Appendix for the Paper “Hybrid Reasoning about Relative Position and Orientation of Objects and Navigating Agents using Answer Set Programming”

Yusuf Izmirlioglu

December 15, 2024

1 Proof of Theorem 1

Theorem 1 states that checking consistency of a set of *HOPA* constraints is an NP-complete problem.

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

Proof. We prove NP-membership and NP-hardness of $H = (C, V, D_{s,t,z})$ as below:

NP-membership: We first note that the domain of *HOPA* constraints $D_{s,t,z}$ is finite, thus a nondeterministic Turing machine can decide H in a finite number of steps. Testing an *HOPA* constraint in C between a pair of objects takes $O(1)$ time. This means that given a candidate solution A of H, it takes $O(|C|)$ time to verify all constraints in C. Hence, $H \in NP$.

NP-hardness: To prove NP-hardness, we use the methodology in Theorem 8 and Corollary 12 of [2]. We consider another decision problem (consistency checking) $S = (C, V)$ where V is a set of oriented points (such that the intrinsic orientation ϕ can be either 0^o or 180^o) and C is a set of single-sided *OPRA*$_1$ constraints between these oriented points over the continuous space $\mathbb{R}^2$. Namely constraints in C indicate whether an object v_k is to the right, left, linear or same location of another object v_l. Next we make reduction from the problem of realizability problem for oriented matroids (ROM) [2] to S. For this, we consider an oriented matroid $\chi :\ \{1,\ ...,\ |V|\}^3 \mapsto \{-1,\ 0,\ +1\}$. Then we make the following transformation: For a differential OPRA constraint $v_k\ {}_1\angle^i\ v_l$ in C, we assign $\chi(k,l) = +1$ if $i = 1$ (left), $\chi(k,l) = -1$ if $i = 3$ (right) and $\chi(k,l) = 0$ if $i \in \{0, 2\}$ (linear). For a same OPRA constraint $v_k\ {}_1\angle i\ v_l$ in C, we assign $\chi(k,l) = 0$ (same location). The number of such $\chi(k,l)$ pairs is $O(|C|)$, hence complexity of this transformation is polytime. This manner, we have reduced the ROM problem (realizability of matroid χ) to an instance S_χ. As in Theorem 8, 9 of [2], the oriented matroid χ is realizable if and only if the corresponding set of *OPRA*$_1$ constraints is satisfiable. That is, the answer of ROM problem of χ is Yes if and only if the answer of S_χ is Yes. Therefore S is an NP-hard problem.

In the next step, we show that $S = (C, V)$, consistency checking problem of single-sided *OPRA*$_1$ constraints, can be solved over the discrete space. We consider the discrete space $D_{s,t,z} = \Xi_{s,t} \times \Phi_z$ where $s = t = |V|$ and $z = 2$. Recall that intrinsic orientation ϕ of an object can take two values (0^o or 180^o), hence orientation vector is parallel to the horizontal axis. *OPRA*$_1$ relations can only distinguish left/right/linear. In the worst case, the location of each object is a different point on the horizontal (or vertical axis). Thus if C is consistent, then *OPRA*$_1$ constraints in C can be realized over a grid of size $s = t = |V|$ and $z = 2$. For the reverse case, if *OPRA*$_1$ constraints in C can be realized over the finite discrete space $D_{s,t,z}$, then clearly they are realized over the continuous space $\mathbb{R}^2$. Namely, the answers of the original problem $S = (C, V)$ and discretized problem $S_d = (C, V, D_{s,t,z})$ are the same. Therefore $S_d = (C, V, D_{s,t,z})$ is also NP-hard.

Notice that S_d is a special case of *HOPA* consistency problem $H = (C, V, D_{s,t,z})$ with parameters $s = t = |V|$ and $z = 2$. Thus consistency checking problem $H = (C, V, D_{s,t,z})$ is NP-hard.

□

2 Full ASP Program for *HOPA*

Let $H = (C, V, D_{s,t,z})$ be a consistency checking problem where C is a set of *HOPA* constraints, V is a set of objects and $D_{s,t,z}$ is the discrete domain.

We develop an ASP program for *HOPA* consistency checking problem as below.

Represent the input. The input of the problem H is represented by a set of facts in ASP. The grid is specified by *xcoord*$(1..\overline{x})$, *ycoord*$(1..\overline{y})$ and the angular resolution by *angle_res*(z) atoms. 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 denote point identifier of an object with another atom *point*(v, p). v and p are just variables and we enumerate them with $1..|V|$, $1..|P|$. In general, if there is no information that objects share the same location, then point identifier can be chosen the same as the 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.

Disjunctive Constraints. Disjunctive *HOPA* constraints represent uncertainity 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, 6:intrinsic orientation range constraint, 7: intrinsic orientation vector constraint, 8:quantitative precise ψ constraint, 9:quantitative ψ range constraint, 10:quantitative precise δ constraint, 11:quantitative δ range constraint, 12:quantitative location constraint, 13:quantitative distance range constraint, 14:quantitative precise distance constraint.

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 heterogenous disjunctive constraints where the pair of object(s) or 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.

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

Then we generate the basic *HOPA* constraint corresponding to the selected disjunct. The rule below generates the basic constraint for each of the 14 type of disjunct.

$$\begin{array}{l}
\textit{same\_opra(N,L,K,I,G)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,1,K,L,I,G)}.\\
\textit{one\_sided\_diff\_opra(N,K,L,I,G)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,2,K,L,I,G)}.\\
\textit{diff\_opra(N,K,L,I,J,G)} \leftarrow \textit{chosen(N,D)}, \textit{disjrelation(N,D,3,K,L,I,J,G)}.\\
\textit{dist\_rel(N,K,L,U)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,4,K,L,U)}.\\
\textit{direction(N,K,A)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,5,K,A)}.\\
\textit{direction\_range(N,K,A1,A2)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,6,K,A1,A2)}.\\
\textit{direction(N,K,P,R)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,7,K,P,R)}.\\
\textit{psi\_precise(N,K,L,A)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,8,K,L,A)}.\\
\textit{psi\_range(N,K,L,A1,A2)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,9,K,L,A1,A2)}.\\
\textit{delta\_precise(N,K,L,A)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,10,K,L,A)}.\\
\textit{delta\_range(N,K,L,A1,A2)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,11,K,L,A1,A2)}.\\
\textit{location(N,K,X,Y)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,12,K,X,Y)}.\\
\textit{distance\_range(N,K,L,Z1,Z2)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,13,K,L,Z1,Z2)}.\\
\textit{distance(N,K,L,Z)} \leftarrow \textit{chosen(N,D)},\ \textit{disjrelation(N,D,14,K,L,Z)}.
\end{array} \tag{2}$$

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

$$\begin{array}{l}
\textit{single\_diff\_opra}(N,K,L,I,G) \leftarrow \textit{diff\_opra}(N,K,L,I,J,G).\\
\textit{single\_diff\_opra}(N,L,K,J,G) \leftarrow \textit{diff\_opra}(N,K,L,I,J,G).\\
\textit{single\_diff\_opra}(N,K,L,I,G) \leftarrow \textit{one\_sided\_diff\_opra}(N,K,L,I,G).
\end{array} \tag{3}$$

$\textit{single\_diff\_opra}(N,K,L,I,G)$ reads as object K is on sector I of L with granularity G.

