Algorithm Outline for Sketch Map Drawing from Spatial Data Distilled from Natural Language Descriptions

Marek Menšík, Petr Rapant, Adam Albert
VSB - Technical University of Ostrava, Ostrava, Czech Republic
marek.mensik@vsb.cz, petr.rapant@vsb.cz, adam.albert@vsb.cz

Much knowledge about the real world is recorded in plain text, e.g., as messages on social networks. These messages contain, among others, also spatial information, and it can be distilled from these messages by natural language processing. The extracted information can be represented as a plain topological graph stored as tuples describing individual edges. This paper presents an outline of an algorithm that uses these tuples for creating a sketch map.

Keywords:
motion verbs, TIL, spatial data, topology, sketch map

University of Maribor Press

DOI https://doi.org/10.18690/um.feri.5.2023.15
ISBN 978-961-286-745-4

1 Introduction

Much knowledge about the real world is recorded in plain text, e.g., messages on social networks. These texts contain, among other things, spatial data. Our research aims to extract these spatial data from plain text, compile a topological graph of the described area, and then visualize it as a sketch map. A sketch map is defined as '*an outline map drawn from observation rather than from exact survey measurements and showing only the main features of the area*'.[1] It is usually a hand drawing of an area drawn without scale, and it usually shows the main characteristics of an area and is not cluttered with unnecessary detail. The sketch map has a low degree of positional accuracy and therefore does not correctly represent the distances, dimensions, and shapes of objects. On the other hand, it can have a high degree of logical accuracy, meaning that the spatial relationships (topology, spatial order) between objects are correctly represented[2].

The process of creating a sketch map from plain text data consists of three steps [1]:

(i) identification of spatial entities and their spatial relations by natural language processing,

(ii) creation of a plain topological graph that captures identified spatial entities and their spatial relations, and (iii) conversion of this graph into a sketch map. This paper deals with the first results related to the third step.

Converting a topological graph into a sketch map involves, in principle, the dynamic placing of spatial entities on the canvas in a way that all the known spatial relations between them are kept. Some authors dealt with this process. The authors [1] represented each entity as a rectangle, the size and position of which are adjusted stepwise to fit all spatial relations. They mainly dealt with data that describe the urbanized area. On the other hand, the authors [2] focused on creating a sketch map of an open landscape using descriptions of individual routes created by orienteers. The resulting sketch map captured the relative position of each spatial entity in relation to other entities. Their approach was based on a genetic algorithm.

[1] See https://www.merriam-webster.com/dictionary/sketch%20map
[2] See https://www.tariffnumber.com/info/abbreviations/12485

The described approach is different. We use TIL constructions to represent captured spatial data and create a plain topological graph of the described area. Mathematical logic tools are used to process this graph to compile individual circles in the graph, and these circles are then concatenated to create the final sketch map.

The following chapters are organized as follows. Chapter 2 provides a detailed description of the input data format utilized by the algorithm, along with essential definitions. In Chapter 3, we focus on the numbering of directions and the identification of the bounding circle in our dataset. The modification of data required for the application of the algorithm is discussed in Chapter 4, while Chapter 5 presents a thorough description of the algorithm itself. Chapter 6 is dedicated to a case study that showcases the application of the algorithm. Finally, Chapter 7 concludes the paper.

2 How We Obtain Our Data

We start with descriptions of the agents' journeys, which we consider coherent both in space and time.

In [3], we have introduced heuristic functions that manipulate these descriptions of journeys. These functions incrementally build a TIL construction describing spatial data.

TIL is a typed hyperintensional λ-calculus of partial functions found by Pavel Tichý in the early 1970s. TIL exploits procedural semantics, i.e., natural language expressions encode *algorithmically structured procedures* as their meaning. Tichý defined six kinds of such meaning procedures that he coined TIL *constructions* as the centerpiece of his system; see [4]. Constructions produce extensional or intensional entities, or even lower-order procedures, as their products or, in well-defined cases, fail to produce anything. TIL has been introduced and thoroughly described in numerous papers, such as [4], [5], [6], [7].

A journey can be informally described as a sequence of natural language sentences, and each sentence contains information about part of an agent's journey. To formalize the sentences, we exploit a class of *motion verbs* (e.g., to go, to walk, to cross, to turn) that bind other sentences' constituents to them via valency. The valence of

the verb is described in valency frames using *functors*. Functors define the semantic-syntactic relationship between the verb and its complement. Details are given, for example, in [8], [9], [10], [1].

