Generalizing Precisiated Natural Language: A Formal Logic as a Precisiation Language

Takehiko Nakama[1] Enrique Muñoz[1] Enrique Ruspini[1]

[1]European Center for Soft Computing

Abstract

We generalize precisiated natrual language by establishing a formal logic as a generalized precisiation language. In this formal logic, each proposition has a form that reflects a syntactic structure observed in natural language. Various syntactic structures are incorporated in the formal logic so that it precisiates not only perceptual propositions but also action-related propositions. The syntax of the formal logic allows us to create infinitely many precisiated propositions while ensuring that every proposition in it is precisiated. We discuss how our formal logic can effectively mediate human-robot interaction.

Keywords: Precisiated natural language, precisiation language, formal logic, propositional logic, predicate logic, quantificational logic, computational theory of perceptions, human-robot interaction

1. Introduction and summary

A precisiated natural language (PNL), introduced by Zadeh (e.g., [1], [2], [3]), is a subset of a natural language that consists of propositions that can be precisiated through translation into a precisiation language, which is considered a set of elements that can serve as objects of computation and deduction. The propositions in PNL are supposed to represent human perceptions, and PNL is an integral part of the computational theory of perceptions ([1], [3]). This theory enables artificial intelligence to operate on and reason with perception-based information, which is intrinsically imprecise, uncertain, or vague. In Zadeh's framework, the primary function of natural language is characterized as describing perceptions, and each precisiable perceptual proposition drawn from a natural language is precisiated by translating it into a generalized-constraint language, which is a precisiation language. Each element in a generalized-constraint language is a generalized constraint on a variable, which has the form

$$X \ isr \ R, \tag{1}$$

where X denotes the constrained variable, R denotes the constraining relation, and r identifies the modality of the constraint ([1], [3]).

We maintain that it is important, both theoretically and practically, to extend precisiation language and PNL to other types of proposition. When we communicate using a natural language, we describe not only perceptions but also actions, for instance, and describing each action as a general constraint on a variable can be ineffective or inefficient; the form (1) is suitable for perceptual propositions, but we claim that there exist syntactic structures other than (1) that are more suitable for precisiating propositions that involve actions; see Section 3.

One of the major fields that require the precisiation of action-related propositions in natural language is robotics. Recently, many studies have been conducted to develop robotic systems in which humans and robots work as true team members, requiring peer-to-peer human robot interaction (e.g., [4], [5], [6]). By interacting with humans, robots can perform a wide range of practical tasks—assistance to people with disabilities (e.g., [7], [8], [9],[10]), search and rescue (e.g., [11], [12], [13], [14]), and space exploration (e.g., [15], [16]), for instance. The mode and degree of human-robot interaction vary considerably (see [5] for review). Robot operations in the teleoperation mode of human-robot communication require continuous low-level inputs from humans; as a result, even slight lapses in communications can degrade performance substantially, and the workload of the human operator can be undesirably high (e.g., [17], [18]). At the other extreme, autonomous robots that operate without any human input often perform complex tasks poorly compared to robots that collaborate with humans interactively (e.g., [19], [20], [21]). The peer-to-peer mode of human-robot communication has been developed to overcome the disadvantages of the fully teleoperational and autonomous approaches. One of the major challenges of this approach is the increased complexity of the human-robot interactions (e.g., [5]).

Various schemes for achieving extensive human-robot interactions have been developed. Many natural-language-based interaction schemes have been proposed (e.g., [22], [23], [24]). Although natural language is clearly a desirable medium for human communications, it presents several major problems when used for human-robot communications. Descriptions in natural language tend to be notoriously underspecified, diverse, vague, or ambiguous, so they often lead to errors that are hard to overcome (e.g., [25], [26], [27], [28], [29]). Restricting natural language to a small finite set of linguistic expressions mitigates these problems but does not fully support the extensive human-robot interactions required for complex tasks. Highly mathematical or symbolic expressions may be easy for experts (e.g., mathematicians, logicians, programmers, and system administrators) to understand, but naive users cannot be expected to be able to understand them; consequently, they are unsuitable for the variety of situations in which robotic systems need to interact with naive users as well as experts. Low-level sensory and motor signals and executable code are easy for robots to interpret, but they are cumbersome for humans and thus cannot, on their own, create an effective human-robot interface.

