Efficient, Situated and Ontology based Referring Expression Generation for Human-Robot collaboration

Guilhem Buisan, Guillaume Sarthou, Arthur Bit-Monnot, Aurélie Clodic, Rachid Alami

Efficient, Situated and Ontology based Referring Expression Generation for Human-Robot collaboration

Guilhem Buisan[1*], Guillaume Sarthou[1*], Arthur Bit-Monnot[1], Aurélie Clodic[1,2] and Rachid Alami[1,2]

***Abstract*— In Human-Robot Interaction (HRI), ensuring non-ambiguous communication between the robot and the human is a key point for carrying out fluently a collaborative task. With this work, we propose a method which allows the robot to generate the optimal set of assertions that are necessary in order to produce an unambiguous reference. In this paper, we present a novel approach to the Referring Expression Generation (REG) problem and its integration into a robotic system. Our method is a domain-independent approach based on an ontology as a knowledge base. We show how this generation can be performed on an ontology which is not dedicated to this task. We then validate our method through simulated situations, compare it with state of the art approach and on a real robotic system.**

I. Introduction

Communication, be it verbal or not, is a key aspect for the success of a Human-Robot collaborative task. This becomes especially important in complex environments with a wide variety of entities: objects, places, people. Referring to a specific entity in such an environment can become challenging when one must account for the context of the task, the variety of facts that can be extracted from the situation depending on available perception modalities and the common ground between the robot and its human partners. The need to communicate on particular objects happens in many everyday tasks and it is important to endow the robot with the ability to estimate if and how referring to an object is feasible as well to assess the intelligibility of a referring expression (RE). This need is also important since deictic pointing is not always available in cooperative tasks when hands might be occupied to do something else.

Consider the situation where a robot needs to ask for a given pen which is not reachable by it but which is visible and reachable by its human partner. The action to refer to the pen can occur in situations of different complexities (Fig. 1). When only one pen is present (Fig. 1a), the reference is obvious to produce. If however there are two pens in two pencil boxes of different colors (Fig. 1f), the robot has to generate an expression referring to one of the pencil boxes in order to refer to the target pen.

Until now, we considered that the robot knows the concepts of pen and pencil box as well as their names in natural language to speak about them. However, if our robot has to speak French and does not know the translation of pencil box, it will have to resort to a more generic term, such as "container". This may raise new ambiguities which will have to be solved, e.g. in the case where other containers are visible to the human. The robot must also pay attention to the relationships it uses to refer to an entity. For instance, using the exact weight of an object wouldn't be useful and using the color will not be the most suitable property for a color blind person. This means that the robot has to only use relations known and perceived by the human partner. This is done by taking into account the theory of mind and therefore by generating the RE on the estimation by the robot of the knowledge base of the human partner. In the rest of the paper, the knowledge base considered will therefore always be the set of the human beliefs which the robot estimates, which ensures that all of the concepts and relationships used are known to the partner and take into account his perspective of the current state of the world.

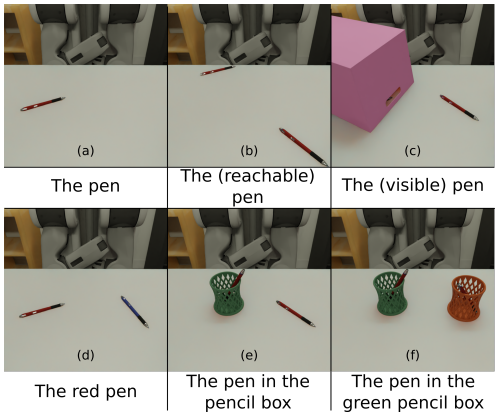

Fig. 1. Six situations where referring to a pen leads to different mechanisms to deal with ambiguities. The views are shown from the side of the Human, the Robot is placed in the other side of the table

The underlying process of all of the previous examples is what is commonly called a *Referring Expression Generation* task. It is often composed of two sub-tasks: the content determination (determining the relations to use) and the linguistic realization (choosing the words to use to communicate the content) [8]. While the focus of this paper is on the content determination, it is impossible to consider these two sub-tasks entirely independently. Indeed, the generation and the valuation of the content depends strongly on the concepts usable in natural language. This can be ensured by using a dedicated knowledge base, containing only verbalizable concepts, but such knowledge bases can be hard to maintain during the interaction. In a cooperative scenario, one should also ensure that the referring expression does not rely on any

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France `firstname.surname@laas.fr`
[2]Artificial and Natural Intelligence Toulouse Institute (ANITI)
*These authors contributed equally to this work

fact unknown to the human partner.

The main contribution of this paper is an **ontology-based and domain-independent algorithm for the generation of referring expressions**. Using a customizable cost function estimating the cognitive load required for a human to interpret the RE, it produces **the optimal set of verbalizable assertions**, allowing to **discriminate unambiguously** a given entity referred to by the robot. We show that our method outperforms state-of-the-art methods on existing benchmarks and scales to realistically sized knowledge-bases (77 objects distributed in a three-room apartment) when integrated in a complete robotic architecture.

In §II, we briefly discuss related work and how our method addresses new issues. The problem is defined in §III. We provide in §IV a description of our resolution procedure. Comparisons with state of the art algorithms and an integration of the method on a real robotic system are respectively presented in §V and §VI.

