Definition 4). Meta-concepts (Cm) provide a context-oriented *interpretation* (i) (see Definition 6) of concepts and might be optionally associated with specific contexts (the square brackets "[]" mean "optional"). Meta-concepts help ontologies to be viewed from different context perspectives by establishing different meanings for some of the key concepts. This is a powerful construct providing for interpretations of a concept and its derived concept tree depending on the current context.

Concept trees (Ct) consist of semantically related *concepts* (C) and/or explicit *concepts* (Ce). Every *concept tree* (ct) has a *root concept* (tr) because the architecture ultimately must reference a single concept that is the connection point to concepts that are outside the concept tree. A root concept may *optionally* inherit a meta-concept, which is denoted $[tr \twoheadrightarrow cm]$ (see Definition 8). The square brackets "[]" mean "optional" and "$\twoheadrightarrow$" is the *inherits* relation. Every concept has a set of *properties* (P) and optional sets of functionalities (F), *parent concepts* (Pr) and *children concepts* (Ch) (see Definition 10).

Explicit concepts are concepts that **must** be presented in the knowledge representation of a cognitive robotic system. Explicit concepts are mainly intended to support 1) the autonomic behavior of the SCs; and 2) distributed reasoning and knowledge sharing among the robots of multi-robot systems. These concepts might be *policies* (Π), *events* (E), *actions* (A), *situations* (Si) and *groups* (Gr) (see Definition 13), i.e., they allow for quantification over such concepts.

FORMAL REPRESENTATION OF KNOWLANG

$Kb := \{Kc,\ Op,\ Ip\}$	(*Knowledge Base*)	(1)
$Kc := \{O,\ Cx,\ Lf\}$	(*Knowledge Corpus*)	(2)

DOMAIN ONTOLOGIES

$O := \{o_{sc}, o_{sce}, o_{env}, o_{si}\}$	(*Ontologies*)	(3)
$o := \{Cm,\ Ct, Ot, R\},\ o \in O$	(*Ontology*)	(4)
$Cm := \{cm_0, cm_1, \ldots, cm_n\}$	(*Meta-concepts*)	(5)
$cm := \{[cx],\ i\}$	(*Meta-concept, cx – Context*)	(6)
$i \in Icx$	(i – *Interpretation*)	
$Ct := \{ct_0,\ ct_1, \ldots, ct_n\}$	(*Concept Trees*)	(7)
$ct := \{tr,\ C,\ [Ce]\}$	(*Concept Tree*)	(8)
$tr \in (C \cup Ce),\ [tr \twoheadrightarrow cm]$	(tr – *Tree Root*)	
$C := \{c_0,\ c_1, \ldots, c_n\}$	(*Concepts*)	(9)
$c := \{P, [F], [Pr], [Ch]\}$	(*Concept*)	(10)
$Pr \subset (C \cup Ce),\ c \twoheadrightarrow Pr$	(Pr – *Parents*)	
$Ch \subset (C \cup Ce),\ c \twoheadleftarrow Ch$	(Ch – *Children*)	
$P := \{p_0, p_1, \ldots, p_n\}$	(*Properties*)	(11)
$F := \{f_0, f_1, \ldots, f_n\}$	(*Functionalities*)	(12)
$Ce := \{\Pi, E, A, Si, Gr\}$	(*Explicit Concepts*)	(13)

Policies (Π) are responsible for the *autonomic behavior* of the cognitive robot (see Section V). A policy π has a *goal* (g), *policy situations* (Si_π), *policy-situation relations* (R_π), and *policy conditions* (N_π) *mapped* to *policy actions* (A_π), where the evaluation of N_π may imply the evaluation of actions (denoted with $N_\pi \longrightarrow A_\pi$) (see Definition 15). A *condition* is a Boolean function over ontology (see Definition 17) or the occurrence of specific events or situations in the system. Thus, policy conditions may be expressed with *policy events*. Policy situations (Si_π) are situations (see Definition 22) that may trigger a policy π, which implies the evaluation of the policy conditions N_π (denoted with $Si_\pi \longrightarrow \pi \longrightarrow N_\pi$). A policy may also comprise optional *policy-situation relations* (R_π) justifying the relationships between a policy and the associated situations. The presence of probabilistic belief in those relations justifies the probability of policy execution, which may vary with time.

