Spatial knowledge in planning language

Lamia Belouaer, Maroua Bouzid, Abdel-Illah Mouaddib

SPATIAL KNOWLEDGE IN PLANNING LANGUAGE

Lamia Belouaer, Maroua Bouzid, Abdel-Illah Mouaddib
GREYC - CNRS (UMR0672),
Université de Caen, Basse-Normandie, ENSICAEN
Boulevard du Maréchal Juin 14032 Caen CEDEX, France
{*first_name.last_name*}*@info.unicaen.fr*

Keywords: Spatial Representation and Reasoning, Spatial Ontolgy, Fuzzy Information, Rules, Task and Path Planning.

Abstract: This paper describes the integration of spatial knowledge in a planning problem. For example, in the case of evacuation problem of a building during a fire, we must know the fastest paths (the shortest, least congested, ...). Such applications require spatial knowledge to perform spatial planning. This planning taste becomes more complex when knowledge are shared by many actors. Indeed, our interest concerns collaborative work between agents, in particular the case of human-robot interaction. In such contexts, considering the space information in planning, we use a spatial ontology called *SpaceOntology* which handles different representations and abstraction levels of spatial information. We integrate this ontology in the planning by defining a formal language planning: Spatial-PDDL. Spatial-PDDL combines PDDL concepts with this ontology. Also, we distinguish between three types of actions: non spatial, spatial and navigation actions.

1 INTRODUCTION

In this paper, we focus on the spatial knowledge management in planning problem where the tasks consider spatial and non-spatial representations. Such problems could be met in human-robot interaction. Most typical scenarios include a human instructing a robot to perform some actions on some objects in the environment. To achieve this task, the human and the robot establish a similar description of the concerned environment. This requires not only that the scene descriptions created by the robot and the human match but also needs a system which can successfully mediate between these descriptions. However, the space representation from human's and robot's points of view is not the same. Indeed, the robot's spatial representation is numeric and the human's spatial representation uses natural language. The natural language employs incomplete and fuzzy information about space (for example, the book is near the bottle). Thus, we propose to find a representation for planning problems that can be used by robots to communicate with humans, analyze scenes and build maps.

To do this, we use an ontology called *SpaceOntology* (Belouaer et al., 2010). It permits a knowledge representation for modeling space, spatial relations and fuzzy information. This ontology allows us: a complex spatial description, an easy management of various spatial aspects, an easily extensible structure and to infer spatial information. It provides a structured knowledge about the environment to explore in planning process.

Planning process in artificial intelligence is decision making about the actions to be taken. As input, this process takes a planning problem (initial state and goal state) returns a plan (a sequence of ordered actions) to solve this problem (achieve the goal state from the initial state). To express spatial knowledge in planning process, we need a formal language. We use language PDDL (Ghallab et al., 1998) (Planning Domain Definition Language).

We propose a planner incorporating spatial representation and reasoning by integrating *SpaceOntology*. *SpaceOntology* in the planning, is not used for verification or control, but as information needed for planning. We adapt the PDDL actions and concepts definition to express spatial information if it is necessary. In other words, we extend PDDL to spatial knowledge: Spatial-PDDL.

The rest of paper is organized as follows. Sec-

tion 2 gives an overview about different approaches to solve planning problems taking spatial knowledge into account. Section 3 describes the spatial representation and reasoning. Section 4 describes the planning process with spatial knowledge and its impact on complexity. Section 5 presents some results. Finally, we conclude in Section 6.

2 RELATED WORK

Different techniques for spatial representation and reasoning have been proposed. Most are based on constraints, logical and algebraic approaches (Balbiani et al., 1999). However, these approaches can not manage numeric, topological and imprecise knowledge at the same time. In planning, we need to combine these knowledge. Thus, we must use an unified framework to cover large classes of problems and to give rise to instantiations adapted to each particular application. Ontologies appear as an appropriate tool toward this aim.

Spatial ontologies can be found in some fields such as Virtual Reality (Dasiopoulou et al., 2005), Robotics (Dominey et al., 2004), etc. All these ontologies are focused on the representation of spatial concepts according to the application domains. A major weakness of usual ontological technologies is their inability to represent and to reason with imprecision. An interesting work presented in (Hudelot et al., 2003) overcomes this limit.

The concept of exploiting ontology to represent spatial knowledge in planning has been explored. (Bouguerra et al., 2008) describes a new approach for monitoring the execution of plans where spatial knowledge is defined by an ontology called LOOM. This ontology permits a region classification and verifies the plan execution. However, it does not describe spatial relations and it is not used in planning.

To express spatial knowledge in planning process, we use PDDL (Ghallab et al., 1998). The first version of PDDL is extended to consider several aspects in a planning problem definition that improve the quality of the solution, such as the temporal dimension (Fox and Long, 2003). Despite the importance of the spatial dimension in planning problem, PDDL with spatial definition is not yet integrated.

Our main contribution is the extension of PDDL to spatial knowledge: Spatial-PDDL. To do this, we use *SpaceOntology* that considers several spatial knowledge aspects; the space hierarchy, spatial relations (numeric, topological and fuzzy).

3 SPATIAL REPRESENTATION AND REASONING

In the following, we present *SpaceOntology* (Belouaer et al., 2010), a framework for spatial representation and reasoning.

3.1 Spatial Representation

We opted for OWL[1] DL[2] (Baader, 2003) as a formal language[3]. An important characteristic of DL is its expressivity and its capability of inferring knowledge from the known information.

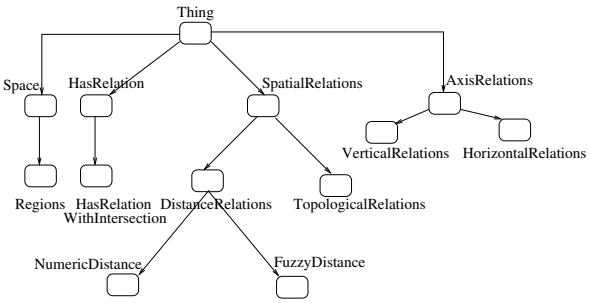

Figure 1: Graphical representation of *SpaceOntology*.

A spatial entity is described by two concepts `Regions` and `Space`. The concept `Space` (`Space`$\sqsubseteq$ `Thing`) is used to model space: the name of places, the hierarchical level, the boundaries definition, etc. It represents a global environment (a country, a city, a building, ...). The concept `Regions` (`Regions`$\sqsubseteq$ `Space`) is a sub-space included in a given space. A region is itself considered as a space that can be decomposed into different sub-regions. From a geometric point of view, a region *rg* is defined by the smallest rectangle *rect* corresponding to its axis-aligned bounding rectangles. These concepts include some attributes. The hierarchical space organization has an impact on the distance evaluation. Indeed, the distance of 10 meters in a city is considered as *close*, however in a corridor this distance is considered as *far*. Each region has the attributes `alpha` and `beta` depending on the scale of its level. These two parameters evaluate the distance between two spatial objects (Belouaer et al., 2010) (section 3.2.3).

A spatial relation is defined by the concept `SpatialRelations` (`SpatialRelations`$\sqsubseteq$ `Thing`) which includes `TopologicalRelations` and `DistanceRelations`. A spatial relation is a concept on its own. A Topological relation

[1] Web Ontology Language

[2] Description Logics

[3] All notations used here are defined in the DL.