To detect inconsistency faster, we check whether OPRA constraints specified for the same pair of objects are compatible with each other. In particular, same OPRA constraint and differential OPRA constraint must not be specified for the same pair.

$$\begin{array}{l}
\leftarrow\ \textit{single\_diff\_opra}(M,K,L,I,G1),\ \textit{same\_opra}(N,L,K,J,G2).\\
\leftarrow\ \textit{single\_diff\_opra}(M,K,L,I,G1),\ \textit{same\_opra}(N,K,L,J,G2).\\
\leftarrow\ I \neq J,\ \textit{single\_diff\_opra}(M,K,L,I,G),\ \textit{single\_diff\_opra}(N,K,L,J,G).\\
\leftarrow\ I,\ J \neq 0,\ I+J \neq 4G,\ \textit{same\_opra}(M,K,L,I,G),\ \textit{same\_opra}(N,L,K,J,G).
\end{array} \tag{4}$$

Note that only the objects and their points which appear in at least one constraint must be instantiated. We identify the points to be instantiated by the $\textit{instantiate}(P)$ atoms.

$$\begin{array}{l}
\textit{instantiate}(P) \leftarrow \textit{point}(K,P),\ \textit{diff\_opra}(N,K,L,I,J,G).\\
\textit{instantiate}(R) \leftarrow \textit{point}(L,R),\ \textit{diff\_opra}(N,K,L,I,J,G).\\
\textit{instantiate}(P) \leftarrow \textit{point}(K,P),\ \textit{same\_opra}(N,K,L,I,G).\\
\textit{instantiate}(R) \leftarrow \textit{point}(L,R),\ \textit{same\_opra}(N,K,L,I,G).
\end{array} \tag{5}$$

and similar rules for other *HOPA* constraints.

We also identify objects whose intrinsic orientation (heading) is necessary[1] and needs to be generated:

$$\begin{array}{l}
\mathit{generate\_orient}(K) \leftarrow \mathit{diff\_opra}(N,K,L,I,J,G). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{diff\_opra}(N,K,L,I,J,G). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{one\_sided\_diff\_opra}(N,K,L,I,G). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{psi\_precise}(N,K,L,A). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{psi\_range}(N,K,L,A). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{delta\_precise}(N,K,L,A). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{delta\_precise}(N,K,L,A). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{delta\_range}(N,K,L,A1,A2). \\
\mathit{generate\_orient}(L) \leftarrow \mathit{delta\_range}(N,K,L,A1,A2). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{direction}(N,K,A). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{direction}(N,K,P,R). \\
\mathit{generate\_orient}(K) \leftarrow \mathit{direction\_range}(N,K,A1,A2).
\end{array} \tag{6}$$

We process qualitative and quantitative *HOPA* constraints to identify whether location, orientation or distance of objects are already known (or can be deduced). If there is a quantitative location constraint, then the coordinates of the object is already known. In the case of quantitative delta constraint or same OPRA constraint, we know objects share the same location, thus location of one object can be deduced from the other. Without loss of generality, we deduce the location of the object with the smaller index.

$$\begin{array}{l}
\mathit{loc\_known}(P) \leftarrow \mathit{point}(K,P),\ \mathit{location}(N,K,X,Y). \\
\mathit{loc\_known}(P) \leftarrow P<R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{loc\_known}(R) \leftarrow P>R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{loc\_known}(P) \leftarrow P<R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{delta\_precise}(N,K,L,A). \\
\mathit{loc\_known}(R) \leftarrow P>R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{delta\_precise}(N,K,L,A). \\
\mathit{loc\_known}(P) \leftarrow P<R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{delta\_range}(N,K,L,A1,A2). \\
\mathit{loc\_known}(R) \leftarrow P>R,\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{delta\_range}(N,K,L,A1,A2).
\end{array} \tag{7}$$

If there is a quantitative orientation constraint or linear OPRA constraint or quantitative relative position constraint, then the orientation (heading) of the object is already known. Likewise, if there is a quantitative distance constraint, we designate that the Euclidean distance between these points is known.

$$\begin{array}{l}
\mathit{orient\_specified}(K) \leftarrow \mathit{direction}(N,K,A). \\
\mathit{orient\_specified}(K) \leftarrow \mathit{direction}(N,K,P,R). \\
\mathit{orient\_specified}(L) \leftarrow I\ \%\ 2=0,\ \mathit{single\_diff\_opra}(N,K,L,I,G). \\
\mathit{orient\_specified}(K) \leftarrow L>K,\ I\ \%\ 2=0,\ \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{orient\_specified}(L) \leftarrow K>L,\ I\ \%\ 2=0,\ \mathit{same\_opra}(N,K,L,I,G). \\
\mathit{orient\_specified}(L) \leftarrow \mathit{psi\_precise}(N,K,L,T). \\
\mathit{orient\_specified}(K) \leftarrow L>K,\ \mathit{delta\_precise}(N,K,L,A). \\
\mathit{orient\_specified}(L) \leftarrow K>L,\ \mathit{delta\_precise}(N,K,L,A). \\
\mathit{dist\_known}(P,R) \leftarrow \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance}(N,K,L,D).
\end{array} \tag{8}$$

Instantiation of Object Location. A candidate solution of H is an instantiation of objects $u \in V$ by an oriented point in $D_{s,t,z}$. An oriented point is specified by its location and orientation (heading). Note

[1]For example, in a single-sided differential OPRA constraint, the orientation of the first object is not necessary unless it is relevant in other constraints

that only the objects and their points which appear in at least one constraint must be instantiated

We nondeterministically generate a location $(x, y) \in \Xi_{r,s}$ 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 and objects cannot occupy these locations.

$$\begin{array}{l} \{xloc(P,X):\ 0 \leq X < r\} = 1 \leftarrow\ not\ loc\_known(P),\ instantiate(P). \\ \{yloc(P,Y):\ 0 \leq Y < s\} = 1 \leftarrow\ not\ loc\_known(P),\ instantiate(P). \\ \leftarrow\ invalid\_loc(X,Y),\ xloc(P,X),\ yloc(P,Y),\ instantiate(P). \end{array} \tag{9}$$

The rule below assigns the location of the objects whose numerical value is given by the user.

$$\begin{array}{l} xloc(P,X) \leftarrow point(K,P),\ location(N,K,X,Y). \\ yloc(P,Y) \leftarrow point(K,P),\ location(N,K,X,Y). \end{array} \tag{10}$$