We claim that PNL can effectively mediate the peer-to-peer mode of human-robot interaction. Consider describing tasks that require peer-to-peer human-robot interaction. Those task descriptions must be easily interpretable for both humans and robots. If the tasks are described in PNL, then the task descriptions can be easily understood by humans because they are propositions in natural language. At the same time, each PNL proposition is precisiable, so each task description can be unambiguously interpreted and executed by robots. Thus, PNL can serve as a middle ground between the natural-language-based mode of human communication and the low-level mode of robotic communication. Since task descriptions inherently involve actions, we must extend PNL to action-related propositions.

In this paper, we generalize PNL by establishing a formal logic as a precisiation language. Our formal logic can create infinitely many precisiated propositions, just as infinitely many propositions can be created in natural language, while ensuring that every proposition in the formal logic is precisiated. In Section 2, we explain why we want to treat the precisiation language as a formal language. As described by Zadeh, bivalence is not desirable for the semantics of precisiation language, so our formal logic is many-valued. We explain the semantics of our formal logic in Section 8.

Since the precisiation of perceptual propositions has been examined in detail (e.g., [1], [3]), we focus on examining how to precisiate action-related propositions in this paper. To explain our formal scheme, we will first appeal to ordinary practices and then move on to formal considerations so that the reader can understand it intuitively. To generate examples of ordinary practice, we consider establishing task descriptions for human-robot interaction. As mentioned earlier, task descriptions inherently involve actions, so the precisiation of task descriptions is an important step toward generalizing PNL.

2. Why formal logic?

As mentioned in Section 1, we establish a formal logic as a generalized precisiation language. In this section, we briefly discuss properties of formal logic that are suitable for precisiation language.

The application of formal logic to natural language is a paradigm of logical analysis [26]; it provides genuine insight into the syntactic structures of natural-language sentences and the consequential characters of assertions expressed by them. This analysis is important for precisiating propositions in natural language. In order for PNL to have high expressive power, it is desirable that precisiation language can generate infinitely many precisiated propositions while ensuring that every proposition in it is precisiated. Formal logic achieves these properties by a recursive definition of its syntax; it can generate infinitely many well-formed formulas while ensuring that every formula in it is well-formed.

Our scheme also reflects the theory of descriptions in formal logic, which was introduced by Russell [30]. He claimed that the reality consists of logical atoms, which can be considered indecomposable, self-contained building blocks of all propositions in formal logic, and that logical analysis ends when we arrive at logical atoms. In our precisiation language, precisiation ends when we arrive at logical atoms, which will be represented by atomic propositions at the lowest level of a hierarchy of propositions. See Section 7.

3. Propositions in the formal logic

In this section, we describe propositions that compose our formal logic. As mentioned in Section 1, we will explain our formal logic using examples of task description for human-robot interaction, but our scheme is not limited to precisiating propositions that describe tasks (nonetheless notice that, since tasks typically involve actions, we will be extending PNL to action-related propositions). We will discuss the generality of our formal logic in Section 5. In this formal logic, each proposition has a syntactic form observed in natural language. Our formal logic generalizes generalized-constraint language by incorporating multiple syntactic forms so that it can deal not only with perceptual propositions but also with action-related propositions.

In Section 3.1, we describe the components of such propositions. In Section 3.2, we describe how to form an atomic proposition. In Section 3.3, we describe how to form a compound proposition.

In these sections, we will provide examples of rather simple task descriptions and explain our scheme in an intuitive manner. However, our scheme can be easily applied to robotic systems that require more intricate task descriptions. See Section 5.

3.1. Components of propositions

First, we define sets whose elements constitute propositions. These sets will be called component sets. To provide concrete examples, we consider the following component sets:

- S denotes the set of all agents that can perform a task. For instance, we can have
$$S = \{robot1, robot2, robot3,\ user\}. \tag{2}$$
- V denotes the set of all verbs that characterize actions required by tasks. For instance, we can have
$$V = \{call, find, deliver, go, move, press\}\,. \tag{3}$$
- O denotes the set of all objects that may receive an action in V or compose an adverbial phrase [see (5)]. For example, we can have
$$O = \{box, button, table, room1, room2, building, robot1, robot2, robot3, user, null\}\,. \tag{4}$$
Here the three robots and the user in S are also in O because they can also receive an action in V. The element denoted by "null" will be called the null element. In Section 3.2, we will explain how it is used in forming a task description.
- A denotes the set of all adverbial phrases that can be included in task descriptions. For instance, we can have
$$A = \{in\ \gamma, from\ \gamma, from\ \gamma_1\ to\ \gamma_2, to\ \gamma, null \mid \gamma, \gamma_1, \gamma_2 \in O\}\,. \tag{5}$$
Again, the null element is denoted by "null," and its use is explained in Section 3.2.
- C denotes the set of all connectives that can be used to combine multiple propositions in forming compound propositions (see Section 3.3 for compound propositions). For instance, we can have
$$C = \{and, if, or, then, until, whenever\}. \tag{6}$$

The elements in these component sets are combined in a specified manner (see Section 3.2) to form propositions in our formal logic.

