Hybrid Reasoning About Relative Position and Orientation of Objects and Navigating Agents Using Answer Set Programming

Yusuf Izmirlioglu

University of Roehampton, Computer Science Department, London, UK
New Mexico State University, Computer Science Department, Las Cruces, USA
yusuf.izmirlioglu@roehampton.ac.uk

Abstract

We study reasoning about relative position, orientation and distance of moving objects in 2D space. We first construct a new hybrid calculus *HOPA* by augmenting qualitative distance and quantitative constraints into Oriented Point Relation Algebra (*OPRA*). Then we develop a framework for consistency checking and reasoning with *HOPA* using Answer Set Programming. This framework can also explain the source of inconsistency, infer new knowledge and generate a layout of objects and their orientation in the discrete space. The framework is capable of reasoning with (un)certain, partial and presumed information. We evaluate efficiency and scalability of our method by computational experiments, and illustrate its applications with sample scenarios from robotic perception and marine navigation.

Introduction

Reasoning about position and distance is vital for cognitive robotics, land/marine navigation and surveying applications. Motion planning and collecting survey knowledge requires identifying relative position and orientation of static objects as well as navigating agents in the environment. In marine navigation, vessels need to position themselves with respect to other agents and the port, for example vessel 4 reports vessel 2 is on its starboard 147^o, vessel 3 states vessel 6 is at its behind and near to the port, vessel 6 observes 7 on clock angle 9. As another example, in the robotic sensing and motion planning problem of Moratz, Dylla, and Frommberger (2005), Dylla and Moratz (2004), Moratz and Ragni (2008), the human user instructs the robot "to navigate to the red cube behind the blue cube" (Figure 1). To achieve this goal, the robot needs to identify which red cube is behind the blue cube (C). In a similar vein, oriented objects and relative position are also prevalent in surveying, localization/mapping and street networks, as in the statements "The cafeteria is in the front-left of the library and very near to it", "when you go down the Kingston road and reach the junction, Wimbledon avenue will be on your left".

In these contexts, moving agents have some bearing and pivot objects may have some intrinsic orientation. Navigating robots and vessels need to know relative position of other agents for collision avoidance, motion/task planning and fleet management. Such relative location and distance information can be quantitative (e.g. the tanker robot is located at 12 m distance and 53^o angle of robot 2) as well as qualitative or coarse (e.g. robot 5 is to the left-back and near to robot 7, ship 3 is at starboard of ship 4). The latter is relevant in contexts with human presence or data obtained from sensors is imprecise or exact location of an extended object is hard to determine. In marine exploration, metric data about location and heading can be obtained from GPS and measurement devices, but additional complex calculations are necessary to determine the relative location of the agent with respect to the port and other agents. In these situations, marine people often use simple view or deck sight to locate themselves, e.g. "we are near and southwest of Patara marina", or "the yacht is starboard and behind us". Besides such qualitative direct information is beneficial for verification of sensor data, in case of fault or calibration. Until now, robotic and marine navigation literature mostly focused on collision and motion planning (Maaref and Barret 2002; Frommberger 2008; Miguel-Tomé 2021), rather than reasoning about knowledge.

Figure 1: Robotic sensing and motion planning scenario

In this paper, we study reasoning about relative position, orientation and distance of point objects in 2D space. We start with Oriented Point Relation Algebra (*OPRA*) of Moratz, Dylla, and Frommberger (2005) which describes qualitative position of oriented points on the plane. We construct a new calculus *HOPA* (Hybrid Oriented Point Algebra) by augmenting qualitative direction and quantitative location, distance, orientation information to *OPRA*. Then we develop a framework for reasoning with this hybrid formalism using Answer Set Programming. The framework can check consistency of a set of qualitative and quantitative constraints, explain the source of inconsistency, handle uncertain and presumed information, infer new knowledge and generate a configuration of objects. We evaluate efficiency

of our method by computational experiments and illustrate its applications with sample scenarios.

Preliminaries

OPRA Formalism

OPRA (Moratz, Dylla and Frommberger 2005) describes qualitative position of two oriented points on 2D space with adjustable granularity g. In $OPRA_g$, there are $4g$ relations indexed from 0 to $4g-1$. The even indexed relations are linear (along a line) whereas odd indexed relations are angular. The length of an angular sector is $180^o/g$. Let (x_K, y_K) denote the location and ϕ_K denote the intrinsic orientation (heading) of object K. $\varphi_{KL} = tan^{-1}(y_L - y_K)/(x_L - x_K)$ denotes the absolute angle of the vector $\overrightarrow{KL}$ directed from K to L. If the location of K and L are different, the $OPRA_g$ relation is shown as $K\ {}_g\angle^i_j\ L$ which reads K is on sector i w.r.t. viewpoint of L and L is on sector j w.r.t. viewpoint of K. Namely, K falls onto sector i of L and L falls onto sector j of K. $\psi_{KL} = \varphi_{KL} - \phi_K$ denotes the relative angular position of L w.r.t. K (and vice versa for ψ_{LK}). We name these relations *differential* *OPRA* relations. For example, in Figure 2(a) below $K\ {}_4\angle^2_{11}\ L$.

When the location of K and L coincide, the relative angular position of L with respect to K is defined as the difference between intrinsic orientations, i.e. $\delta_{LK} = \phi_L - \phi_K$. In this case, the $OPRA_g$ relation is shown as $K\ {}_g\angle i\ L$ which reads L is on sector i w.r.t. viewpoint of K. We name this type of relation as *same* *OPRA* relation. To exemplify, $K\ {}_2\angle 3\ L$ in Figure 2(b). Note that specifying j is unnecessary since $j = 4g - i$.

Qualitative interval-based distance relation has been introduced (Clementini, Di Felice, and Hernández 1997) as a symbolic binary relation (e.g. *overlap*, *near*, *far*) with adjustable granularity. Each distance relation is associated with an interval, for example *near* corresponds to $[2.5, 4.7)$. If the Euclidean distance between two objects is in this range, they are *near* to each other. We create a new hybrid calculus named *HOPA* by adding distance relation $K\ \beta\ L$ and quantitative constraints into *OPRA*. Quantitative constraints are of the form

(1) $\phi(K) = a^o$, (2) $\phi(K) \in [a^o, b^o]$, (3) $\phi(K) = \overrightarrow{PR}$,
(4) $\psi(K, L) = a^o$, (5) $\psi(K, L) \in [a^o, b^o]$,
(6) $\delta(K, L) = a^o$, (7) $\delta(K, L) \in [a^o, b^o]$,

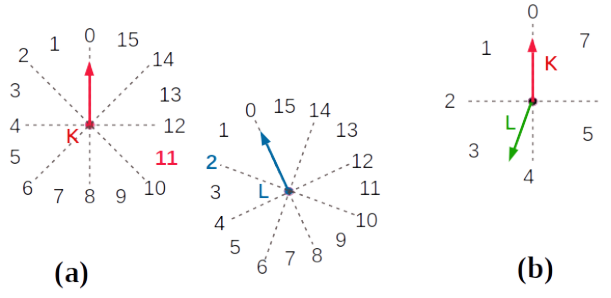

Figure 2: Example for *OPRA* relations

(8) *loc(K)* $= (x, y)$, (9) *dist(K,L)* $= c$,
(10) *dist(K,L)* $\in [c, d]$.