II. Related Work

Referring Expression Generation has been studied for decades and "is concerned with how we produce a description of an entity that enables the hearer to identify that entity in a given context" [13]. Criteria for good RE include the referential success (the target entity is unambigously identified), the ease of comprehension (the RE must be understood quickly by the receiver, note that it depends on person) and the avoidance of false implications[1] (especially be as short as possible) [4]. Moreover, the computational complexity in term of generation time should also be considered, as they could be generated in a dynamic and changing environment.

Two pioneer fundamental approaches to the REG problem are the Depth First Search (DFS) [2] and the Full Brevity [3] algorithms. While the first does not always find an optimal solution, the second does at the cost of an exhaustive search. Early in the field, the notion of preference over features has been highlighted in [4] with the Incremental Algorithm (IA) in order to promote the use of certain features such as the color or the shape of an object rather than spatial relations. However, due to an attribute-value pair representation, these solutions can only refer to entity attributes and cannot refer to relations between entities.

The Graph-Based Algorithm (GBA) REG proposed in [7] introduces a new representation in the form of a labeled directed multi-graph also known as the REG graph. This representation allows to use relations between entities to generate a RE and integrates the preferences over features by assigning costs to each edge of the graph. These costs are called Preference Ordering (PO) and aim for physiological realism. On top of this representation a branch&bound algorithm is used to find the optimal RE. Many extensions of the GBA have been made with for example a basic-level category descriptor [8] which also addresses the problem of the hierarchy of entity types. [10] proposed some optimizations on the original GBA allowing an efficiency gain close to 56% but since their task involves only cubes, they do not take into account the types of objects. An other interesting GBA is the Longest First (LF) algorithm [16] but, as we will show later, its exhaustive search entails poor performance when used on larger realistic knowledge bases.

Other approaches have been proposed like the belief network based disambiguation method, introduced in [20] with the ability to work on several object attributes. However, the authors indicate that a specific belief network should be constructed and therefore trained for each attribute. This limitation reduces the genericity of the method. [5] face the same problem with a log-linear model trained from a corpus of the probability distribution of REs. An important aspect of the REG problem is highlighted in [20]: by working on belief bases, their algorithm runs on the human partner's estimated belief base which ensures that the robot generate a referring solution compatible with concepts known by the human.

All solutions mentioned above are highly domain-dependent, whether through training on corpus or dedicated representations integrating only relations relevant to the task. In [18], the authors presented a hybrid approach between domain-dependent and domain-independent with the DIST-PIA, a distributed Incremental Algorithm. The idea is to have domain-dependent consultants [17] on each of the distributed knowledge bases in order to have a domain-independent IA querying these consultants. While this is a good solution for distributed knowledge bases, it still raises some issues regarding the hand made ordering of relations in the consultants and the impact of the order of the consultants in the IA. However, it is worth mentioning that this method has been successfully integrated into a robotic architecture [19].

The closest work to ours is introduced in [14] and an integration in a robotic system is presented in [9]. Like ours, this method is based on a knowledge base coded as an ontology and is independent of the perception modality (i.e. the kind of relation present in the knowledge base) making it domain-independent. However, it only supports entity attributes and not relations between entities.

A common point between the presented methods is that they all consider the linguistic realisation [8] as perfect in the sense that every concept available in their knowledge bases has a word in natural language. As we focus on determining the content using a domain-independent knowledge base, with this work we want to make a first step in taking into account the language in the REG by not assuming that all the concepts in the knowledge base can be used in natural language.

To sum up, we identified a number of key desired features for a REG to be used by a robot in human-robot collaboration contexts and which are presented in Table I: **Domain independent**: it is important to that the REG can be performed on standard and domain-independent knowledge bases which can allow easy extension and use of the available inference engines. **Representation type**: For a dynamic robotic application, the REG must be done on a data type expressive enough to represent a large variety of situations and which can be easily updated through perception and inference. **Use**

[1] Saying "the green pen" implies the presence of another non-green pen.

TABLE I

SUMMARY OF THE FIVE MAJOR FEATURES IDENTIFIED FOR THE MOST REPRESENTATIVE CONTRIBUTIONS IN THE FIELD

	[2]	[3]	[4]	[7]	[16]	[20]	[18]	[14]	Our
Domain independent	No	No	No	No	No	No	Yes	Yes	Yes
Representation type	knowledge base entity	-	attribute-value pairs	REG graph	REG graph	Belief network	Distributed KBs	Ontology	Ontology
Use of types	No	-	Yes	No	No	No	No	No	Yes
Preference ordering	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Referring to other entities	No	No	No	Yes	Yes	No	Yes	No	Yes

of types: The type of an entity is the minimal information to use to refer an entity. Without type, linguistic realisation can not be done. **Preference ordering**: As discussed in early works on REG [4], some attributes are better and faster understood than others. Being able to order relations ensures finding efficient referring expressions. **Referring to other entities**: It is important to be able, when necessary (Fig. 1(f)), to refer to an entity by referring to another one.

III. PROBLEM DEFINITION

In this section we present the chosen knowledge representation as an ontology and the specific extensions targeting the REG problem in a HRI context. We then discuss how the situation in which REG is performed can influence the resolution process. Finally we formally define a solution to a REG problem.

A. Knowledge representation

We have chosen to use an ontology as a knowledge base since it is a largely used representation in many fields and also to ensure for future extensions and inclusion of standardized ontologies and inference engines. Additionally, efforts have been and will be dedicated to create ontologies for robotic applications with the IEEE-SA P1872.2 Standard for Autonomous Robotics (AuR) Ontology.