For example, the sentence *"John walks swiftly 15 minutes from the gym to home."* can be analyzed by exploiting the valency frame of the verb *to walk* as follows:

- ACT (who): John
- DIR1 (from where): gym
- DIR3 (to where): home
- EXT (for how long/ how far): 15 minutes
- MANN (manner): swiftly

Using valency frames and the information that follows from them, we might obtain a formal description of one's journey by formalizing it in the natural language formalized in the expressive language of TIL. [3]

$$\lambda w \lambda t\ [[{}'ACT_{wt}\ {}'John\ {}'walk] \wedge [{}'DIR1_{wt}\ {}'gym\ {}'walk] \wedge [{}'DIR3_{wt}\ {}'home\ {}'walk] \wedge [{}'EXT_{wt}\ {}'15\ {}'walk] \wedge [{}'MANN_{wt}\ {}'quickly\ {}'walk]] \quad (1)$$

Types:

DIR1, DIR3/$(o\pi v)_{\tau\omega}$*; ACT/* $(otv)_{\tau\omega}$*; MANN/* $(oav)_{\tau\omega}$*; EXT/; Tom/* ι*; home, school /*π*; walk/v, where* π *is a type of places; v is a type of the activity denoted by a verb.*

This approach is based on our previous research. In that, we introduced an algorithm of symbolic supervised machine learning that incrementally builds an explication of vague or inaccurate expressions into an adequately accurate one. For more information, see [11], [7].

[3] For the sake of readability, we will use just TIL language to display examples. Our computations are executed over TIL-Script constructions.

In this paper, we process the data obtained from the TIL constructions. These data contain information about the places the agents visited and the directions of movement between pairs of these places they took. In [3], we introduced several definitions that identify spatial data in TIL constructions. The following two ones help us identify the places visited by agents.

Definition 1 (*node, edge*)**.** Let V be a *motion verb*, let $S = \{B \mid (\text{'}DIR1 \text{ or } \text{'}DIR3)$ and V are constituents of B $\}$ and let $D^V = \{C \mid V$ is a constituent of C $\} \setminus S$ then D^V is a set of ***edges*** and S is a set of ***nodes***.

Definition 2 (*place, functor, value*)**.** Let $[a\ x\ v]$ be a *node* and let $[\beta\ y\ v_1]$ be the *edge description*. Then x is a ***place***, a, β are ***functors***, and y, v, v_1 is ***values***.

A simple sentence might connect two places by verb valency with additional information. In the case of construction 1, by definition 1, $[\text{'}DIR1_{wt}\ \text{'}gym\ \text{'}walk]$ and $[\text{'}DIR3_{wt}\ \text{'}home\ \text{'}walk]$ are *nodes*. The rest, $[\text{'}ACT_{wt}\ \text{'}John\ \text{'}walk]$,$[\text{'}EXT_{wt}\ \text{'}15\ \text{'}walk]$, $[\text{'}MANN_{wt}\ \text{'}quickly\ \text{'}walk]$, is an edge. By definition 2, *home*, and *gym* are *places*; they are constituents of *nodes*.

Definition 3 (*relative direction RD*)**.** Let $[\text{'}MANN\ x\ v]$ be an *edge description*, and the value x be one of the following *'straight*, *'slightly–left*, *'left*, *'sharp–left*, *'back*, *'sharp–right*, *'right* and *'slightly–right*. Then x is the ***relative direction***.

Using definitions 1, 2 and 3, we extract the spatial information used for map sketching, namely, a place that is the origin of the agent's movement, his direction of movement, and the place to which the agent is heading. We obtain input data as a tuple *[DIR1, direction, DIR3]*.

3 Numbering

Directions are essential elements in map sketching. We encode the relative directions from the input data using numbers. The mapping of relative directions into natural numbers is shown in table 1.

The data in table 1 represent the basic eight directions that we obtain, i.e. Straight (F), slightly left (FL), left (L), sharp left (BL), back (B), sharp right (BR), right (R) and slightly right (FR). TIL constructions which are

Table 1: TIL representations (TIL), relative directions (RD) and their numerical representation.

TIL	RD	Encoding		TIL	RD	Encoding
'straight	F	0		*'back*	B	4
'slightly–left	FL	1		*'sharp–right*	BR	5
'left	L	2		*'right*	R	6
'sharp–left	BL	3		*'slightly–right*	FR	7

Let [*'MANN x v*] be an edge description, and the value x be one of the following , *'slightly–left*, , *'sharp–left*, *'back*, *'sharp–right*, *'right* and *'slightly–right*. Then x is the **relative direction**.