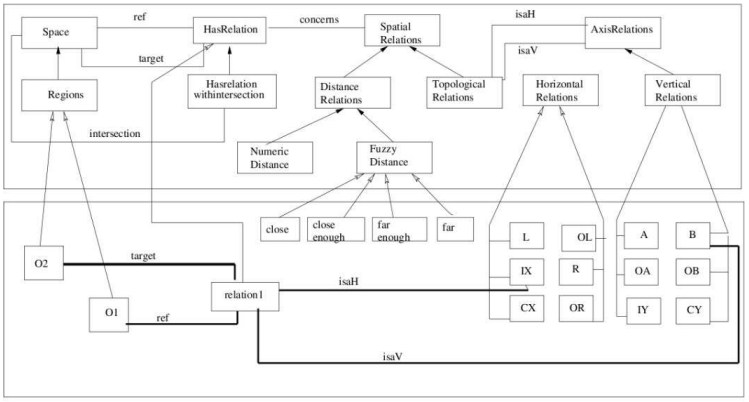

Figure 2: Graphical illustration of *SpaceOntology* to describe structures in specific domains.

(TopologicalRelations) is an ABLR relation (Laborie et al., 2006). This relation is a couple $\langle r_X, r_Y \rangle$, where: $r_X \in \{L, O_L, C_x, I_x, O_R, R\}$ and $r_Y \in \{A, O_A, C_y, I_y, O_B, B\}$. A Distance relation (DistanceRelations) can be an Euclidean distance (NumericDistance) between the symmetry points of two rectangles representing two regions (r referent object and ε target object) noted $d(\varepsilon, r)$ (in $\mathbb{R}^+$) or fuzzy distance (FuzzyDistance) evaluating a distance between the symmetry points of two rectangles representing two regions. A fuzzy distance is defined by four linguistic variables: close to, close enough, far enough and far.

To define a spatial relation between spatial entities, we use the concept HasRelation. This concept links with a spatial relation between a reference region and one or more target regions (Example 1).

There are several relations (or predicates) to link these concepts to provide a spatial semantic. For example to express a hierarchical relation between two regions, we consider the links:

- consistsOf:

$$\texttt{Space} \sqsubseteq \texttt{Thing} \sqcap \exists\ \texttt{consistsOf.Regions} \sqcap \geqslant 1\ \texttt{consistsOf}$$

- isPartOf:

$$\texttt{Regions} \sqsubseteq \texttt{Space} \sqcap \exists\ \texttt{isPartOf.Space} \sqcap \geqslant 1\ \texttt{IsPartOf}$$

Example 1 *Let us consider a part of a floor with two offices ($O1$, $O2$) and a hall ($H1$) (Fig. 3).*

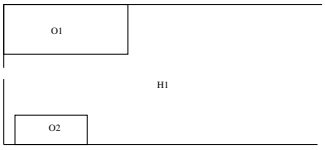

Figure 3: Partial Map.

Figure 2 models[4] the space domain defined in Figure 3 by using *SpaceOntology*. We consider that each spatial entity is a Regions concept. Then, spatial relation between these different spatial entities are described by combining SpatialRelations and HasRelation.

In Example 1, the relation1 is an instance of the concept HasRelation describing the topological relation between offices $O2$ and $O1$:

$$\textit{relation1}:\texttt{HasRelation} \sqcap \exists\ \texttt{concernsSpatialRelation}.\textit{inside\_below} \sqcap \exists\ \texttt{hasReferent}.O1 \sqcap \exists\ \texttt{hasTarget}.O2$$

The relation *inside_below* is defined as follow:

$$\textit{inside\_below}:\texttt{TopologicalRelations} \sqcap \exists\ \texttt{isAnHorizontalRelation}.I_x \sqcap \exists\ \texttt{isAVerticalRelation}.B$$

3.2 Spatial Reasoning

The reasoning methods infer spatial information in order to enrich the description of the environment and/or to manage the lack of information.

3.2.1 Spatial Hierarchical Reasoning

To express a hierarchical relation between two regions (for example, a region includes another region, or a region is in another one), we use the predicates consistsOf and isPartOf. The predicate isPartOf is the inverse of the relation consistsOf. These relations are transitive. The transitivity rules for the predicate consistsOf and isPartOf are:

$$\texttt{consistsOf(r\_1,r\_2)} \sqcap \texttt{consistsOf(r\_2,r\_3)} \longrightarrow \texttt{consistsOf(r\_1,r\_3)}$$

[4]For more readability, we omit certain classes and attributes.

```
isPartOf(r_1,r_2) ⊓ isPartOf(r_2,r_3)
        ⟶ isPartOf(r_1,r_3)
```

Example 2 *The building B6 consists of five floors. We consider the forth floor E4_B6. Also, the office O43_B6 is in the floor E4_B6.*

```
<Regions rdf:ID="B6">
 <consistsOf><Regions rdf:ID="E4_B6"/></consistsOf>
</Regions>
<Regions rdf:ID="O43_B6">
 <isPartOf rdf:resource="#E4_B6"/>
</Regions>
```

Because the predicate `consistsOf` (`isPartOf`) is transitive and the predicate `isPartOf` is the inverse of the relation `consistsOf`, we can deduce from *SpaceOntology* that the office *O*43_*B*6 is part of *B*6:

```
<Regions rdf:ID="B6">
 <consistsOf>
  <Regions rdf:ID="E4_B6"/>
  <Regions rdf:ID="O43_B6"/>
 </consistsOf>
</Regions>
```

To consider a space as a whole with the different abstraction levels is not always necessary and makes spatial reasoning very expensive. The hierarchical organization can simplify this reasoning, depending on abstraction level. To do so, we exploit the two predicates: `consistsOf` and `isPartOf` in order to define the function *hierarchical interest zone* denoted $Z^l(r)$. This function $Z^l(r)$ defines an *interest zone* (An *interest zone* defines a region in a considered space to be reached and/or localized) of a region *r* in the hierarchical level *l*. To compute this function we query *SpaceOntology* (details in Section 4.2.1).

3.2.2 Spatial Reasoning Based on Topological Relations

In order to enrich *SpaceOntology* with new knowledge, we consider two different aggregations of spatial relations: combination $\oplus$ and composition $\otimes$.

The combination mechanism The combination concerns topological relations. As mentioned, a topological relation is a couple $\langle r_X, r_Y \rangle$. This mechanism consists of combining horizontal relation r_X with vertical one r_Y ($r_X \oplus r_Y$). This mechanism is defined by a set of rule proposed in *SpaceOntology*. Rules definition for the combination mechanism is used in order to define the 36 topological relations of ABLR (*Neighborhood graph* (Laborie et al., 2006)), dynamically. These relations express topological relations (overlaps, touch, ...) and orientation relations. Indeed, by this mechanism we can represent the 8 cardinal relations (left, above, above left, ...) and other directional relations as at the same level, between, etc. Figure 4 presents a rule which allows the combination of the horizontal relation $L(Left)$ and the vertical relation $A(Above)$.

```
isATopologicalRelation(?t_relation)
⊓ isAVerticalRelation(?t_relation,"A")
⊓ isAnHorizontalRelation(?t_relation,"L")
⟶ is(?t_relation,"left_above")
```

Figure 4: Rule definition for `left_above`.

The composition mechanism The composition mechanism infers new spatial information. Let us consider three regions defined by three rectangles *rect*1, *rect*2 and *rect*3 (Fig. 5).

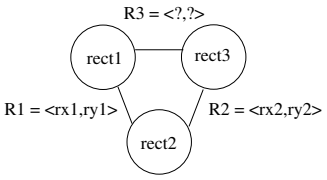

Figure 5: Spatial Relation Network.