If two objects are related a same OPRA constraint or delta constraint, then their locations must be identical. Thus location of one of them (the lower indexed one) can be deduced from the other:

$$\begin{array}{l} xloc(P,X) \leftarrow P < R,\ xloc(R,X),\ point(K,P),\ point(L,R),\ same\_opra(N,K,L,I,G). \\ xloc(R,X) \leftarrow P > R,\ xloc(P,X),\ point(K,P),\ point(L,R),\ same\_opra(N,K,L,I,G). \\ yloc(P,Y) \leftarrow P < R,\ yloc(R,Y),\ point(K,P),\ point(L,R),\ same\_opra(N,K,L,I,G). \\ yloc(R,Y) \leftarrow P > R,\ yloc(P,Y),\ point(K,P),\ point(L,R),\ same\_opra(N,K,L,I,G). \\ xloc(P,X) \leftarrow P < R,\ xloc(R,X),\ point(K,P),\ point(L,R),\ delta\_precise(N,K,L,A). \\ xloc(R,X) \leftarrow P > R,\ xloc(P,X),\ point(K,P),\ point(L,R),\ delta\_precise(N,K,L,A). \\ yloc(P,Y) \leftarrow P < R,\ yloc(R,Y),\ point(K,P),\ point(L,R),\ delta\_precise(N,K,L,A). \\ yloc(R,Y) \leftarrow P > R,\ yloc(P,Y),\ point(K,P),\ point(L,R),\ delta\_precise(N,K,L,A). \\ xloc(P,X) \leftarrow P < R,\ xloc(R,X),\ point(K,P),\ point(L,R),\ delta\_range(N,K,L,A1,A2). \\ xloc(R,X) \leftarrow P > R,\ xloc(P,X),\ point(K,P),\ point(L,R),\ delta\_range(N,K,L,A1,A2). \\ yloc(P,Y) \leftarrow P < R,\ yloc(R,Y),\ point(K,P),\ point(L,R),\ delta\_range(N,K,L,A1,A2). \\ yloc(R,Y) \leftarrow P > R,\ yloc(P,Y),\ point(K,P),\ point(L,R),\ delta\_range(N,K,L,A1,A2). \end{array} \tag{11}$$

Angle Computation. For a differential OPRA constraint, we need to compute the vectors PR, RP and the angle of these vectors ϕ_{PR}, ϕ_{RP}. Note that $\phi_{LK} = (\phi_{KL} + 180) mod 360$ so it is sufficient to compute only one of them. We determine the pair of points (P, R) to compute their vector and angle φ_{PR}. By convention, we compute ϕ_{PR} if $P > R$.

$$\begin{array}{l} compute\_vector(P,R) \leftarrow P > R,\ point(K,P),\ point(L,R),\ diff\_opra(N,K,L,I,J,G). \\ compute\_vector(R,P) \leftarrow P < R,\ point(K,P),\ point(L,R),\ diff\_opra(N,K,L,I,J,G). \\ compute\_vector(P,R) \leftarrow point(K,P),\ point(L,R),\ one\_sided\_diff\_opra(N,K,L,I,G). \\ compute\_vector(P,R) \leftarrow point(K,P),\ point(L,R),\ psi\_precise(N,K,L,A). \\ compute\_vector(P,R) \leftarrow point(K,P),\ point(L,R),\ psi\_range(N,K,L,A1,A2). \\ compute\_vector(R,P) \leftarrow point(K,P),\ point(L,R),\ direction(N,K,P,R). \end{array} \tag{12}$$

In case of a two-sided differential OPRA constraint, the reverse angle φ_{RP} also needs to be computed. The below rule determines the pair of points to compute the reverse angle.

$$\begin{array}{l} compute\_reverse(P,R) \leftarrow P > R,\ point(K,P),\ point(L,R),\ diff\_opra(N,K,L,I,J,G). \\ compute\_reverse(R,P) \leftarrow P < R,\ point(K,P),\ point(L,R),\ diff\_opra(N,K,L,I,J,G). \end{array} \tag{13}$$

For a qualitative or quantitative distance constraint, we need to compute the difference between coordinates for the pair, unless it is already computed for vector calculation.

$$\begin{array}{l} compute\_loc\_diff(P,R) \leftarrow not\ compute\_vector(P,R),\ point(K,P),\ point(L,R),\ dist\_rel(N,K,L,U). \\ compute\_loc\_diff(P,R) \leftarrow not\ compute\_vector(P,R),\ point(K,P),\ point(L,R),\ distance(N,K,L,D). \\ compute\_loc\_diff(P,R) \leftarrow not\ compute\_vector(P,R),\ point(K,P),\ point(L,R),\ distance\_range(N,K,L,D1,D2). \end{array} \tag{14}$$

Having generated locations, we compute φ_{PR} for every differential OPRA constraint. For this, we first calculate the difference between coordinates and their sign.

$$\begin{array}{l} \mathit{xdist}(P, R, X1 - X2) \leftarrow \mathit{xloc}(P, X1),\ \mathit{xloc}(R, X2),\ \mathit{compute\_vector}(P, R). \\ \mathit{ydist}(P, R, Y1 - Y2) \leftarrow \mathit{yloc}(P, Y1),\ \mathit{yloc}(R, Y2),\ \mathit{compute\_vector}(P, R). \\ \mathit{xdist}(P, R, X1 - X2) \leftarrow \mathit{xloc}(P, X1),\ \mathit{xloc}(R, X2),\ \mathit{compute\_loc\_diff}(P, R). \\ \mathit{ydist}(P, R, Y1 - Y2) \leftarrow \mathit{yloc}(P, Y1),\ \mathit{yloc}(R, Y2),\ \mathit{compute\_loc\_diff}(P, R). \end{array} \tag{15}$$

We also compute the absolute value and sign of the distance over x,y axis for the vector.

$$\begin{array}{l} \mathit{xdist\_abs}(P, R, |DX|) \leftarrow \mathit{xdist}(P, R, DX),\ \mathit{compute\_vector}(P, R). \\ \mathit{ydist\_abs}(P, R, |DY|) \leftarrow \mathit{ydist}(P, R, DY),\ \mathit{compute\_vector}(P, R). \\ \mathit{xdist\_positive}(P, R) \leftarrow DX > 0,\ \mathit{xdist}(P, R, DX),\ \mathit{compute\_vector}(P, R). \\ \mathit{ydist\_positive}(P, R) \leftarrow DY > 0,\ \mathit{ydist}(P, R, DY),\ \mathit{compute\_vector}(P, R). \end{array} \tag{16}$$

Tangent of φ_{PR} is equal to ratio of the vector length in y and x axes i.e. $\mathit{tan}(\varphi_{PR}) = |PR|_y / |PR|_x$. We represent tan function in range $[0, 90^o]$ by internal ASP atoms[2]. To examplify, the range of $\mathit{tan}4^o$ is between $\mathit{tan}(3.5^o) = 0.068$ and $\mathit{tan}(4.5^o) = 0.078$ and this is represented by by $\mathit{tan\_range}(4, 0.062, 0.078)$ atom[3]. Considering the fact that $|PR|_y$ or $|PR|_x$ can have negative values, we first estimate the angle in $[0, 90^o]$ using the absolute value of $|PR|_y / |PR|_y$. Then we determine φ_{PR} based on the sign of $|PR|_y$, $|PR|_x$.