(1-3) are about intrinsic orientation (heading), (4-5) are about relative position of objects whose location are different, (6-7) about relative position of objects with same location, (8) numerical location and (9-10) numerical distance.

In *HOPA*, we also introduce *single-sided* differential *OPRA* relation $K\ {}_g\not\angle^i\ L$. A single-sided *OPRA* relation specifies only the relative position of K w.r.t. L, and not the other way around. The intuition is agent L may not know how it is observed from viewpoint of other agent K. In *HOPA*, we allow for heterogeneous relations: an agent K observes relative position of L with granularity g_1, while agent L observes K or M with granularity $g_2 \neq g_1$.

Answer Set Programming

Answer Set Programming (ASP) is a logic programming paradigm based on answer set semantics (Gelfond and Lifschitz 1988, 1991). It provides a formal framework for knowledge reasoning and declaratively solving computationally hard problems. ASP models a problem by a set of logical formulas (called rules), so that its models (called answer sets) characterize the solutions. ASP provides logical formulas, called rules, of the form

$$Head \leftarrow L_1, \ldots, L_k, not\ L_{k+1}, \ldots, not\ L_l\,. \quad (1)$$

where $l \geq k \geq 0$, *Head* is a literal and each L_i is a literal. A literal is an atom A or its negation $\neg A$ or $\bot$. A rule is called a *constraint* if *Head* is $\bot$, and a *fact* if $l = 0$. A set of rules is called a *program*.

Choice rules (a special type of logical formula) allows to make nondeterministic choices. For example, the rule below chooses at least 1 and at most 5 interns among candidates, for each lab:

$$1 \leq \{intern(L,C) : candid(C)\} \leq 5 \leftarrow lab(L). \quad (2)$$

Hard constraint rules eliminate those answer sets which do not qualify as a solution. For example the constraint below prohibits the cases where a candidate is assigned to multiple labs:

$$\leftarrow\ L1 \neq L2,\ intern(L1, C), intern(L2, C). \quad (3)$$

Weak constraint rules express preferences and are used for optimization. The first rule below adds 2 unit of cost (penalty) for each intern and course, if an intern has not taken a preliminary course. The priority of this rule is 1, which is the highest priority. The ASP solver tries to minimize the total cost in a lexicographic manner, starting from the highest priority.

$$\leftarrow\ intern(L, I),\ not\ taken(I, Z),\ prelim(Z).\ [2@1, I, Z] \quad (4)$$

Further information about ASP can be found in Baral (2003), Gebser et al. (2012), Gelfond and Kahl (2014), Lifschitz (2019).

HOPA Formalism

HOPA already includes *OPRA* relations, in addition it also includes qualitative and quantitative constraints about orientation, distance and location. *OPRA* relations are defined

over the continuous space $\mathbb{R}^2 \times [0, 360^o)$. However instantiation and search for spatial variables over continuous domain are not feasible, besides real data from measurement devices has finite precision and the location of an extended object (like robot, ship) can be determined within some bounds. Thereby we define semantics of *HOPA* relations over a discrete space. Let $\Xi_{s,t} = [x_1, ..., x_s] \times [y_1, ..., y_t]$ be equally spaced x and y coordinate values and $\Phi_z = [f_1, ..., f_z]$ be the angular degree values. For example, if the range in both axis is 8 km, the precision is 0.1 km and angular resolution is 2^o, then $\Xi = [0, 0.1, 0.2, ..., 8.0]^2$ and $\Phi = [0, 2^o, 4^o, ..., 358^o]$. We can further represent the discrete domain by a grid such that $\Xi_{s,t} = [1, ..., s] \times [1, ..., t]$ and $\Phi_z = [1, ..., z]$. Then the domain of *HOPA* relations become $D_{s,t,z} = \Xi_{s,t} \times \Phi_z$.

We say that a given set C of *HOPA* constraints is consistent over domain $D_{s,t,z}$ if there is an instantiation of variables $E : V \mapsto D_{s,t,z}$ which satisfies constraints in C, and we call E a solution of C. We define the consistency checking problem H in *HOPA* as deciding whether the input constraint network C is consistent or not.

Consistency problem of *OPRA* calculus itself is *NP*-hard and $\exists\mathbb{R}$-complete (Lee 2014), and it is not in *NP* unless these complexity classes coincide. Theorem 1 states that complexity of consistency checking in *HOPA* is NP-complete.[1] The reason for *OPRA* having higher complexity is that it is defined over continuous space: A nondeterministic Turing machine may not search over an infinite space.

Theorem 1. *The consistency checking problem $H = (C, V, D_{s,t,z})$ in HOPA is NP-complete.*

Reasoning with *HOPA* Using ASP

We first study consistency checking problem $H = (C, V, D_{s,t,z})$ in *HOPA* using Answer Set Programming. Due to limited space, we explain the core ASP rules, the full ASP code is available at the web repository.[1]

Represent the input. The input of the problem H is represented by a set of facts in ASP. An oriented spatial object $v \in V$ is represented by *object*(v) atom. In some domains (like surveying and our robotic scenario), it is known that two objects are located at the same point (with or without knowing their numerical coordinates). Hence we distinguish points and objects; and we denote point identifier of an object with another atom *point*(v, p). v and p are just variables and we enumerate them with $1..|V|$ and $1..|P|$. In general, if point location of objects is unknown, then their point identifier can be chosen the same as their object identifier.

We encode *OPRA* constraints by *same_opra(N,K,L,I,G)*, *diff_opra(N,K,L,I,J,G)*, *one_sided_diff_opra(N,K,L,I,G)* atoms. Here N denotes the constraint number, K, L objects, I, J sectors, G granularity. Qualitative distance relation is encoded by *dist_rel(N,K,L,U)* where U is the relation between K, L. We represent the quantitative constraints by *direction(N,K,A)*, *direction_range(N,K,A1,A2)*, *direction(N,K,P,R)*, *psi_precise(N,K,L,A)*, *psi_range(N,K,L,A1,A2)*, *delta_precise(N,K,L,A)*, *delta_range(N,K,L,A1,A2)*, *location(N,K,X,Y)*, *distance(N,K,L,D)*, *distance_range(N,K,L,D1,D2)* atoms.