Based on Table 1, we define the relative direction number (RDN):

Definition 4 (*relative direction number RDN*)**.** Let *C* be a graph circle. ***RDN*** is the value assigned to the edge in *C* based on algorithm 1.

RDNs are together with the relevant relative direction visually represented in Figure 1.

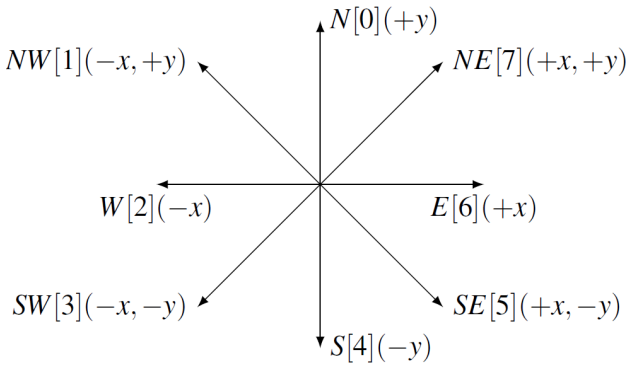

Figure 1: Absolute [RDN/ASDN] and ⊕ function for computation of individual nodes coordinates

Source: own.

In the first step of the map sketching algorithm, we assign RDNs to edges according to Algorithm 1.

Algorithm 1. *Relative direction number (RDN) assignment*
Require: $\circ$ - numbering function (Table 2),
G - set of tuples [a DIR1 RD DIR3] representing the graph,
$C \subseteq G$ represents the circle,
S - starting node, where $RD_1 = front$

$RDN_1 \leftarrow 0$
for $i := 2$ to $|C|$ **do**
$RDN_i = RDN_{i-1} \circ RD_i$
end for

Table 2: Computation of RDN: Application of the $\circ$ function3

$\circ$	F	FL	L	BL	B	BR	R	FR
F	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

Afterward, we identify all bounding circles defined by definition 5 in our input data.

Definition 5 (*bounding circle BC*)**.** Let G be a plane graph and C be a graph circle. If each edge of the circle C has the maximal number Cn calculated by the equation:

$$Cn = ((RDN^{out} - RDN^{in} + 8) \mod 8) \quad (2)$$

Then C is called the ***bounding circle***.[4]

[4] $ASDN^{out}$ is ASDN of outgoing incident edge and $ASDN^{in}$ represents ASDN of the edge entering into the node.

There may be an edge in multiple circles with different RDNs. To deal with this situation, we need to run the *rotation* algorithm (see Section 4.1) to obtain ASDN (further definition) to unify all RNDs.

For example, assume that we have a segment of a description of some bounding circle in which an *agent went right from A to B* and assume that the agent came to node *A* from relative direction 0. Our input data would be in the form *[A, right, B]*. Therefore, we can assign the edge from A to B with RND 6.

4.1 Bounding Circle Adjustment

Before we can join two bounding circles in their common sections, it is often necessary to modify them. In general, there are two situations that we need to consider. In the first one, the same sections of the bounding circles are described by agents from different directions. Therefore, it is necessary to rotate one bounding circle so that their common sections lead in the same direction. The second occurs when the distances between nodes in the common sections are unequal. In this case, the nodes of one bounding circle must be moved so that the distances of the nodes in the common sections are the same, but the directions of edges in the modified bounding circle remain the same.

4.1 Rotation

The common edges of two bounding circles can be described in different directions. One agent can turn on a street from the left, and another can turn on the same street from the right. Therefore, the unification of the RDNs of two bounding circles with common edges is necessary for the merging of those two bounding circles. The unification is done by rotation, and rotation is achieved by recalculating all RDNs using the equation 3.[5]

$$RDN_{i+1} = (RDN_i + 1) \mod 8 \quad (3)$$

Definition 6 (*absolute sketch direction number ASDN*)**.** Let *e* be an edge, then the *RDN* of the edge *e* is called ***ASDN*** if for every circle where the edge *e* is part of it, the *e* has the same *RDN*.

[5] The rotation is used as many times as it is necessary to unify the RDN.

Remark: If all the data are consistent, then *rotation* gives *ASDNs*. If some *RDN* is not possible to transform into *ASDN*, then the data are not consistent and the user can be notified which edge is not consistent.