We first calculate the absolute value of the tangent for the vector $\overrightarrow{PR}$.

$$\mathit{tan\_abs}(P, R, DY/DX) \leftarrow DX > 0,\ DY > 0,\ \mathit{xdist\_abs}(P, R, DX),\ \mathit{ydist\_abs}(P, R, DY),\ \mathit{compute\_vector}(P, R). \tag{17}$$

The next rule computes the arctangent of the absolute value $|PR|_y$ / $|PR|_x$. Therefore $\mathit{qangle}(P, R, A)$ atom describes the corresponding value of the angle in the range in range $[0, 90^o]$.

$$\begin{array}{l} \mathit{qangle}(P, R, A) \leftarrow H \geq T1,\ H \leq T2,\ \mathit{tan\_abs}(P, R, H),\ \mathit{tan\_range}(A, T1, T2),\ \mathit{compute\_vector}(P, R). \\ \mathit{qangle}(P, R, 0) \leftarrow \mathit{ydist\_abs}(P, R, 0),\ \mathit{compute\_vector}(P, R). \\ \mathit{qangle}(P, R, 90) \leftarrow \mathit{xdist\_abs}(P, R, 0),\ \mathit{compute\_vector}(P, R). \end{array} \tag{18}$$

The rule below computes the vector angle φ_{PR}, using the angle $\mathit{qangle}(P, R, A)$ in the range $[0, 90^o]$ and the sign over x,y axis.

$$\begin{array}{l} \mathit{varphi}(P, R, A) \leftarrow \mathit{qangle}(P, R, A),\ \mathit{xdist\_positive}(P, R),\ \mathit{ydist\_positive}(P, R),\ \mathit{compute\_vector}(P, R). \\ \mathit{varphi}(P, R, 180 - A) \leftarrow \mathit{qangle}(P, R, A),\ \mathit{not}\ \mathit{xdist\_positive}(P, R),\ \mathit{ydist\_positive}(P, R),\ \mathit{compute\_vector}(P, R). \\ \mathit{varphi}(P, R, 360 - A) \leftarrow \mathit{qangle}(P, R, A),\ \mathit{xdist\_positive}(P, R),\ \mathit{not}\ \mathit{ydist\_positive}(P, R),\ \mathit{compute\_vector}(P, R). \\ \mathit{varphi}(P, R, 180 + A) \leftarrow \mathit{qangle}(P, R, A),\ \mathit{not}\ \mathit{xdist\_positive}(P, R),\ \mathit{not}\ \mathit{ydist\_positive}(P, R),\ \mathit{compute\_vector}(P, R). \end{array} \tag{19}$$

For a differential OPRA constraint, the reverse angle φ_{RP} can be computed from φ_{PR}.

$$\mathit{varphi}(R, P, (A + 180)\%360) \leftarrow \mathit{varphi}(P, R, A),\ \mathit{compute\_reverse}(P, R). \tag{20}$$

[2] An alternative approach is to make external function call to Python or Lua to compute arctangent but this yields higher computation time

[3] If the ASP solver does not allow real numbers, the tangent value can be multiplied by 10^b to convert to integer with b significant digits

Instantiation of Object Orientation (Heading). We also need to assign an orientation (heading) $\phi \in \Phi_z$ to the objects whose orientation is unknown. The angular resolution specifies possible values of the orientation. To examplify, if angular resolution is 3^o, then possible values are $\{0, 3, 6, ..., 357\}$. For this, we generate a value in $\Omega = \{0, 1, 2, ..., 119\}$ and multiply by 3. The intrinsic orientation of an object is represented by the *orient*(K, A) atom.

$$\{\mathit{degree}(K,E):\ E \in \Omega\} = 1 \ \leftarrow\ \mathit{not\ orient\_specified}(K),\ \mathit{generate\_orient}(K). \tag{21}$$

$$\mathit{orient}(K, E.Z)\ \leftarrow\ \mathit{degree}(K,E),\ \mathit{angle\_res}(Z),\ \mathit{not\ orient\_specified}(K),\ \mathit{generate\_orient}(K). \tag{22}$$

The next rule computes the orientation of objects whose value is directly specified by a quantitative constraint, or can be implicitly deduced from linear OPRA constraints or *delta_precise*(N, K, L, A), *psi_precise*(N, K, L, T) constraints.

$$\begin{array}{l}
\mathit{orient}(K,A) \leftarrow \mathit{direction}(N,K,A).\\
\mathit{orient}(K,A) \leftarrow \mathit{varphi}(R,P,A),\ \mathit{direction}(N,K,P,R).\\
\mathit{orient}(L,(A-90.I/G)\%360) \leftarrow I\,\%\,2=0,\ \mathit{varphi}(P,R,A),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G).\\
\mathit{orient}(L,(A-T)\%360) \leftarrow \mathit{varphi}(P,R,A),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_precise}(N,K,L,T).\\
\mathit{orient}(K,(T-I.90/G)\%360) \leftarrow\ I\,\%\,2=0,\ L>K,\ \mathit{orient}(L,T),\ \mathit{same\_opra}(N,K,L,I,G).\\
\mathit{orient}(L,(T+I.90/G)\%360) \leftarrow\ I\,\%\,2=0,\ K>L,\ \mathit{orient}(K,T),\ \mathit{same\_opra}(N,K,L,I,G).\\
\mathit{orient}(K,(T-A)\%360) \leftarrow\ L>K,\ \mathit{orient}(L,T),\ \mathit{delta\_precise}(N,K,L,A).\\
\mathit{orient}(L,(T+A)\%360) \leftarrow\ K>L,\ \mathit{orient}(K,T),\ \mathit{delta\_precise}(N,K,L,A).
\end{array} \tag{23}$$

We insist that orientation information for the same object L obtained from various qualitative and quantitative constraints must be consistent.

$$\begin{array}{l}
\leftarrow A1 \neq A2,\ \mathit{orient}(L,A1),\ \mathit{orient}(L,A2),\ \mathit{object}(L).\\
\leftarrow I\,\%\,2=0,\ (A-T)\,\%\,360 \neq I.90/G,\ \mathit{direction}(N,L,T),\ \mathit{varphi}(P,R,A),\\
\qquad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G).\\
\leftarrow I\,\%\,2=0,\ (A-T)\,\%\,360 \neq I.90/G,\ \mathit{varphi}(Z,Y,T),\ \mathit{varphi}(P,R,A),\\
\qquad \mathit{direction}(N,L,Y,Z),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G).\\
\leftarrow T \neq A,\ \mathit{varphi}(R,P,A),\ \mathit{direction}(N,K,T),\ \mathit{direction}(M,K,P,R).
\end{array} \tag{24}$$

OPRA Constraints We first impose the relative angular position information in the same OPRA constraints. We have already processed linear same OPRA constraints (even indexed) above. The rules below handle the case for same OPRA constraints with angular sectors.

$$\begin{array}{l}
\leftarrow\ I\,\%\,2=1,\ (T-B)\,\%\,360 \leq (I-1).90/G,\ \mathit{orient}(K,B),\ \mathit{orient}(L,T),\ \mathit{same\_opra}(N,K,L,I,G)\\
\leftarrow\ I\,\%\,2=1,\ (T-B)\,\%\,360 \geq (I+1).90/G,\ \mathit{orient}(K,B),\ \mathit{orient}(L,T),\ \mathit{same\_opra}(N,K,L,I,G).
\end{array} \tag{25}$$

For a differential OPRA constraint, we first impose that the locations of the objects are different.

$$\leftarrow\ \mathit{xdist}(P,R,0),\ \mathit{ydist}(P,R,0),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G). \tag{26}$$