A *goal* is a desirable transition from a *state* to another *state* (denoted $s \Longrightarrow s'$) (see Definition 18). The system may occupy a state (s) when the *properties* of an object are updated (denoted $Tell \rhd ob.P$), the *properties* of a set of objects get updated, or some events have occurred in the system or in the environment (denoted with $Tell \rhd E_s$) (see Definition 19). Note that *Tell* is a KB Operator involving knowledge inference (see Definition 46).

A *situation* is expressed with a state (s), a *history of actions* ($A_{si}^{\leftarrow}$) (actions executed to get to state s), *actions* A_{si} that can be performed from state s and an optional *history of events* $E_{si}^{\leftarrow}$ that eventually occurred to get to state s (see Definition 23).

$$\Pi := \{\pi_0, \pi_1, \dots, \pi_n\} \quad (Policies) \tag{14}$$

$$\pi := \{g,\ Si_\pi, [R_\pi], N_\pi,\ A_\pi,\ map(N_\pi,\ A_\pi)\}\ (Policy) \tag{15}$$

$$A_\pi \subset A_{si},\ N_\pi \longrightarrow A_\pi \quad (A_\pi - Policy\ Actions)$$

$$E_\pi \subset E, E_\pi \subset N_\pi \quad (E_\pi - Policy\ Events)$$

$$Si_\pi \subset Si, Si_\pi \longrightarrow \pi \longrightarrow N_\pi \quad (Si_\pi - Policy\ Situations)$$

$$R_\pi \subset R, r_\pi := \{\pi, [rn], [Z], si_\pi\} \quad (R_\pi - Policy - Situation\ Realtion)$$

$$N_\pi := \{n_0, n_1, \dots, n_n\} \quad (Policy\ Conditions) \tag{16}$$

$$n := bf(O) \mid E_\pi \mid Si_\pi \quad (Condition - Boolean\ Statement,\ Ev.,\ Situations) \tag{17}$$

$$g := (s \Longrightarrow s') \quad (Goal) \tag{18}$$

$$s := \langle Tell \rhd ob.P \rangle \mid \langle Tell \rhd \{ob_0.P,\ ob_1.P, \dots,\ ob_n.P\} \rangle \mid \langle Tell \rhd E_s \rangle \quad (State) \tag{19}$$

$$E_s \subset E \quad (E_s - State\ Events)$$

$$E := \{e_0, e_1, \dots, e_n\} \quad (Events) \tag{20}$$

$$A := \{a_0, a_1, \dots, a_n\} \quad (Actions) \tag{21}$$

$$Si := \{si_0, si_1, \dots, si_n\} \quad (Situations) \tag{22}$$

$$si := \{s, A_{si}^{\leftarrow}, [E_{si}^{\leftarrow}],\ A_{si}\} \quad (Situation) \tag{23}$$

$$A_{si}^{\leftarrow} \subset A \quad (A_{si}^{\leftarrow} - Executed\ Actions)$$

$$A_{si} \subset A \quad (A_{si} - Possible\ Actions)$$

$$E_{si}^{\leftarrow} \subset E \quad (E_{si} - Situation\ Events)$$

A *group* involves objects related to each other through a distinct set of relations (see Definition 25). Note that groups are an explicit concept intended to (but not restricted to) represent knowledge about the structure of the system.

Object trees (Ot) are conceptualization of how objects existing in the world of interest are related to each other. The relationships are based on the principle that objects have properties, where sometimes the value of a property is another object, which in turn also has properties. Such properties are termed *object properties* (Pb). An object tree consists of a root object (Pb) and an optional set of object properties (Pb) (see Definition 27). An *object* (ob) has a set of *properties* (P) including object properties (Pb) and is an instance of a concept (denoted as $instof(c)$ - see Definition 28).

$$Gr := \{gr_0, gr_1, \dots, gr_n\} \quad (Groups) \tag{24}$$

$$gr := \{Ob_{gr}, R_{gr}\} \quad (Group) \tag{25}$$

$$Ob_{gr} \subset Ob \quad (Ob_{gr} - Group\ Objects,\ Ob - Objects)$$

$$R_{gr} \subset R \quad (R_{gr} - Group\ Relations)$$

$$Ot := \{ot_0, ot_1, \dots, ot_n\} \quad (Object\ Trees) \tag{26}$$

$$ot := \{ob, [Pb]\} \quad (Object\ Tree) \tag{27}$$

$$ob := \{instof(c), P\}, ob \in Ob \quad (Object) \tag{28}$$

$$Pb := \{ob_0, ob_1, \dots, ob_n\}, Pb \subset P \quad (Object\ Properties) \tag{29}$$