3.2. Atomic propositions

In our formal logic, an atomic proposition is defined to be a tuple that consists of elements in component sets. Each admissible tuple structure is specified in the form of a cartesian product of component sets. To develop propositions that can be considered natural-language sentences, we employ tuple structures that reflect syntactic structures observed in natural language. For instance, using the component sets described in Section 3.1, we can define each atomic proposition to be an SVOA clause (The S, V, O, and A in SVOA stand for subject, verb, object, and adverbial phrase, respectively) by setting the admissible tuple structure to $S \times V \times O \times A$. The SVOA structure is observed in many languages, including English, Russian, and Mandarin. Using the null element in O and A, we can also generate SVO, SVA, and SV clauses. See the following examples of atomic propositions, which describe tasks for a robotic system, resulting from the component sets (2)–(5):

- $\frac{robot2}{S}\ \frac{find}{V}\ \frac{ball}{O}\ \frac{null}{A}$.
We will omit instances of the null element and express it more simply as
$\frac{robot2}{S}\ \frac{find}{V}\ \frac{ball}{O}$.
- $\frac{robot1}{S}\ \frac{deliver}{V}\ \frac{box}{O}\ \frac{from\ room1\ to\ room2}{A}$.

For humans, these propositions (task descriptions) are easy to specify and understand. Meanwhile, the structural and lexical constraints substantially limit the diversity and flexibility of everyday language, so we can ensure that the resulting propositions are unambiguously interpretable for robots (i.e., the specified tasks can be precisely interpreted and executed by robots). As will be explained in Section 7, we establish a hierarchy of propositions, and at the lowest level of the hierarchy, each atomic proposition is directly associated with an indecomposable, self-contained executable code. Thus, these atomic propositions can be considered building blocks of all propositions, and they constitute each proposition at higher levels of the hierarchy.

There are several ways to deal with the undesirable or nonsensical atomic propositions that can be formed in $S \times V \times O \times A$. We can remove all such propositions from the cartesian product to ensure that each resulting atomic proposition is a precisiated proposition. (In this case, we abuse the notation and let $S \times V \times O \times A$ denote the "cleaned" cartesian product.) Also, we can consider them as always false so that they will never be executed in practice (see Section 8).

Notice that the atomic propositions are not expressed as generalized constraints on variables. We can precisiate action-related propositions rather naturally and effectively using the SV, SVO, SVA, and SVOA structures described in this example, and other syntactic structures can be incorporated in our formal logic. See Section 5.

3.3. Compound propositions

A compound proposition consists of multiple atomic propositions combined by one or more connectives in the component set C. With the examples (2)–(6) in Section 3.1, we can form the following compound propositions:

- $\dfrac{\frac{robot1}{S}\ \frac{put}{V}\ \frac{ball}{O}\ \frac{in\ box}{A}}{\text{atomic proposition}}\ \dfrac{if}{C}\ \dfrac{\frac{user}{S}\ \frac{call}{V}\ \frac{robot1}{O}}{\text{atomic proposition}}.$

- $\dfrac{\frac{robot2}{S}\ \frac{go}{V}\ \frac{to\ room1}{A}}{\text{atomic proposition}}$
$$\dfrac{whenever}{C}\ \dfrac{\frac{user}{S}\ \frac{press}{V}\ \frac{button}{O}}{\text{atomic proposition}}.$$

When necessary, we use parentheses to disambiguate the manner in which atomic tasks are performed:

- $\dfrac{\frac{robot1}{S}\ \frac{go}{V}\ \frac{to\ room1}{A}}{\text{atomic proposition}}$
$$\dfrac{if}{C}\left(\dfrac{\frac{user}{S}\ \frac{call}{V}\ \frac{robot1}{O}}{\text{atomic proposition}}\quad \dfrac{or}{C}\ \dfrac{\frac{user}{S}\ \frac{press}{V}\ \frac{button}{O}}{\text{atomic proposition}}\right). \tag{7}$$

These compound propositions are still quite easy for humans to specify and understand, and the syntactic structures imposed on the clauses and the compositions ensure effective interpretation and execution by robots.