The idea is to infer which relation *R*3 links between *rect*1 and *rect*3 where *R*1 is a relation between *rect*1 and *rect*2 and *R*2 is a relation between *rect*2 and *rect*3. The topological relation R_3 is defined by $R3 = R1 \otimes R2$, then

$$\langle rx3, ry3 \rangle = \langle rx1, ry1 \rangle \otimes \langle rx2, ry2 \rangle \quad (1)$$
$$= \langle rx1 \otimes rx2, ry1 \otimes ry2 \rangle \quad (2)$$

The composition results of $(rx1 \otimes rx2)$ and $(ry1 \otimes ry2)$ are given by the composition table of rectangle algebra by preserving the directionality property as defined in ABLR. Formally, we define in *SpaceOntolgy* a set of rules reflecting the composition table. The composition mechanism returns a disjunction of possible relations between two rectangles. However, not all inferred relations are valid. To reduce the number of possible relations between two rectangles, by keeping only the valid relations, we proceed as follows. First, we select the necessary knowledge for the inference and we formalize it as a constraint network called spatial relation network (Fig. 5). Then, we apply the required rules, and we complete this network. Thus, a network is defined by :

- *V* a set of nodes representing rectangles describing regions,
- C a set of constraints representing topological relations. An arc is labeled by a topological relation (constraint) between two rectangles connected to this arc. Arcs labeled with disjunctions are arcs that are inferred by composition.

The idea is to remove all spatial relations does not ensure the consistency of the network. Thus, we aim to establish arc consistency in this network. We produce an equivalent network to the first one which checking the arc consistency property (Def. 1).

Definition 1 *A network is arc consistent if all its arcs are arc consistent*

Thus, the network produced is arc-consistent, *i.e.* it does not contain topological relations does not ensure the arc consistency of the network. Practically, for the rules application, we use the Pellet inference engine (Sirin et al., 2007), and for the verification of the arc consistency, we rely on the MAC algorithm (Prosser, 1995).

Example 3 *Let us consider the following problem, a robot is in the office o2. Its mission is to go to the office o3 . However, it has no spatial information enabling it to define its trajectory between o2 and o3.*

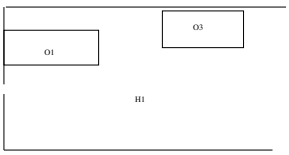

Figure 6: Spatial Relation Between $o1$ and $o3$.

We have a set of spatial information, depicted in Figure 6: the office $o3$ is right and is overlaps above the office $o1$: $o3\langle R, O_A\rangle o1$. Also, in Figure 3, the office $o2$ is inside and is below of the office $o1$: $o2\langle I_x, B\rangle o1$.

Our goal is to found the spatial relation between the offices $o3$ and $o2$. First, we apply a symmetry rule, then, $o1\langle L, O_B\rangle o3$ is the inverse of $o3\langle R, O_A\rangle o1$. Second, we apply composition rule between $o2\langle I_x, B\rangle o1$ and $o1\langle L, O_B\rangle o3$ (Fig. 7).

```
isATopologicalRelation(?t_relation1)
⊓ isAnHorizontalRelation(?t_relation1,"I_X")
⊓ isAVerticalRelation(?t_relation1,"B")
⊓ isATopologicalRelation(?t_relation2)
⊓ isAnHorizontalRelation(?t_relation2,"L")
⊓ isAVerticalRelation(?t_relation2,"O_B")
⟶ [isATopologicalRelation(?t_relation3)
⊓ isAnHorizontalRelation(?t_relation3,"L")
⊓ isAVerticalRelation(?t_relation3,"B")]
```

Figure 7: Rule definition for `left_below` with composition.

Thus, from this information we can plan our trajectory to reach the office $o3$ without considering the entire second floor but only the northeastern part of the floor from the office $o2$. The information $o2\langle L, B\rangle o3$ is added in *SpaceOntology*.

3.2.3 Spatial Fuzzy Reasoning

In this work, the fuzzy representation concerns only distance relations. Indeed, we use linguistic variables to evaluate a distance (close, close enough, far enough and far). We aim to give a numeric representation to each variable and vice versa. *SpaceOntology* allows this translation, by defining some rules based on functions $N_{\alpha,\beta}$ and $F_{\alpha,\beta}$ (Schockaert, 2008). The function $N_{(\alpha,\beta)}(p,q)$ (resp. $F_{(\alpha,\beta)}(p,q)$) measures the degree with which p is close to q (resp. the degree with which p is far from q) ($\alpha, \beta > 0$).

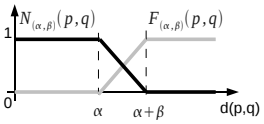

Figure 9: Distance evaluation. (1) if $d \in [0,\alpha]$, then d is close , (2) if $d \in]\alpha, \alpha + \frac{\beta}{2}]$ then d is close enough, (3) if $d \in]\alpha + \frac{\beta}{2}, \alpha + \beta]$ then d is far enough, (4) if $d \in]\alpha + \beta, +\infty[$ then d is far.

Figure 9 illustrates the translation of a distance from a numeric definition to symbolic definition and vice versa. The parameters α and β allow us to evaluate the numeric distance. These parameters depend on the current region. Figure 8 is a rule allowing us to evaluate a distance relation.

The rule *Def_close_relation* (fig. 8) translates a numeric distance (`NumericDistance`) (*distanceValue (?relation)*) in symbolic evaluation (`FuzzyDistance`) given by the instance *close* which indicates the close relation. This rule is apply for distance (*distanceValue (?relation)*) between 0 and α.

4 *SPACEONTOLOGY* BASED PLANNING

Now, we explore the potential uses of spatial representation and reasoning offered by *SpaceOntology* in planning. Our goal is to propose a planner incorporating spatial representation and reasoning: *Spatial Planner* (Fig. 10).

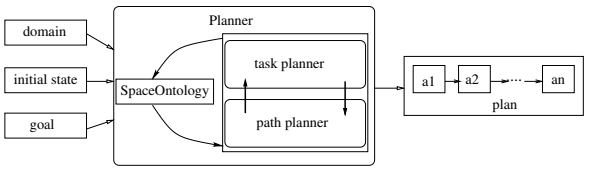

Figure 10: Spatial Planner

As input, the spatial planner (Fig. 10) takes a *planning problem* defined by:

```
translate(?link,?t_link) ⊓ translate(?relation, ?t_relation) ⊓ concerns(?link,?relation)
⊓ hasTarget(?link,?tgt) ⊓ hasReferent(?link,?ref) ⊓ isANumericDistance(?relation)
⊓ isPartOf(?ref,?region) ⊓ isPartOf(?tgt,?region)
⊓ swrlb:greaterThan(distanceValue(?relation),0)
⊓ swrlb:lessThanOrEqual(distanceValue(?relation),alpha(?region))
⟶
[concerns(?t_link,?t_relation) ⊓ hasTarget(?t_link,?tgt) ⊓ hasTarget(?t_link,?tgt)
⊓ hasReferent(?t_link,?ref) ⊓ isAFuzzyDistance(?t_relation) ⊓ is(?t_relation,"close")]
```

Figure 8: Rule Definition of *Def_close_relation*.

- a *domain* is a set of actions. Each action specifies *preconditions* that must be present in the current state so it can be applied, and *effects* on the current state,
- a description of the *initial state* of a world,
- a description of a *goal*.