[1]The proofs and code are at github.com/yizmirlioglu/HOPA

Because a differential OPRA constraint involves two different relative position information, we transform it into two single-sided constraint.

single_diff_opra(N,K,L,I,G) ← *diff_opra(N,K,L,I,J,G)*.
single_diff_opra(N,L,K,J,G) ← *diff_opra(N,K,L,I,J,G)*.
single_diff_opra(N,K,L,I,G) ← *one_sided_diff_opra(N,K,L,I,G)*. (5)

Here *single_diff_opra(N,K,L,I,G)* reads as K is on sector I of L with granularity G. We process quantitative constraints to identify location and orientation (heading) of objects.

xloc(P,X) ← *point(K,P)*, *location(N,K,X,Y)*.
yloc(P,Y) ← *point(K,P)*, *location(N,K,X,Y)*.
orient(K,A) ← *direction(N,K,A)*.
loc_known(P) ← *point(K,P)*, *location(N,K,X,Y)*.
orient_specified(K) ← *direction(N,K,A)*.
dist_known(P,R) ← *point(K,P)*, *point(L,R)*, *distance(N,K,L,D)*. (6)

Disjunctive Constraints. Disjunctive *HOPA* constraints represent uncertainty in qualitative or quantitative spatial information. A disjunctive *HOPA* constraint is of the form *disjrelation(N,D,Y,_)* where N is the constraint number, D is the disjunct number, Y denotes the type of the constraint and the rest are constraint parameters. We enumerate the type of the disjunctive constraint as 1:same *OPRA* constraint, 2:one-sided differential *OPRA* constraint, 3:differential *OPRA* constraint, 4:qualitative distance constraint, 5:intrinsic orientation constraint and so on. For example, the 23^{th} constraint "Agent 11 is located on the sector 7 of agent 9 (with granularity 6), or moving at direction 73^o, or agent 13 is near to agent 18" is represented by the set of *disjrelation(23,1,2,11,9,7,6)*, *disjrelation(23,2,5,11,73)*, *disjrelation(23,3,4,13,18,near)* atoms. Notice that *HOPA* formalism allows for heterogeneous disjunctive constraints where the pair of objects or the type of constraint can differ among disjuncts.

If the N^{th} constraint is disjunctive, the rule below nondeterministically picks one disjunct and *chosen*(N, D) atom indicates its index.

$$\{\textit{chosen(N,D)} : \textit{disj\_index(N,D)}\} = 1 \leftarrow \textit{existdisj}(N). \quad (7)$$

Then we generate the basic constraint corresponding to the selected disjunct. As an example, the rule for type 3 is given below.

$$\textit{diff\_opra(N,K,L,I,J,G)} \leftarrow \textit{chosen(N,D)}, \textit{disjrelation(N,D,3,K,L,I,J,G)}. \quad (8)$$

Instantiation of spatial variables. A candidate solution is characterized by an instantiation of objects $u \in V$ as an oriented point in $D_{s,t,z}$. An oriented point is specified by its location and orientation. We nondeterministically generate a location $(x, y) \in \Xi_{s,t}$ on the grid for those points whose location is unknown. Forbidden locations and obstacles are designated by *invalid_loc*(X, Y) atoms in the input.

$\{\textit{xloc(P,X)} : 1 \le X \le s\} = 1 \leftarrow \textit{not loc\_known(P)}, \textit{point(P)}$.
$\{\textit{yloc(P,Y)} : 1 \le Y \le t\} = 1 \leftarrow \textit{not loc\_known(P)}, \textit{point(P)}$.
$\leftarrow \textit{invalid\_loc}(X, Y), \textit{xloc}(P, X), \textit{yloc}(P, Y)$. (9)

We nondeterministically assign an orientation (heading) $\phi \in \Phi_z$ to the objects whose orientation is unknown. To

exemplify, if the angular resolution is 3^o, then the possible values are $\{0, 3, 6, ..., 357\}$. For this, we generate a value in $\Omega = \{0, 1, 2, ..., 119\}$ and multiply by 3.

$$\begin{array}{l} \{\mathit{degree}(K,E) :\ E \in \Omega\} = 1 \ \leftarrow \\ \quad \mathit{not\ orient\_specified}(K),\ \mathit{point(P)}. \\ \mathit{orient}(K, E.Z) \ \leftarrow \ \mathit{degree}(K,E),\ \mathit{angle\_res}(Z), \\ \quad \mathit{not\ orient\_specified}(K),\ \mathit{point(P)}. \end{array} \tag{10}$$

Next we determine which vectors and angles to compute for *OPRA* and quantitative constraints. We compute the numerical distance between coordinates and find the angle φ_{PR} by arctangent. We represent *tan* function by internal ASP atoms, for example $\mathit{tan\_range}(4, 0.062, 0.078)$ shows the range of 4^o. For differential *OPRA* constraints and quantitative ψ constraints, we need to compute the vectors $\overrightarrow{PR}$, $\overrightarrow{RP}$ and the angle of these vectors ϕ_{PR}, ϕ_{RP}.

$$\begin{array}{l} \mathit{compute\_vector(P,R)} \leftarrow \mathit{point(K,P)},\ \mathit{point(L,R)}, \\ \quad \mathit{diff\_opra(N,K,L,I,J,G)}. \\ \mathit{xdist(P,R,X1\text{-}X2)} \ \leftarrow \ \mathit{xloc(P,X1)},\ \mathit{xloc(R,X2)}, \\ \quad \mathit{compute\_vector(P,R)}. \\ \mathit{ydist(P,R,Y1\text{-}Y2)} \ \leftarrow \ \mathit{yloc(P,Y1)},\ \mathit{yloc(R,Y2)}, \\ \quad \mathit{compute\_vector(P,R)}. \\ \mathit{tan}(P,R,DY/DX) \ \leftarrow \ DX \neq 0, \\ \quad \mathit{xdist(P,R,DX)},\ \mathit{ydist(P,R,DY)}. \\ \mathit{varphi}(P,R,A) \ \leftarrow \ B \geq T1,\ B \leq T2, \\ \quad \mathit{tan(P,R,B)},\ \mathit{tan\_range(A,T1,T2)}. \end{array} \tag{11}$$

OPRA Constraints. We add ASP rules to impose and process differential *OPRA* and same *OPRA* constraints.

