Online ISSN 1848-3380 , Print ISSN 0005-1144
ATKAFF 52(3), 256–268(2011)

Fulvio Mastrogiovanni, Antonello Scalmato, Antonio Sgorbissa, Renato Zaccaria

Robots and Intelligent Environments: Knowledge Representation and Distributed Context Assessment

UDK 004.434.046:004.82.031.6
IFAC 2.8.1

Original scientific paper

Ubiquitous Robotics is a novel paradigm aimed at addressing the coordinated behaviour of robots in environments that are intelligent *per se*. To this aim, suitable methods to enforce cooperative activities must be assessed. In this article, a formalism to encode spatio-temporal situations whose occurrences must be detected by a context-aware system is introduced. The Situation Definition Language is a tool used to specify relationships among classes of sensory data in distributed systems (such as those adhering to the Ubiquitous Robotics paradigm), without posing any assumption on how data themselves are acquired. The capabilities offered by the language are discussed with respect to a real-world scenario, where a team of mobile robots cooperates with an intelligent environment to perform service tasks. Specifically, the article focuses on the system ability to combine in a centralized representation information originating from distributed sources, either *mobile* (i.e., the robots) or *fixed* (i.e., the intelligent devices in the network).

Key words: Ubiquitous robotics, Context assessment, Distributed systems

Roboti i inteligentni prostori: prikazivanje znanja i procjena konteksta u raspodijeljenim sustavima. Sveprisutna robotika nova je paradigma namijenjena koordiniranom ponašanju robota u prostorima koji su sami po sebi inteligentni. Radi toga, nužna je procjena primjerenih metoda za provedbu kooperativnih aktivnosti. U ovome se članku predstavlja formalizam za zapisivanje prostorno-vremenskih događaja koji moraju biti detektirani u kontekstno osviještenom sustavu. Situation Definition Language je alat koji se koristi za specifikaciju odnosa među klasama senzorskih podataka u raspodijeljenim sustavima (poput onih koji se pridržavaju paradigme sveprisutne robotike), bez ikakvih pretpostavki nad načinom prikupljanja podataka. Mogućnosti koje jezik nudi analizirane su za stvarne slučajeve, gdje je tim mobilnih robota surađivao s inteligentnim prostorom radi izvođenja uslužnih zadataka. Članak se posebice fokusira na mogućnost sustava da na centraliziran način objedini informacije iz raspodijeljenih izvora, bili oni mobilni (tj. od robota) ili stacionarni (tj. od inteligentnih uređaju na mreži).

Ključne riječi: sveprisutna robotika, procjena konteksta, raspodijeljeni sustavi

1 INTRODUCTION

According to the Ubiquitous Robotics paradigm [1, 2], mobile robots are part of a fully *networked* system that is *populated* by intelligent devices (both sensors and actuators) distributed throughout the environment. Robots cooperate with intelligent devices to perform tasks that require sophisticated physical interaction capabilities, such as remote sensing and interaction with *quasi static* parts of the environment (e.g., automated doors, elevators, loading/unloading stations) or with other appliances. In these cases, highly specialized intelligent devices can provide with information exchange for the lack of physical interaction capabilities traditionally exhibited by robots: "autonomy" and "situatedness" refer thus to the whole system. Robotic architectures proposed in the literature lack in addressing how this novel paradigm affects knowledge representation and context awareness. A promising approach is to describe contexts using high-level symbolic frameworks [3]. In Ubiquitous Robotics the need arises for a theoretical framework able to deal with the following issues: (i) design and implement context assessment strategies for robots and intelligent devices; (ii) assess the trade-off between situation-dependent knowledge representation and intelligent behaviour.

This article introduces the Situation Definition Language (henceforth referred to as $\mathcal{SDL}$) aimed at making the process of representing contextual knowledge easier. $\mathcal{SDL}$ allows users to express models for context-awareness that are organized in a hierarchical fashion and encoded through ontologies [4]. Although different formalisms have been proposed to this end, such as the Situation Calculus [5, 6], the Linear Temporal Logic [7] and

the Allen's Interval Algebra [8,9], the use of temporal constraints in this work is rather different. The aim of the envisaged context-assessment system is not only to specify highly expressive formulas and to reason upon their consequences, but also to ground sensory data (originating from different sources) with respect to predefined event templates.

The paper is organized as follows. Section 2 discusses situation languages for context models, as well as relevant Robotics frameworks. Section 3 describes the key components of the proposed model. Section 4 discusses the related context assessment strategy. Section 5 describes $\mathcal{SDL}$ syntax and semantics. Implementation details, experiments and results are discussed in Section 6. Conclusions follow.

2 RELATED WORK

Situation languages. Context-aware systems are subject to three key requirements: *effectiveness* in assessing sensory information; *reusability* for generating composite representation structures; *expressiveness* in representing relational and temporal event patterns. The notion of "context" has been defined as *any information that can be used to characterize the situation of an entity* [10–15]. Four approaches to implement this idea are particularly relevant for this discussion.

The "Context Toolkit" [10] is based on context *widget*, *interpreter* and *aggregator*. A widget corresponds to a real-world artefact acquiring contextual knowledge. Widgets are not application specific. Instead, they can be considered as available services. Interpreters correspond to modules reasoning upon contextual knowledge. They can process several contexts and produce new contextual information. Finally, aggregators are modules used to combine heterogeneous knowledge.

The "Context Broker Architecture" (CoBrA in short) has been designed to deal with three requirements of distributed systems [16]: (i) contextual knowledge must be *shared*; (ii) intelligent agents must be provided with a well-defined information semantics; (iii) interoperability among heterogeneous data must be enforced. Within CoBrA, all these aspects are realized by a *broker*, a software entity maintaining a shared language to model situations, acquiring data from distributed sources and establishing data delivery policies.

The "Aspect-Scale-Context" model and the associated Context Ontology Language (CoOL in short) [17] originate from the *shared understanding* issue of distributed systems: each "actor" must be provided with the same declarative data semantics. CoOL is structured to enforce interoperability: situations are sets of "aspects", each one represented using one or more "scales", relating heterogeneous contextual knowledge. The model has two major constraints: (i) the availability of pairwise mappings between scales; (ii) the existence of a metric to establish the mapping itself.

In [18], a hierarchical temporal situation language has been introduced. A context is a dynamic process where the relationships between semantics and interpretation depend on their own *history*. Therefore, a context is a sequence of "context states" depending on "context features", which are individual and atomic language elements. Unfortunately, context features must be carefully classified and evaluated in order to determine which features are important for a particular context.

Frameworks for Ubiquitous Robotics. Although many Ubiquitous Robotics approaches have been presented in the literature [4], if we focus on systems specifically taking context-awareness into account, this number reduces to a few examples. Within the *Ambience* project [19], robots and smart environments manage human-robot interaction tasks. A principled context model is not explicitly represented within the system, which lacks in reusability and expressiveness. The *Ubibot* paradigm [1] is based on three main constituents: *Mobots* (i.e., mobile robots), *Embots* (i.e., embedded robots: agents in charge of data collection) and *Sobots* (i.e., software robots: agents implementing cognitive algorithms). A context model requires a tight integration between Mobots, Embots and Sobots. However, just few contextual information is used by the system, which lacks in effectiveness and reusability. As part of the *PEIS Ecology* framework, and grounded with respect to *cooperative anchoring*, the work in [20] is one of the first attempts to generate distributed context structures. Unfortunately, issues related to reusability and expressiveness are not sufficiently discussed. Finally, with a specific emphasis on human-robot interaction, the work in [21] is aimed at investigating formal models of human behaviour in specific situations, integrating information obtained from distributed sources. Although this is an effective solution to drive robot behaviours, an explicit context model is missing.

Ontologies and logic approaches. Approaches based on *ontologies* are expected to provide a superior expressiveness in describing concepts and relationships [13, 22]. The work in [23] adopts activity models to characterize a situation, which refer to many entities: a situation is originated from contexts affecting different individuals. Crowley and colleagues [24] extend this principle by considering contexts as networks of building blocks: different entity properties are obtained by considering relationships among different contexts. The network model abstracts from the source of a particular information, therefore allowing to consider distributed sources. Although these systems offer

a basic framework for supporting reasoning, they are characterized by two major drawbacks: (i) they tend to produce highly specialized terminologies, and (ii) context models can not be easily aggregated to build composite representations.