Note that these compound propositions are precisiated propositions although they are not expressed as generalized constraints on variables. In fact, it can be quite difficult or ineffective to translate them into generalized constraints.

There are many tasks that we can describe using conditions and imperatives. For instance, consider the compound task (7). In this description, the clause "*user call robot1*" or "*user press button*" represents a condition that must be checked in determining whether to send the robot to room1, and the clause "*robot go to room1*" is an imperative. Our task descriptions consist of atomic propositions described in Section 3.2, and each atomic proposition will be either a condition or an imperative. The type of each atomic proposition in a compound task description is unambiguously determined by the logical connective that connects it and by the location of the atomic proposition relative to the connective. For instance, an atomic proposition that forms a subordinate clause immediately following the connective "if" is considered a condition, whereas an atomic proposition that forms a clause immediately preceding the connective is considered an imperative. If the proposition is a condition, the robotic system monitors the described condition. If it is an imperative, the system executes the described action provided that all the required conditions are satisfied.

4. Well-formed formulas of the formal logic

In this section, we provide the syntax of our formal logic, which allows us to create infinitely many precisiated propositions while ensuring that each proposition in the formal logic is precisiated. We will provide a recursive definition of the syntax using the framework discussed in Section 3, but it can be easily generalized; we will address this issue in Section 5.

Using the component sets S, V, O, A, and C described in Section 3.1, we can define the syntax of a propositional logic as follows:

1. Any $x \in S \times V \times O \times A$ is an atomic well-formed formula.
2. If α and β are well-formed formulas, then $\alpha\, c\, \beta$, where $c \in C$, is also a well-formed formula.
3. Nothing else is a well-formed formula.

This recursive definition allows us to generate infinitely many well-formed formulas, i.e., precisiated propositions. With sufficiently rich component sets, we can describe any task that must be performed by the robotic system, and, using the syntax, we can ensure that the robotic system does not operate on any ill-formed task descriptions (i.e., task descriptions that are not interpretable).

5. Generality of our framework

Although we considered rather simple component sets and syntactic structures in Sections 3–4 to facilitate the exposition of our formal logic, we can easily extend our scheme to more sophisticated component sets and syntactic structures so that our formal logic can deal with complex actions and perceptions. Each component set can be made as large as necessary, and other component sets or clause structures can be incorporated in the formal logic. For instance, in addition to the SV, SVO, SVA, and SVOA structures described and used in Sections 3–7, we can also use other commonly observed clause structures (see, for instance, [31]), such as the SVC, SVOC, and SVOO structures, in forming atomic propositions. Furthermore, we can extend the clause structure so that a phrase can be used as the subject or the object in an atomic proposition. Negation, a unary logical connective, can certainly be incorporated in the formal logic. For perceptual propositions, we can incorporate generalized constraints in our formal logic; each generalized constraint can be considered an atomic proposition that has the SVC structure, and it can be combined with other propositions by connectives to form a compound proposition.

We can also establish a quantificational logic as a precisiation language. Quantificational logic fully incorporates quantifiers and predicates in well-formed formulas. Since propositions that describe perceptions often include quantifiers (see, for instance, [2, 3]), it is desirable to develop a quantificational logic as a generalized precisiation language that covers both actions and perceptions.

6. Inference and reasoning in the formal logic

As in other formal logics, we can infer and reason in our formal logic by adding a deductive apparatus to it. Syntactically, a set of inference rules such as modus ponens and modus tollens can be constructed, and axioms can also be established. (The hierarchy described in Section 7 represents non-logical, domain-dependent axioms.) Thus, in our formal logic, we can form a sequent, which consists of a finite (possibly empty) set of well-formed formulas (the premises) and a single well-formed formula (the conclusion), and examine its provability. As will be described in Section 8, we can employ fuzzy relations to establish the semantics of our formal logic, so we can also investigate the truth conditions and the semantic validity of each proposition. Space limitations do not allow us to elaborate on these matters in this paper; we will describe them fully in our full-length papers.