Firstly, if two objects are related by a same *OPRA* constraint then their locations must be identical. Thus location of one of them can be deduced from the other, without loss of generality we calculate the lower-indexed one.

$$\begin{array}{l} \mathit{xloc}(P,X) \leftarrow P < R,\ \mathit{xloc}(R,X), \\ \quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra(N,K,L,I,G)}. \\ \mathit{xloc}(R,X) \leftarrow P > R,\ \mathit{xloc}(P,X), \\ \quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra(N,K,L,I,G)}. \\ \mathit{yloc}(P,Y) \leftarrow P < R,\ \mathit{yloc}(R,Y), \\ \quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra(N,K,L,I,G)}. \\ \mathit{yloc}(R,Y) \leftarrow P > R,\ \mathit{yloc}(P,Y), \\ \quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra(N,K,L,I,G)}. \end{array} \tag{12}$$

For a differential *OPRA* constraint, if the sector is linear (even indexed), the orientation of object L can be calculated from *varphi(P,R,A)*. Likewise, for a linear same *OPRA* constraint, orientation of one object (the lower indexed one) can be calculated from the other.

$$\begin{array}{l} \mathit{orient}(L, (A - 90.I/G)\%360) \leftarrow I\ \%\ 2 = 0,\ \mathit{varphi(P,R,A)}, \\ \quad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{single\_diff\_opra(N,K,L,I,J,G)}. \\ \mathit{orient}(K, (T - I.90/G)\%360) \leftarrow \ I\ \%\ 2 = 0,\ L > K, \\ \quad \mathit{orient}(L,T),\ \mathit{same\_opra}(N,K,L,I,G). \\ \mathit{orient}(L, (T + I.90/G)\%360) \leftarrow \ I\ \%\ 2 = 0,\ K > L, \\ \quad \mathit{orient}(K,T),\ \mathit{same\_opra}(N,K,L,I,G). \end{array} \tag{13}$$

For same OPRA constraints with angular sectors, we use the rules below to check them.

$$\begin{array}{l} \leftarrow \ I\ \%\ 2 = 1,\ (T - B)\ \%\ 360 \leq (I-1).90/G,\ \mathit{orient(K,B)}, \\ \quad \mathit{orient(L,T)},\ \mathit{same\_opra(N,K,L,I,G)}. \\ \leftarrow \ I\ \%\ 2 = 1,\ (T - B)\ \%\ 360 \geq (I+1).90/G,\ \mathit{orient(K,B)}, \\ \quad \mathit{orient(L,T)},\ \mathit{same\_opra(N,K,L,I,G)}. \end{array} \tag{14}$$

For an angular differential OPRA constraint, the guessed value of ϕ_L must be compatible with φ_{LK}. For example, if $\varphi_{LK} = 250^o$ and $I = 3$, $G = 4$ then each angular segment is 45^o and ϕ_L must be in the range $(250 - 2 \times 45^o, 250 - 45^o) = (160^o, 205^o)$. We must also handle the case when bounds are reverse (i.e. change due to modulo 360). For example, if $\varphi_{LK} = 280^o$ and $I = 13$, $G = 4$ then ϕ_L must be in the range $(280 - 7 \times 45^o, 280 - 6 \times 45^o) = (325^o, 10^o)$. In this case, ϕ_L must not be less than 325 and greater than 10.

$$\begin{array}{l} \leftarrow \ I\ \%\ 2 = 1,\ T \leq (A - (I+1).90/G)\ \%\ 360, \\ \quad (A - (I-1).90/G)\ \%\ 360 > (A - (I+1).90/G)\ \%\ 360, \\ \quad \mathit{orient(L,T)},\ \mathit{varphi(P,R,A)},\ \mathit{point(K,P)}, \\ \quad \mathit{point(L,R)},\ \mathit{single\_diff\_opra(N,K,L,I,G)}. \\ \leftarrow \ I\ \%\ 2 = 1,\ T \geq (A - (I-1).90/G)\ \%\ 360, \\ \quad (A - (I-1).90/G)\ \%\ 360 > (A - (I+1).90/G)\ \%\ 360, \\ \quad \mathit{orient(L,T)},\ \mathit{varphi(P,R,A)},\ \mathit{point(K,P)}, \\ \quad \mathit{point(L,R)},\ \mathit{single\_diff\_opra(N,K,L,I,G)}. \\ \leftarrow \ I\ \%\ 2 = 1,\ T \leq (A - (I+1).90/G)\ \%\ 360, \\ \quad T \geq (A - (I-1).90/G)\ \%\ 360, \\ \quad (A - (I-1).90/G)\ \%\ 360 < (A - (I+1).90/G)\ \%\ 360, \\ \quad \mathit{orient(L,T)},\ \mathit{varphi(P,R,A)},\ \ \mathit{point(K,P)}, \\ \quad \mathit{point(L,R)},\ \mathit{single\_diff\_opra(N,K,L,I,G)}. \end{array} \tag{15}$$

Quantitative Constraints. We process quantitative *HOPA* constraints to restrict the intrinsic orientation and the location of objects, by using a similar method to the *OPRA* constraints. For example, the ASP rules below handle the $\phi(K)$, $\psi(K,L)$, $\delta(K,L)$ constraints. In particular, the range constraints for $\phi(K)$, $\psi(K,L)$, $\delta(K,L)$ can be imposed by ASP rules. In the case of a precise $\psi(K,L)$, $\delta(K,L)$ constraint, orientation of one object can be deduced from the other.

$$\begin{array}{l} \leftarrow T < A1,\ \mathit{orient(K,T)},\ \mathit{direction\_range(N,K,A1,A2)}. \\ \leftarrow T > A2,\ \mathit{orient(K,T)},\ \mathit{direction\_range(N,K,A1,A2)}. \\ \mathit{orient}(L, (A - T)\%360) \leftarrow \mathit{varphi(P,R,A)}, \\ \quad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{psi\_precise(N,K,L,T)}. \\ \mathit{orient}(K, (T - A)\%360) \leftarrow \ L > K,\ \mathit{orient(L,T)}, \\ \quad \mathit{delta\_precise(N,K,L,A)}. \\ \mathit{orient}(L, (T + A)\%360) \leftarrow \ K > L,\ \mathit{orient(K,T)}, \\ \quad \mathit{delta\_precise(N,K,L,A)}. \\ \leftarrow \ (T - B)\%360 < A1,\ \mathit{orient(K,B)},\ \mathit{orient(L,T)}, \\ \quad \mathit{delta\_range(N,K,L,A1,A2)}. \\ \leftarrow \ (T - B)\%360 > A2,\ \mathit{orient(K,B)},\ \mathit{orient(L,T)}, \\ \quad \mathit{delta\_range(N,K,L,A1,A2)}. \end{array} \tag{16}$$