As output, it generates a *plan*: a sequence of actions.

This planner consists of a task planner and a path planner and *SpaceOntology*. The task planner's aim is to define the sequence of actions which will be executed to achieve the goals of the entry problem. Path planner defines possible paths to reach the execution of tasks to accomplish. It finds a path between two positions (fuzzy positions or well defined positions). *SpaceOntology* manages spatial information about the space.

There is a communication between these components (Fig. 10). This communication is in the form of queries/answers. These one is in charge of planners which at some steps require spatial preconditions to be satisfied. To this end, it sends a query to *SpaceOntology* to answer to the request.

Example 4 (Study Case) *A team (human and robot) is in the office $O_{11}^{B_1}$ in the first floor of the building B_1 (the human position is h) (Fig.11). The human asks the robot to pick up a book (b) located in the office (o)* close to *the coffee machine. This machine is on the second floor of the building B_6 (Fig.11).*

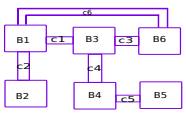

Figure 11: Abstract Map.

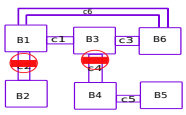

Figure 12: Deadlocks.

The robot's goal is to fulfill this mission. A classical task planner provides: (1) `go-to`$(o_{11}^{B_1}, o)$, (2) `take`(b), (3) `go-to`(o,h), (4) `give`(b) as a plan. Several problems requiring spatial information are identified: (1) in which position the robot should execute an action (`go-to`$(o_{11}^{B_1}, o)$). The office o is defined as the office *close to* the coffee machine, i.e. how to evaluate the relation *close to the coffee machine*? (2) how to move between different areas to accomplish actions? i.e to define the robot's navigation during and between each action; for example, finding a path between the current position o and the human position h. (3) how to order the action execution with respect to navigation?

To solve this problem, we use the *Spatial Planner* (Fig. 10). This planner requires different types of knowledge encoded in Spatial-PDDL. It is an extension of PDDL with spatial information made by the communication between planners (task and path planner) and *SpaceOntology* (Fig. 10). We focus here on Spatial-PDDL definition and the planner operation.

4.1 From PDDL to Spatial-PDDL

We extend PDDL with spatial knowledge using *SpaceOntology* in order to improve the expressiveness of the planning language. Formally, *SpaceOntology* is a triple (Φ, R, I), where Φ is a set of concepts, R is a set of relations (or predicates) between concepts and I is a set of instances which depends on the described environment.

Our planner takes three inputs (Fig. 10). (1) An initial state S_0, (2) a goal state S_G; these states describe a set of objects (O) by specifying their type (T) and relations between them (Π), (3) A set of actions (A) described in a domain (D). We extend PDDL language with *SpaceOntology*.

4.1.1 Notations

The main extensions are summarized in Table 1.

Notions	PDDL	Extend PDDL with *SpaceOntology*
Predicate	Π	$\Pi \leftarrow \Pi \cup R$
Type	T	$T \leftarrow T \cup \Phi$
Domain	D	use of extends and @ns
Objects	O	spatial/non spatial objects
Actions	A	non spatial/spatial/navigation actions

Table 1: Extended Notions with *SpaceOntology* Elements.

A concept, a relation or an instance always has a prefix. (Dou, 2008) expresses that the domain is extended by an ontology, where the notion of " : *extends*" and "@$ns1$: " were introduced. Each concept in *SpaceOntology* defines a spatial type. All the concepts defined in *SpaceOntology* (Φ) are introduced and exploited for extending PDDL (Table. 1).

A predicate π ($\pi \in \Pi$) expresses a relation. All the relations (R) defined in *SpaceOntology* are introduced and exploited for extending PDDL (Table. 1).

Our contributions concerning the action definition consist of the extension of geometric preconditions/effects in order to take the different spatial representations (topological and fuzzy information) into account. In addition, it consists of the definition of the spatial fuzzy actions (Def. 2).

Definition 2 (A Spatial Fuzzy Action) *A spatial fuzzy action a is defined by* `a = ( at , pr, q, e)` *, where :* `at, pr` *and* `e` *are respectively parameters for the action, preconditions, effects as defined in the extended action (Guitton, 2010) and* `q` *is a set of queries.*

Example 5 *The semantic of the action* `take` *(Fig. 13) is: the robot* `r` *takes the book* `b` *located in the office defined in a fuzzy way* `?o`*. The office* `?o` *is defined in Spatial-PDDL as a query variable (Dou, 2008).*

```
take(r,b,?o)
(robot r) (book b)(empty-arm r)
(¬(has r b))(region ?o)(At r ?o)
(:query
 (freevars (?o - Region ?α ?β -@xsd:int
    ?rel - SpatialRelations ?hr - HasRelation)
  (and(@ns:isPartOf ?o Z^{Level(CM)-1}(CM))
    (@ns:hasTarget ?hr ?o)(@ns:hasReferent ?hr CM)
    (@ns:concernsRelation ?hr ?rel)
   (if(instanceOf ?rel NumericDistance)
      (and(owl:DatatypeProperty rdf:ID="α" ?α  Z^{Level(CM)}(CM))
         (owl:DatatypeProperty rdf:ID="β" ?β Z^{Level(CM)}(CM))
         (bounded-int owl:DatatypeProperty rdf:ID="distance" ?α (?α+ ?β))))
   (if(instanceOf ?rel FuzzyDistance) (= ?rel  "close_to")))))
(At r p_r)(@ns:isPartOf p_r ?o)
(bounded-int distance(p_r r.arm , b), ?α_o, ?β_o)
(:query
  (freevars (?α_o ?β_o -@xsd:int)
    (and(owl:DatatypeProperty rdf:ID="α" ?α_o  ?o)
        (owl:DatatypeProperty rdf:ID="β" ?β_o ?o))))
(frontOf r.arm, b) (sameLevel r.arm,b)
(¬(empty-arm r) (has r b)
```

Figure 13: Modeling the Action `take` in Spatial-PDDL.

To delimit this region, we query *SpaceOntology* (details in Section 4.2.1). The fuzzy relation *close* can be described in a numeric way. So, we seek an office satisfying the fuzzy relation close by translating it in a numerical representation (Belouaer et al., 2010). Moreover, the action execution requires that the precondition `(At r` p_r`)` is satisfied. However, the satisfaction of this precondition requires the definition of the position p_r. This position is defined as follows. First, it must be in the office `?o`. Then, it must allow us to reach the target; the book. This requires that the robot is at a position p_r close to the book. The evaluation of this distance is made by the function `bounded-int`. Also, there are two functions like `frontOf` and `sameLevel` which must be satisfied in order to grab the book.

4.1.2 Integrating *SpaceOntology* in PDDL: Spatial-PDDL

We translate any spatial knowledge described in *SpaceOntology* to the domain and problem definitions in PDDL. *SpaceOntology* has an open world assumption and PDDL a closed one. To overcome this difficulty, the translation concerns only known types (Example 6).

Example 6 (Spatial Type and Predicate Extension) *Figure 14 show a translation of an extract from SpaceOntology to PDDL in order to define Spatial-PDDL.*

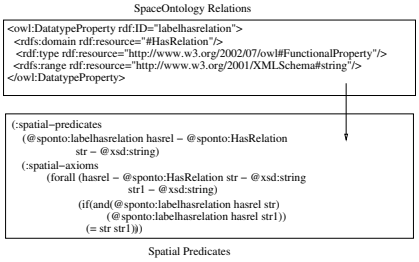

Figure 14: Extension of PDDL with Spatial Predicates.