Logic approaches manage contexts integrating sensory data with axioms and rules. Two inspiring principles deserve special attention [25]: (i) *locality*: reasoning must occur within a domain defined by a context; (ii) *cross-domains bridging*: relationships can occur between reasoning activities belonging to different contexts. Logic-based architectures have been proposed for smart environments [26]. However, they show a number of limitations with respect to expressiveness, since they lack in representing compact models.

3 A CONTEXT MODEL FOR UBIQUITOUS ROBOTS

The proposed model takes inspiration from *computational functionalism*, a *theory of mind* considering mental states as functional relationships [27]. Mental states are defined using mental and *non*mental states, such as sensory information. Functional states are defined in abstract. Their effective realization is independent from the underlying *embodiment*. Functional states can be originated from distributed and heterogeneous sources of information, possibly remotely located and only loosely coupled. The main hypothesis of this work is to consider contexts as special mental states. A system based on functionalist considerations is characterized by very desirable properties, such as the possibility of being easily implemented, the support for distributed cognition and the lack of assumptions about physical properties of information sources. The proposed context model adheres to a number of design principles, which are introduced as follows.

Proposition 1 Contexts are functional states emerging from the interaction between sensory data and the functional structures used to assess information.

Proposition 2 Functional structures and the corresponding relationships are defined in abstract and then grounded with respect to the particular scenario.

Proposition 3 Physically distributed as well as abstract sources of information are considered: a context can be shared among many distributed entities, each one possibly contributing to its assessment.

Proposition 4 Contexts are hierarchically structured in order to maintain knowledge and allow for reasoning at the proper level of abstraction.

On the basis of these principles, the proposed model is described as follows (Figure 1 on the bottom).

Proposition 5 Symbols σ_k, $k = 1, \ldots, |\sigma_k|$ are iteratively defined by numerical or ordered values provided by distributed sources using other symbols σ_k, whose semantics is grounded with respect to an ontology Σ.

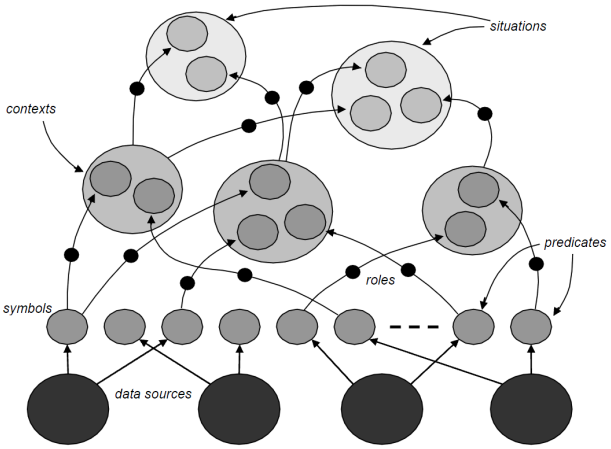

Fig. 1. A graphical sketch of the proposed context assessment model.

Proposition 6 Each symbol in Σ is defined by a vector $\mathbf{r}$ of n functional roles r_i, each one possibly filled by a vector ϕ_i of one or more symbol fillers ϕ_i^j, according to the role definition: $\sigma_k = \wedge\{r_i\} \doteq r_1 \wedge r_2 \wedge \ldots \wedge r_n$. Roles implement functional relationships among symbols.

Proposition 7 A context $\Sigma \models \sigma_k(\hat{\sigma})$ is a grounded symbol originating from the closure of the functional relationship between roles and fillers (which is referred to in the following as a *variable assignment* α), according to the definition of $\sigma_k : \hat{\sigma} \doteq \wedge\{r_i \bigotimes \phi_i\} = r_1 \otimes \phi_1 \wedge r_2 \otimes \phi_2 \wedge \ldots \wedge r_n \otimes \phi_n$.

From Propositions 6 and 7 it can be seen how a grounded symbol $\hat{\sigma}$ is recursively structured: each filler ϕ_i^j of a certain role r_i is a grounded symbol. Since each *symbol* ultimately represents real-world *events* the system can observe, Propositions 5–7 define a *computational process* extracting base predicate symbols from sensory data and to semantically assess information from symbol combinations.

Proposition 8 Given the symbol $\hat{\sigma}$ grounded with respect to a context σ described in an ontology Σ, and given a variable assignment α under a specific interpretation $\mathcal{I}$, then the proposed context model can be seen as the satisfiability procedure $(\Sigma, \mathcal{I}, \alpha) \models \hat{\sigma}$.

This model requires to identify an interpretation $\mathcal{I} \doteq (\Sigma, \cdot^{\mathcal{I}})$ for contexts in Σ.

Proposition 9 A context model Σ^{cm} is defined such that $\Sigma^{cm} \doteq \{\mathcal{P}, \mathcal{C}, \mathcal{S}\} \subset \Sigma$, where cm stands for "context model", $\mathcal{P}$ is a set of *predicates* P_p, $p = 1, \ldots, |\mathcal{P}|$, used to build more complex structures, $\mathcal{C}$ is a set of *contexts* C_c, $c = 1, \ldots, |\mathcal{C}|$ and, finally, $\mathcal{S}$ a set of *situations* S_s, $s = 1, \ldots, |\mathcal{S}|$.

In the following paragraphs, we adopt the syntax of Description Logic formalisms and the related nomenclature.

Proposition 10 *Instance checking*. A binary operator is introduced, henceforth called *instance checking*, such that $\{true|false\} \leftarrow \Sigma \models_? \sigma_l^{cm}(\hat{\sigma}_j^{cm})$, where Σ is an ontology, σ_l^{cm} is a non grounded symbol (i.e., a concept) and $\hat{\sigma}_j^{cm}$ is a grounded symbol (i.e., an individual) of the context model Σ^{cm}. Given a variable assignment α, the operator returns true if $\hat{\sigma}_j^{cm}$ is an *instance* of (i.e., it can be classified as) σ_l^{cm}, or false otherwise.

Although the model can manage hierarchical contexts, in practice only 3-layer structures are needed, namely *predicate*, representing information about sensory data, *context*, aggregating predicates dealing with the same entity and *situation*, considering different contexts as a whole. Common logic operators are provided by the underlying ontology. However, to deal with *temporal* event patterns, the context Σ^{cm} is augmented with a number of temporal operators.

Proposition 11 *Introduction of the temporal dimension*. Given a time instant τ, elements of the context model σ_l^{cm}, where $l = 1, \ldots, |\mathcal{P}| + |\mathcal{C}| + |\mathcal{S}|$, are satisfied in τ (and we write $\sigma_{l,\tau}^{cm}$) if there is an interpretation $\mathcal{I}$ and a variable assignment α_τ such that $(\Sigma, \mathcal{I}, \alpha_\tau) \models \mathcal{P}(\hat{\sigma}_{l,\tau}^{cm}) \cup \mathcal{C}(\hat{\sigma}_{l,\tau}^{cm}) \cup \mathcal{S}(\hat{\sigma}_{l,\tau}^{cm})$.

The use of temporal information is coupled in Σ with a number of operations on symbols. Of particular relevance are the *derivative* and the *length* of symbols, as well as the *timeline*. The derivative δ is an operator returning true when the truth value of the operated grounded symbol changes in two subsequent time instants. The length λ returns true if the truth values (either true or false) of the operated grounded symbols are in a particular temporal relationship, such as $<, \leq, =, \geq, >$. Finally, the timeline $\prec$ expresses precedence relationships between two grounded symbols, returning true if the first symbol actually precedes the second.

Different techniques can be adopted to implement temporal relationships. To this aim, symbols in the model use two specific roles, namely *startsAt* and *endsAt*, which specify the temporal interval when the corresponding $\hat{\sigma}_l^{cm}$ definition has been last satisfied by a variable assignment. It is possible to encode advanced temporal relationships by properly reasoning on actual *startsAt* and *endsAt* values. However, the focus of this work is on strict precedence relationships, i.e., $\hat{\sigma}_1 \prec \hat{\sigma}_2$ holds iff $\hat{\sigma}_1$ ends before $\hat{\sigma}_2$ starts.