Distance Constraints. For a qualitative distance constraint or a quantitative distance range constraint, we check whether the numerical distance between the objects lies inside the respective interval. If the quantitative distance information is precise, then we ensure that the guessed coordinates satisfy the numerical distance.

$$\begin{array}{l} \mathit{ndist}(P,R,X^2 + Y^2) \leftarrow \mathit{xdist}(P,R,X),\ \mathit{ydist}(P,R,Y). \\ \leftarrow D < D1,\ \mathit{lower\_bound(U,D1)},\ \mathit{ndist(P,R,D)}, \\ \quad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{dist\_rel(N,K,L,U)}. \\ \leftarrow D > D2,\ \mathit{upper\_bound(U,D2)},\ \mathit{ndist(P,R,D)}, \\ \quad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{dist\_rel(N,K,L,U)}. \\ \leftarrow \ Y^2 \neq D^2 - X^2,\ \mathit{xdist(P,R,X)},\ \mathit{ydist(P,R,Y)}, \\ \quad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{distance(N,K,L,D)}. \end{array} \tag{17}$$

Henceforth the ASP program Π for checking consistency of *HOPA* constraints is composed of the rules (5)-(17) and the facts that represent the input. An answer set of the program Π specifies a configuration of oriented objects on the

grid, identified by $xloc(P,X)$, $yloc(P,Y)$, $orient(K,A)$ atoms. Theorem 2 states that Π is a sound and complete solution for H over domain $D_{s,t,z}$. However, if the resolution parameters s,t,z are less than the sensor precision (that constraints are measured) i.e. the grid size is smaller than actual, then theoretically Π may not yield an answer set though constraints are satisfiable (this never happened in our experiments).

Theorem 2. *Let $H=(C,V,D_{s,t,z})$ be an HOPA consistency problem and $\mathcal{O}_{s,t,z}$ denote the set of all $xloc(P,X)$, $yloc(P,Y)$, $orient(K,A)$, $point(K,P)$ atoms. An assignment X from $D_{s,t,z}$ to objects in V is a solution of H if and only if X can be represented as $X=Z\cap\mathcal{O}_{s,t,z}$ for some answer set Z of Π. Moreover, every solution of H can be represented in this form in only one way.*

Inferring New Knowledge

If the given information is incomplete, agents may need to deduce new knowledge. The desired relations to infer can be specified at the input by $toinfer\_opra(K,L,G)$, $toinfer\_distrel(K,L)$, ... type atoms. The unknown relations can be inferred from the generated location and orientation of objects in the answer set. For example,

$$\begin{aligned}
&inferred_orient(K,A) \leftarrow orient(K,A),\ toinfer_orient(K).\\
&inferred_same_opra(K,L,Z/(90/G),G) \leftarrow\\
&\quad Z\%(180/G)=0,\ orient_diff(K,L,Z),\\
&\quad same_loc(K,L),\ toinfer_opra(K,L,G).\\
&inferred_distrel(K,L,U) \leftarrow D\geq D1,\ D\leq D2,\\
&\quad upper_bound(U,D1),\ upper_bound(U,D2),\\
&\quad ndist(P,R,D),\ point(K,P),\ point(L,R),\\
&\quad toinfer_distrel(K,L).
\end{aligned} \tag{18}$$

Note that there may be multiple configuration of objects that satisfy the given constraints and thus inferred relations may not be unique. All possible inferred relations can be obtained by computing all answer sets with the ASP solver.

Explaining the Source of Inconsistency

It might be the case that the *HOPA* constraints in C are inconsistent with each other, i.e. C is unsatisfiable. However, when we exclude some constraints from C, it may become satisfiable. In that sense, the set of excluded constraints explain a source of inconsistency in the original set C. To identify the source of inconsistency, we nondeterministically select which constraints to drop. $drop(N)$ atom indicates that the constraint with ID number N is dropped. The mandatory (non-defeasible) constraints are specified by *mandatory(N)* atom in the input. Each dropped constraint adds 1 unit cost and the ASP solver tries to minimize the total cost. We also revise the ASP rules so that the dropped constraints do not apply.

$$\begin{aligned}
&\{drop(N)\}\leq 1 \leftarrow not\ mandatory(N),\ constraint_id(N).\\
&\overset{\sim}{\leftarrow}\ drop(N)\quad [1@1,N].
\end{aligned} \tag{19}$$

Presumed Information

In some situations, agents might have prior or presumed information, for example the port station knows that ship 4 planned to sail from Izmir port to Gocek Marina, so its heading is presumably towards Gocek. We call them presumed *HOPA* constraints. The reasoning agent should consider these (qualitative or quantitative) presumed constraints unless they conflict with the existing *HOPA* constraints. In this sense, the presumed constraints are defeasible. We represent presumed constraints by *presumed_diff_opra(N,K,L,I,J,G)*, *presumed_same_opra(N,K,L,I,G)*, *presumed_orient(N,K,A)* and similar atoms. We minimize the number of violated presumed constraints by

$$\begin{aligned}
&\{violated(N)\}\leq 1 \leftarrow presumed_constraint_id(N).\\
&\overset{\sim}{\leftarrow}\ violated(N)\quad [1@2,N].
\end{aligned} \tag{20}$$

The priority of the weak constraint in rule (19) is higher than that of rule (20) because satisfiability of original *HOPA* constraints is more important. If the presumed constraint applies, the corresponding atoms are generated. For example,

$$\begin{aligned}
&diff_opra(N,K,L,I,J,G) \leftarrow not\ violated(N),\\
&\quad presumed_diff_opra(N,K,L,I,J,G).
\end{aligned} \tag{21}$$

Applications of *HOPA*

Marine Navigation

We first consider fleet management of a group of six vessels sailing in the Aegean Sea and the port station at Dalyan. The ships report the following data to the station: Vessel 4 observes vessel 2 near and on its starboard 147^o and 2 is staying motionless at Kosedere marina. 5 reports its location is $(18,11)$ and he observes 4 on its right-front. Vessel 6 reports that it is distant from the station by 20-25 km. Vessel 3 states that 5 is on its port direction between 70^o-100^o, and 6 is on his behind and not near. 7 observes 6 on his clock angle 9 and very near to it.