7. Hierarchy of propositions

In our formal logic, we can establish a hierarchy of propositions that enhances the expressive power and the interactivity of the pecisiation language. This hierarchy also results in domain-dependent axioms, which can be used for inference and reasoning in the formal logic. We explain the hierarchy using the task description scheme described in Sections 3–4.

A complex task can be described rather concisely, i.e., it can be expressed by an atomic proposition. In many cases, naive users will prefer specifying a complex task using an atomic proposition as opposed to a more lengthy compound proposition. On the other hand, in order for a robotic system to actually execute a complex task, the task must be broken down into simpler subtasks, and the manner in which the subtasks are performed must be specified. Consequently, the atomic proposition describing a complex task can be reexpressed as a compound proposition that consists of atomic propositions describing the required subtasks. Some of the subtasks may have to be further decomposed in order to fully specify how to execute them. Expert users may want to establish and combine these subtasks carefully so that the robotic system can perform the task effectively and efficiently.

Thus, we can precisiate an atomic proposition by reexpressing it as a compound proposition consisting of atomic propositions that precisiate it. This process also leads to flexibility in the level of detail. In the task description example, the flexibility gives naive users an efficient, user-friendly interface with robots while giving expert users the power to customize tasks. As a result, the hierarchy allows human-robot interactions to take place at various levels of detail, and it also helps determine the appropriate level of detail for each human-robot interaction; the hierarchy determines whether, from the point of view of a given user, a given task is "atomic" or "compound."

To explain this hierarchy, we first explore it informally, appealing to intuitions, and then move on to formal characterizations. Consider the following task description:

$$\frac{robot}{S}\ \frac{bring}{V}\ \frac{box}{O}\ \frac{from\ room1\ to\ room2}{A}. \tag{8}$$

This atomic proposition can be reexpressed as a compound proposition that consists of three atomic propositions representing subtasks that must be performed to accomplish the task:

$$\begin{gathered} \frac{\frac{robot}{S}\ \frac{find}{V}\ \frac{box}{O}\ \frac{in\ room1}{A}}{\text{atomic proposition 1}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{collect}{V}\ \frac{box}{O}}{\text{atomic proposition 2}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ room2}{A}}{\text{atomic proposition 3}}. \end{gathered} \tag{9}$$

Atomic propositions 1 and 2 in (9) can also be reexpressed as compound propositions that clarify how they are performed; atomic proposition 1 in (9) can be defined as

$$\frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ room1}{A}}{\text{atomic proposition}}\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{search}{V}\ \frac{box}{O}}{\text{atomic proposition}}, \tag{10}$$

and atomic proposition 2 in (9) can be defined as

$$\frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ box}{A}}{\text{atomic proposition}}\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{grasp}{V}\ \frac{box}{O}}{\text{atomic proposition}}. \tag{11}$$

Therefore, using (10)–(11) and atomic proposition 3 in (9), we can reexpress (8) as

$$\begin{gathered} \frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ room1}{A}}{\text{atomic proposition}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{search}{V}\ \frac{box}{O}}{\text{atomic proposition}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ box}{A}}{\text{atomic proposition}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{grasp}{V}\ \frac{box}{O}}{\text{atomic proposition}} \\ \frac{then}{C}\ \frac{\frac{robot}{S}\ \frac{go}{V}\ \frac{to\ room2}{A}}{\text{atomic proposition}}. \end{gathered} \tag{12}$$

Figure 1 visualizes this hierarchy, which consists of three levels (levels 0, 1, and 2). For simplicity, each atomic proposition is represented by its verb; for instance, the atomic proposition at the highest level (level 2), "Robot *bring* box from room1 to room2," is represented by "bring." The task expressed by the atomic proposition at level 2 is described in more detail at the intermediate level (level 1), where the atomic propositions that involve the verbs "find," "collect," and "go" describe the subtasks that constitute the task. These subtasks are described in more detail at the lowest level (level 0), where they are expressed by the atomic propositions that involve the verbs "go," "search," and "grasp."

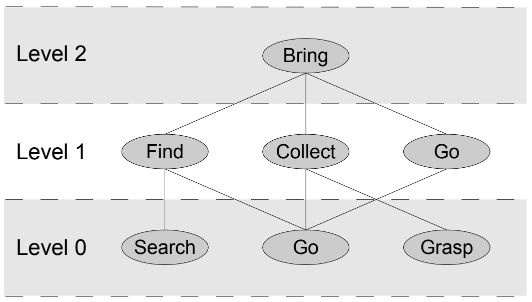

Figure 1: Hierarchical task description. At the highest level (level 2), the task is expressed as atomic proposition (8), represented by "bring." At the intermediate level (level 1), the task is expressed as compound proposition (9), which consists of atomic propositions represented by "find," "collect," and "go." At the lowest level (level 0), the task is expressed as compound proposition (12), which consists of atomic propositions represented by "go," "search," and "grasp."