4 CONTEXT RECOGNITION AS A COMPUTATIONAL PROCESS

Theorem 1 *Given a finite-length sequence of grounded symbols* $\hat{\sigma}_1, \ldots, \hat{\sigma}_K$ *such that* $\Sigma^{cm} \models \mathcal{P}(\hat{\sigma}_k)$, $\Sigma^{cm} \models \mathcal{C}(\hat{\sigma}_k)$ *or* $\Sigma^{cm} \models \mathcal{S}(\hat{\sigma}_k)$, $\forall k \in 1, \ldots, K$, *it is possible to uniquely determine if such a sequence is composed of actual symbols, to retrieve the number of involved symbols, and to identify symbols.*

Proof: Each grounded symbol $\hat{\sigma}_k$ can be associated with a finite-length tree $\mathcal{T}_k$, which represents the process of functional relationships by recursively inspecting filler symbols until reaching elements in $\mathcal{P}$, i.e., *leaves*. The process of building the associated tree $\mathcal{T}_k$ can be achieved using the following steps: (i) the tree root is labelled with $\hat{\sigma}_k$; (ii) if a node is such that $\Sigma^{cm} \models \mathcal{P}(\hat{\sigma}_k)$, it is a leaf; (iii) if a node is such that $\Sigma^{cm} \models \sigma_k(\hat{\sigma}_k) = \wedge\{r_i\}$, then it has n child nodes, which are labelled, respectively, using $\phi_i^1, \ldots, \phi_i^m$. If it is possible to build $\mathcal{T}_k$ for each $\hat{\sigma_k}$, then it is possible to inspect the sequence $\hat{\sigma}_1, \ldots, \hat{\sigma}_K$ by exhaustively parsing all the trees. ■

In virtue of Theorem 1, we can draw a parallelism between finite-length trees and elements in the context model. Specifically, building the finite-length tree $\mathcal{T}_k$ for a given $\hat{\sigma}_k$ is equivalent to fill the corresponding roles r_i with symbolic structures belonging to Σ^{cm}. Parsing the tree is equivalent to traverse functional relationships among functional states. On the basis of this result, context awareness in Ubiquitous Robotics scenarios is reduced to many *satisfiability* procedures carried out over instances of Σ^{cm}.

Proposition 12 Given a finite-length tree $\mathcal{T}_k$, an interpretation $\mathcal{I}$, a variable assignment α_τ, then satisfied symbols $\hat{\sigma}_k$ are such that $(\mathcal{T}_k, \mathcal{I}, \alpha_\tau) \vdash_k \Sigma_\tau^{cm}(\hat{\sigma}_k)$, i.e., $\hat{\sigma}_k$ can be derived with the derivation k.

Strictly speaking, symbols in $\mathcal{C}$ or $\mathcal{S}$ are encoded as finite-length trees $\mathcal{T}_k$ which – on their turn – rely on definitions of concepts in Σ^{cm}. Context awareness is realized by aggregating instances of symbols in $\mathcal{P}$ in order to satisfy symbolic structures C_c and S_s represented within Σ (see Proposition 9 onwards). The aggregation assumes the form of the *history* of the n most recent instances of symbols in $\mathcal{P}$ P_p^i, $i = 1, \ldots, n$, which are stored within Σ in a *first-in-first-out* approach (see Figure 1 on the bottom).

Every time instant τ, when updated sensory information is available, a classification process is carried out over instances of symbols in $\mathcal{P}$, thus – possibly – modifying their truth value. In particular, the overall history description $\mathcal{D}_p$ is considered; $\mathcal{D}_p$ is obtained by joining the description of each P_p^i:

$$\mathcal{D}_p \doteq \mathcal{D}(P_p^1 \sqcap \ldots \sqcap P_p^n) \quad p = 1, \ldots, |\mathcal{P}|. \qquad (1)$$

$\mathcal{D}_p$ represents the system state with respect to both the current situation and the most recent past events. For Theorem 1, the system can infer what symbols C_c are satisfied. This is accomplished by checking for instance relationship

between $\mathcal{C}$ and $\mathcal{D}_p$. Specifically, $\mathcal{C}_v(\tau)$ is the collection of C_c symbols classifying $\mathcal{D}_p$:

$$\mathcal{C}_v(\tau) = \{C_c : \Sigma^{cm} \models_? C_c(\mathcal{D}_p)\}. \quad (2)$$

Therefore, all the satisfied symbols in $\mathcal{C}$, namely $\mathtt{c}_c \subseteq \mathcal{C}_v(\tau)$ are occurring in τ (Figure 1 on the mid). This mechanism is easily iterated for symbols in $\mathcal{S}$. Since $\mathcal{C}_v(\tau)$ varies at each τ, the history description $\mathcal{D}_c$ is considered, obtained by superimposing the description of each $C_c^j \subseteq \mathcal{C}_v(\tau)$:

$$\mathcal{D}_c \doteq \mathcal{D}(C_c^1 \sqcap \ldots \sqcap C_c^{|\mathcal{C}_v(\tau)|}) \quad c = 1, \ldots, |\mathcal{C}|. \quad (3)$$

Analogously to satisfied symbols in $\mathcal{C}$, using $\mathcal{D}_c$ the system can infer what S_s are satisfied in τ (see Figure 1 on top). Again, this is managed by instance checking between $\mathcal{S}$ and $\mathcal{D}_c$. Current situations are stored in $\mathcal{S}_v(\tau)$, which is the collection of instances S_s subsuming $\mathcal{D}_c$, where

$$\mathcal{S}_v(\tau) = \{\mathtt{s}_s : \Sigma^{cm} \models_? S_s(\mathcal{D}_c)\}. \quad (4)$$

Remark 1 By explicitly taking time into account, temporal instances of *Situation* can be handled as relational symbols: the only difference is that satisfiability periods may not overlap. For each temporal *Situation*, the number of constituent *Context* elements to be considered over time is limited. The reason is twofold: (i) since efficiency in logic inferences depends on description complexity, an increased number of *Context* elements in defining a *Situation* can lead to dramatic response times; (ii) correlation among different instances of *Context* fades away with time.

Remark 2 The state at time instant τ can not completely represent the whole evolution of the system, because temporal *Situation* symbols require an explicit representation of previously satisfied symbols. In other words, the context model does not comply with the *First Order Markov Assumption*: the approach implements a process of "symbolic smoothing" over a temporal window that is characterized by the history length n. As long as n increases, the rate of successful recognition of *Situation* instances increases, since relevant *Predicate* symbols are still represented within the system, thus contributing to build the necessary *Context* elements. However, computational load exponentially increases with n , thus requiring a trade-off between history length and real-time recognition requirements.

5 A FORMAL LANGUAGE FOR SITUATION SPECIFICATION

5.1 Syntax and Definitions

A formal language is introduced that builds upon the knowledge representation model described in the previous Section. $\mathcal{SDL}$ is used to build abstract sentences correlating different base symbols, which correspond to sensory data. As a consequence, the language poses constraints on the construction of symbolic structures in the ontology, which correspond to named *Contexts* and *Situations*.

Proposition 13 $\mathcal{SDL}$ is defined as $\mathcal{SDL} = L(\mathcal{SDL}_{st})$, where L is a generative process that operates on a grammar $\mathcal{SDL}_{st}$. As usual, $\mathcal{SDL}_{st}$ is formally defined as a 2-ple $\mathcal{SDL}_{st} =< A, R >$, where A is a finite *alphabet* and R is a binary relationship over A^* (the set of finite, possibly empty, strings over A) such that $R \subseteq A^* \times A^*$.

The grammar specifies how to build well-formed sentences belonging to $\mathcal{SDL}$ starting from an initial sentence in A^* and iteratively applying *rewrite* rules belonging to R until the intended sentence is *derived* in n steps. In particular, sentences s_i in $\mathcal{SDL}$ are elements of the set A^*, whereas rewrite rules $r_a \rightarrow r_c$ are defined as binary elements $(r_a, r_c) \in R$, where r_a and $r_c \in A^*$ are called, respectively, the *antecedent* and the *consequent* of the rule. The generative process L must be precisely defined.

Proposition 14 Given a_1 and $a_2 \in A^*$, a one-step rewrite relation $\Rightarrow_R$ over an alphabet A^* can be defined such that, given x, y, r_a and $r_c \in A^*$, then $a_1 \Rightarrow_R a_2$ holds if and only if $a_1 = xr_ay$, $a_2 = xr_cy$ and $(r_a, r_c) \in R$.