1) Ontology definition: Let us define a semantic knowledge base K represented as an ontology by $K = \langle \mathbb{A}, \mathbb{T}, \mathbb{R} \rangle$ as defined in [6]. The TBox $\mathbb{T}$ defines a type hierarchy as a finite directed acyclic graph $\mathbb{T} = \langle T, H \rangle$ with T the set classes (types) and H the directed edges representing the inheritance/inclusion links between theses classes, commonly referred as "isA" links (*e.g. (Dog, isA, Animal)*). The RBox $\mathbb{R} = \langle P, Incl, Inv \rangle$ is the roles/properties tuple, with P the set of properties, $Incl$ the finite directed acyclic graph edges of the properties representing the inheritance/inclusion links between the properties and $Inv = \{(p_i, p_j) \in P^2\}$ the set representing the properties inverses (*e.g.* $(isHeldBy, isHolding) \in Inv$). Finally, the ABox $\mathbb{A} = \langle A, C_0, R \rangle$, with A the set of individuals/entities, C_0 the set of direct types of A such as $C_0 = \{(a, t) | a \in A, t \in T\}$ (an instance a can have several direct types) and R the set of relations between entities (i.e. the triplets) such as $R = \{(s, p, o) | (s, o) \in A^2, p \in P\}$ where s is the subject, p the property and o the object. With theses relations we can both represent the attributes of an entity but also represent relations between entities such as spatial relations relating to other entities.

While C_0 contains the direct types of entities, we use C to denote set of direct and inherited types. For instance, an entity *dog23* with a direct *Dog* type $((dog23, Dog) \in C_0)$ would also inherit the *Animal* type $((dog23, Dog), (dog23, Animal) \in C)$ through the type hierarchy of $\mathbb{T}$. When appropriate, the REG will take benefit of this inheritance to refer to entities with a higher level of abstraction.

Note that to fully match the Fokue definition [6], we also need to add the disjunctive, transitive, reflexive and chain relations declarations in $\mathbb{R}$ and disjunctive class declarations in $\mathbb{T}$. Since we will not use them in here we chose to not take them into account in this definition. We also assume that all inference processes have already been made resulting in complete and consistent knowledge base to be used by the REG.

2) Ontology extension for the REG problem: To represent writable/speakable names, we define a class labeling function $\mathcal{L}_t : T \rightarrow str \cup \perp$ where str denotes a set of strings to be used as words in the vocabulary. This can be done using the rdf:label property commonly used in ontologies. We say that a class $t \in T$ is labeled if $\mathcal{L}_t(t) \neq \perp$ and refers to $\mathcal{L}_t(t) \in str$ as the label of t. We further require labels to be unique, i.e., for any two *labeled* classes $t, t' \in T^2$, $t \neq t' \Leftrightarrow \mathcal{L}_t(t) \neq \mathcal{L}_t(t')$. Similarly, we define an entity labeling function $\mathcal{L}_a : A \rightarrow str \cup \perp$, which associates unique labels to a subset of the entities.

We also define a *comprehension cost function* which aims at ordering relations to reflect that some are harder/longer for a human to understand than others. These "preferences" have been identified early in REG studies. Dale and Reiter [4] identified that the use of some properties or types should be preferred to others and defined a *preference ordering* on properties (*e.g.* it is preferable to use the color of an object than its size). This is intended to account for the ease with which a human can include a relation in his mental representation of an object. Moreover, this preference ordering can change from one person to another. Persons subject to mild color vision defect can prefer and be more efficient when presented with size or shape than color properties. The comprehension cost function is defined as $\mathcal{C} : P \rightarrow \mathbb{R}^{+*}$.

3) Ontology in the context of Human-Robot Interaction: Since we are dealing with HRI applications, it is pertinent to maintain a knowledge base per agent in order to implement theory of mind decisional mechanisms. This is provided by the system we use, Ontologenius [15] which is an efficient framework designed for robotic applications and

which allows to manage several ontologies corresponding to the knowledge of the robot and to its estimated knowledge of its human partners.

We define the robot's own knowledge base $K_{RO} = \langle \mathbb{A}_{RO}, \mathbb{T}_{RO}, \mathbb{R}_{RO} \rangle$. The robot maintains, for each agent AG it knows of, an estimation of this agent's knowledge $K_{AG} = \langle \mathbb{A}_{AG}, \mathbb{T}_{AG}, \mathbb{R}_{AG} \rangle$. The robot's knowledge thus encompasses both its own perception of the environment as well as an estimation of the other agent's knowledge. In the rest of this paper, we use only the knowledge base - simply noted K - corresponding to the estimation of the human knowledge for the generation of referring expressions. This is to account for perspective-taking and to ensure that REG will only use the concepts and relations it believes the human understands and is aware of.