The hierarchy clearly shows how atomic proposition 8 is precisiated. At level 0, we have atomic propositions that are not decomposable; each of them is directly associated with a self-contained executable code that is run to perform the corresponding task. Thus, atomic propositions at level 0 can be considered logical atoms described in Section 2, and they precisiate each proposition at higher levels.

The suitability of a given proposition depends on the level of granularity required for it. As regards the task description scheme, naive users will most likely prefer describing tasks at level 2, thus preferring (8). For expert users, there may be situations where they prefer specifying a given task step by step or reconfiguring its subtasks according to various circumstances; in such cases, interacting with robots at level 1 using (9) or at level 0 using (12) will be desirable. Thus, the hierarchy allows a variety of users to interact with robots at various levels of detail.

The hierarchy of propositions can be characterized more formally as follows. Let S_i, V_i, O_i, A_i, and C_i denote the component sets for level i of the hierarchy ($i \geq 0$). The elements in these sets reflect the degree of detail suitable for level i. Then at level i, we establish a formal logic with these component sets, as described in Sections 3–4. Atomic propositions in $S_0 \times V_0 \times O_0 \times A_0$ are the logical atoms and the building blocks of all propositions; each of them is indecomposable and directly associated with a self-contained computational unit. We will call such a computational unit as a computational atom. For each $i \geq 1$, every atomic proposition in $S_i \times V_i \times O_i \times A_i$ can be decomposed into atomic propositions in $S_{i-1} \times V_{i-1} \times O_{i-1} \times A_{i-1}$.

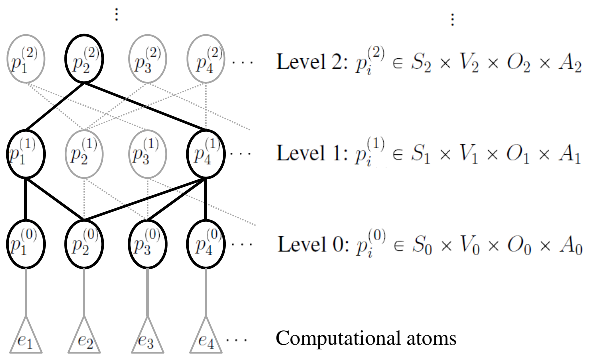
$p_1^{(2)}$ $p_2^{(2)}$ $p_3^{(2)}$ $p_4^{(2)}$ ⋯
Level 2: $p_i^{(2)} \in S_2 \times V_2 \times O_2 \times A_2$
$p_1^{(1)}$ $p_2^{(1)}$ $p_3^{(1)}$ $p_4^{(1)}$ ⋯
Level 1: $p_i^{(1)} \in S_1 \times V_1 \times O_1 \times A_1$
$p_1^{(0)}$ $p_2^{(0)}$ $p_3^{(0)}$ $p_4^{(0)}$ ⋯
Level 0: $p_i^{(0)} \in S_0 \times V_0 \times O_0 \times A_0$
e_1 e_2 e_3 e_4 ⋯
Computational atoms

Figure 2: Hierarchy of propositions. At level i, the formal logic described in Sections 3–4 is established with component sets S_i, V_i, O_i, A_i, and C_i. Atomic propositions in $S_0 \times V_0 \times O_0 \times A_0$ are the logical atoms and the building blocks of all propositions, and each of them is directly associated with a computational atom. For each $i \geq 1$, every atomic proposition at level i is connected to atomic propositions at level $i-1$ that precisiate it.

Figure 2 visualizes a typical form of this hierarchy. For each i and j, $p_j^{(i)}$ denotes an atomic proposition at level i, and e_j denotes a computational atom associated with $p_j^{(0)}$. In the figure, $p_j^{(0)}$ is connected to e_j for each j, and for each $i \geq 1$ and j, $p_j^{(i)}$ is connected to atomic propositions at level $i-1$ that precisiate $p_j^{(i)}$. The figure shows that, for instance, $p_2^{(2)}$ can be expressed as a compound proposition consisting of two atomic propositions ($p_1^{(1)}$ and $p_4^{(1)}$) at level 1 and also as a compound proposition consisting of four atomic propositions ($p_1^{(0)}$, $p_2^{(0)}$, $p_3^{(0)}$, and $p_4^{(0)}$) at level 0. As regards formal logic, these hierarchies represent non-logical, domain-dependent axioms.