Relations connect two concepts, two objects, or an object with a concept and may have probability distribution Z (e.g., over time, over situations, over concepts' properties, etc.). A relation has an optional name, i.e., when the name is missing

we have the *implication* relation. Probability distribution is provided to support *probabilistic reasoning*. By specifying relations with probability distributions we actually specify Bayesian networks connecting the concepts and objects of an ontology. Note that we consider *binary relations* only, but may have multiple relations relating same the concepts/objects.

$$R := \{r_0, r_1, \dots, r_n\} \quad (\textit{Relations}) \tag{30}$$

$$r := \{c_k, [rn], [Z], c_n\} \mid \{ob_k, [rn], [Z], ob_n\} \quad (\textit{Relation, } rn \textit{ – rel. name, } Z \textit{ – probability distr.}) \tag{31}$$

CONTEXTS

$$Cx := \{cx_0, cx_1, \dots, cx_n\} \quad (\textit{Contexts}) \tag{32}$$

$$cx := \{At, [Icx]\} \quad (\textit{Context}) \tag{33}$$

$$At := \{at_0, at_1, \dots, at_n\} \quad (\textit{Ambient Trees}) \tag{34}$$

$$at := \{ct, Ca, [i]\} \quad (\textit{Ambient Tree}) \tag{35}$$

$$ct \in Ct \quad (\textit{Concept Tree described by an ontology})$$

$$Ca \subset C \quad (Ca - \textit{Ambient Concepts})$$

$$i \in Icx \quad (i - \textit{Ambient Tree Interpretation})$$

$$Icx := \{i_0, i_1, \dots, i_n\} \quad (\textit{Context Interpretations}) \tag{36}$$

Contexts are intended to extract the relevant knowledge from an ontology. Moreover, contexts carry interpretation for some of the meta-concepts (see Definition 6), which may lead to new interpretation of the *descendant concepts* (derived from a meta-concept – see Definition 8). We consider a very broad notion of context, e.g., the environment in a fraction of time or a generic situation such as currently-ongoing important system function such as observing, listening, etc. Thus, a context must emphasize the key concepts in an ontology, which helps the inference mechanism narrow the domain knowledge (domain ontology) by exploring the concept trees down only to the emphasized key concepts.

Depending on the context, some low-level concepts might be subsumed by their upper-level parent concepts, just because the former are not relevant for that very context. For example, a robot wheel can be considered as a thing or as an important part of the robot's motion system. As a result, the context interpretation of knowledge will help the system deal with "clean" knowledge and the reasoning will be more efficient. A context (cx) consists of *ambient trees* (At) and optional *context interpretations* (Icx) (see Definition 33). An *ambient tree* (at) consists of a real concept tree (ct) described by an ontology (o), *ambient concepts* (Ca) part of the concept tree and optional *context interpretation* (i).

The ambient concepts (see Definition 35) explicitly determine new level of deepness for their *original concept tree*, i.e., ambient concepts subsume all of their *child concepts* (if any). As result, when a robot reasons about a particular context (expressed with ambient trees), the reasoning process does not consider those *child concepts*, but their *ambient parents*, which are far more generic, and thus less detailed. This technique reduces the size of the relevant knowledge, by temporarily removing from the concept trees all the ambient concepts' children (child concepts). We may think about ambient trees as filters the system applies at runtime to reduce the visibility of concepts of a concept tree.

LOGICAL FRAMEWORK

$$Lf := \{Fa, Rl, Ct\} \quad (\textit{Logical Framework}) \tag{37}$$

$$Fa := \{fa_0, fa_1, \dots, fa_n\} \quad (\textit{Facts}) \tag{38}$$

$$fa := bf(O) \longrightarrow \mathbf{T} \quad (\textit{Fact – True statement over ontology}) \tag{39}$$

$$Rl := \{rl_0, rl_1, \dots, rl_n\} \quad (\textit{Rules}) \tag{40}$$

$$rl := if\ fa_1\ then\ fa_2\ [else\ fa_3] \quad (\textit{Rule}) \tag{41}$$

$$Ct := \{ct_0, ct_1, \dots, ct_n\} \quad (\textit{Constraints}) \tag{42}$$

$$ct := \langle if\ fa_1\ then\ MUST\ fa_2\rangle \mid \langle if\ fa_1\ then\ MUST\ \neg fa_2\rangle \quad (\textit{Constraint}) \tag{43}$$
$$fa_1, fa_2 \in Fa$$