Proposition 15 An n-derivation in $\mathcal{SDL}$ is defined as a finite sequence of sentences $s_0, \ldots, s_n$ that are produced starting from an initial sentence s_0 belonging to A^* and rewriting it by means of rewrite rules in R.

Proposition 16 The generative process L can be defined such that $\mathcal{SDL} = \{s_i \in A^* | s_0 \Rightarrow_R s_i\}$, from which it can be argued that $\mathcal{SDL}$ is a Semi-Thue system.

Sentences are not atomic elements of $\mathcal{SDL}$. On the contrary, they can be decomposed into many parts, which are explained in the following paragraphs.

Proposition 17 The alphabet A is defined as a set of symbol classes such that $A = \{s_0, s, \emptyset, l, f, \oplus, t, v, c\}$, where: s_0, s and $\emptyset$ are, respectively, the initial sentence, the generic sentence and the null sentence; l represents *labels*, meant at providing each sentence with a name; f represents *formulas*, i.e., building blocks upon which sentences can be built; $\oplus$ represents both unary and binary connecting *operators*; t represents *terms*, i.e., everything else apart from sentences, and more precisely language *variables* v and *constants* c.

Proposition 18 A sentence s is an element of A^* in the form $l \equiv l(f);$, where l represents suitable labels and f is a formula.

Proposition 19 A formula f is an element of A^* that can be a label, a term, a juxtaposition or a composition of subformulas by means of connecting operators.

Proposition 20 A connecting symbol $\emptyset$ is an element of A that contributes to formulas by composing one or more

subformulas using, respectively, unary symbols $\oplus_u$ and binary symbols $\oplus_b$.

Proposition 21 The set of rewrite rules R is defined as follows: $s_0 \rightarrow s \mid ss_0 \mid \emptyset$; $s \rightarrow l \equiv l(f)$; $f \rightarrow l \mid t \mid \oplus_u f \mid f \oplus_b f$; $l \rightarrow$ *string*; $t \rightarrow v \mid c$; $v \rightarrow$ *string*; $c \rightarrow$ *string*; $\oplus_u \rightarrow \neg \mid \equiv \mid \delta \mid ()$; $\oplus_b \rightarrow \sqcap \mid \sqcup \mid \lambda_\omega \mid \prec \mid$.

The syntax of $\mathcal{SDL}$ is completely specified on the basis of the alphabet A and the rewrite rules in R. However, what is unspecified is how to ground the language with respect to the underlying ontology Σ, and in particular to descriptions $\mathcal{D}_k$. In particular, it is necessary to map language elements to symbol classes within the ontology.

Proposition 22 The rule $v \rightarrow$ *string* maps a set of variables v_k, $k = 1, \ldots, |v_k|$, to corresponding symbols within Σ that are related to *Basic Types* $\mathcal{B}$ such that $\Sigma \models \mathcal{B}(\mathcal{D}_k)$.

Variables v_k correspond to descriptions $\mathcal{D}_k$ that refer to basic types. In particular, *string* refers to the name of the related σ_k.

Proposition 23 The rule $c \rightarrow$ *string* maps a set of constants c_l, $l = 1, \ldots, |c_l|$, to symbols within the ontology that are related to instances of *Basic Types* $\mathcal{B}$ such that $\Sigma \models \mathcal{B}(\hat{\mathcal{D}}_l)$.

Differently from variables, constants c_l correspond to descriptions $\hat{\mathcal{D}}_l$ that refer to instances of basic types. In this case, *string* refers to the name of the related $\hat{\sigma}_l$.

Proposition 24 The rule $l \rightarrow$ *string* maps labels to a set of named non-grounded σ_k or grounded $\hat{\sigma}_l$ symbols within the ontology Σ, each one related to either *Predicates*, *Contexts* or *Situations*.

All the symbols above contribute to the definition of formulas.

Proposition 25 The rule $f \rightarrow l \mid t \mid \oplus_u f \mid f \oplus_b f$ maps formulas f_j, $j = 1, \ldots, |f_j|$ to corresponding either non-grounded or grounded descriptions within Σ, which are related to either *Predicates*, *Contexts* or *Situations*.

Proposition 26 The rule $s \rightarrow l \equiv l(f)$; maps a set of sentences $p_j \in A^*$, $j = 1, \ldots, |p_j|$ to corresponding *Predicate* symbols within Σ.

Sentences p_j correspond to descriptions related to basic types in conjunctive normal form that describe "facts" about an entity.

Proposition 27 The rule $s \rightarrow l \equiv l(f)$; maps a set of sentences $c_j \in A^*$, $j = 1, \ldots, |c_j|$ to corresponding *Context* symbols within Σ.

Sentences c_j correspond to descriptions related to predicates in conjunctive normal form that refer to the same entity.

Proposition 28 The rule $s \rightarrow l \equiv l(f)$; maps a set of sentences $s_j \in A^*$, $j = 1, \ldots, |s_j|$, to corresponding *Situation* symbols within Σ.

Sentences s_j correspond to descriptions related to predicates in normal conjunctive form that refer to different entities.

The introduced mapping defines a translation mechanism to map $\mathcal{SDL}$ sentences in structures that are represented within an ontology Σ. The set of connecting operators is not described in details, as actual translation mechanisms are ontology-dependent (since they rely on the underlying operators) and – as such – out of the scope of this paper[1].

5.2 Semantics

Every time instant τ, grounded formulas originate from a variable assignment α_τ according to a specific interpretation $\mathcal{I}$. Sentences in $\mathcal{SDL}$ assert which symbols must be grounded in τ for the represented sentence to be satisfied, given sensory information acquired from some instant in the past up to the present time. How the grounding process is performed depends on the constituent formulas. Ultimately, this requires to identify a proper interpretation $I \doteq (\Sigma, \cdot^I)$ for symbols belonging to the underlying ontology Σ, thereby imposing a semantics to the corresponding *formulas*.

In order to ground *Basic Types* $\mathcal{B}$, it is necessary to ground symbols with respect to actual percepts corresponding to sensory data. In this work, we refer to the mapping discussed in [28], and we focus on how semantics is propagated to abstract structures by means of sentences. In particular, the effects of the variable assignment α over $\mathcal{SDL}$ formulas must be defined.

Proposition 29 The *substitution* of an actual numerical, ordered or non-ordered value α_i to all the occurrences of a variable v in a term t, that is referred to as $t[v/\alpha_i]$, is recursively defined as: (i) if t is a variable s.t. $v \neq t$, then $t[v/\alpha_i] \rightarrow t$; (ii) if t is a variable s.t. $v = t$, then $t[v/\alpha_i] \rightarrow \alpha_i$; (iii) if t is a constant, then $t[v/\alpha_i] \rightarrow t$.

Proposition 30 The *substitution* of a term t to all the occurrences of a variable v in a formula f, that is referred to as $f[v/t]$, is recursively defined as: (i) if $f_{\oplus_u}$ is a formula in the form $\oplus_u f$, then $f_{\oplus_b}[v/t] \rightarrow \oplus_u f[v/t]$; (ii) if $f_{\oplus_b}$ is a formula in the form $f_1 \oplus_b f_2$, then $f_{\oplus_b}[v/t] \rightarrow f_1[v/t] \oplus_b f_2[v/t]$.

As soon as new or updated information is available at the time instant τ, a new variable assignment α_τ is defined. Accordingly, variables are updated. As a consequence, corresponding formulas are given proper truth values, thereby satisfying more complex sentences. Furthermore, the meaning associated with connecting operators

[1] The interested reader could guess how to implement connecting operators by inspecting their semantics in the next Section. However, in Description Logics based ontologies, the mapping can be easily realized using some sort of extra logic "trick".

must be clearly assessed. In order to recursively compose formulas, unary and binary symbols can be better classified in *relational* and *temporal* operators, that are characterized by an intuitive correspondence with common logic and mathematical operators. Specifically, common logic operators have been added.

Proposition 31 *Assignment*. The unary operator $\doteq \in \oplus_u$ produces a formula $f_{\doteq} \leftarrow \doteq f$ that is the copy of the subformula f. Given an interpretation $\mathcal{I}$ and a variable assignment α, the grounded formula $\doteq \hat{f}$ is satisfied if and only if the logical expression $\hat{f}$ is satisfied.