If the sector is angular, the guessed value of ϕ_L must be compatible with φ_{KL}. 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 or equal to 325 and greater than or equal to 10.

$$\begin{array}{l}
\leftarrow\ I \,\%\, 2 = 1,\ T \leq (A - (I+1).90/G) \,\%\, 360,\ (A - (I-1).90/G) \,\%\, 360 > (A - (I+1).90/G) \,\%\, 360, \\
\quad \mathit{orient}(L,T),\ \mathit{varphi}(P,R,A),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G). \\
\leftarrow\ I \,\%\, 2 = 1,\ T \geq (A - (I-1).90/G) \,\%\, 360,\ (A - (I-1).90/G) \,\%\, 360 > (A - (I+1).90/G) \,\%\, 360, \\
\quad \mathit{orient}(L,T),\ \mathit{varphi}(P,R,A),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G). \\
\leftarrow\ I \,\%\, 2 = 1,\ T \leq (A - (I+1).90/G) \,\%\, 360,\ T \geq (A - (I-1).90/G) \,\%\, 360, \\
\quad (A - (I-1).90/G) \,\%\, 360 < (A - (I+1).90/G) \,\%\, 360,\ \mathit{orient}(L,T),\ \mathit{varphi}(P,R,A), \\
\quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{single\_diff\_opra}(N,K,L,I,G).
\end{array} \tag{27}$$

Quantitative Constraints The next set of rules impose the quantitative constraints about orientation and relative angular position. We restrict the intrinsic orientation of objects based on quantitative position information, similar to differential *OPRA* constraints. Note that there might be modulo 360 change in intrinsic orientation range and relative orientation range constraints.

$$\begin{array}{l}
\leftarrow T < A1,\ A2 \geq A1,\ \mathit{orient}(K,T),\ \mathit{direction\_range}(N,K,A1,A2). \\
\leftarrow T > A2,\ A2 \geq A1,\ \mathit{orient}(K,T),\ \mathit{direction\_range}(N,K,A1,A2). \\
\leftarrow T > A2,\ T < A1,\ A2 < A1,\ \mathit{orient}(K,T),\ \mathit{direction\_range}(N,K,A1,A2). \\
\leftarrow\ \mathit{xdist}(P,R,0),\ \mathit{ydist}(P,R,0),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_precise}(N,K,L,T). \\
\leftarrow\ \mathit{xdist}(P,R,0),\ \mathit{ydist}(P,R,0),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_range}(N,K,L,A1,A2). \\
\leftarrow\ T < (A - A2) \,\%\, 360,\ (A - A1) \,\%\, 360 \geq (A - A2) \,\%\, 360,\ \mathit{orient}(L,T),\ \mathit{varphi}(P,R,A), \\
\quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_range}(N,K,L,A1,A2). \\
\leftarrow\ T > (A - A1) \,\%\, 360,\ (A - A1) \,\%\, 360 \geq (A - A2) \,\%\, 360,\ \mathit{orient}(L,T),\ \mathit{varphi}(P,R,A), \\
\quad \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_range}(N,K,L,A1,A2). \\
\leftarrow\ T < (A - A2) \,\%\, 360,\ T > (A - A1) \,\%\, 360,\ (A - A1) \,\%\, 360 < (A - A2) \,\%\, 360,\ \mathit{orient}(L,T), \\
\quad \mathit{varphi}(P,R,A),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{psi\_range}(N,K,L,A1,A2). \\
\leftarrow\ (T - B) \,\%\, 360 < A1,\ A2 \geq A1,\ \mathit{orient}(K,B),\ \mathit{orient}(L,T),\ \mathit{delta\_range}(N,K,L,A1,A2). \\
\leftarrow\ (T - B) \,\%\, 360 > A2,\ A2 \geq A1,\ \mathit{orient}(K,B),\ \mathit{orient}(L,T),\ \mathit{delta\_range}(N,K,L,A1,A2). \\
\leftarrow\ (T - B) \,\%\, 360 < A1,\ (T - B) \,\%\, 360 > A2,\ A2 < A1,\ \mathit{orient}(K,B),\ \mathit{orient}(L,T), \\
\quad \mathit{delta\_range}(N,K,L,A1,A2).
\end{array} \tag{28}$$

Distance Constraints For qualitative and quantitative distance constraints, we first determine the pair of points to calculate the (square of) Euclidean distance (unless specified by a quantitative constraint).

$$\begin{array}{l}
\mathit{compute\_dist}(P,R) \leftarrow \mathit{not\ dist\_known}(P,R),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{dist\_rel}(N,K,L,U). \\
\mathit{compute\_dist}(P,R) \leftarrow \mathit{not\ dist\_known}(P,R),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance\_range}(N,K,L,D1,D2).
\end{array} \tag{29}$$

The rules in (30) makes the necessary computations for distance constraints. The first rule calculates the (square of the) numerical Euclidean distance between the pair of objects whose distance is unknown and indicated by the *compute_dist*(P, R) atom. The second rule utilizes the exogenous precise numerical distance information provided by the user. Note that when precise numerical distance is given, we can use the Euclid equation to find the distance on y axis from the distance on x axis. Indeed the third rule below calculates the distance between the target object and the reference object on y axis, which will be used for checking the quantitative distance constraint.

Note that *ndist*(P, R, D) atom holds the **square** of the distance value, because the distance between P, R (which is equal to $\sqrt{D}$) may not be an integer value. For the same reason, the D parameter in *distance*(N, K, L, D), *lower_bound*(U, D), *upper_bound*(U, D) atoms and $D1$, $D2$ parameters in *distance_range*$(N, K, L, D1, D2)$ atom hold the square of the respective distance value.

$$\begin{array}{l}
\mathit{ndist}(P,R,X^2+Y^2) \leftarrow \mathit{xdist}(P,R,X),\ \mathit{ydist}(P,R,Y),\ \mathit{compute\_dist}(P,R). \\
\mathit{ndist}(P,R,D) \leftarrow \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance}(N,K,L,D). \\
\mathit{ydist\_known}(P,R,\sqrt{D-X^2}) \leftarrow\ \mathit{xdist}(P,R,X),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance}(N,K,L,D).
\end{array} \tag{30}$$

To detect inconsistency faster, we check whether qualitative distance constraints are symmetric.

$$\begin{array}{l} \leftarrow\ U1 \neq U2,\ \mathit{dist\_rel}(M,K,L,U1),\ \mathit{dist\_rel}(N,K,L,U2). \\ \leftarrow\ U1 \neq U2,\ \mathit{dist\_rel}(M,K,L,U1),\ \mathit{dist\_rel}(N,L,K,U2). \end{array} \tag{31}$$

We ensure that the Euclidean distance between objects is inside the interval of the qualitative distance relation.

$$\begin{array}{l} \leftarrow D < D1,\ \mathit{lower\_bound}(U,D1),\ \mathit{ndist}(P,R,D),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{dist\_rel}(N,K,L,U). \\ \leftarrow D > D2,\ \mathit{upper\_bound}(U,D2),\ \mathit{ndist}(P,R,D),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{dist\_rel}(N,K,L,U). \end{array} \tag{32}$$