4.2 Node adjustment

The node adjustment consists of verifying that the node is consistently positioned to neighboring nodes with respect to the RDN and moving in the direction of the RDN if necessary. If a node is moved, the consistency of neighboring nodes is verified. The function $\oplus$ is used to place the nodes in particular coordinates consistently.

For example: If node A has coordinates [0,0] and node B is in direction 7, that is, edge (A, B) has RDN = 7. Then according to the function represented by figure 1, node B has coordinates [n,n], where $n > 0$.

5 Algorithm for Computing Coordinates of Nodes

1. Sort all *bounding circles* (BCs).
2. Pick the longest one and place any node to coordinate [0,0].
3. Go through all the nodes in the circle and according to their ASDN calculate the coordinates. Check whether there are no edges intersections. If so, make *adjustments*.[6]
4. Withdraw the used circle from the set.
5. From the rest of the circles find the one with the longest common part with the already processed circle. If there are known coordinates, use them. Otherwise, recalculate and *adjust* all affected BCs. [7]
6. Set one common node of the new circle to already known coordinates and continue by step [3.]

6 Case Study

In our case study, we will demonstrate the outline of the algorithm functionality. First, we will present the input data visualized using the topological graph. We will identify bounding circles from input data and from this point we will visualize

[6] Move nodes to other coordinates in the same direction as described in chapter 4.
[7] The longest common part means BC with the most common edges.

processed data using graph visualization for simplicity. The next step will demonstrate the adjustment of bounding circles and subsequently merging them into a sketch map.

Table 3 represents our input data. Column *Places* contain triplets (*H, A, F*) meaning an agent went from place *H* via place *A* to place *F*. *Absolute* and *Relative* columns are values of absolute directions, and relative direction, respectively. For example, a pair (*ne*, *sw*) means that an agent came to A from the northeast (place *H*) and continued to the southwest (place *F*). Therefore, the relative direction is straight.

The absolute directions in Table 3 are generated from the map data and are mentioned here only to verify the correctness of our algorithm, they are not required for the correct functionality of our algorithm.

Table 3: Input data

Places	Abs	Rel	Street		Places	Abs	Rel	Street
(H,A,F)	ne,sw	F	'Bowery		(F,A,H)	sw,ne	F	'Bowery
(A,F,G)	ne,sw	F	'Bowery		(G,F,A)	sw,ne	F	'Bowery
(F,G,E)	ne,sw	F	'ChathamSQ		(E,G,F)	sw,ne	F	'Bowery
(G,E,J)	ne,sw	F	'ChathamSQ		(J,E,G)	sw,ne	F	'ChathamSQ
(E,J,D)	ne,sw	F	'ChathamSQ		(D,J,E)	sw,ne	F	'ChathamSQ
(J,D,B)	ne,nw	R	'MottST		(B,D,J)	nw,ne	L	'ChathamSQ
(D,B,M)	se,ne	R	'MottST		(M,B,D)	ne,se	L	'MottST
(B,M,I)	sw,se	R	'BayardST		(I,M,B)	se,sw	L	'MottST
(M,I,H)	nw,se	F	'BayardST		(H,I,M)	se,nw	F	'BayardST
(I,H,A)	nw,sw	R	'Bowery		(A,H,I)	sw,nw	L	'BayardST
(D,B,C)	se,e	BR	'PellST		(C,B,D)	e,se	BL	'MottST
(B,C,O)	w,e	F	'PellST		(O,C,B)	e,w	F	'PellST
(C,O,L)	w,e	F	'PellST		(L,O,C)	e,w	F	'PellST
(O,L,P)	w,e	F	'PellST		(P,L,O)	e,w	F	'PellST
(L,P,A)	w,e	F	'PellST		(A,P,L)	e,w	F	'PellST
(C,B,M)	e,ne	BR	'MottST		(M,B,C)	ne,e	BL	'PellST
(P,A,H)	w,ne	FL	'Bowery		(H,A,P)	ne,w	FR	'PellST
(P,A,F)	w,sw	BR	'Bowery		(F,A,P)	sw,w	BL	'PellST
(K,G,E)	nw,sw	R	'ChathamSQ		(E,G,K)	sw,nw	L	'DoyerST
(K,G,F)	nw,ne	L	'Bowery		(F,G,K)	ne,nw	R	'DoyerST
(L,K,G)	n,se	FL	'DoyerST		(G,K,L)	se,n	FR	'DoyerST
(O,L,K)	w,s	R	'DoyerST		(K,L,O)	s,w	L	'PellST
(P,L,K)	e,s	L	'DoyerST		(K,L,P)	s,e	R	'PellST

From Table 3, we identify three bounding circles (according to definition 5), namely BC 1: L–O–C–B–M–I–H–A–P; BC2: L–O–C–B–D–J–E–G–K and BC 3: L–K–G–F–A–P.