Proposition 32 *Parentheses*. The unary operator $() \in \oplus_u$ produces a formula $f_{()} \leftarrow (f)$ that specifies the level of precedence in the parsing process for the subformula f. Given an interpretation $\mathcal{I}$ and a variable assignment α, the grounded formula $(\hat{f})$ is satisfied if and only if the logical expression $\hat{f}$ is satisfied.

Proposition 33 *Conjunction*. The binary operators $\{\sqcap, ,\} \in \oplus_b$ produce formulas $f_{\{\sqcap,,\}} \leftarrow f_1\{\sqcap, ,\}f_2$ that are the juxtaposition of two subformulas f_1 and f_2. Given an interpretation $\mathcal{I}$ and a variable assignment α, the grounded formulas $\hat{f}_1\{\sqcap, ,\}\hat{f}_2$ are satisfied if and only if the logical expression $\hat{f}_1$ *and* $\hat{f}_1$ is satisfied.

It is worth noting that the correspondence of meaning between the two latter connecting operators originates from the mapping process of formulas to structures within the ontology. However, this is permitted to guarantee compatibility with commonly used logic formalisms.

Proposition 34 *Disjunction*. The binary operator $\sqcup \in \oplus_b$ produces a formula $f_{\sqcup} \leftarrow f_1 \sqcup f_2$ that is the disjunction of two subformulas f_1 and f_2. Given an interpretation $\mathcal{I}$ and a variable assignment α, the grounded formula $\hat{f}_1 \sqcup \hat{f}_2$ is satisfied if and only if the logical expression $\hat{f}_1$ *or* $\hat{f}_2$ is satisfied.

Proposition 35 *Negation*. The unary operator $\neg \in \oplus_u$ produces a formula $f_{\neg} \leftarrow \neg f$ that is the negation of the subformula f. Given an interpretation $\mathcal{I}$ and a variable assignment α, the grounded formula $\neg\hat{f}$ is satisfied if and only if the logical expression *not* $\hat{f}$ is satisfied.

Temporal operators require to take several time instants τ into account. As a consequence, they contribute to formulas that are satisfied by a sequence of several variable assignments α_τ. Three operators have been defined.

Proposition 36 *Derivative*. The unary operator $\delta \in \oplus_u$ produces a formula $f_\delta \leftarrow \delta(f)$ that is the "derivative" of the subformula f. Given two time instants τ and $\tau - 1$, an interpretation $\mathcal{I}$ and two assignments α_τ and $\alpha_{\tau-1}$, the grounded formula $\delta_\tau(\hat{f}_\tau)$ is satisfied if and only if $\hat{f}_\tau = \neg\hat{f}_{\tau-1}$.

For simplicity, two operators, namely *positive derivative* $\delta^{t\rightarrow f}$ and *negative derivative* $\delta^{f\rightarrow t}$ are used in place of the general operator δ, in order to differentiate the case in which the argument of the derivative changes from *satisfied* to *not satisfied* or vice versa.

Proposition 37 *Length*. The family of binary operators $\lambda_\omega \in \oplus_b$, $\omega \in \{<, \leq, =, \geq, >\}$, produces a formula $f_\lambda \leftarrow \lambda_\omega(f_1, f_2)$ that compares the temporal duration of two subformulas f_1 and f_2. Given three time instants τ, tau_1 and τ_2, an interpretation $\mathcal{I}$ and three variable assignments α, α_{τ_1} and α_{τ_2}; given that $\tau_1 < \tau$ is the most recent time instant such that $\delta(f_1)$ holds, and $\tau_2 < \tau$ is the most recent time instant such that $\delta(f_2)$ holds; then the grounded formula $\lambda_{\omega,\tau}(\hat{f}_{1,\tau}, \hat{f}_{2,\tau})$ is satisfied if and only if $(\tau - \tau_1)\omega(\tau - \tau_2)$ is satisfied, whereas it is not satisfied otherwise.

In most cases, one of the constituent formulas may simply correspond to actual time intervals (e.g., "20 minutes" or "8 hours"), and the operator is currently used to check whether a formula lasts less, equally, or more than the specified interval.

Proposition 38 *Precedence*. The binary operator $\prec \in \oplus_b$ produces a formula $f_\prec \leftarrow \prec (f_1, f_2)$ that expresses the temporal precedence relationship between two subformulas f_1 and f_2. Given two instants τ_1 and τ_2, an interpretation $\mathcal{I}$ and two variable assignments α_{τ_1} and α_{τ_2}; given that $\delta^{t\rightarrow f}(\hat{f}_{1,\tau_1})$ and $\delta^{f\rightarrow t}(\hat{f}_{2,\tau_2})$ hold; then the grounded formula $\hat{f}_{1,\tau_1} \prec \hat{f}_{2,\tau_2}$ is satisfied if and only if $\tau_1 < \tau_2$ holds (i.e., if the interval in which f_1 is satisfied strictly precedes the interval in which f_2 is satisfied), and not satisfied otherwise.

6 IMPLEMENTATION AND DISCUSSION

$\mathcal{SDL}$ has been used to model the behaviour of a team of service robots in a hospital environment. All the experiments have been performed using Merry Porter, a commercially available robotic platform (Figure 6). In the following paragraphs, implementation details and an overall discussion about system performance are presented.

6.1 Implementation Details

At the hardware level, Merry Porter is characterized by an unicycle kinematics with an active front steering wheel, three different systems for outdoor or indoor localization, respectively based on GPS, laser rangefinders and active beacons (referred to as DLPS system), a safety laser rangefinder for obstacle detection and two cameras for surveillance. Merry Porter has a battery working time of about 4 hours with only 30 minutes charge time.

At the software level, Merry Porter exploits a multiagent architecture that is based on the ETHNOS framework [29,30], where each module is in charge of executing well-defined tasks, for instance mission management [31], localization [32], trajectory planning [33], see Figure 2.

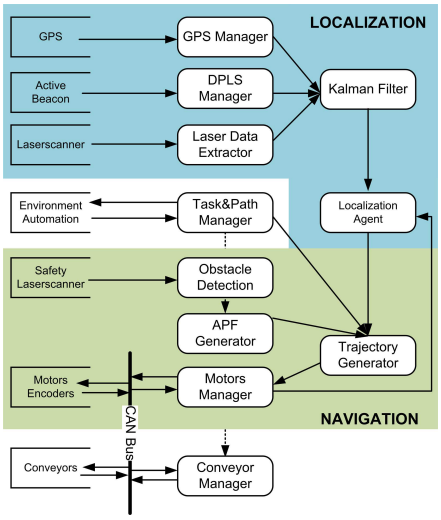

Fig. 2. The robot software architecture.

The theoretical model described in this paper has been implemented as part of the *Task Manager* module. It receives as input information originating from sensors distributed in the environment as well as on board the robots, such as position, detected obstacles or batteries level. The model has been implemented using the CLASSIC knowledge representation framework, which has been selected for its efficiency in the reasoning scheme [34]. As a matter of fact, CLASSIC guarantees a number of desirable properties like reasoning complexity in polynomial time, at the price of few limitations with respect to base construct expressiveness.

In order to obtain an efficient structure for the ontology, specifically in a reasoning perspective, a number of rule-of-thumb and best practices related to the field of Ontology Engineering have been considered [35]. In particular, a first classical division in *upper* and *domain* ontologies has been considered. On the one hand, the upper ontology represents general-purpose concepts related to information (e.g., data) and physical (e.g., space, areas, places, time) domains, as well as general environment and robot related characteristics. On the other hand, the domain ontology is related to specific concepts holding in typical Ubiquitous Robotics scenarios, in particular in the form of further specification of concepts both in the information space (e.g., data types, such as those related to laser, GPS or device activity) and in the physical space (e.g., automated doors, elevators, load/unload stations and other relevant objects), as well as specific features of robots and intelligent devices.