B. Contextualization and Restrictions for Situated REG

We are aiming to unambiguously designate, through its relations to other entities, an entity $a_t \in A$ in a knowledge base K. However, the RE is meant to be used in the context of a task and its generation has to take this into account. When a Human-Robot collaborative task concerns object on a table between the human and the robot, the other entities in the room are clearly out of context. In the example of Fig. 1a, the human knows about the object on the table but may also be aware of other pens in the room. Despite this, if the robot has to designate to the human the pen which is on the table, the assertion "take the pen" will not lead to any ambiguous situation because in the context of the task, it is obvious that the robot is currently speaking about the pen which is on the table and which is visible by the human. This is why the problem must be given a **context** $Ctx = (R_{ctx}, C_{ctx})$, a set of relations and direct types that are implicit in the current situation, which will be used to reference a_t, but not included in the generated RE. For the table-top interactions of Fig. 1, the context could be defined as $Ctx = (\{\langle a_t, isOn, Table1 \rangle, \langle a_t, isVisibleBy, Bob \rangle, \langle a_t, isReachableBy, Bob \rangle\}, \emptyset)$ where a_t is the entity to be referenced and *Table1* the identifier of the table where the task is performed. With this context, we restrict the disambiguation to the entities present on the table $Table1$ and visible and reachable by Bob, the human partner.

Finally, in a more general case, some properties might be present in the knowledge base, but cannot be used in the discrimination process. For example, the *hasMesh* property should not be used for a verbal communication with humans. Thus, the problem must be provided with a set of so-called **usable properties** $U \subseteq P$. Because of properties inheritance $Incl$ all the properties inheriting from the ones in U are usable in the problem. Thus, the REG problem is a tuple $\mathcal{P} = \langle a_t, K, Ctx, U \rangle$.

C. Solution: structure and validity criteria

A solution to the REG problem is a set of relations which could be verbalized afterwards. Because some entities are *not* labeled with a unique name (anonymous) and thus cannot be referred to directly, some of the relations might be under-specified. For instance, the sentence "the pen is blue" is under-specified in that "the pen" does not identify a unique entity but any entity with the class "pen". In addition, it might be the case that a unique, anonymous, entity participates in more than one relation, e.g., "the pen is blue and on the table". To keep track of anonymous entities in underspecified relations, we introduce a variable set X, representing the anonymous entities. By convention, variables will be prefixed with a question mark (e.g. $?y \in X$). An underspecified relation is thus a triple $(s, p, o) \in (X \cup A) \times U \times (X \cup A)$, e.g., *(?y, Color, red)* where $?y \in X$ is a variable and $red \in A$ is a labeled entity in the knowledge base.

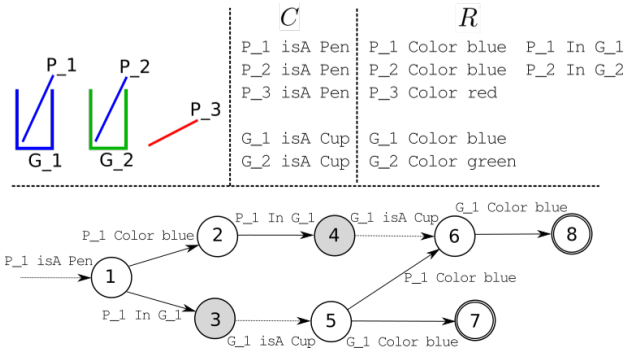

Fig. 2. Disambiguation of a scene with three pens, the two blue ones being in a green and a blue cup respectively. Given is the set of classes C and relations R of the knowledge base and the search space of the algorithm to reference entity P_1. Hashed arrows correspond to typing actions and grayed states do not respect the constraints C1 or C2.

When speaking about anonymous entities, one must know its type to serve as a placeholder in sentences (e.g. "the pen"). Thus, the solution should associate each variable and a type. For simplicity, we chose to represent them also as triplets: $X \times "isA" \times T$ (e.g. *(?y, isA, Pen)*).

Thus, a **reference** E is a set of triplets, each triplet in E being either an under-specified relation in $(X \cup A) \times U \times (X \cup A)$ or a type ascription in $(X \times "isA" \times T)$.

A **valid reference** must respect three constraints:

C1. **Nameability of entities**. Each entity $a \in A$ present in any tuple of E (as first or third component) must have a label: $\mathcal{L}_a(a) \neq \perp$.

C2. **Nameability of the variables**. For each variable $x \in X$ present in any tuple of E (as first or third component) there must also be a unique tuple in E specifying one of its labeled type $((x, "isA", t) \in E$ with $t \in T$ and $\mathcal{L}_t(t) \neq \perp$.

C3. **Correct instantiation of variables**. The reference E is valid if there exists at least one substitution function $f : X \to A$ of the variables in E into entities in A such that the types and relations linking entities in E are still present in T and R once f has been applied. In practice, f transforms the underspecified relations of E into fully specified ones that must appear in the knowledge base.

In the situation of Fig. 2, let the references $E_1 =$

Fig. 3. A scene where disambiguating between the two cups by referring to the pens can be done either by leaving the ambiguity on the pen (R1) or also disambiguating the color of the pen (R2).

$\{(P\_1, Color, blue)\}$, $E_2 = \{(?y, Color, blue)\}$ and $E_3 = \{(?y, isA, Pen), (?y, Color, green)\}$. These are not valid because: (1) since P_1 is not labeled E_1 violates C1, (2) the variable $?y$ has no type in E_2, so it violates C2 and (3) there is no green pen in the knowledge base K, thus the mapping of $?y$ is not possible and E_3 violates C3.

To a REG problem $\mathcal{P} = \langle a_t, K, Ctx, U \rangle$ we now define a **solution** $S = \langle E, x_g \rangle$ composed of E a **valid reference** and x_g a variable designating the target entity a_t in E. Moreover a solution must respect the following requirement:

R1. **Unambiguity of the target entity**. For all the mapping function f respecting C3 for E, we have $f(x_g) = a_t$.