Different levels of granularity may require different component sets, but the same syntactic structure is enforced at all levels. Using the hierarchy, we can ensure that all the resulting propositions remain precisiated at each level, and we can attain flexibility in the level of detail.

8. Semantics of the formal logic

The semantics of formal logic specifies how to determine the truth value of each proposition. In twovalued logics, for instance, the truth value is either 1 (true) or 0 (false). As described by Zadeh (e.g., [1], [3]), this bivalence is not suitable for PNL, so we develop a many-valued semantics for our formal logic. The meaning of the truth value depends on the context. For the task description scheme described in Sections 3–7, for instance, one can evaluate each proposition and let its truth value reflect the feasibility of the corresponding task specification; 1 indicates that the task certainly can be carried out whereas 0 indicates that it certainly cannot be. In this case, it is more realistic and practical to let the degree of feasibility take on not only the values 0 and 1 but also other values between 0 and 1. In real-world problems, it can be highly practical to evaluate the feasibility of a task description before any serious attempt is made to execute it. If robotic systems interact with a variety of users, including users who have no knowledge of these systems, some users may describe tasks that are virtually impossible to accomplish. It is more desirable to disregard such highly infeasible tasks immediately than to waste resources by attempting to realize them. Also, the user may want to be informed of the degree of feasibility of the task that he specifies before the system attempts to perform it. When multiple options are available for performing a specified task, the user may want to compare their degrees of feasibility before determining which option to take.

To determine the truth value of each proposition in our formal logic systematically and effectively, we use a fuzzy relation, which is a generalization of a classical ("crisp") relation (see, for instance, [32]). It is a mapping from a Cartesian product to the set of real numbers between 0 and 1. While a classical relation only expresses the presence or absence of some form of association between the elements of factors in a Cartesian product, a fuzzy relation can express various degrees or strengths of association between them. (Hence a classical relation can be considered a "crisp" case of a fuzzy relation.) In our formal logic, each proposition consists of prespecified components, so a function that assigns a degree of feasibility to each proposition can be represented by a fuzzy relation on a Cartesian product of the components. Using some of the operations defined on fuzzy relations, we can systematically and economically determine the truth value of each proposition. Space limitations on this paper do not allow us to fully describe the semantics of our formal logic; we will provide details of the semantics in our full-length paper.

9. Conclusions

This paper is a first step toward generalizing PNL (and precisiation language) and establishing a formal logic as a generalized precisiation language. Various syntactic structures in natural language can be incorporated in our formal logic so that it precisiates not only perceptual propositions but also action-related propositions. The high expressive power of the formal logic is achieved by its syntax, which enables us to create infinitely many precisiated propositions while ensuring that every proposition is precisiated. Our generalized precisiation language can serve as a middle ground between the natural-language-based mode of human communication and the low-level mode of robotic communication.

Acknowledgements

This research is supported by the Spanish Ministry of Economy and Competitiveness through the project TIN2011-29824-C02-02 (ABSYNTHE).

References

[1] Lotfi A Zadeh. A new direction in ai: Toward a computational theory of perceptions. *AI magazine*, 22(1):73, 2001.

[2] L. A. Zadeh. Some reflections on information granulation and its centrality in granular computing, computing with words, the computational theory of perceptions and precisiated natural language. *Studies in Fuzziness And Soft Computing*, 95:3–22, 2002.

[3] L. A. Zadeh. Precisiated natural language (PNL). *AI Magazine*, 25(3):74–92, 2004.

[4] M. Bernardine Dias, Thomas K. Harris, Brett Browning, Edward Gil Jones, Brenna Argall, Manuela M. Veloso, Anthony Stentz, and Alexander I. Rudnicky. Dynamically formed human-robot teams performing coordinated tasks. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 30–38, 2006.

[5] M. A. Goodrich and A. C. Schultz. Human-robot interaction: a survey. *Found. Trends Hum.-Comput. Interact.*, 1:203–275, 2007.

[6] M. Johnson, P. J. Feltovich, J. M. Bradshaw, and L. Bunch. Human-robot coordination through dynamic regulation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2159–2164, 2008.