BCs are presented in Table 4, where the edges of bounding circles are in the form of [[*X,Y*], *RDN*]. By algorithm 1 we assign RDN to all edges in the bounding circles.

Table 4: Three identified BCs.

BC 1	BC 2	BC 3
[[a,p],0]	[[l,o],2]	[[a,p],0]
[[p,l],0]	[[o,c],2]	[[p,l],0]
[[l,o],0]	[[c,b],2]	[[l,k],2]
[[o,c],0]	[[b,d],5]	[[k,g],3]
[[c,b],0]	[[d,j],7]	[[g,f],5]
[[b,m],5]	[[j,e],7]	[[f,a],5]
[[m,i],3]	[[e,g],7]	
[[i,h],3]	[[g,k],1]	
[[h,a],1]	[[k,l],0]	

The visualization of bounding circles from Table 4 is presented in Figure 2.

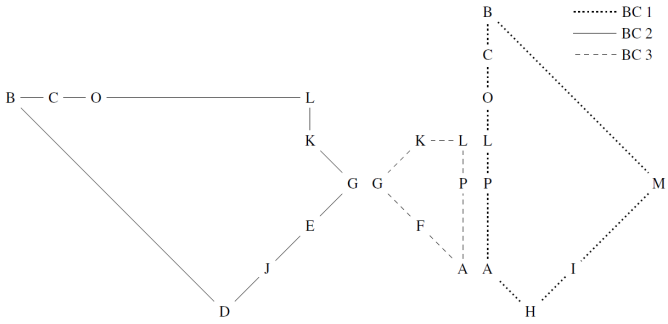

Figure 2: Three unrotated BCs are sketched according to Table 4
Source: own.

Because there are bounding circles that share some nodes, we can merge them. Firstly, we merge BCs that share the most nodes. In this case, we will merge BC 1 and BC 2 first (L-O-C-B are shared). To do so, we need to unify RDN of the same edges in both BCs. The edge L-O in BC 1 has RDN = 0 and the same edge in BC 2 has RDN = 2. Therefore, it is necessary to adjust (rotate) BC 2 until the RDNs are the same. Rotation is visualized in figure 3.

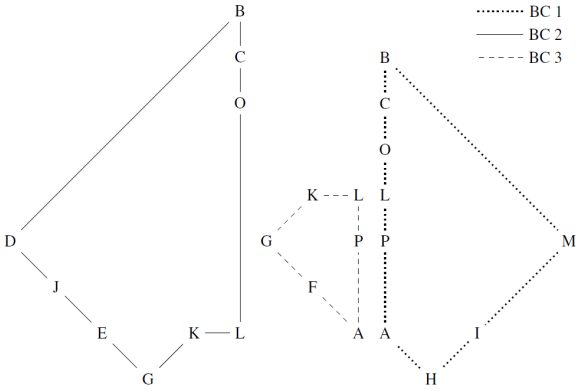

Figure 3: Three BCs sketched according to Table 4
Source: own.

Now, all bounding circles are oriented in the same direction. However, as seen from the visualizations of BC 1 and BC 2 in Figure 3, the distances between the shared nodes *L* and *O* are not equal. It is necessary to adjust the nodes of one of the BCs to match the nodes of the other BC. In figure 4, we can see that in BC 1, we have adjusted the coordinates of node *L*, so the distance from node *L* to node *O* is the same in both BCs. The adjustment of the coordinates of node *L* compromised the RND of the other nodes in BC 1. Therefore, the coordinates of the nodes *P*, *A*, *H*, *I*, and node *M* were adjusted, respectively, so the original RND of the edges remained unchanged.

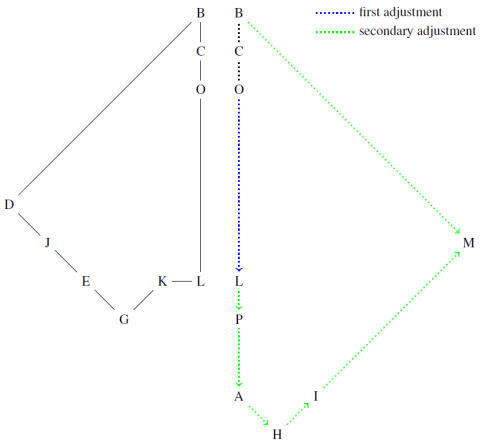

Figure 4: Node adjustment of BC 1
Source: own.