A spatial relation in *SpaceOntology* is described by the concept *HasRelation*. This concept connects two regions by a spatial relation. Each relation instantiating this concept is identified by a label (*labelhasrelation*). This label is defined as a predicate in Spatial-PDDL (Fig. 14). The axioms concerning (Fig. 14) this predicate express that each label corresponds to a single instantiation of the concept *HasRelation*.

Our model requires as an input : (1) a planning domain describes each action of a system in Spatial-PDDL, (2) a planning problem describing the initial state and goals in Spatial-PDDL, (3) *SpaceOntology* gives spatial description of space, and (4) some required information.

Definition 3 (A Domain) *A planning domain* $D = (A, \Pi)$*, where* $A = \{a_1, a_2, \ldots, a_n\}$ *a set of actions, where* a_i *is an action and* Π *set of relations between objects O.*

Definition 4 (A Spatial Planning Problem) *A Spatial Planning Problem* $P = < O, S_0, S_G >$*, where O is a set of spatial and non-spatial objects,* S_0 *is an initial state that could be associated to fuzzy position and* S_G *is the goal state that could be associated to fuzzy position.*

In Figure 15, some objects are spatial (`:regions` O_{11}^{B1} *CM* `?o`), others are non spatial (`:objects human robot book`). The initial state expresses an initial symbolic situation *human has not the book* `(not (has human book))`. Also, it expresses the

initial spatial situation *the robot is in the office* O_{11}^{B1} `(At robot` O_{11}^{B1}`)`.

```
(define (problem bookmissionpb)
 (:domain  bookmission)
 (:objects
       Arnaud - human
       Bobo - robot
       book  table - item
       o11b1 - region
       ?target - region ;; fuzzy region)
 (:init
      (on book table)
      (At Arnaud o11b1)
      (At Bobo o11b1)
      (At table  ?target)
      (Empty Arnaud)
      (Empty Bobo)
      (= (maxhdistance) 1)
      (= (minhdistance) 0.5)
      (not (Holding Arnaud book)
)
 (:query
    (freevars (?target - region ?α ?β -@xsd:int)
    (and ( In ?target Z^{Level(cm)-1}(cm)
     (owl:DatatypeProperty rdf:ID="α" ?α  Z^{Level(cm)}(cm))
     (owl:DatatypeProperty rdf:ID="β" ?β Z^{Level(cm)}(cm))
     (bounded-int distance(?o,cm) ?α (?α+ ?β)))))
 )
 (:goal ( and
     (not(on book table))
     (Holding Arnaud book))))
```

Figure 15: An Example of Spatial-PDDL Problem.

Note by "?α" and "?β" query variables to determine the α and β of the current level. The relation `owl:DatatypeProperty rdf:ID="β"` identifies for a given region the β (Fig. 15).

The fuzzy spatial information *the office close to the coffee machine* is defined by a query variable: "?*o*". There are some functions. $(distance?ip?tp - position)$ used to compute the Euclidean distance between two positions. $(maxhdistance)$ (resp. $(minhdistance)$) indicates the maximum distance (resp. minimum) from which the interaction with humans can not be executed. *path* (Fig. 16) checks the existence of a path and computes it.

4.2 Communication Between Components

The planner has three components: task planner, path planner and *SpaceOntology*. The communication between them is in form of request/response. Planners (task and path) formulate queries to spatial ontology. They are encapsulated in the preconditions of the actions or they are directly in the problem definition. In this section, we start by specifying the types of queries necessary for the planner function. Then, we describe the operation.

4.2.1 Requests for *SpaceOntology*

Several types of queries that are needed for the system operation. We can classify them into three categories: **Adjacency query**, **Proximity Query** and **Hierarchical Interest Zone Query**

```
(define (domain bookmission)
 (:requirements :typing :fluents)
 (:extends
    (uri ".../SpaceOntology.owl" :prefix sponto))
 (:types actor - object
 item - object
 human robot - actor)
 (:spatial-types  region - @sponto:Regions)
 (:predicates (Empty a - actor)(Holding a - actor i - item)
      (on i - item t - item)))
 (:spatial-predicates (At o - object p - region))
 (:functions (distance ip tp - position)(maxdistance)(mindistance)
             (maxhdistance)(minhdistance)(path irg trg - region))
 (:action go
    :parameters(r - robot irg - region trg - region )
    :precondition (and(At r irg)(not(At r trg)))
    :spatial-precondition (=(path irg trg)true))
    :effect(and(At r trg) (not(At r irg))))
 (:action pick-up  ...)
 (:action put-down ...)
 (:action give  ...)
 (:action take ...))
```

Figure 16: An Example of Spatial-PDDL Domain.

Adjacency query We assume that each region is adjacent to its neighbors with access points (doors, intersection of corridors, ...) called *gates*. A gate allows transitions between adjacent regions (in the same hierarchical level or in different hierarchical levels). A gate allows us to connect more than two regions.

Two regions are adjacent if there is a/or a set of transition between these two regions. Two types of adjacency: *direct adjacency* (i.e, the longer of the path between these two regions is equal to 1 (number of gates), of course several possibilities may exist, this represents a multiple choice of paths) and *indirect adjacency* (i.e, the longer of the path between these two regions is more than 1 (number of gates))

To extract the gates between regions or neighbors for a given region, we use adjacency query to *SpaceOntology*.

```
SELECT ?referent ?target "
WHERE {"?x ns:label ?relation."
       "?x ns:hasintersection \"gate\"."
       "?x ns:hasReferent ?referent. "
       "?x ns:hasTarget ?target."}
```

This query takes as parameter a gate (*gate*) and defines the regions connected to this gate. The *indirect adjacency* is inferred by the existence of a path (of length greater than one see next section).

Proximity query A proximity query defines regions that are close to a given region or if a region is near or far from another region.

```
SELECT ?referent ?target "
```

```
WHERE {"?x ns:concerns \"close\"."
       "?x ns:hasReferent ?referent. "
       "?x ns:hasTarget ?target."}
```

This query takes a parameter relation (*close*) and defines the regions that are close.

Hierarchical Interest Zone Query To define the function $Z^l(r)$, we define the hierarchical interest zone query.

```
SELECT * "
WHERE {"?region rdf:type ns:Regions ."
       "?region ns:label \"" + rg +"\" ."
       "?region ns:isPartOf ?father . "
       "?father ns:level ?level." }
       "Filter ( ?level = " + level + " ).
```

This request is used to return the area of interest for the region *rg* level hierarchical *level*.

4.3 Spatial Planning Process

During the selection of these actions, the planner asks the satisfaction of these preconditions using *SpaceOntology*. Then, it calculates the required displacements and returns an answer to the result thereof. If the navigation is possible so action can be executed. Otherwise another action must be found to achieve the mission. The execution of the action *take* the book (Fig. 13) requires that the robot is in the office close to the coffee machine (in which the book is). However, the robot is in another office : O_{11}^{B1}. Thus, a navigation between the current position and target position is required. The target position is defined in a fuzzy way. Formally, we assume that the general planning problem includes a *fuzzy path planning problem*.

Definition 5 (Fuzzy Path Planning Problem) *Let us consider two fuzzy positions ($\tilde{i_p}$, $\tilde{t_p}$) and a spatial description given by SpaceOnology (Φ,R,I). A path solution Pth between $\tilde{i_p}$, $\tilde{t_p}$ is a continuous sequence of positions/regions that allows the robot to move from the initial state to the goal state while avoiding obstacles.*

4.3.1 Path Planning

Our main contributions to the path planning focus on using space hierarchy, topological spatial relations and spatial fuzzy information.