We say a solution is **complete** if it respects the stronger R2:

R2. **Unambiguity of all the entities**. The mapping function f respecting C3 for E is unique.

In Fig. 3, to designate the cup B, the solution {*(?y, isA, Cup), (?z, isA, Pen), (?y, hasIn, ?z)*} and $x_g = ?y$ violates R2, because it is unclear which pen is ?z. However, ?y resolves only to the cup B (here a_t), for every ?z possible, it is thus a valid solution as it respects R1. Another solution {*(?y, isA, Cup), (?z, isA, Pen), (?z, Color, Red), (?y, hasIn, ?z)*} respects R2 and is complete: ?z resolves unambiguously to the red pen, and ?y to the cup B.

Finally, we define an optimal solution $S^* = \langle E^*, x_g \rangle$ as being a solution minimizing $\sum_{r \in E^*} \mathcal{C}(r)$ over the set of all the possible solutions for a REG problem. In case of equal costs, S^* would be the reference with the smallest number of triplets.

IV. Algorithm

A. Formalisation as a search problem

In the REG problem, we define a **state** $\mathbb{s}$ as a set $\mathcal{T} \subseteq R \cup C$ of relation r representing relations present in the referenced knowledge base K. The **initial state** is specified by the user's query through the *context* of the problem.

To find all substitutions defined in III-C (C3), and thus, all the entities which can be bound to the variables in the reference, we use SPARQL queries. SPARQL is a query language allowing to retrieve information from an ontology. From any state $\mathbb{s}$ we can easily construct a SPARQL query, and submit it on the knowledge base to know how many entities can bound to the variables of the request. A state $\mathbb{s}$ is a **target state** if a_t is the only solution to the variable x_g of the SPARQL query created from the state (R1), and possibly all the variables in the SPARQL query have only one assignation (R2).

An **action** $\mathbb{o}$ in the unambiguous reference generation problem consists in the insertion of a new triplet (s, p, o) to the set $\mathcal{T}$ of a state $\mathbb{s}$ resulting in the creation of a new state $\mathbb{s}'$. The inserted relation in a state $\mathbb{s}$ can be a typing relation $(p \equiv isA)$ or a relation which differs between ambiguous entities in $\mathbb{s}$. We define two kinds of difference between ambiguous entities. 1) **Hard difference** (a_i, Δ, a_j) exists when two entities own the same property towards a different entity (i.e $(a_i, p, o_i) \in R \wedge (a_j, p, o_j) \in R | o_i \neq o_j$). 2) **Soft difference** (a_i, δ, a_j) exists when an entity owns a property that is not owned by another ambiguous entity (i.e $(a_1, p, o_i) \in R \wedge (a_j, p, \cdot) \notin R$).

Finally, the **cost** of a state is the sum of the cost of each action leading to this state. If we assume that each action $\mathbb{o}_j$ corresponds to the fact of adding a relation r_j to the state $\mathbb{s}$ with a cost $\mathcal{C}(r_j)$, the cost to $\mathbb{s}$ is $\mathbb{c}_\mathbb{s} = \sum^{\mathcal{T}_\mathbb{s}} \mathcal{C}(r_j)$. As the hard differences respect the **open-world assumption** but the soft differences do not, we propose to encourage the use of hard difference when possible by adding an extra-cost to actions coming from soft differences.

B. Algorithm presentation

Our REG algorithm performs a graph search in the space of states (Alg. 1). From an initial state built from the context of the query, the algorithm generates new states by adding possibly disambiguating relation to the current state. We use an uniform-cost search which is **optimal** and **complete** with positive action costs and a finite number of entities and properties in K. Just like Dijkstra's algorithm, it expands the states in increasing cost order until a solution is discovered or the search space is exhausted.

ToVariable: We globally define a symbol table $\mathcal{S}$ to keep track of the variables assigned to the entities without labels. When needed, this symbol table assigns a unique variable identifier to an entity a. We note $\mathcal{S}^{-1}$ the inverse table, allowing to retrieve the entity from an existing variable.

ToQuery: Performs a direct translation of a set of triplets into a SPARQL query. For each triplet, its subject and object are given a string representation with the function $v(a) : A \times T \mapsto str$:

$$v(a) = \begin{cases} str(a), & \text{if } \mathcal{L}_a(a) \neq \bot \\ \mathcal{S}(a), & \text{otherwise} \end{cases}$$

SparqlResult: The function that takes a SPARQL query as input and returns a match table $\mathcal{M}$ in the way that $\mathcal{M}(x)$ is the set of entities matching the variable x in the given query.

GoalTest: The test succeeds for the degraded solution if the target entity a_t is the only solution to the variable $v(a_t)$ in $\mathcal{M}$. For the complete solution, the test succeed if all the variables of $\mathcal{M}$ have exactly one solution.