A KR *Logical Framework* (Lf) is composed of *facts* (Fa), *rules* (Rl) and *constraints* (Ct) (Definition 37). As shown in Definitions 37 through 43, the Lf's KR structures are built with ontology terms:

- *facts* – define *true statements* in the ontologies (O);
- *rules* – relate *hypotheses* to *conclusions*;
- *constraints* – used to validate knowledge, i.e., to check its consistency; can be *positive* or *negative*.

The KR Logical Framework helps developers realize the explicit representation of particular and general factual knowledge, in terms of predicates, names, connectives, quantifiers and identity. It provides computational structures (additional to the ontology) that basically determine logical foundations helping a robot reason and infer knowledge.

The *Knowledge Base Operators* (Op) can be grouped into three groups: Ask operators (retrieve knowledge from a *knowledge corpus* Kc), $Tell$ operators (update a Kc) and *inter-ontology operators* (Oop) intended to work on one or more ontologies (see Definitions 44 through 47). Such operators can be, *merging*, *mapping*, *alignment*, etc. Note

that all the *Knowledge Base Operators* (Op) may imply the use of *inference primitives*.

KNOWLEDGE BASE OPERATORS

$Op := \{Ask,\ Tell,\ Oop\}$ (*Knowledge Base Operators*) (44)

$Ask := retrieve\ (Kc) \longrightarrow Ip\ \lhd\ Kc$ (*query knowledge base*) (45)

$Tell := update\ (Kc) \longrightarrow Ip\ \rhd Kc$ (*update knowledge base*) (46)

$Oop := fo\ (Oi\ ) \longrightarrow Ip\ \rhd Kc,\ Oi\ \subset O$ (*Inter-ontology Operators*) (47)

INFERENCE PRIMITIVES

$Ip := \{ip_0, ip_1, \ldots., ip_n\}$ (*Inference Primitives*) (48)

$ip := impl\ (FOL)\ |\ impl\ (FOPL)\ |\ impl\ (DL)$ (*Inference Primitive*) (49)

The *Inference Primitives* (Ip) are intended to specify algorithms for reasoning and knowledge inference. The inference algorithms are based on reasoning algorithms relying on FOL (and its extensions), First Order Probabilistic Logic (FOPL) [13] and on DL. FOPL increases the power of FOL by allowing us to assert in a natural way "likely" features of objects and concepts via a probability distribution over the possibilities that we envision. Having logics with semantics gives us a notion of deductive entailment.

V. AUTONOMIC BEHAVIOR

A cognitive robotic system is considered to be a *self-adaptive system* that changes its behavior in response to stimuli from its execution and operational environment [14]. Such behavior is considered *autonomic* and *self-adaptive*. Any long-running system is subject to uncertainty in its execution environment due to potential changes in requirements, business conditions, available technology, etc. Thus, it is important to capture and cater for uncertainty as part of the development process. Failure to do so may result in systems that are too rigid to be fit for purpose, which is of particular concern for the domains that typically make use of self-adaptive technology, e.g., cognitive robotics. We hypothesize that modeling uncertainty and developing mechanisms for managing it as part of KR&R will lead to systems that are:

- more expressive of the real world;
- fault tolerant due to fluctuations in requirements and conditions being anticipated;
- flexible and able to manage dynamic changes.