[7] G. Lacey and K. M. Dawson-Howe. The application of robotics to a mobility aid for the elderly blind. *Robotics and Autonomous Systems*, 23:245–252, 1998.

[8] I. Shim, J. Yoon, and M. Yoh. A human robot interactive system "RoJi". *International Journal of Control, Automation, and Systems*, 2:398–405, 2004.

[9] D. Feil-Seifer and M. J. Mataric. Defining socially assistive robotics. In *Proceedings of the International Conference on Rehabilitation Robotics*, pages 465–468, 2008.

[10] V. Kulyukin, C. Gharpure, J. Nicholson, and G. Osborne. Robot-assisted wayfinding for the

visually impaired in structured indoor environments. *Autonomous Robots*, 21:29–41, 2006.

[11] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahsahi, A. Shinjou, and S. Shimada. RoboCup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 739–743, 1999.

[12] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33:367–385, 2003.

[13] M. B. Dias, B. Kannan, B. Browning, E. G. Jones, B. Argall, M. F. Dias, M. Zinck, M. M. Veloso, and A. J. Stentz. Evaluation of human-robot interaction awareness in search and rescue. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2327–2332, 2008.

[14] I. R. Norbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion. Human-robot teaming for search and rescue. *Pervasive Computing*, pages 72–79, 2005.

[15] T. Fong and C. Thorpe. Vehicle teleoperation interfaces,. *Autonomous Robots*, 11:9–18, 2001.

[16] T. Fong, I. Nourbakhsh, C. Kunz, L. Flückiger, J. Schreiner, R. Ambrose, R. Burridge, R. Simmons, L.M. Hiatt, and A. Schultz. The peer-to-peer human-robot interaction project. *Space*, 6750, 2005.

[17] M. Anderson, C. Conner, V. Daniel, M. McKay, and N. Yancey. Demonstration of the robotic gamma locating and isotopic identification device. In *Proceedings of the he American Nuclear Society Spectrum*, 2002.

[18] D.J. Bruemmer, J.L. Marble, D.D. Dudenhoeffer, M. Anderson, and M.D. Mckay. Mixed-initiative control for remote characterization of hazardous environments. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.

[19] K. A. Abbott, S. M. Slotte, and D. K. Stimson. Federal aviation administration human factors team report on: The interfaces between flightcrews and modern flight deck systems, June 1996.

[20] J. A. Espinosa, J. Cadiz, L. Rico-Gutierrez, R. E. Kraut, W. Scherlis, and G. Lautenbacher. Coming to the wrong decision quickly: Why awareness tools must be matched with appropriate tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 392–399. ACM, 2000.

[21] D.J. Bruemmer, D.A. Few, R.L. Boring, J.L. Marble, M.C. Walton, and C.W. Nielsen. Shared understanding for collaborative control. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(4):494–504, 2005.

[22] T. C. Lueth, T. Laengle, G. Herzog, E. Stopp, and U. Rembold. KANTRA: human-machine interaction for intelligent robots using natural language. In *Proceedings of the IEEE International Workshop on Robotand HumanCommunication*, pages 106–111, 1994.

[23] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Toward conversational human-computer interaction. *AI Magazine*, 22:27–37, 2001.

[24] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein. Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 38:171–181, 2002.

[25] T. Winograd and F. Flores. *Understanding computers and cognition: A new foundation for design.* Ablex Pub, 1986.

[26] Paul Tomassi. *Logic.* Routledge, 1999.

[27] B. Shneiderman. The limits of speech recognition. *Communications of the ACM*, 43(9):63–65, 2000.

[28] M. Forsberg. Why is speech recognition difficult. *Chalmers University of Technology*, 2003.

[29] P. Gieselmann and P. Stenneken. How to talk to robots: Evidence from user studies on human-robot communication. *How People Talk to Computers, Robots, and Other Artificial Communication Partners*, page 68, 2006.

[30] B. Russell. Lectures on the philosophy of logical atomism. In R. C. Marsh, editor, *Logic and Knowledge Essays 1901–1950.* George Allen & Unwin, London, 1984.

[31] Douglas Biber, Susan Conrad, and Geoffrey Leech. *A student grammar of spoken and written English.* Pearson ESL, 2002.

[32] George J Klir and Tina A Folger. *Fuzzy sets, uncertainty, and information.* Prentice Hall, 1988.