Let us consider that the initial position i_p is accurate[5] and the target position $\tilde{t_p}$ is fuzzy. The path planning operation is summarized in the main algorithm 1.

Algorithm 1 main

Require: (Φ,R,I), i_p, $\tilde{t_p}$, KC_Σ
1: $\Gamma_G \longleftarrow develop\Gamma_G(\Phi,R,I,i_p,\tilde{t_p},KC_\Sigma)$
2: $P \longleftarrow search(\Gamma_G,i_p,\tilde{t_p})$
3: target zone $\tilde{t_p}$ delimitation
4: **return** A best path P from i_p, $\tilde{t_p}$

- Modeling the environment; it is based on *SpaceOntology* (Φ,R,I), for spatial reasoning in order to define a structure called *Crossing Network Graph*, Γ_G (line 1).
- In this graph, we try to find a path between two positions i_p, $\tilde{t_p}$ according to the optimality criteria (line 2). A path solution is a list of regions that the robot must visit to reach its target position.
- then defining the target position (line 3).

Generation of Crossing Network Graph A *Crossing Network Graph* Γ_G is a directed graph. Nodes are organized hierarchically. Edges represent hierarchical relation between nodes. A node in this graph represents a *Crossing Network* or a region.

Definition 6 (Crossing Network Γ) *A graph Γ = (V,E), where; $V = (v_1,\ldots,v_n)$ is a set of vertices, each v_i is a crossing network and $E = (e_1,\ldots,e_n)$ is a set of edges, each e_i defines an adjacency relation, a gateway and a congestion.*

Algorithm 2 developΓ_G

Require: (Φ,R,I), ip,$\tilde{tp}$,KC_Σ
1: $\Gamma_G \longleftarrow lowLevelNetwork((\Phi,R,I),ip,\tilde{tp},KC_\Sigma)$
2: $l \longleftarrow \Gamma_G.getLowLevel()$
3: $highlevel \longleftarrow \tilde{tp}.getLevel()$
4: **while** $l < highlevel$ **do**
5: $\quad g \longleftarrow develop(\Gamma_G.getGraph(l),ip,\tilde{tp},\Phi,R,I,KC_\Sigma)$
6: $\quad$ add g to Γ_G in level $l+1$
7: $\quad l \longleftarrow l+1$
8: **end while**
9: **return** Γ_G

The algorithm 2 constructs the Crossing Network Graph Γ_G. Each element of this graph is related to the mission. This graph defines all possible paths between ip and $\tilde{tp}$. The function $develop\Gamma_G$ is executed in two steps. The first, generation of the abstract graph (lines 1, 2). The second, incremental development of the abstract graph (lines 3 to 7).

Abstract Graph Generation : The idea consists in finding the first same ancestor of the two places i_p,$\tilde{t_p}$ and returning the hierarchical level l of

[5]To simplify the explanation, the initial position is crisp. This work may well be applied in the context of precise positions.

this ancestor. After this, the abstract Crossing Network is generated (algorithm3).

Algorithm 3 lowLevelNetwork

Require: $(\Phi, R, I), i_p, \tilde{t_p}, KC_\Sigma$
1: $l \leftarrow lowLevel(\Phi, R, I, i_p, \tilde{t_p}) - 1$
2: $N \longleftarrow \emptyset$, $E \longleftarrow \emptyset$
3: $initialzone \longleftarrow Z^l(i_p)$, $targetzone \longleftarrow Z^l(\tilde{t_p})$
4: $N \longleftarrow initialzone$, $N \longleftarrow N \cup targetzone$
5: $rg \longleftarrow targetzone$
6: $R \longleftarrow$ { regions form I in level l }
7: **while** $rg \neq initialzone$ **do**
8: Select r from R |
9: r check that r is concerned by the mission
10: r is not a dead end
11: rg and r are directly connected
12: $gateways \longleftarrow (rg, r)$ | they satisfy KC_Σ
13: $N \longleftarrow N \cup r$
14: $E \longleftarrow E \cup Edge(rg, r)$
15: $rg \longleftarrow r$
16: **end while**
17: $g \longleftarrow (N, E)$
18: **return** $g = (N, E)$

Note, by $g = (N, E)$ the abstract Crossing Network. N is a sub set of regions in the abstract level (lines 9, 10). For each $n \in N$, n must satisfy the selection criteria (lines 14 to 18). E is a set of edges. Edges represent spatial adjacency relations between two nodes giving gateways. The selection of gateways depends on kinematic constraints of the robot. The function `lowLevel` gives the level of the most same abstract ancestor of $i_p, \tilde{t_p}$. This algorithm is a query submitted to the ontology. In *SpaceOntology*, the relation *isPartOf* is transitive. The inference mechanisms provided by the ontology help to deduce for a given region, the region encompassing it.

Hierarchical Development : The algorithm 4 develops each node of level l until reaching the target hierarchical level *highlevel*. This taking the various constraints into account . Graph generation is mainly based on topological relations. Based on the notion of *hierarchical interest zone*, we manage the fuzzy information. We consider the most abstract region encompassing the region in which the coffee machine is. But, this is not enough. Indeed, we will generate all possible paths between buildings $B1$ and $B6$ by considering $B2$,$B3$, $B4$ and $B5$. However, the passing through the buildings $B2$, $B4$ and $B5$ is not necessary (Fig. 12). For $B2$ and $B5$ they are not included in the mission; they are a deadlocks regions (one gate to exit/entrance). For $B4$, it is not included in the mission. Also, this region is connected to a deadlock region $B5$. So, $B4$ is also a deadlock region. By combining all these concepts we can generate the crossing network graph (Fig. 17).

Example 7 *In example 4, the team is in office* O_{11}^{B1}

Algorithm 4 develop

Require: $graph = (N, V)$, ip,$\tilde{tp}$,Φ, R, I, KC_Σ
1: $l \longleftarrow graph.getLevel()$
2: $nodes \longleftarrow \emptyset$, $edges \longleftarrow \emptyset$
3: **for** $n \in N$ **do**
4: $gates \longleftarrow n.getVertex.getGateways()$
5: **if** $(n == Z^{l+1}(ip))$ **then**
6: $i_z \longleftarrow Z^{l+1}(ip)$
7: $gr \longleftarrow graph(i_z, gates, KC_\Sigma, \Phi, R, I,)$
8: $nodes \longleftarrow gr.nodes$
9: $edges \longleftarrow gr.edges$
10: **else if** $(n == Z^{l+1}(\tilde{tp}))$ **then**
11: $t_z \longleftarrow Z^{l+1}(\tilde{tp})$
12: $gr \longleftarrow graph(t_z, gates, KC_\Sigma, \Phi, R, I,)$
13: $nodes \longleftarrow nodes \cup gr.nodes$
14: $edges \longleftarrow edges \cup gr.edges$
15: **else**
16: $g \longleftarrow gates$
17: $gr \longleftarrow graph(g, g, KC_\Sigma, \Phi, R, I,)$
18: $nodes \longleftarrow nodes \cup gr.nodes$
19: $edges \longleftarrow edges \cup gr.edges$
20: **end if**
21: **end for**
22: $graph_{l+1} \longleftarrow (nodes, edges)$
23: **return** $graph_{l+1} = (N_{l+1}, V_{l+1})$

and the target is in floor E_2^{B6}. *These offices are not in the same hierarchical level. In addition, these regions* (O_{11}^{B1} *and* E_2^{B6}) *are not adjacent. Thus, finding a path between these two regions exploits the space hierarchy and adjacency links defined in SpaceOntology.*