The agent at port station also knows that vessel 6 had a plan to go to Kumburun port. Based on the collected information, the port station wants to (1) verify that all measured data are correct and (2) find out the orientation of vessel 3 and its position w.r.t. 4. For this, the station agent uses the main ASP program augmented with subprograms for inference and presumed constraints. The grid is 40x30 with each slot 1 km and the angular resolution 3^o. The combined program yields an answer set (Fig. 3), and from the *inferred_orient(3,245)*, *inferred_diff_opra(3,4,15,4)* atoms, the agent infers that 3 is moving 245^o and right-front of 4.

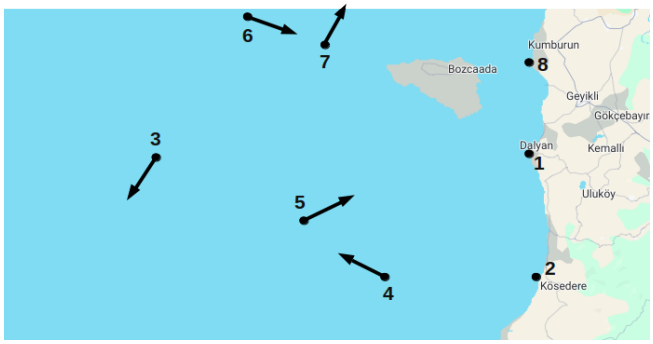

Figure 3: ASP output of marine navigation scenario

Cognitive Robotics

Next we study the robotic sensing problem mentioned in the introduction. The task of the robot is to navigate to the red cube behind the blue cube. To achieve this, the robot first needs to identify which red cube is behind the blue cube (C). In this mission, the robot makes two perceptions at point R1 and R2 with its sensor. The sensor data at these points are stated as *OPRA* relations (Moratz, Dylla, and Frommberger 2005) as below. Here P_R denotes the object located at point P, oriented toward R and ${}_R P$ denotes the object located at point P, whose orientation is vector $\overrightarrow{RP}$.
$(R1)\ R1_{R2\ 4}\angle 1\ R1_C\ ,\ R1_{R2\ 4}\angle 1\ R1_{B1}\ ,\ R1_{R2\ 4}\angle 15\ R1_{B2}$
$(R2)\ R1_{R2\ 4}\angle_0^8{}_{R1}\ R2\ ,\ {}_{R1}\ R2_{\ 4}\angle 4\ R2_C\ ,$
${}_{R1}\ R2_{\ 4}\angle 1\ R2_{B1}\ ,\quad {}_{R1}\ R2_{\ 4}\angle 13\ R2_{B2}$

Based on perceived information, the robot must determine which object(s) are behind the cube i.e. check whether the following (disjunctive) constraints hold: $B1_{\ 4}\not\angle^{\{7,8,9\}}\ C_{R1}\ ,\quad B2_{\ 4}\not\angle^{\{7,8,9\}}\ C_{R1}$
One method to solve this problem is inference as in the first scenario, an alternative method we present here is checking (in)consistency. We impose the physical sensor data as mandatory *OPRA* constraints; we add the candidate relations above as non-mandatory constraints. Then we utilize the main ASP program together with the subprogram for inconsistency which tries to satisfy the maximum number of constraints. At an optimal answer set, the first disjunctive constraint is satisfied but the second is not. Thus object B1 is behind C but B2 cannot be. The layout of objects in this solution is shown below. With composition and inversion based reasoning, Moratz, Dylla, and Frommberger (2005) could not reject that B2 is behind C. This constitutes another example that path-consistency type approaches are insufficient for *OPRA*. Thus our model is better suited for checking consistency and reasoning with *OPRA* and *HOPA*. The setup of this scenario is also relevant in other applications such as surveying, mapping, landmark localization. An example of surveying street networks using *OPRA* is Lücke, Mossakowski, and Moratz (2011).

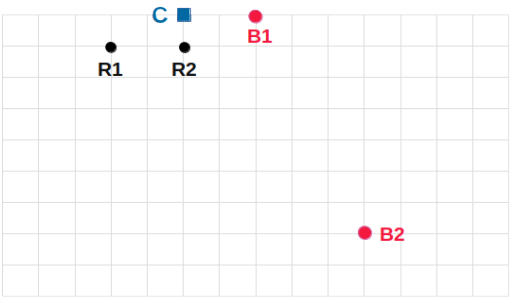

Figure 4: ASP output of cognitive robotics scenario

Experimental Evaluations

We perform experiments to assess computational efficiency of our ASP formulation and observe the impact of input size, grid, granularity and type of constraints. All tests are performed on a Linux workstation with Intel i9-9900K 3.6GHz CPU, 64GB memory using the ASP solver Clingo 5.4.0 on a single core. We first make experiments to evaluate the performance of consistency checking with basic *HOPA* constraints. For a parameter combination, number of objects (N), constraints (C) and grid size (S), we have created 100 consistent and 100 inconsistent random benchmark instances (samples). We take grid size equal in horizontal and vertical dimensions i.e. $s = t$ and angular resolution as 1^o while creating instances. Consistent problem instances are created by randomly picking location and heading of objects on the respective grid. The pair of objects and the type of constraints (qualitative/quantitative, orientation, distance, location) in the constraint set are randomly chosen and the true relations are extracted from the layout. Inconsistent instances are created by assigning random relations or numbers to those object pairs. We run the ASP program Π to check consistency of each problem instance, we have used angular resolution $z = 3^o$ in the experiments. We have recorded time statistics (in seconds) over 100 samples: Grnd shows average grounding time while Mean, Max, Std shows average, maximum and standard deviation of the solving time (grounding+search).

Instance			Consistent				Inconsistent			
N	C	Grid	Grnd	Mean	Max	Std	Grnd	Mean	Max	Std
40	40	50	2.41	2.86	7.15	0.88	0.66	0.67	2.79	0.45
40	40	100	3.70	7.27	45.8	6.42	1.27	1.30	5.44	0.68
40	60	50	3.41	6.00	76.9	7.88	0.89	0.89	2.91	0.25
40	60	100	5.48	18.0	94.9	16.6	1.71	1.71	4.76	0.45
60	60	50	3.72	4.85	19.8	1.93	0.86	0.87	2.68	0.23
60	60	100	5.83	11.9	65.1	8.93	1.79	1.79	6.13	0.63
60	100	50	5.91	11.5	45.0	7.40	1.39	1.40	2.04	0.20
80	80	50	4.91	6.35	13.8	1.59	1.14	1.15	1.74	0.16
80	120	50	7.23	11.4	27.3	3.83	1.70	1.71	2.19	0.21
100	100	50	6.15	8.10	19.8	1.95	1.40	1.40	1.85	0.17
100	150	50	9.22	15.2	36.6	5.32	2.13	2.13	2.69	0.22
120	120	50	7.68	10.2	17.2	2.04	1.73	1.74	2.16	0.19
120	150	50	9.35	14.2	39.2	4.86	2.16	2.17	2.80	0.23

Table 1: Results for basic *HOPA* constraints

According to the results in Table 1, the grounding time and total computation time increase as the number of objects, constraints and the grid size increase. In general, the computation time and timeout values for inconsistent instances are lower compared to consistent instances. The reason is that for inconsistent instances, the relations and numerical values in constraints are randomly generated, thus it is faster to eliminate invalid values over the search tree. On the other side, for a consistent instance, the solver needs to search for value of each parameter to satisfy all constraints.