We also add rules to impose the quantitative distance constraints to be compatible with the Euclidean distance between points. For a quantitative distance range constraint, we check whether the numerical distance between generated coordinates is actually inside the range. If the quantitative distance information is precise, then the guessed y coordinates of the target and the reference object must satisfy the calculated distance along the y axis.

$$\begin{array}{l} \leftarrow\ Y1 \neq Y2 - DY,\ Y1 \neq Y2 + DY,\ \mathit{ydist\_known(P,R,DY)},\ \mathit{yloc(P,Y1)},\ \mathit{yloc(R,Y2)}, \\ \qquad \mathit{point(K,P)},\ \mathit{point(L,R)},\ \mathit{distance(N,K,L,D)}. \\ \leftarrow D < D1,\ \mathit{ndist}(P,R,D),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance\_range}(N,K,L,D1,D2). \\ \leftarrow D > D2,\ \mathit{ndist}(P,R,D),\ \mathit{point}(K,P),\ \mathit{point}(L,R),\ \mathit{distance\_range}(N,K,L,D1,D2). \end{array} \tag{33}$$

Henceforth the ASP program Π for checking consistency of *HOPA* constraints consists of the facts that represent the input and the rules (1)-(33) described above. An answer set of the program Π specifies a configuration of objects in the grid, identified by $\mathit{xloc}(P,X)$, $\mathit{yloc}(P,Y)$, $\mathit{orient}(K,A)$ atoms.

3 Proof of Theorem 2

Theorem 2 below states that Π is a sound and complete solution for consistency checking problem H of *HOPA* constraints, over domain $D_{s,t,z}$.

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 $\mathit{xloc}(P,X)$, $\mathit{yloc}(P,Y)$, $\mathit{orient}(K,A)$, $\mathit{point}(K,P)$ 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.*

Proof. The proof of Theorem 2 uses the following results in [1].

Splitting Set Theorem [1]. Let U be a splitting set for a program Π. A consistent set of literals is an answer set for Π if it can be written as $X \cup Y$ where X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$.

Intuitively, the bottom part $b_U(\Pi)$ of a program Π consists of the rules whose literals are contained in the splitting set U. Once an answer set X for the bottom part is computed, it is "propagated" to the rest of the program (called the top part) and the answer set Y is computed for the top part. The theorem ensures that $X \cup Y$ is an answer set for the whole program.

Proposition 2 of [1]. For any program Π and formula F, a set Z of literals is an answer set for $\Pi \cup \{\leftarrow F\}$ if Z is an answer set for Π and does not satisfy F.

Intuitively, Proposition 2 of [1] express that adding constraints to an ASP program eliminates its answer sets that violate these constraints.

The proof of Theorem 1 is by repeated application of Splitting Set Theorem. We create the ASP subprograms as below:

P^0 is the subprogram that consists of the facts in the input representation,
P^1 is the subprogram that consists of the rules (1),
P^2 is the subprogram that consists of the rules (2),
P^3 is the subprogram that consists of the rule (3),
P^4 is the subprogram that consists of the rules (4),
P^5 is the subprogram that consists of the rules (5)-(6),
P^6 is the subprogram that consists of the rules (7)-(8),
P^7 is the subprogram that consists of the rules (9)-(11),
P^8 is the subprogram that consists of the rules (12)-(13),
P^9 is the subprogram that consists of the rules (14),
P^{10} is the subprogram that consists of the rules (15),
P^{11} is the subprogram that consists of the rules (16),
P^{12} is the subprogram that consists of the rules (17),
P^{13} is the subprogram that consists of the rules (18),
P^{14} is the subprogram that consists of the rules (19),
P^{15} is the subprogram that consists of the rules (20),
P^{16} is the subprogram that consists of the rules (21),
P^{17} is the subprogram that consists of the rules (22)-(23),
P^{18} is the subprogram that consists of the rules (24),
P^{19} is the subprogram that consists of the rules (25)-(28),
P^{20} is the subprogram that consists of the rules (29),
P^{21} is the subprogram that consists of the rules (30),
P^{22} is the subprogram that consists of the rules (31)-(33).

We define the union of subprograms as $\Pi^k = \cup_{i=0}^{k} P^i$. Notice that $\Pi = \Pi^{22}$.

We construct the splitting set for each program Π^{k+1} as U_k, $1 \leq k \leq 21$ where U_k consists of all atoms in the subprograms P^0 up to P^k. Namely, U_0 is the set of facts in the input representation and U_k includes all atoms in the program Π^k, $1 \leq k \leq 21$. Observe that the bottom part of Π^{k+1} with respect to U_k is Π^k, and the top part is P^{k+1}.

Note that the atoms in the head of the rules in Π^i do not appear in the rules of subprograms P^0 to P^{i-1}. The answer set X_0 exists which describes the input description of the *HOPA* consistency problem.

We analyze the subprograms P^i and answer sets X_i as below.

- We apply the splitting set theorem [1] to Π^1: U_0 is a splitting set for Π^1, the bottom part of Π^1 with respect to U_0 is Π^0, and the top part is P^1.

  Let T_1 be an answer set for $e_{U_0}(P^1, X_0)$ i.e. T_1 is partial evaluation of the rules in (1) with respect to X_0. Namely, T_1 describes the index of the chosen disjunct from each disjunctive constraint with the *chosen(N,D)* atom.

Then by the Splitting Set Theorem, $X_1 = X_0 \cup T_1$ is an answer set for Π^1.

- U_1 is a splitting set for Π^2, the bottom part of Π^2 with respect to U_1 is Π^1, and the top part is P^2.

  Let T_2 be an answer set for $e_{U_1}(P^2, X_1)$ i.e. T_2 is partial evaluation of the rules in (2) with respect to X_1. Namely, T_2 describes the chosen disjunct with *diff_opra(N,K,L,I,J,G)*, *same_opra(N,K,L,I,G)*, *one_sided_diff_opra(N,K,L,I,G)*, *dist_rel(N,K,L,U)* atoms.

  By the Splitting Set Theorem, $X_2 = X_1 \cup T_2$ is an answer set for Π^2.

- U_2 is a splitting set for Π^3, the bottom part of Π^3 with respect to U_2 is Π^2, and the top part is P^3.

  Let T_3 be an answer set for $e_{U_2}(P^3,\ X_2)$ i.e. T_3 is partial evaluation of the rules in (3) with respect to X_2. Namely, T_3 describes the single-sided differential OPRA constraints with the *single_diff_opra*(N, K, L, I, G) atoms.

  Then by the Splitting Set Theorem, $X_3 = X_2 \cup T_3$ is an answer set for Π^3.

- The subprogram P^4 prohibits same OPRA constraint and differential OPRA constraint to be specified for the same pair.

  Proposition 2 of [1] implies that adding rule (4) into Π^3, the answer sets of Π^3 which violate the rules in (4) are eliminated.

  Hence an answer set X_4 for Π^4 involves compatible OPRA constraints.

- U_4 is a splitting set for Π^5, the bottom part of Π^5 with respect to U_4 is Π^4, and the top part is P^5.