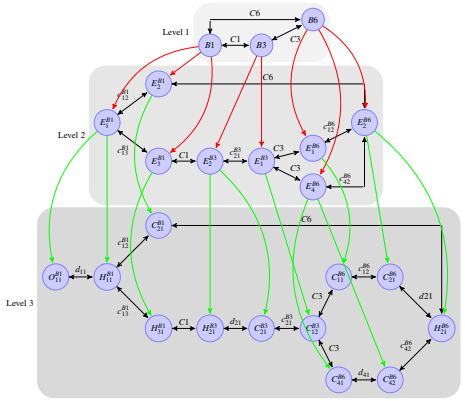

Figure 17: Crossing Network Graph

First, we consider the most abstract regions for O_{11}^{B1} and E_2^{B6} in the same level, there are $B1$ and $B6$. Then, we look for the adjacency links between $B1$ and $B6$. We assume that each region is adjacent to its neighbors with *gates*. To extract gates between regions, we query the ontology. So, we generate all possible paths between buildings $B1$ and $B6$ by considering $B2$,$B3$, $B4$ and $B5$. However, the passing through the buildings $B2$, $B4$ and $B5$ is not necessary (Fig. 12). For $B2$ and $B5$ they are not included in the mission; they are a deadlocks regions (one gate to exit/entrance). For $B4$, it is not included in the mission. Also, this region is connected to a deadlock re-

gion $B5$. So, $B4$ is also a deadlock region. By combining all these concepts we generate a directed graph: *Crossing Network Graph* (Fig. 17). Nodes are organized hierarchically. Edges represent hierarchical relation between nodes. A node in this graph represents a *Crossing Network* (a graph given the adjacency relations between regions in the same hierarchical level). Each node of one level l is developed until reaching the target hierarchical level. We assume that N^l is a subset of regions in the level l. E^l is a set of edges. Edges represent spatial adjacency relations between two nodes giving gateways. For each $n \in N^l$, n must satisfy the selection criteria.

Path Computing We search a path between two positions i_p (accurate position) and $\tilde{t_p}$ (fuzzy position) in the Crossing Network Graph, based on two criteria measures (Mouaddib, 2004): (1) minimize the Euclidean distance between i_p and $\tilde{t_p}$ and (2) facilitate access to a region, to ensure fast and simple actions. We combine these concepts in order to improve the path solution quality. These criteria, allow the robot to execute its actions in an easy and quick way (the choice of paths that are short in terms of distance) and to maintain the speed of navigation. We define a measure: $\Theta(p_1,p_2) = (d(p_1,p_2), c(p_1,p_2))$, where $d(p_1,p_2)$ is the euclidean distance between p_1 et p_2 and $c(p_1,p_2)$ measures the congestion in the region between p_1 and p_2. Pathfinding in crossing network graph consists of path search in crossing network for each hierarchical level. A path solution is a sequence of paths $pth_0,\ldots,pth_k$, where pth_i contains nodes of the i-th level and that are a specialization of nodes on pth_{i-1} (i.e the path solution in level 1 is $B1-C6-B6$, the pathfinding in level 2 will not consider the sub nodes of the node $B3$ (Fig. 17). In each level, we use multi-criteria Dijkstra's algorithm with a single source $Z^i(i_p)$. It returns the shortest path between the source node and the target one $Z^i(\tilde{t_p})$ (i is the abstract level). The shortest path is the path minimizing the measure $\Theta^i = (d,c)^i$, where d is the Euclidean distance and c is a congestion measure. In order to minimize Θ^i , we compute it as described in (Mouaddib, 2004): $\Theta^i = \sum_{crit\in\{d,c\}} Regret(crit)$. The concept is to minimize the regret (Fig. 18).

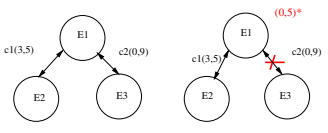

Figure 18: Regret Based Path Selection.

We compute θ^*, where this measure is $\theta^* = (0,5)^* = [Min(3,0), Min(5,9)]$. Then, for each edge e, we compute the distance between θ^* and θ_e :

$$Min(Dist) = \begin{cases} Dist(\theta_{c_{12}^{B6}}, \theta^*) = Dist((3,5),(0,5)) = 3 \\ Dist(\theta_{c_{42}^{B6}}, \theta^*) = Dist((0,9),(0,5)) = 4 \end{cases}$$

Finally, we select the edge allowing us the minimum distance with θ^* : edge c_{12}^{B6} (Fig. 18).

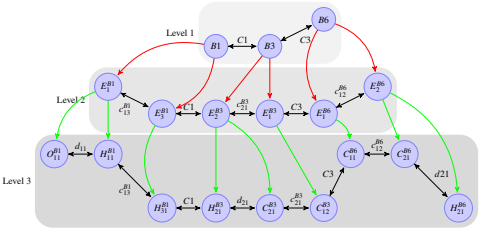

Figure 19: Hierarchical Path Solution.

We are not interested in the global network in level i. Indeed, we consider that the nodes and edges belonging to the path solution of level $i-1$. We can generate the path for each node in the next level and exploit the hierarchical aspect of space which reduces the search time. Indeed, only nodes of the at level i are extended in level $i+1$ and thus some edges in the graph are not explored (Fig. 19).

Delimitation of the Fuzzy Region This step uses the *proximity queries* that allows for a region to define the set of regions that are close/far to it. Let us consider our scenario (Example 4). The robot found the coffee machine. However, it does not reach its target position.

```
SELECT ?target "
WHERE {"?x ns:concerns \"close\"."
       "?x ns:hasReferent \"CM\". "
       "?x ns:hasTarget ?target."}
```

Figure 20: Proximity Query.

To achieve/define its target, we use the *proximity query* defined in Figure 20. This query defines the set of regions which verifies the close (*close*) relation with the coffee machine (*CM*).

4.3.2 Plan Solution

A plan solution is sequence of a set actions where each set of primitive actions is a partial plan performed on a single hierarchical level to reach the state goal starting from a certain initial condition. For the generation of the plan several techniques are possible, mainly *GraphPlan*, *HTN*. According to the reasoning described here, we have the plan solution depicted in Figure 21 for the mission defined in the Example 4. This plan is generated by applying the algo-

rithm Graphplan[6]. The satisfaction of spatial preconditions is based on querying *SpaceOntology*. The disambiguation of fuzzy positions is also based on this ontology. Indeed, in Example 4, the target office is defined in a fuzzy way. Using our model we found the office $o22b6$.

```
go(Bobo, o11b1, o22b6)
pick-up(Bobo, book, table, o22b6)
go(Bobo, o22b6, o11b1)
give(Bobo, book, Arnaud, o21b1)
```

Figure 21: Solution Plan.

The robot moves between $011b1$ and $022b6$. The navigation action respect the path solution given in Figure 19. Then, it takes the book, it joins the human and finally, it gives the book to the human. The execution of the action *give* requires that the robot is at a distance between 0.5 and 1 meters as specified in the definition of the problem (Fig.15).

5 EXPERIMENTS

We present the resolution complexity and some experimental results to solve planning described with Spatial-PDDL.