In order to ground the $\mathcal{SDL}$ language, the model has been implemented as a specific module within the ontology. Following the subdivision in upper and domain ontology, an upper specification of $\mathcal{SDL}$ comprises those concepts representing language elements in the form of *Predicates*, *Contexts* and *Situations*. The domain specification of $\mathcal{SDL}$ represents all the predicates, contexts and situations that can be defined using the $\mathcal{SDL}$ formalism as introduced in Section 5. For instance, the *Context* concept is represented in Description Logics formalism as:

$$\begin{aligned} \mathit{Context} \;\doteq\; & \exists \mathit{madeOf}.\mathit{Predicate} \sqcap \geq_1 \mathit{madeOf} \sqcap \\ & \exists \mathit{startsAt}.\mathit{Integer} \sqcap =_1 \mathit{startsAt} \sqcap \\ & \exists \mathit{endsAt}.\mathit{Integer} \sqcap =_1 \mathit{endsAt}, \end{aligned}$$

where *startsAt* and *endsAt* are – respectively – the start and end time instants associated with a given truth value for the context (time is modelled using integers). As a major consequence of organizing predicates, contexts and situations in a hierarchical fashion, a bottom-up approach to the continuous update of the ontology is achieved: sensory data affects instances of basic of domain elements, which are used to determine truth values associated with domain predicates, to be used by higher-level concepts such as contexts and situations to define complex abstractions.

All the $\mathcal{SDL}$ operators have been implemented as specific concepts in the ontology subsumed by *SDLOperator*. For instance, given the *Context* concept, the *Precedence* operator is defined as:

$$\begin{aligned} \mathit{Precedence} \;\doteq\; & \mathit{SDLOperator} \sqcap \\ & \exists \mathit{arg1}.\mathit{Context} \sqcap =_1 \mathit{arg1} \sqcap \\ & \exists \mathit{arg2}.\mathit{Context} \sqcap =_1 \mathit{arg2} \sqcap \\ & (\mathit{arg1.endsAt} < \mathit{arg2.startsAt}), \end{aligned}$$

where the test in the last row of the definition is used to characterize precedence between occurrences of contexts in *arg1* and *arg2*, respectively. Similar considerations hold for other concepts and operators.

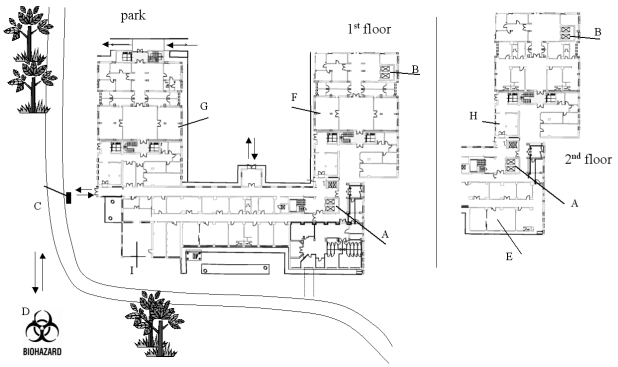

Fig. 3. Map of the robot operating environment.

The approach described in this paper is currently under validation at the Polyclinic of Modena, a hospital located

in Italy. Robots are required to transport – either according to a time schedule or on demand – biologic waste and other material between areas located at different floors of one of the buildings of the Polyclinic complex (Figure 3). In the following paragraphs, two test cases is reported, discussing different examples of use.

6.2 Example 1: Mission Feasibility

Rationale. Mobile robots need to know both *in advance* and at run-time the availability of elevators (e.g., *A* and *B* in Figure 3), the crowding of areas to traverse (e.g., *F*, *H* and other areas $A_1, \ldots, A_n$), the type of material being delivered (in order to avoid possible contacts with food provisions and people) and the charge status of the *BatteryPack*.

In our set-up, this data is provided by proper information sources: an *Environment Automation* module running on the network device associated with elevators *A* and *B* continuously provides information about their availability; a *Task Manager* module running on a networked workstation determines the sequence of areas $A_1, \ldots, A_n$ to be visited in order to carry out the mission [31]; algorithms (running on networked workstations) process camera images and passive infra red data (managed through sensor nodes in the network) in order to determine whether areas $A_1, \ldots, A_n$ are crowded or not, specifically using state of the art background subtraction algorithms [36], whereas distributed sensors provide information to track movements of other vehicles (through RFID tags) throughout the hospital; the integration with the hospital information system allows to access data about the type of *material* being handled and the privileged routes for other delivery activities; finally, an on board battery system module monitors battery status, whereas other agents cooperate to perform self-localization [32], obstacle avoidance [37] and navigation [33].

A mission is considered feasible if a path between the initial and goal areas exist, if on board batteries are sufficiently charged, and either the material to be delivered is safe, or – in case of risky material – the elevators are immediately available and the areas to be traversed are free of people.

Modelling. Assuming that *F* and *H* are – respectively – the initial and goal areas, and that $A_1, \ldots, A_n$ are the areas to be traversed by the robot in order to move from *F* to *H*, mission feasibility is modelled as the *FeasibleMission* situation, and in particular using the following collection of sentences (or "source code") in $\mathcal{SDL}$:

1. *Path* $\doteq$ *Connected(f, h)*;
2. *IsSafe* $\doteq$ *Safe(m)*;
3. *IsUnsafe* $\doteq \neg$*IsSafe*;
4. *NotCrowded* $\doteq \neg$(*Crowded*(a_1)$\sqcap \ldots \sqcap$*Crowded*(a_n));
5. *AreAvailable* $\doteq$ *Available(a)* $\sqcup$ *Available(b)*;
6. *AreCharged* $\doteq \delta^{f \to t}$*Charged(bp)*;
7. $\sigma_1 \doteq$ *FeasibleMission* $\equiv$ *Path* $\sqcap$ *AreCharged* $\sqcap$ (*IsSafe* $\sqcup$ (*IsUnsafe* $\sqcap$ *NotCrowded*)) $\sqcap$ *AreAvailable*;

Specifically, *FeasibleMission* is the actual situation name, *Path*, *IsSafe*, *IsUnsafe*, *NotCrowded*, *AreAvailable* and *AreCharged* are *Contexts*; *Connected*, *Safe*, *Crowded*, *Available* and *Charged* are *Predicates*; f, h, m, $a_1, \ldots, a_n$, a, b and bp are variables which names correspond to environmental elements represented in Figure 3. For a given interpretation $\mathcal{I}$ and time instant τ, an α_τ can be defined such that *Connected[f/F, g/G]*, *Safe[m/laundry]*, $\neg$*Crowded*[$a_1/A_1, \ldots, a_n/A_n$], *AreCharged[bp/BatteryPack]*, *Available[a/A]* and $\neg$*Available[b/B]*, then the corresponding grounded sentence $\hat{\sigma}_1$ satisfies σ_1.

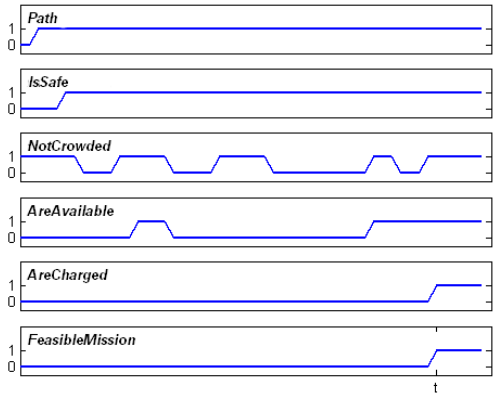

Fig. 4. A variable assignment satisfying the FeasibleMission situation.

Figure 4 shows a real truth value log that satisfies this situation (opportunely "stretched" to fit space limits): *FeasibleMission* holds at time instant t when all the constituting instances of context elements are satisfied at the same time. In this example, it can be noticed that, when the robot is requested to perform an actual mission, it is in battery charge mode: as soon as batteries are charged enough to complete the mission, then it becomes feasible. Furthermore, since the material is safe, the availability of elevators and the crowding of the areas to be visited do not really matter.