Actions: At each step, we consider two kinds of possible actions. The TypingActions function (Alg. 2) consisting in the addition of an inheritance relation if at least one entity has no label and no inheritance relation in $\mathcal{T}$. Otherwise, the **hard difference actions** (Alg. 3) and the **soft differences** (Alg. 3 with the δ operator at line 6)

Algorithm 1 Uniform-Cost Search for unambiguous reference generation

function DISAMBIGUATE($problem$)
  $state \leftarrow$ CREATE-INITIAL-STATE($problem$.context)
  $frontier \leftarrow$ a priority queue o states ordered by their *cost*, with $state$ as only element
  $explored \leftarrow$ an empty set
  loop
    if EMPTY($frontier$) **then return** failure **end if**
    $state \leftarrow$ POP($frontier$)
    if GOALTEST($problem$, TOQUERY($state$)) **then return** SOLUTION($state$) **end if**
    add $state$ to $explored$
    for all $action$ in ACTIONS($state$) **do**
      $child \leftarrow state \cup \{action\}$
      if $child$ is not in $explored$ or $frontier$ **then** $frontier \leftarrow$ INSERT($child$, $frontier$) **end if**

add relations that differ as hard and soft differences between ambiguous entity for each variable in $\mathcal{M}$.

In the TYPINGACTIONS function, the function USABLECLASSES returns the most specific *labeled* classes of an entity a, i.e., the set of classes $t \in T$ such that $(a, t) \in C$, t is labeled and there are no labeled subclasses of t.

Algorithm 2 Typing actions pseudocode

function TYPINGACTIONS($state$)
  for all (s, p, o) **in** $state$ **do**
    if $\nexists x$ s.t. $(s$,"isA",$x) \in$ *state* $\wedge\ \mathcal{L}_a(s) = \perp$ **then**
      return $\{\ (s$,"isA",$t)\ |\ t \in$ USABLECLASSES(s)$\ \}$
  return OK ▷ every anonymous entity is typed

This strategy differs from the one of [4] that prefers the least specific types (so called basic-level classes). However, in domain-independent knowledge bases such as ours their scheme could often resolve to "Object" or "Thing" which can lead to confusion. Furthermore, by being conservative in our estimation of the receiver's knowledge base, we can guarantee that the labels of the considered classes are known to the human partner. Finally, using the most specific classes might reduce the ambiguities, and thus the branching factor early in the search, without impacting completeness.

The TYPINGACTIONS function stops at the first entity which has no label nor type. This specificity reduces the branching factor while ensuring that each entity has a label or at least a type. Since typing actions are the first tested in the ACTIONS function, all entities not tested during a first execution will be performed during the next ones.

The Δ (resp. δ) operator returns all the relations that are hard differences (resp. soft) between two entities as defined in IV-A. In the difference actions algorithm, an action can be added only once and must not be present in the current state to avoid redundancy. The inverse relation to the one added by the action is also retrived from the Inv set defined in the knowledge base and checked if not present in the current state and in the current actions set, again to avoid redundancy. In the example of Fig. 2 the relation $(P\_1, isIn, G\_2)$ will be redundant if the relation $(G\_2, hasIn, P\_2)$ has been already used in the current state.

Algorithm 3 Hard difference actions pseudocode

1: **function** HARDDIFFERENCEACTIONS($state$)
2: $actions \leftarrow$ an empty set of actions
3: $matches \leftarrow$ SPARQLRESULT(TOQUERY($state$))
4: **for all** (x, a) **in** matches **do**
5: **if** $a \neq \mathcal{S}^{-1}(x)$ **then**
6: **for all** $r = (\mathcal{S}^{-1}(x), p, o)$ **in** $\mathcal{S}^{-1}(x)\,\Delta\,a$ **do**
7: r_{inv} = $(o, Inv(p), \mathcal{S}^{-1}(x))$
8: **if** $r \notin sate \wedge r_{inv} \notin state \wedge p \in U$ **then**
9: *actions* $\leftarrow$ *actions* $\cup\ \{r\}$
  return $actions$

V. RESULTS

We present hereafter the solutions given by our algorithm to the illustrative examples. Then we provide results involving a large scale knowledge base describing a full apartment in terms of time-execution, solution length and composition. Finally, we provide comparative performance measures with two state-of-the-art methods on their own domains.

A. Solutions analysis

In order to familiarize with solutions, we propose to present some of them. For every presented solution, the variable denoting the entity to refer to will be x_g =?0. The first setup is for illustration purposes, and operates on the static knowledge base illustrated in Fig. 2. Since this setup is really small, the context is always empty, all the relations are usable and no entity is labeled. We only tested with two interesting entities since the others present similar characteristics. The solutions for P_1 and G_1 are respectively {*(?0, isA, Pen), (?0, isIn, ?1), (?1, isA, Cup), (?1, Color, blue)*} and {*(?0, isA, Cup), (?0, Color, blue)*}, which can be read respectively as "the pen in the blue cup" and "the blue cup". These two solutions are R2 (allowing to read "the" and not "a" in the verbalization), as ?0 and ?1 bind to only one entity. Here, we see how referring to another entity lead to interesting solutions.

In order to give the reader a sense of how the context is useful as defined in the problem, we propose to come back to the Fig. 1. In a knowledge base describing Fig. 1(b), with a labeled entity *Bob*, representing the human, giving a

empty context to the problem would lead to the solution {*(?0, isA, Pen), (?0, isReachableBy, Bob)*}, which would read as "The pen reachable by Bob". Whereas, if the robot wants the human to give it the pen, the reachablity of the pen is obvious. So the context would become: {*(pen0, isReachableBy, Bob)*}, the ensuing solution would be {*?0, isA, Pen)*}, simply verbalizable as "the pen", as taking into account the given context resolve the ambiguity.