5.1 Complexity Analysis

The global complexity of the plan generation of a plan depends on the complexity of inference, the complexity of the graph generation and the complexity of plan solution generation. In this section, we present the complexity analysis for the generation of the graph and path computation. The complexity of plan generation depends on the algorithm used, we use Graphplan. The inference complexity depends on the number of rules. It correpsonds to the complexity given by the reasoner Pellet (Sirin et al., 2007).

5.1.1 Generation of Crossing Network

The Crossing Network generation is based on the adjacency relations between the regions of mission. Two ways to define an adjacency relation with *SpaceOntology* (Fig 22): (1) by exploiting hierarchical relations or (2) by exploiting the adjacency relations.

Figure 22: In the left, the adjacency relation is presented by hierarchical relation and in the right, by topological one.

[6]http://www.cs.cmu.edu/ avrim/graphplan.html.

Formally, this crossing network generation depends on the gates number used; $|G|$. In the case of hierarchical relation, the complexity is $O(|G|)$. In the case of topological relation, the complexity is $O(|G|^2)$. So, we use inclusion relation.

5.1.2 Path Computing

Let N be the nodes number in the Crossing Network Graph generated. The branching factor b_n corresponds to the number of sub-nodes of an abstract node n. We assume that b_{n_l} gives the number of sub-nodes of a node n in level l. So $N = \sum_{l=min_l}^{max_l} \sum_{i=1}^{n_{min_l-1}} bi_l$, where; (1) n_{min_l-1} is the set of nodes in the most abstracted level in the Crossing Network Graph, (2) min_l is the most abstracted level in the Crossing Network Graph, (3) max_l is the most detailed level in the Crossing Network Graph, (4) bi_l is the number of new nodes that can be generated from a node i in the level l and (5) $\sum_{i=1}^{n_{min_l-1}} b_{i_{min_l-1}}$ corresponds to the number of nodes in the level min_l.

We used multi-criteria Dijkstra's single-source shortest path algorithm. In fact, implementing the priority queue with a Fibonacci heap makes the complexity $O(E + V \log V)$, where E is the number of edges and V the nodes number. So in each Level , l the complexity of path finding is $O(E_l + \sum_{i=1}^{n_{l-1}} b_{i_{l-1}} log(\sum_{i=1}^{n_{l-1}} bi_{l-1}))$. The full complexity is: $O(\sum_{l=min_l}^{max_l} E_l + \sum_{l=min_l}^{max_l} (\sum_{i=1}^{n_{l-1}} b_{i_{l-1}} * \log(\sum_{i=1}^{n_{l-1}} bi_{l-1})))$. The impact of the hierarchical multi-criteria Dijkstra's single-source approach in the pathfinding, is depicted in Table 2. In our context, real scenarii do not exceed 6 levels and maximum branching factor of 5 sub-nodes (b). According to table 2, we noted that our approach returns a path solution in a time (in seconds) $\in [0,100]$ which is acceptable. Finally, the proposed pathfinding method in an environment, based on its description via *SpaceOntology* is not complex or expensive (in our application context). Whatever, it depends on several factors namely, the Crossing Network Graph generation time (due to quantity / quality and structure of spatial information) and the size of this graph.

5.2 Experimental Results

The computation time to solve a spatial planning problem described in Space-PDDL depends on computing time of answering to queries (Table 2).

Table 3 shows that the computation time depends on the number of requests and the response time. The latter depends on the rules number defined in *SpaceOntology*. This computing time is very reasonable even if we have a large number of requests. In-

l , b	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$	$b=7$	$b=8$	$b=9$	$b=10$
$l=2$	<1	<1	<1	<1	<1	<1	1	2	40
$l=3$	<1	<1	<1	<1	1.05	10	35	62.5	70
$l=4$	<1	<1	<1	<1	85	120	342.1	427.2	600
$l=5$	9	35	50	62.1	160	210.2	300.5	400.2	581.2
$l=6$	13.2	64.2	80.03	100.2	195.2	281.3	450.5	907.2	990.5
$l=7$	320	480.2	600.1	900.2	1050.4	1700	1970.01	-	-
$l=8$	340 s	650.8	801.5	957	995.9	2000.01	-	-	-
$l=9$	1200.1	2400.4	5000.7	9007	2400.4	-	-	-	-
$l=10$	8000.4	14000	-	-	-	-	-	-	-

Table 2: Time Computation for Hierarchical Pathfinding. The size of graph depends on hierarchy levels and branching factor. The most abstract level contains 6 nodes. The symbol (-) represents very high computing time ($>$ 30000 seconds).

Queries		Solving problem
Queries number	Answer to query	
[2-8]	<1	<1
[9-13]	[1-2.2]	[1 - 3.08]
[14-20]	[3.01-4.8]	[4.01 - 7]
[21-30]	[5.05-10.09]	[8.07 - 15.01]

Table 3: Computation Time of Plan Solution Generation Solving the problem in Example 4 (seconds). The path computation time is less then one second.

deed, in Table 3, for a query number between 21 and 30, solving the problem is done in 15.01 seconds.

6 CONCLUSION

Our main contribution was the extension of PDDL to spatial knowledge: Spatial-PDDL. To do this, we uses *SpaceOntology* that considers several spatial knowledge aspects; the space hierarchy, spatial relations (numeric, topological and fuzzy). This permits to consider not only numeric spatial information in preconditions but also topological and fuzzy information. Moreover, spatial reasoning (the inference mechanisms) provided by this ontology permits a complex reasoning. *SpaceOntology* in the planning, is not used for verification or control, but as information needed for planning.

Future work will concern the improvement of spatial interaction between human and robots. In other words, we aim to formalize communication between them, especially in the case where the set of fuzzy rules are not sufficient to find a solution.

REFERENCES

Baader, F. (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge Univ Pr.

Balbiani, P., Condotta, J.-F., and del Cerro, L. F. (1999). A new tractable subclass of the rectangle algebra. *IJCAI*.

Belouaer, L., Bouzid, M., and Mouaddib, A. (2010). Ontology based spatial planning for human-robot interaction. *TIME*.

Bouguerra, A., Karlsson, L., and Saffiotti, A. (2008). Monitoring the execution of robot plans using semantic knowledge. *Robotics and Autonomous Systems*.

Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V., and Strintzis, M. (2005). Knowledge-assisted semantic video object detection. *IEEE Transactions on Circuits and Systems for Video Technology*.

Dominey, P., Boucher, J., and Inui, T. (2004). Building an adaptive spoken language interface for perceptually grounded human–robot interaction. In *IEEE-RAS/RSJ international conference on humanoid robots*.

Dou, D. (2008). The Formal Syntax and Semantics of Web-PDDL.

Fox, M. and Long, D. (2003). PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *JAIR*.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL the planning domain definition language. *AIPS*.

Guitton, J. (2010). Architecture hybride pour la planification d'actions et de dplacemen.

Hudelot, C., Atif, J., and Bloch, I. (2003). Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets and Systems*.

Laborie, S., Euzenat, J., and Layaıda, N. (2006). A spatial algebra for multimedia document adaptation. *SMAT*.

Mouaddib, A.-I. (2004). Multi-objective decision-theoretic path planning. In *ICRA*.

Prosser, P. (1995). Mac-cbj: maintaining arc consistency with con ict-directed backjumping. *Research report*, 177.

Schockaert, S. (2008). *Reasoning about fuzzy temporal and spatial information from the web*. PhD thesis, Ghent University.

Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.