Mapping. Within the ontology, the sentence σ_1 is mapped to a concept *FeasibleMission* subsumed by *Situ-*

ation, and defined as:

$$\begin{aligned} \textit{FeasibleMission} \doteq\ & \textit{Situation} \sqcap \\ & \textit{Path} \sqcap \textit{AreCharged} \sqcap \\ & (\textit{IsSafe} \sqcup \\ & \textit{IsUnsafe} \sqcap \textit{NotCrowded}) \sqcap \\ & \textit{AreAvailable}. \end{aligned}$$

Specifically, *FeasibleMission* is made up of six *Context* concepts, which are connected through the *and/or* operators provided by the underlying ontology. Each *Context* is conventionally provided with a role *madeOf* that is filled by a proper set of *Predicate* concepts. In this example, a one-to-one mapping exists between *Contexts* and *Predicates*, e.g., *AreCharged* is associated with *Charged*:

$$\textit{AreCharged} \doteq \textit{madeOf.Charged}.$$

Predicate concepts are related to their entities and grounded instances through their *arg* roles. In this case:

$$\textit{Charged} \doteq \textit{arg1.batteryPack} \sqcap \textit{arg2.batteryLevel}.$$

Predicate concepts have two child concepts that are used to designate their truth value. For instance, *Charged* subsumes *ChargedTrue* and *ChargedFalse*, which differ on the basis of the value of *batteryLevel*. When this value is over a given threshold, the instance of *Charged* is subsumed by *ChargedTrue*, thereby contributing to *FeasibleMission* to hold. Finally, analogous mechanisms hold for the other *Context* instances as well.

Fig. 5. Merry Porter performing a delivery task.

6.3 Example 2: Loading an Object to Deliver

Rationale. When loading an object from loading stations, mobile robots can start their mission *strictly after* the overall loading phase has been carried out. This can be achieved by a tight interaction between robots and loading conveyor stations (Figure 5 on the bottom): the robot MP_1 must properly dock close to the loading conveyor station, the station must be activated, the material to be delivered must be successfully detected on board the robot, and the station must notify the successful conclusion of the overall procedure. Again, this information is provided by proper information sources: a *Localization* module running on board the robot provides both metric and topological information about the robot position [32]; a *Conveyor Manager* module running on the network device associated with loading stations S_1 and S_2 continuously provides information about their status (e.g., *on*, *off*, *active*); finally, pressure sensors located on the robot base are managed by a *Base System* agent to detect when the material is loaded. The loading procedure is considered accomplished if the robot enters the *DockingArea*, then if the station is activated, if the material is detected on board, and finally if the conveyor station stops. Obviously enough, further information can be obtained by the rear camera mounted on the robot pole (Figure 5).

Modelling. Assuming that $\texttt{MP}_1$ is the robot in charge of loading the material from S_1 in area *F*, *LoadMaterial* can be modelled using the following $\mathcal{SDL}$ code:

1. $\textit{Docking} \doteq \delta^{f\rightarrow t}\textit{IsIn(r, da)}$;
2. $\textit{StationOn} \doteq \delta^{f\rightarrow t}\textit{Active(ls)}$;
3. $\textit{StationOff} \doteq \delta^{t\rightarrow f}\textit{Active(ls)}$;
4. $\textit{Load} \doteq \delta^{f\rightarrow t}\textit{Present(m)}$;
5. $\sigma_2 \doteq \textit{LoadMaterial} \equiv \textit{Docking} \prec \textit{StationOn} \prec \textit{Load} \prec \textit{StationOff}$;

Specifically, *LoadMaterial* is the actual situation name, *Docking*, *Load*, *StationOff* and *StationOn* are *Contexts*, *IsIn*, *Active* and *Present* are *Predicates*, whereas *r*, *da*, *ls* and *m* are variables. For a given interpretation $\mathcal{I}$ and a sequence of time instants $\tau_1, \ldots, \tau_7$, if it is possible to define a number of variable assignments $\alpha_{\tau_1}, \ldots, \alpha_{\tau_7}$, where α_{τ_1} is such that $\neg$*IsIn[r/MP*$_1$*, da/F]*, α_{τ_2} such that *IsIn[r/MP*$_1$*, da/F]* (and therefore the corresponding derivative $\delta^{f\rightarrow t}$ holds), α_{τ_3} such that $\neg$*Active[ls/S*$_1$*]*, α_{τ_4} such that *Active[ls/S*$_1$*]* (therefore satisfying the corresponding derivative $\delta^{t\rightarrow f}$), α_{τ_5} such that $\neg$*Present[m/laundry]*, α_{τ_6} such that *Present[m/laundry]* (and therefore the corresponding derivative $\delta^{f\rightarrow t}$ holds), and finally an assignment α_{τ_7} such that $\neg$*Active[ls/S*$_1$*]*, then the corresponding grounded sentence $\hat{\sigma}_2$ satisfies σ_2.

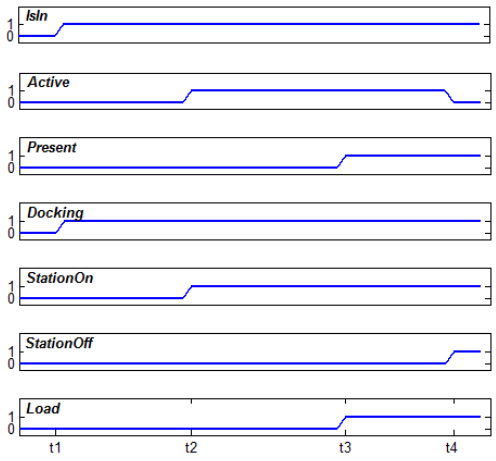

Fig. 6. Merry Porter performing a delivery task.

Figure 6 shows a truth value log satisfying *Load*: in order for the situation to be detected, four time instants, $t_1, \ldots, t_4$, are important: t_1 corresponds to the true switching instant of predicate *IsIn*: MP_1 enters area F. This triggers the context *Docking* to hold. After some time, as a consequence of a direct communication between MP_1 and S_1, the latter activates in order to load the robot base in t_2. MP_1 detects an object on its base in t_3, when predicate *Present* becomes true. In t_4, after S_1 deactivates, the procedure is complete.

Mapping. Within the ontology, sentence σ_2 corresponds to a concept *LoadMaterial* subsumed by *Situation*, and defined as:

$$\begin{aligned} \mathit{LoadMaterial} \doteq\ & \mathit{Situation} \sqcap \\ & \mathit{PrecDockingStationOn} \sqcap \\ & \mathit{PrecStationOnLoad} \sqcap \\ & \mathit{PrecLoadStationOff}. \end{aligned}$$

Specifically, *LoadMaterial* is made up of three *Precedence* concepts, each one responsible for detecting precedence between *Contexts*. For instance, *PrecDockingStationOn* can be modelled as follows:

$$\begin{aligned} \mathit{PrecDockingStationOn} \doteq\ & \mathit{Precedence} \sqcap \\ & \exists \mathit{arg1.Docking} \sqcap \\ & \exists \mathit{arg2.StationOn}. \end{aligned}$$

As it can be noticed from the previous definition, each trend similar to that of Figure 6 satisfies the formula, because $t_1 < t_2$.

7 CONCLUSIONS

In this paper, a formal language suitable to model context-aware behaviours has been presented and discussed. In order to ground actual descriptions of patterns of events to detect, an example borrowed from the Ubiquitous Robotics paradigm has been discussed, which is currently experimented in a real-world scenario. With respect to the three fundamental requirements that context models must adhere to, it is possible to conclude that: (i) *effectiveness*: sensory data are mapped to symbolic representation that can be directly operated upon; (ii) *reusability*: since symbolic representations are maintained within an ontology in a hierarchical fashion, selected concepts and relationships can be differently composed to build different representations; (iii) *expressiveness*: since tractability of inference is a fundamental prerequisite, the system is limited to simple temporal relationships among events; however, practice suggests that it is possible to model a wide range of contexts and situations for a broader spectrum of artificial cognitive systems.

REFERENCES

[1] T. Kim, S. Choi, and J. Kim, "Incorporation of a software robot and a mobile robot using a middle layer," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 37, no. 6, 2007.

[2] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "Context assessment strategies for ubiquitous robots," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA'09)*, (Kobe, Japan), May 2009.

[3] H. Nakashima, H. Aghajan, and J. A. (Eds), *Handbook on Ambient Intelligence and Smart Environments*. Berlin, Heidelberg: Springer–Verlag, 2008.

[4] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "From autonomous robots to artificial ecosystems," in *Handbook on Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan, and J. Augusto, eds.), Berlin, Heidelberg: Springer–Verlag, 2008.

[5] J. McCarthy and P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Machine Intelligence IV* (B. Meltzer and D. Michie, eds.), Edinburgh: Edinburgh University Press, 1969.

[6] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, MA: The MIT Press, 2001.

[7] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundation of Computer Science (FOCS-77)*, (Providence, Rhode Island, USA), May 1977.

[8] J. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, November 1983.