  Let T_5 be an answer set for $e_{U_4}(P^5,\ X_4)$ i.e. T_5 is partial evaluation of the rules in (5)-(6) with respect to X_4. Namely, T_5 designates the points whose coordinates to be instantiated with the *instantiate*(P) atom, and the objects whose orientation be generated with the *generate_orient*(K) atom.

  Then by the Splitting Set Theorem, $X_5 = X_4 \cup T_5$ is an answer set for Π^5.

- U_5 is a splitting set for Π^6, the bottom part of Π^6 with respect to U_5 is Π^5, and the top part is P^6.

  Let T_6 be an answer set for $e_{U_5}(P^6,\ X_5)$ i.e. T_6 is partial evaluation of the rules in (7)-(8) with respect to X_5. Namely, T_6 indicates those points whose location is already known (or can be deduced) with the *loc_known*(P) atom, the objects whose orientation is already known (or can be deduced) with the *orient_specified*(K) atom, and the point pairs whose distance is already known with the *dist_known*(P, R) atom.

  Then by the Splitting Set Theorem, $X_6 = X_5 \cup T_6$ is an answer set for Π^6.

- U_6 is a splitting set for Π^7, the bottom part of Π^7 with respect to U_6 is Π^6, and the top part is P^7.

  Let T_7 be an answer set for $e_{U_6}(P^7,\ X_6)$ i.e. T_7 is partial evaluation of the rules in (9)-(11) with respect to X_6. Namely, T_7 describes an instantiation of point locations for each object with *xloc*(P, X), *yloc*(P, Y) atoms.

The rule (9) generates x,y coordinates for points whose location is unknown, (10) states the location of those points that are already known (given by user), and (11) finds out the coordinates of those points that are implicit (can be deduced from others).

According to the Splitting Set Theorem, $X_7 = X_6 \cup T_7$ is an answer set for Π^7.

- U_7 is a splitting set for Π^8, the bottom part of Π^8 with respect to U_7 is Π^7, and the top part is P^8.

  Let T_8 be an answer set for $e_{U_7}(P^8,\ X_7)$ i.e. T_8 is partial evaluation of the rules in (12)-(13) with respect to X_7.

  The rule (12) designates the pair of points (P, R) to compute their vector and angle φ_{PR}. (13) indicates those pairs to compute the reverse angle φ_{RP} (due to two-sided differential OPRA constraint).
  Thus T_8 describes the pair of points to compute their vector angle, with the *compute_vector*(P, R) and *compute_reverse*(P, R) atoms.

  Then by the Splitting Set Theorem, $X_8 = X_7 \cup T_8$ is an answer set for Π^8.

- U_8 is a splitting set for Π^9, the bottom part of Π^9 with respect to U_8 is Π^8, and the top part is P^9.

  Let T_9 be an answer set for $e_{U_8}(P^9,\ X_8)$ i.e. T_9 is partial evaluation of the rules in (14) with respect to X_8.

  The rule (14) states the pair of points (P, R) to compute the difference between their x,y coordinates (for distance constraint) which are not already computed for vector calculation.

  Then, T_9 describes the additional pair of atoms to compute the difference in their x,y coordinates, with the *compute_loc_diff*(P, R) atom.

  According to the Splitting Set Theorem, $X_9 = X_8 \cup T_9$ is an answer set for Π^9.

- U_9 is a splitting set for Π^{10}, the bottom part of Π^{10} with respect to U_9 is Π^9, and the top part is P^{10}.

  Let T_{10} be an answer set for $e_{U_9}(P^{10},\ X_9)$ i.e. T_{10} is partial evaluation of the rules in (15) with respect to X_9. Namely, T_{10} states the difference between x,y coordinates (i.e. distance over respective axis) of a pair of points (P, R) with the *xdist*(P, R, X), *ydist*(P, R, Y) atoms.

  By the Splitting Set Theorem, $X_{10} = X_9 \cup T_{10}$ is an answer set for Π^{10}.

- U_{10} is a splitting set for Π^{11}, the bottom part of Π^{11} with respect to U_{10} is Π^{10}, and the top part is P^{11}.

  Let T_{11} be an answer set for $e_{U_{10}}(P^{11},\ X_{10})$ i.e. T_{11} is partial evaluation of the rules in (16) with respect to X_{10}. Namely, T_{11} describes the absolute value and sign of coordinate difference over x,y axis with *xdist_abs*(P, R, X), *ydist_abs*(P, R, Y), *xdist_positive*(P, R), *ydist_positive*(P, R) atoms, for those pair of points (P, R) to compute their vector and angle φ_{PR}.

  Then by the Splitting Set Theorem, $X_{11} = X_{10} \cup T_{11}$ is an answer set for Π^{11}.

- U_{11} is a splitting set for Π^{12}, the bottom part of Π^{12} with respect to U_{11} is Π^{11}, and the top part is P^{12}.

Let T_{12} be an answer set for $e_{U_{11}}(P^{12},\ X_{11})$ i.e. T_{12} is partial evaluation of the rules in (17) with respect to X_{11}.

The rule (17) calculates the ratio of absolute value of the distance over y axis to the distance over x axis, for the pair of points indicated by *compute_vector*(P, R). T_{12} describes the absolute value of the tangent for the vector $\overrightarrow{PR}$, with the *tan_abs*(P, R, Z) atom.

By the Splitting Set Theorem, $X_{12} = X_{11} \cup T_{12}$ is an answer set for Π^{12}.

- U_{12} is a splitting set for Π^{13}, the bottom part of Π^{13} with respect to U_{12} is Π^{12}, and the top part is P^{13}.

  Let T_{13} be an answer set for $e_{U_{12}}(P^{13},\ X_{12})$ i.e. T_{13} is partial evaluation of the rules in (18) with respect to X_{12}.

  The rule (18) computes the arctangent of the absolute value of ratio of vector over y and x axis. Therefore T_{13} describes the corresponding value of the angle in the range in the range $[0, 90^o]$, with the *qangle*(P, R, A) atom.

  According to the Splitting Set Theorem, $X_{13} = X_{12} \cup T_{13}$ is an answer set for Π^{13}.

- U_{13} is a splitting set for Π^{14}, the bottom part of Π^{14} with respect to U_{13} is Π^{13}, and the top part is P^{14}.

  Let T_{14} be an answer set for $e_{U_{13}}(P^{14},\ X_{13})$ i.e. T_{14} is partial evaluation of the rules in (19) with respect to X_{13}.

  The rule (19) finds out the vector angle φ_{PR}, based on the angle *qangle*(P, R, A) in the range $[0, 90^o]$ and by the atoms *xdist_positive*(P, R), *ydist_positive*(P, R) which show the sign over each axis. Thus T_{14} describes the vector angle φ_{PR} with *varphi*(P, R, A) atom, for the vector $\overrightarrow{PR}$.

  Then by the Splitting Set Theorem, $X_{14} = X_{13} \cup T_{14}$ is an answer set for Π^{14}.

- U_{14} is a splitting set for Π^{15}, the bottom part of Π^{15} with respect to U_{14} is Π^{14}, and the top part is P^{15}.