B. Scaling up

To assess the relevance of our approach, we created a larger, realistically-sized, knowledge base (101 entities, 36 classes, 40 properties and 497 relations), describing an apartment with three rooms including several furniture (tables, shelves) and objects (cups, boxes) linked through geometrical relations (atLeftOf, onTopOf) and attributes (color, weight). We ran our algorithm over all the 77 entities inheriting from the "Object" class, representing physical entities.

As this algorithm must be used in a human robot interaction application, we want it not to spoil the interaction when the robot is computing an explanation. In this setup, 100% of the entities have been referred in under 4.33ms that is well bellow 100ms which is the maximum system response time for the user to get a feeling of instantaneity [12]. More over, 50% are referred under 357μs and 75% under 772μs. On average, 10.6 nodes are explored to refer to an object with an average of 67.35μs/node explored.

Over the 77 entities, 32 (41.56%) are referred with 2 or less relation meaning that only the type of the entity and one relation is needed to refer to them. We can also note that 25 entities (32.46%) are referred using 4 or more relations with a maximum of 6 for one of them. Finally, 49.4% need to be referred by referring to another entity and two of them need to be referred by referring to two other entities. This mean that 49.4% of the entities can not be referred using approaches like [14] or [4]. For this reason we will not compare more of these two works.

These results over a large scale knowledge base highlight the need to be able to refer to an entity through the use of relation linking it with other entities. They also shows that the use of the type of an entity is often sufficient with the use of only one attribute. With this experiment we also demonstrate that our algorithm is suitable for a use with a realistic large scale knowledge base.

C. Comparisons with other state-of-the-art algorithms

1) Longest First: The Longest First (LF)[2] algorithm [16] has been tested on the GRE3D3 Corpus composed of 20 scenes with three objects with different spatial relations relative to one another (onTopOf, atLeftOf). Each object can be referenced by its color, its size (large or small) and its type (cube or ball). The target referent is marked by an arrow and is always in a direct adjacency relation (onTopOf or inFrontOf). Among the 20 scenes, 8 target objects can be referenced without any ambiguity using only their type, 7 can be referenced using only their type in addition to an attribute (color or size) and the other five can be referenced using their types and both color and size attribute. This means that spatial relations are never necessary to reference the target object. We perform the comparison on the 19$^{\text{th}}$ case which consists of a small green cube on a large green cube and a small blue cube to the right of the green cubes. We chose this case with only cubes because the LF algorithm does not consider the types when generating the RE and adds them only as a post-process. The other cases requiring only the type are resolved in less than 100μs and those requiring the type and an attribute are resolved in less than 250μs with our algorithm.

Because of their objective of obtaining an over-specification of the RE, their results are strongly impacted by the maximum length parameter. By setting it to 4 as recommended, we get the result which we can read as *"The small green cube on top of a cube"* in 311ms. By setting the maximum length to 3 we obtain the shortest admissible result which can be read as *"The small green cube"* in 109ms. This last result is the one given by our algorithm in just 0.87ms.

We see here that the results given by the LF algorithm largely depend on the maximum length parameter. This parameter also has a significant impact on the execution time. Besides, in the realistic scenario presented previously, 13% of the entity need a reference expression length greater than 4. Thus, even if the over-specification is the goal of the LF approach, it can hardly scale-up. Moreover, for a maximum length fixed at the optimal length, the two approaches give identical results.

2) Graph Based Algorithm: A speed up of the original GBA [16] is presented in [10]. It aims at extracting, from a dedicated entities relations graph G, the lowest cost subgraph which is graph isomorphic to one and only one subgraph in G containing the entity to refer to. Their approach is evaluated on a corpus containing multiple tabletop scenes [11], presenting numerous cubes of different colors.

We generated the graph (relations and costs) used for the scene 1, converted it into an ontology, and ran our algorithm on it. This scene contains 15 cubes, GBA algorithm and ours are able to find a solution for the same 10 of them. In all the 10 cases, as we used the same costs, both algorithms returned the same solution (with the types of used entities added in our approach). For the other 5 cases, the two algorithms detect the absence of a solution in a few milliseconds.

On all the 10 cases with a solution, our approach performs faster than theirs (29.4 times faster in average). We can note that the speed increase is more important in cases where there are many solutions (under 4 times faster on 50% of the cases, but more than 50 times faster for 25% of the cases, up to 130 times faster). Indeed, the GBA approach uses a branch and bound algorithm where the search graph is bounded if the branch exceeds the cost of the current best found solution. Thus, it can explore a large part of the graph if the optimal solution is not found early in the search. Whereas our approach uses an uniform cost search algorithm, ensuring the first found solution is optimal. Moreover, we think that

[2]http://www.m-mitchell.com/code

on cases where the knowledge base contains entities with different types, our approach should work faster, since we prioritize the use of the type. We were not able to test this, as we could not manage to run their approach on other data than their own corpus.

VI. Integration

Our ontology-based REG method has been integrated on a PR2 robotic platform and used in a tabletop scenario. The objects on the tables are detected with the ROBOSHERLOCK[3] [1] perception system. It provides the position (not used to extract relations), the shape ("circular" or "rectangular"), the color and the size of the objects ("large", "medium" or "small"). Since the types of objects is not determined by the system, all the objects were set with the labeled type "Object". This allows us to challenge our method with situations where the robot is not able to use high-level concepts and where various ambiguities will be raised.