[9] P. V. Beek and D. Manchak, "The design and experimental analysis of algorithms for temporal reasoning," *Journal of Artificial Intelligence Research*, vol. 4, pp. 1–18, 1986.

[10] A. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, 2001.

[11] P. Dourish, "What we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, 2004.

[12] S. Loke, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective," *Knowledge Engineering Review*, vol. 19, no. 3, pp. 213–233, 2005.

[13] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp2004)*, (Nottingham, England), 2004.

[14] J. Ye, "Ontology-based models in pervasive computing systems," *Knowledge Engineering Review*, vol. 22, pp. 315–347, 2007.

[15] R. Krummenacher and T. Strang, "Ontology-based context-modelling," in *Proceedings of the 3rd Workshop on Context Awareness for Proactive Systems (CAPS'07)*, (Guildford, United Kingdom), June 2007.

[16] H. Chen, T. Finin, and A. Joshi, "An ontology for context aware pervasive computing environments," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, (Acapulco, Mexico), August 2003.

[17] T. Strang, C. Linnhoff-Popien, and T. Frank, "CoOL: A context ontology language to enable contextual interoperability," in *Proceedings of the 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, (Paris, France), November 2003.

[18] P. Lonsdale and R. Beale, "Towards a dynamic process model of context," in *Proceedings of the 1st Workshop on Advanced Context Modelling, Reasoning and Management, co-located with UbiComp'04*, (Nottingham, England), September 2004.

[19] A. V. Breemen, "A user-interface robot for ambient intelligent environments," in *Proceedings of the First International Workshop on Advances in Service Robotics (ASER 2003)*, (Bardolino, Italy), March 2003.

[20] K. LeBlanc and A. Saffiotti, "A cooperative anchoring in heterogeneous multi-robot systems," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA'08)*, (Pasadena, CA, USA), May 2008.

[21] F. Yamahoka, T. Kanda, H. Ishiguro, and N. Hagita, "A model of proximity control for information-presenting robots," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 187–195, 2010.

[22] E. Ko, H. Lee, and J. Lee, "Ontology-based context modeling and reasoning for u-healthcare," *IEICE Transactions on Information and Systems*, vol. 8, pp. 1262–12705, 2007.

[23] M. Muehlenbrock, "Learning to detect user activity and availability from a variety of sensor data," in *Proceedings of the 2004 International Conference on Pervasive Computing (PerComm04)*, (Piscataway, NY), 2004.

[24] J. Crowley, G. Coutaz, and P. Reignier, "Perceptual components for context-aware computing," in *Proceedings of the 2002 International Conference on Ubiquitous Computing (UbiComp'02)*, (Goteborg, Sweden), 2002.

[25] F. Giunchiglia and L. Serafini, "Multilanguage hierarchical logics (or: How can we do without modal logics?)," *Artificial Intelligence*, vol. 64, pp. 29–70, 1994.

[26] J. Augusto, "Enhanced healthcare provision through assisted decision-making in a smart home environment," in *Proceedings of the 2nd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI07)*, (Hyderabad, India), January 2007.

[27] T. Polger, "Computational functionalism," in *The Routledge Companion to the Philosophy of Psychology* (P. Calvo and J. Symons, eds.), London: Routledge Press, 2008.

[28] X. Wang, D. Zhang, T. Gu, and H. Pung, "A distributed architecture for symbolic data fusion," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, (Hyderabad, India), January 2007.

[29] M. Piaggio, A. Sgorbissa, and R. Zaccaria, "Preemptive versus non-preemptive real-time scheduling in intelligent mobile robotics," *Journal of Theoretical and Experimental Artificial Intelligence*, vol. 12, no. 2, pp. 235–245, 2000.

[30] M. Piaggio, A. Sgorbissa, and R. Zaccaria, "A programming environment for real-time control of distributed multiple robotic systems," *Advanced Robotics*, vol. 14, no. 1, pp. 75–86, 2000.

[31] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "A system for hierarchical planning in service mobile robotics," in *Proceedings of the Eight Conference on Intelligent Autonomous Systems (IAS-8)*, (Amsterdam, The Netherlands), March 2004.

[32] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "The more the better? a discussion about line features for self-localization," in *Proceeding of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, (San Diego, CA, USA), October 2007.

[33] A. Sgorbissa and R. Zaccaria, "Roaming stripes: Smooth reactive navigation in partially known environments," in *Proceeding of the 2003 IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN 2003)*, (Millbrae, CA, USA), October 2003.

[34] R. Brachman, A. Borgida, D. McGuinness, and P. Patel-Schneider, "Reducing classic to practice: Knowledge representation theory meets reality," *Artificial Intelligence*, vol. 114, no. 1–2, pp. 203–237, 1999.

[35] V. Devedzic, "Understanding ontological engineering," *Communications of the ACM*, vol. 45, no. 4, pp. 136–145, 2002.

[36] M. Brubaker, L. Sigal, and D. Fleet, "Video-based people tracking," in *Handbook on Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan, and J. Augusto, eds.), Berlin, Heidelberg: Springer–Verlag, 2008.

[37] M. Campani, F. Capezio, A. Rebora, A. Sgorbissa, and R. Zaccaria, "A minimalist approach to path following among unknown obstacles," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, (Taipei, Taiwan), October 2010.

Fulvio Mastrogiovanni's research activities expand in various branches of Robotics, specifically Humanoid and Distributed Robotics. Fulvio received his Master degree in Computer Engineering and the Ph.D. in Robotics from University of Genova in 2003 and 2008, respectively. Currently, Fulvio is interested in large scale tactile sensing for Humanoids, cognitive representation mechanisms, and neural-based planning and control. Fulvio received the Best Paper Awards at DARS 2008 and at IEEE RO-MAN 2010.

Antonello Scalmato received the Master degree in Computer Science Engineering from University of Genova in 2008. He is a third year PhD candidate at University of Genoa. His research interests include ontology applications for mobile robotics and ambient intelligence. Currently, his activities are focused on context-aware systems for smart homes and robots in intelligent environments.

Antonio Sgorbissa received the Laurea degree in Electronic Engineering from the University of Genova, Italy, in 1996. In 2000 he received is Ph.D. in Robotics from the University of Genova, Italy, and from 2001 to 2004 he was a Post Doc at Georgia Tech, University of Parma and later at University of Genova. Since 2005 he is Assistant Professor at the Department of Communication, Computer and System Sciences (DIST) of the University of Genova. He currently teaches Ambient Intelligence and Real-Time Operating Systems at the Faculty of Engineering, and Geographic Information Systems and Cognitive Robotics at the Faculty of Humanities. His main research interests are in the area of: mobile robotics, multi-robot systems, and ambient intelligence. His research interests include also planning, knowledge representation, and machine consciousness. He is author/coauthor of more than 100 international scientific papers, and of two international patents in Robotics.

Renato Zaccaria is Full Professor in Computer Science at the Department of Communication, Computer and System Sciences (DIST), University of Genova, Italy. He leads the Mobile Robotics & Ambient Intelligence Research Group in the lab called Laboratorium. He has been researcher and project leader in many national and European projects in Autonomous Robotics, and responsible of many industrial contracts as well.His present activity regards mainly Service Robotics and Ambient Intelligence. In order to purposively carry out technology transfer, in 2000 he founded one of the Department's spin-off companies, Genova Robot, whose present R&D activity is strongly linked to academic research. He is author/coauthor of more than 140 international scientific papers, of two textbooks, and of two international patents in Robotics (guidance of autonomous mobile robots). Presently, he is responsible of the European Master on Advanced Robotics EMARO, coordinator of the Robotics Engineering Master Track, and head of the Commission for the Study Programmes at the Faculty of Engineering. He has been awarded the title Commendatore della Repubblica because of his activity in technology transfer in Robotics.

AUTHORS' ADDRESSES

Fulvio Mastrogiovanni, Ph.D.
Antonello Scalmato, M.Sc.
Asst. Prof. Antonio Sgorbissa, Ph.D.
Prof. Renato Zaccaria, Ph.D.
Department of Communication, Computer and System Sciences,
University of Genova,
Via Opera Pia 13, 16145, Genova, Italy
email: {fulvio.mastrogiovanni, antonello.scalmato, antonio.sgorbissa, renato.zaccaria}@unige.it

Received: 2010-03-29
Accepted: 2011-07-14