Next we perform experiments with disjunctive *HOPA* constraints, to observe the impact of their presence in the constraint set. We have created 100 consistent and 100 inconsistent problem instances with disjunctive constraints as follows: We take a basic (in)consistent instance for a parameter combination N, C, S and convert $\%20$ or $\%40$ of the constraints into disjunctive constraints. In an instance, TxD denotes that there are T disjunctive constraints, each having D disjuncts. For the disjuncts, we keep the original basic relation and add other $D-1$ randomly generated basic *HOPA* relations as disjuncts. In experiments, we have used grid size 50x50, angular resolution 3^o and recorded compu-

tation time. With disjunctive constraints, the grounding and total computation time increase but the increase is not large.

Instance			Consistent				Inconsistent			
N	C	Disj	Grnd	Mean	Max	Std	Grnd	Mean	Max	Std
40	40	8x2	3.12	4.07	24.9	2.69	0.99	1.00	4.58	0.83
40	40	8x4	4.39	5.17	12.9	1.56	1.35	1.37	6.70	1.19
40	40	8x8	6.55	7.35	13.8	1.72	2.13	2.26	19.8	2.47
40	40	16x2	3.88	4.95	34.0	3.18	1.48	1.84	32.6	3.47
40	40	16x4	6.24	7.41	22.1	2.42	2.37	2.68	14.8	2.89
40	40	16x8	10.9	12.0	31.2	2.51	4.04	4.67	31.4	5.33
60	60	12x2	4.80	6.12	21.9	2.53	1.16	1.16	4.54	0.60
60	60	12x4	6.48	7.79	20.4	2.11	1.58	1.59	6.46	0.79
60	60	12x8	9.96	11.4	17.8	1.95	2.42	2.46	12.0	1.55
60	60	24x2	5.94	7.45	19.5	2.21	1.72	1.92	20.9	2.47
60	60	24x4	9.60	11.4	26.9	2.82	2.65	2.81	20.5	2.79
60	60	24x8	16.8	18.8	37.2	2.95	4.65	5.16	31.9	5.81

Table 2: Results for disjunctive *HOPA* constraints

Since consistency checking problem in *OPRA* has no (partial) solution in the literature, we also test performance of our framework for checking consistency of *OPRA* constraints. For this, we have created 100 consistent and 100 inconsistent problem instances with solely basic *OPRA* constraints, for a given number of objects (N) and constraints (C). The method of creating these random benchmark instances is similar to *HOPA*. In order to observe the impact of granularity, we set granularity (G) of *OPRA* constraints uniform in an instance. In the experiments s, aR denotes the grid size is sxs and angular resolution is R^o.

Instance				Consistent				Inconsistent			
N	C	G	Grid	Grnd	Mean	Max	Std	Grnd	Mean	Max	Std
10	10	4	50,a3	1.31	1.66	3.63	0.41	0.69	8.35	94.2	21.4
10	10	4	100,a2	2.39	5.59	33.2	4.83	1.25	2.32	32.5	4.34
10	10	6	50,a3	1.37	2.04	5.45	0.84	0.69	8.78	95.9	21.8
10	10	6	100,a2	2.47	6.97	37.9	5.10	1.23	3.59	81.8	9.66
10	20	4	50,a3	2.81	7.16	41.2	5.29	1.18	1.19	3.04	0.41
10	20	4	100,a2	5.09	35.7	92.6	23.6	2.13	2.16	6.44	0.79
20	20	2	50,a3	2.51	3.26	8.23	0.88	1.40	5.88	90.3	13.5
20	20	4	50,a3	2.88	5.21	23.9	3.36	1.52	6.71	95.7	16.0
20	40	2	50,a3	5.09	24.64	97.5	19.6	2.14	2.15	4.66	0.64
40	40	2	50,a3	5.31	9.90	33.9	4.88	2.80	7.63	87.8	16.2
60	60	2	25,a3	7.21	8.99	15.4	1.54	4.01	6.81	86.9	11.2

Table 3: Results for basic *OPRA* constraints

Note that these are (double-sided) differential *OPRA* constraints, each of them actually includes two constraints. We observe that grounding time and total computation time increase as the number of objects, constraints, granularity and grid size increase. Though average computation times are comparable, inconsistent instances exhibit greater maximum time, standard deviation and timeout values. This is mainly because *OPRA* constraints are qualitative and the solver needs to search many combinations on the search tree to decide inconsistency. Notice that computation time for consistent instances are higher when the number of constraints or granularity is large. Our interpretation is that finding a solution is harder when the constraints are dense (number) or they are fine (granularity).

Related Literature

Relative orientation information with *OPRA* has been utilized in robotic motion (Glez-Cabrera, Álvarez-Bravo, and Díaz 2013), marine navigation (Dylla et al. 2007), topological map learning (Wallgrün 2010), surveying (Lücke, Mossakowski, and Moratz 2011). However reasoning with *OPRA* is NP-hard (Wolter and Lee 2010) and currently there is no mechanism that can decide consistency of even basic *OPRA* constraints. Existing tools use composition and path-consistency for reasoning which is incomplete (Mossakowski and Moratz 2010) and cannot generate configuration of objects.

Answer Set Programming has been applied to spatial reasoning (Baryannis et al. 2018, 2020; Brenton, Faber, and Batsakis 2016; Li 2012). However these approaches are based on path-consistency and do not involve commonsense knowledge or assumptions. Izmirlioglu and Erdem (2023) has proposed a framework for reasoning with relative direction of stationary, extended objects using cardinal directions (north, east, southwest). Their setup involves commonsense knowledge, but it is purely qualitative and does not include distance or location information. ASP has not yet been applied to reasoning with *OPRA* or distance relations.