The ability to specify knowledge providing for autonomic behavior is an important factor in dealing with uncertainty. In our approach, *autonomic self-adapting behavior* is provided by *policies*, *events*, *actions*, *situations*, and *relations* between policies and situations (see Definitions 14 through 23). Ideally, policies are specified to handle specific situations, which may trigger the application of policies. A policy exhibits a behavior via actions generated in the environment or in the system itself. Specific conditions determine, which specific actions (among the actions associated with that policy – see Definition 15) shall be executed. These conditions are often generic and may differ from the situations triggering the policy. Thus, the behavior not only depends on the specific situations a policy is specified to handle, but also depends on additional conditions. Such conditions might be organized in a way allowing for synchronization of different situations on the same policy. When a policy is applied, it checks what particular conditions are met and performs the associated actions (see $map(N_\pi, A_\pi)$ – see Definition 15). The cardinality for the *policy-event relationship* is many-to-many, i.e., a situation might be associated with many policies and vice versa. Moreover, the set of *policy situations* (situations triggering a policy) is open-ended, i.e., new situations might be added or old might be removed from there by the system itself. With a set of *policy-situation relations* we may grant the system with an initial *probabilistic belief* (see Definitions 15 and 31) that certain situations require specific policies to be applied. Runtime factors may change this probabilistic belief with time, so the most likely situations a policy is associated with can be changed. For example, the successful rate of actions execution associated with a specific situation and a policy may change such a probabilistic belief and place a specific policy higher in the "list" of associated policies, which will change the behavior of the system when a specific situation is to be handled. Note that situations are associated with a state (see Definition 23) and a policy has a goal (see Definition 15), which is considered as a transition from one state to another (see Definition 18). Hence, the *policy-situation relations* and the employed *probabilistic beliefs* may help with what state to choose, based on experience.

To illustrate autonomic behavior based on this approach, let us suppose that we have a robot that carries items from point A to point B by using two possible routes - route one and route two (see Figure 2). A situation $si1$:"*robot is in point A and loaded with items*" will trigger a policy $\pi1$:"*go to point B via route one*" if the relation $r(si1, \pi1)$ has the higher probabilistic belief rate (let's assume that such a rate has been initially given to this relation because *route one* is shorter – see Figure 2.a). Any time when the robot gets into situation $si1$ it will continue applying the $\pi1$ policy until it gets into a situation $si2$:"*route one* is blocked" while applying that policy. The $si2$ situation will trigger a policy $\pi2$:"*go back to* $si1$ *and then apply policy* $\pi3$" (see Figure 2.b). Policy $\pi3$ is defined as $\pi3$:"*go to point B via route two*". The unsuccessful application of policy $\pi1$ will decrease the probabilistic belief rate of relation $r(si1, \pi1)$ and the eventual successful application of policy $\pi3$ will increase the probabilistic belief rate of relation $r(si1, \pi3)$ (see Figure 2.b). Thus, if route one continues to be blocked in the future, the relation $r(si1, \pi3)$ will get to have a higher

probabilistic belief rate than the relation $r(si1, \pi1)$ and the robot will change its behavior by choosing *route two* as a primary route (see Figure 2.c). Similarly, this situation can change in response to external stimuli.