The knowledge base is managed using the Ontologenius[4] system [15]. It uses a custom internal structure to store and manipulate assertions as triplets, and offers reasoning capabilities in the form of plugins. Ontologenius provides a low level API allowing to manipulate the knowledge base as a classical data structure in addition to a SPARQL interface. The ontology is dynamically fed to keep it up to date on the basis of a simple situation assessment consisting only of filtering and object tracking.

A simple linguistic realisation has been made, taking as input a SPARQL query and generating an English sentence as output. For example, it transforms the query "?0 isA Cup, ?0 isOn ?1, ?1 isA Table, ?1 hasColor black" into "*the cup on the black table*". It is an ad hoc implementation based on a simple grammar and the labels present in the ontology.

The task [5] involves six objects on a table. The entity to reference is obj_4 (a white mug). The robot generates the solution *"The white circular object"* since there are other non-white circular objects and other white non-circular objects. Then, a human adds a new object which is a white and circular milk bottle (obj_5). When the robot is asked to describe the white cup, it generates the sentence *"The white small circular object"*. With this simple task, we show that our REG algorithm can be used within a robotic architecture, can deal with dynamic environment and can adapt its explanation to the current situation.

VII. Conclusion and future work

In this paper, we formalized the REG problem adapted to hierarchical knowledge bases such as ontologies. We proposed an algorithm solving this problem by searching for a set of expressions using entities, usable attributes and relations to other entities, based on properties costs as an optimisation criterion. The flexibility and efficiency of our method are illustrated by *(i)* comparing its performances with other state-of-the-art approaches, *(ii)* showing its efficiency on a realistically-sized ontology, and *(iii)* integrating it in an operating robotic architecture.

We aim at integrating this method within a cost-based human-aware task planner in order to equip it with an informed criterion to decide if and when to use a communication action including reference to an entity.

[3] http://robosherlock.org/

[4] https://sarthou.github.io/ontologenius/

[5] Commented video available at: https://frama.link/TWU_VE0o

Acknowledgement

Authors want to thank Amandine Mayima for her work on the integrated scenario. This work has been funded by the French Agence Nationale de la Recherche JointAction4HRI project ANR-16-CE33-0017 and the Artificial and Natural Intelligence Toulouse Institute (ANITI).

References

[1] M. Beetz, F. Bálint-Benczédi, N. Blodow, D. Nyga, T. Wiedemeyer, and Z.-C. Márton, "Robosherlock: Unstructured information processing for robot perception," in *IEEE ICRA*, 2015.

[2] R. Dale, "Cooking up referring expressions," in *27th Annual Meeting of the association for Computational Linguistics*, 1989.

[3] ——, *Generating referring expressions: Constructing descriptions in a domain of objects and processes.* The MIT Press, 1992.

[4] R. Dale and E. Reiter, "Computational interpretations of the Gricean maxims in the generation of referring expressions," *Cognitive Science*, vol. 19, no. 2, 1995.

[5] N. FitzGerald, Y. Artzi, and L. Zettlemoyer, "Learning distributions over logical forms for referring expression generation," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2013.

[6] A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas, "The Summary Abox: Cutting Ontologies Down to Size," in *The Semantic Web - ISWC*. Springer Berlin Heidelberg, 2006, vol. 4273.

[7] E. Krahmer, S. v. Erk, and A. Verleg, "Graph-based generation of referring expressions," *Computational Linguistics*, vol. 29, no. 1, 2003.

[8] E. Krahmer and K. van Deemter, "Computational generation of referring expressions: A survey," *Computational Linguistics*, vol. 38, no. 1, 2012.

[9] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, "Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction," *Int. Journal of Social Robotics*, vol. 4, no. 2, 2012.

[10] S. Li, "Automatically evaluating and generating clear robot explanations," Master's thesis, Carnegie Mellon Uni., Pittsburgh, PA, 2017.

[11] S. Li, R. Scalise, H. Admoni, S. Rosenthal, and S. S. Srinivasa, "Spatial references and perspective in natural language instructions for collaborative manipulation," in *IEEE RO-MAN*, 2016.

[12] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, fall joint computer conference, part I.* Association for Computing Machinery, 1968.

[13] E. Reiter and R. Dale, *Building natural language generation systems*. Cambridge University Press, 2000.

[14] R. Ros, S. Lemaignan, E. A. Sisbot, R. Alami, J. Steinwender, K. Hamann, and F. Warneken, "Which one? Grounding the referent based on efficient human-robot interaction," in *IEEE RO-MAN*, 2010.

[15] G. Sarthou, A. Clodic, and R. Alami, "Ontologenius : A long-term semantic memory for robotic agents," in *IEEE RO-MAN*, 2019.

[16] J. Viethen, M. Mitchell, and E. Krahmer, "Graphs and spatial relations in the generation of referring expressions," in *Proceedings of the 14th European Workshop on Natural Language Generation*, 2013.

[17] T. Williams and M. Scheutz, "A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[18] ——, "Referring expression generation under uncertainty: Algorithm and evaluation framework," in *Proc. of the 10th International Conference on Natural Language Generation*, 2017.

[19] T. Williams, F. Yazdani, P. Suresh, M. Scheutz, and M. Beetz, "Dempster-Shafer theoretic resolution of referential ambiguity," *Autonomous Robots*, vol. 43, no. 2, 2019.

[20] Y. Yamakata, T. Kawahara, H. G. Okuno, and M. Minoh, "Belief Network based Disambiguation of Object Reference in Spoken Dialogue System," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 19, 2004.