Other studies which utilize commonsense knowledge for spatial reasoning are Walega, Schultz, and Bhatt (2017) for inertia in topological relations, and Izmirlioglu and Erdem (2020) for cardinal relations over 3D extended objects. These models also use ASP for integrating commonsense knowledge into reasoning.

As for explanation generation in ASP, different topics such as knowledge-based diagnosis, finding conflicts/inconsistencies, justification of answer sets and debugging have been applied to various fields of computational sciences. Review of these approaches is provided by Fandinno and Schulz (2019), Redl (2018), Syrjänen (2006).

Conclusion

Spatial reasoning about orientation and position is beneficial for cognitive robotics, land, marine, UAV navigation, surveying and localization applications. In these domains metric data can be utilized in reasoning when available. Additionally, qualitative or coarse spatial information is also relevant because human agents tend to use symbolic terms of natural language; moreover spatial data obtained from sensors may be imprecise. In this paper, we have proposed a new hybrid formalism *HOPA* for position, orientation, distance and an ASP-based framework for reasoning. This framework can reason with qualitative and quantitative constraints, explain source of inconsistency, incorporate uncertain and presumed information and infer unknown relations. The framework can also provide solution to the consistency problem of *OPRA* for which the existing literature does not have a solution. We have shown benefits and uses of *HOPA* by scenarios from marine navigation and robotic sensing and motion. We have experimentally tested our ASP-based framework for consistency checking of basic and disjunctive *HOPA* and *OPRA* constraints, and measured its efficiency. Future work involves other constraint programming methods to increase scalability of reasoning with *HOPA*.

References

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. New York, NY, USA: Cambridge University Press. ISBN 0521818028.

Baryannis, G.; Tachmazidis, I.; Batsakis, S.; Antoniou, G.; Alviano, M.; and Papadakis, E. 2020. A generalised approach for encoding and reasoning with qualitative theories in answer set programming. *Theory and Practice of Logic Programming*, 20(5): 687–702.

Baryannis, G.; Tachmazidis, I.; Batsakis, S.; Antoniou, G.; Alviano, M.; Sellis, T.; and Tsai, P.-W. 2018. A Trajectory Calculus for Qualitative Spatial Reasoning Using Answer Set Programming. *Theory and Practice of Logic Programming*, 18(3-4): 355–371.

Brenton, C.; Faber, W.; and Batsakis, S. 2016. Answer Set Programming for Qualitative Spatio-Temporal Reasoning: Methods and Experiments. In *OASICS-OpenAccess Series in Informatics*, volume 52. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Clementini, E.; Di Felice, P.; and Hernández, D. 1997. Qualitative representation of positional information. *Artificial intelligence*, 95(2): 317–356.

Dylla, F.; Frommberger, L.; Wallgrün, J. O.; Wolter, D.; Nebel, B.; and Wölfl, S. 2007. SailAway: Formalizing navigation rules. In *Proceedings of the Artificial and Ambient Intelligence Symposium on Spatial Reasoning and Communication, AISB*, volume 7, 1–5.

Dylla, F.; and Moratz, R. 2004. Empirical complexity issues of practical qualitative spatial reasoning about relative position. In *Workshop on Spatial and Temporal Reasoning at ECAI*, volume 2004.

Fandinno, J.; and Schulz, C. 2019. Answering the "why" in answer set programming–A survey of explanation approaches. *Theory and Practice of Logic Programming*, 19(2): 114–203.

Frommberger, L. 2008. Learning to behave in space: A qualitative spatial representation for robot navigation with reinforcement learning. *International Journal on Artificial Intelligence Tools*, 17(03): 465–482.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gelfond, M.; and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. New York, NY, USA: Cambridge University Press.

Gelfond, M.; and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP*, 1070–1080. MIT Press.

Gelfond, M.; and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4): 365–385.

Glez-Cabrera, F. J.; Álvarez-Bravo, J. V.; and Díaz, F. 2013. QRPC: A new qualitative model for representing motion patterns. *Expert systems with applications*, 40(11): 4547–4561.

Izmirlioglu, Y.; and Erdem, E. 2020. Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming. *Theory and Practice of Logic Programming*, 20(6): 942–957.

Izmirlioglu, Y.; and Erdem, E. 2023. Qualitative reasoning about 2D cardinal directions using answer set programming. *Journal of Artificial Intelligence Research*, 77: 1371–1453.

Lee, J. H. 2014. The complexity of reasoning with relative directions. In *Proc. of ECAI 2014*, 507–512. IOS Press.

Li, J. J. 2012. Qualitative spatial and temporal reasoning with answer set programming. In *Proc. of ICTAI*, volume 1, 603–609. IEEE.

Lifschitz, V. 2019. *Answer Set Programming*. Springer. ISBN 9783030246570.

Lücke, D.; Mossakowski, T.; and Moratz, R. 2011. Streets to the OPRA—finding your destination with imprecise knowledge. In *Proc. IJCAI Workshop on Benchmarks and Applications of Spatial Reasoning*, 25–32.

Maaref, H.; and Barret, C. 2002. Sensor-based navigation of a mobile robot in an indoor environment. *Robotics and Autonomous systems*, 38(1): 1–18.

Miguel-Tomé, S. 2021. The Heuristic of Directional Qualitative Semantic: A New Heuristic for Making Decisions about Spinning with Qualitative Reasoning. *Robotics*, 10(1): 17.

Moratz, R.; Dylla, F.; and Frommberger, L. 2005. A relative orientation algebra with adjustable granularity. In *Proceedings of the workshop on agents in real-time and dynamic environments (IJCAI 2005)*, volume 21, 22.

Moratz, R.; and Ragni, M. 2008. Qualitative spatial reasoning about relative point position. *Journal of Visual Languages & Computing*, 19(1): 75–98.

Mossakowski, T.; and Moratz, R. 2010. Qualitative reasoning about relative direction on adjustable levels of granularity. *arXiv preprint arXiv:1011.0098*.

Redl, C. 2018. Inconsistency in Answer Set Programs and Extensions.

Syrjänen, T. 2006. Debugging inconsistent answer set programs. In *Proceedings of NMR*, volume 6, 77–83.

Walega, P. A.; Schultz, C.; and Bhatt, M. 2017. Nonmonotonic spatial reasoning with answer set programming modulo theories. *Theory and Practice of Logic Programming*, 17(2): 205–225.

Wallgrün, J. O. 2010. Qualitative spatial reasoning for topological map learning. *Spatial Cognition & Computation*, 10(4): 207–246.

Wolter, D.; and Lee, J. 2010. On qualitative reasoning about relative point position. *Artificial Intelligence*, 174: 1498–1507.