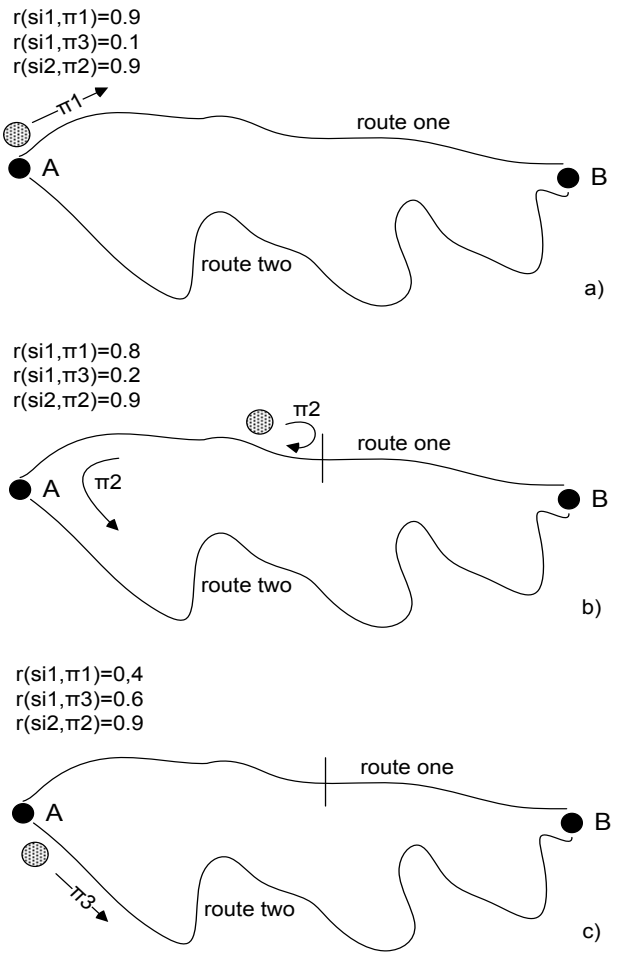

Figure 2. Self-adaptation Case Study

VI. SUMMARY AND FUTURE WORK

This paper has presented an approach to KR&R (Knowledge Representation and Reasoning) in cognitive robotic systems. The problem is tackled by a framework called KnowLang implying a multi-tier specification model that allows for integration of ontologies together with rules and Bayesian networks. The goal is efficient and comprehensive knowledge structuring and awareness based on logical and statistical reasoning. This is provided via 1) explicit representation of domain concepts and relationships; 2) explicit representation of particular and general factual knowledge, in terms of predicates, names, connectives, quantifiers and identity; and 3) handling uncertain knowledge where additive probabilities are used to represent degrees of belief. We hypothesize that modeling uncertainty and developing mechanisms for managing it as part of KR&R will help cognitive systems be flexible and able to manage dynamic changes. To support this, the KnowLang framework provides a mechanism for expressing autonomic behavior with KR&R constructs, where special probabilistic relations are used to connect situations with policies and probabilistic reasoning is used to cater for self-adaptation. Another remarkable feature is related to knowledge cleaning allowing for efficient reasoning.

Note that KnowLang is still under development as part of the ASCENS international European project [15]. Our plans for future work are mainly concerned with further and complete development of KnowLang including a toolset for formal validation. Once fully implemented, KnowLang will be used to specify knowledge representation and autonomic behavior in ASCENS case studies.

ACKNOWLEDGEMENT

This work was supported by the European Union FP7 Integrated Project Autonomic Service-Component Ensembles (ASCENS) and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero-—the Irish Software Engineering Research Centre.

REFERENCES

[1] C. Galindo, J. Fernandez-Madrigal, J. Gonzalez, and A. Saffiotti, "Robot task planning using semantic maps", Robotics and Autonomous Systems, Vol. 56 (11), 2008, pp. 955–966.

[2] O. Mozos, P. Jensfelt, H. Zender, G.-J. M. Kruijff, and W. Burgard, "An integrated system for conceptual spatial representations of indoor environments for mobile robots," Proc. of the IROS 2007 Workshop: From Sensors to Human Spatial Concepts (FS2HSC), San Diego, CA, USA, November 2007, pp. 25–32.

[3] G.-J. M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, and N. Hawes, "Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction," Symposium on Language and Robots, Aveiro, Portugal, 2007.

[4] H. Holzapfel, D. Neubig, and A. Waibel, "A dialogue approach to learning object descriptions and semantic categories," Robotics and Autonomous Systems, vol. 56 (11), 2008, pp. 1004–1013.

[5] R. Neapolitan, Learning Bayesian Networks, Prentice Hall, 2003.

[6] W. Swartout and A. Tate, "Ontologies", IEEE Intelligent Systems, Vol. 14, 1999, pp. 18-19.

[7] John F. Sowa, "Semantic networks", Encyclopedia of Artificial Intelligence (ed. S. C. Shapiro), 2nd ed., Wiley, New York, 1992.

[8] J.L. Gordon, "Creating knowledge maps by exploiting dependent relationships", Journal of Knowledge-Based Systems, Elsevier Science, Vol. 13, 2000, pp. 71-79.

[9] R. J. Brachman and H. J. Levesque, Knowledge representation and reasoning, Elsevier, San Francisco, 2004.

[10] F. Baader and W. Nutt, "Basic Description Logics", The Description Logic Handbook (ed. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, Cambridge University Press, Cambridge, UK, 2003, pp. 43–95.

[11] S. Harnad, "The symbol grounding problem", Physica D, Vol. 42, 1990, pp. 335–346.

[12] P.N. Robinson and S. Bauer, Introduction to Bio-Ontologies, CRC Press, 2011.

[13] J. Y. Halpern, "An analysis of first-order logics of probability", Artificial Intelligence, 46, 1990, pp. 311–350.

[14] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges", ACM Transactions on Autonomous and Adaptive Systems (TAAS), Vol.4(2), May 2009, pp.1-42.

[15] ASCENS – Autonomic Service-Component Ensembles, 2011, http://www.ascens-ist.eu/.