  Let T_{15} be an answer set for $e_{U_{14}}(P^{15},\ X_{14})$ i.e. T_{15} is partial evaluation of the rules in (20) with respect to X_{14}. Namely, T_{15} describes the reverse angle φ_{RP} with *varphi*(R, P, A) atom for two-sided differential OPRA constraints.

  According to the the Splitting Set Theorem, $X_{15} = X_{14} \cup T_{15}$ is an answer set for Π^{15}.

- U_{15} is a splitting set for Π^{16}, the bottom part of Π^{16} with respect to U_{15} is Π^{15}, and the top part is P^{16}.

  Let T_{16} be an answer set for $e_{U_{15}}(P^{16},\ X_{15})$ i.e. T_{16} is partial evaluation of the rules in (21) with respect to X_{15}. Thus T_{16} describes a chosen integer from the set Ω for the objects whose orientation is unknown. The chosen value represents a degree in the range $[0, 360/r)$, with the *degree*(K, E) atoms.

  By the Splitting Set Theorem, $X_{16} = X_{15} \cup T_{16}$ is an answer set for Π^{16}.

- U_{16} is a splitting set for Π^{17}, the bottom part of Π^{17} with respect to U_{16} is Π^{16}, and the top part is P^{17}.

Let T_{17} be an answer set for $e_{U_{16}}(P^{17},\ X_{16})$ i.e. T_{17} is partial evaluation of the rules in (22)-(23) with respect to X_{16}.

The rule (22) generates an orientation for those objects whose orientation is unknown. This is achieved by multiplying the degree value by the resoultion parameter. (23) computes the orientation of objects whose value is directly specified by the user, or implicitly known from linear OPRA constraints or $\mathit{delta\_precise}(N, K, L, A)$, $\mathit{psi\_precise}(N, K, L, T)$ constraints. Hence T_{17} describes intrinsic orientation of each object with the $\mathit{orient}(K, A)$ atom.

Then by the Splitting Set Theorem, $X_{17} = X_{16} \cup T_{17}$ is an answer set for Π^{17}.

- The subprogram P^{18} ensures that every object has unique orientation value and the answer set X_{18} describes intrinsic orientation of each object. The rules in (24) ensure that there is unique orientation value for each object. If there are multiple orientation information for an object, they must be the same.

  Proposition 2 of [1] implies that adding rule (24) into Π^{17}, the answer sets of Π^{17} which violate the rules in (24) are eliminated.

  Hence an answer set X_{19} for Π^{19} describes an instantiation of objects with unique orientation value.

- The subprogram P^{19} ensures that the generated location and orientation of objects satisfy OPRA constraints and quantitative relative position constraints.

  The rules in (25) impose the relative angular position specified by same OPRA constraints. The rules in (26) insists that location of the objects related by a differential OPRA constraint are not the same. The rules in (27) impose the relative position specified by differential OPRA constraints. The rules in (28) impose the quantitative constraints about orientation and relative angular position.

  Proposition 2 of [1] implies that adding rules (25)-(28) into Π^{18}, the answer sets of Π^{18} which violate the rules in (25)-(28) are eliminated.

  Hence an answer set X_{19} for Π^{19} describes an instantiation of objects that satisfy OPRA constraints and quantitative position constraints.

- U_{19} is a splitting set for Π^{20}, the bottom part of Π^{20} with respect to U_{19} is Π^{19}, and the top part is P^{20}.

  Let T_{20} be an answer set for $e_{U_{19}}(P^{20},\ X_{19})$ i.e. T_{20} is partial evaluation of the rules in (29) with respect to X_{19}. Namely, T_{20} describes the pair of points to calculate their numerical Euclidean distance, with $\mathit{compute\_dist}(P, R)$ atom (to check distance constraints). The Euclidean distance between these points are not known apriori.

  Then by the Splitting Set Theorem, $X_{20} = X_{19} \cup T_{20}$ is an answer set for Π^{20}.

- U_{20} is a splitting set for Π^{21}, the bottom part of Π^{21} with respect to U_{20} is Π^{20}, and the top part is P^{21}.

  Let T_{21} be an answer set for $e_{U_{20}}(P^{21},\ X_{20})$ i.e. T_{21} is partial evaluation of the rules in (30) with respect to X_{20}. The first rule in (30) calculates the Euclidean distance between the pair of objects whose distance is unknown and indicated by the $\mathit{compute\_dist}(P, R)$ atom. The second rule in (30) uses exogenous numerical distance information, provided by the user.

Namely, T_{21} describes the numerical Euclidean distance between the pair of objects with *ndist*(P, R, D) atom, and the numerical distance between objects on the y axis with the *ydist_known(P,R,Y)* atom.

According to the Splitting Set Theorem, $X_{21} = X_{20} \cup T_{21}$ is an answer set for Π^{21}.

- The subprogram P^{22} imposes qualitative and quantitative distance constraints.

  The rules in (31) ensures that the distance relations for the same pair of points are symmetric. The rules in (32) ensures that the Euclidean distance between points lies inside the interval corresponding to the qualitative distance constraint. The rules in (33) impose the quantitative distance constraints.

  Proposition 2 of [1] implies that adding rules (31)-(33) into Π^{21}, the answer sets of Π^{21} which violate the rules in (31)-(33) are eliminated.

  Hence an answer set X_{22} for Π^{22} describes an instantiation of objects that satisfy qualitative and quantitative *HOPA* constraints in the given constraint set.

Note that $\Pi = \Pi^{22}$ and if Z is an answer set for Π, then *xloc*(P, X), *yloc*(P, Y), *orient*(K, A) atoms in Z describes an instantiation of objects that satisfy the given *HOPA* constraints in C. Namely, $Z \cap \mathcal{O}_{s,t,z}$ characterize an instantiation X of objects in $D_{s,t,z}$ to objects in V that is a solution of the consistency checking problem H. Then, X is a solution of H if and only if X can be characterized as $Z \cap \mathcal{O}_{s,t,z}$ for some answer set Z of Π.

Uniqueness Proof: To prove uniqueness of representation, suppose that another answer set Z' for Π also characterizes X and $Z' \neq Z$. Z' must include precisely the same *xloc*(P, X), *yloc*(P, Y), *orient*(K, A) atoms as Z, otherwise Z' does not characterize X. Since the *HOPA* constraints in the input are the same, Z' and Z include identical *loc_known*(P), *orient_specified*(K), *dist_known*(P, R), *compute_vector*(P, R), *compute_reverse*(P, R), *compute_loc_diff*(P, R), *compute_dist*(P, R), atoms. Recall that *xdist*(P, R, X), *ydist*(P, R, Y), *ndist*(P, R, D), *ydist_known(P,R,Y)*, *tan_abs*(P, R, Z), *qangle*(P, R, A), *varphi*(P, R, A) atoms are derived from *xloc*(P, X), *yloc*(P, Y), *orient*(K, A) atoms, then these atoms are also the same in Z' and Z. Since all atoms in the two sets coincide, $Z' = Z$.

$\square$

References

[1] Erdogan, S.T., Lifschitz, V.: Definitions in answer set programming. In: Proc. of LPNMR. pp. 114–126 (2004)

[2] Wolter, D., Lee, J.: On qualitative reasoning about relative point position. Artificial Intelligence **174**, 1498–1507 (2010)