Since the sequence L–O–C–B is proportionally the same in both bounding circles 1 and 2, we can proceed to the merging process. The visualization of the merged BCs is presented in Figure 5.

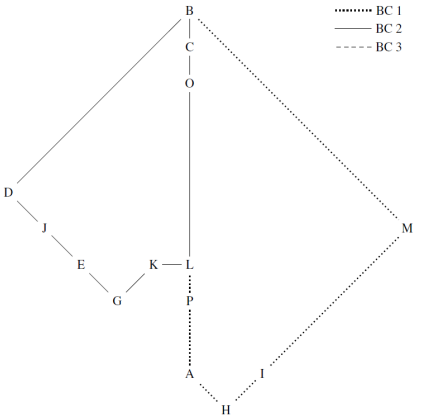

Figure 5: Merge of BC 1 and 2
Source: own.

The same process is applied in the case of merging BC 3 to the merged BC 1 and 2. In this case, there is no rotation or adjustments needed. Figure 6 presents the merged BCs 1, 2, and 3.

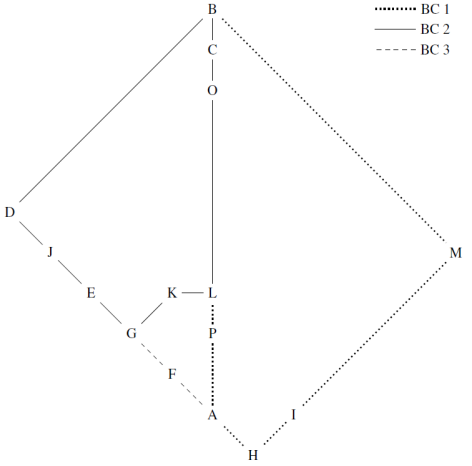

Figure 6: Merge of BC 1, 2 and 3
Source: own.

The resulting image represents a sketch map, where the nodes are positioned according to the direction in which they lie from each other. The distances between the nodes, as mentioned in the introduction, are not relevant, what matters is their relative position on the sketch map.

7 Conclusion

This paper presents an outline of an algorithm that accepts tuples that describe individual edges of a topological graph. The tuples are in the form of [from where (place), via what (place), to where (place), change of direction (direction)]. The output of the algorithm is a sketch map of the topological graph. From the input data, we identify bounding circles that are modified and merged into the sketch map. This outline of the algorithm is the first attempt to solve the problem of drawing a sketch map, i.e. the computation is costly and not optimized. The algorithm was implemented in PROLOG language.

Acknowledgements

This research has been supported by a Grant from SGS No. SP2023/065, VŠB - Technical University of Ostrava, Czech Republic, "Application of Formal Methods in Knowledge Modelling and Software Engineering VI".

References

[1] Maria Vasardani, Sabine Timpf, Stephan Winter, and Martin Tomko. From descriptions to depictions: A conceptual framework. In Thora Tenbrink, John Stell, Antony Galton, and Zena Wood, editors, *Spatial Information Theory*, pages 299–319, Cham, 2013. Springer International Publishing.

[2] Lamia Belouaer, David Brosset, and Christophe Claramunt. From verbal route descriptions to sketch maps in natural environments. SIGSPACIAL '16, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Marek Menšík, Adam Albert, Petr Rapant, and Tomaáš Michalovský. Heuristics for spatial data descriptions in a multi-agent system. *Frontiers in Artificial Intelligence and Applications*, 364:68–80, 2023.

[4] Marie Duží, Bjørn Jespersen, and Pavel Materna. *Procedural semantics for hyperintensional logic*. Springer, New York, 2010.

[5] Marie Duží. *Extensional Logic of Hyperintensions*, pages 268–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[6] Marie Duzi. Communication in a multi-cultural world. *Organon F*, 21:198–218, 01 2014.

[7] Marek Menšík, Marie Duží, Adam Albert, Vojtěch Patschka, and Miroslav Pajr. Refining concepts by machine learning. *Computación y Sistemas*, 23(3):943 – 958, 2019.

[8] Jarmila Panevova. Valency frames and the meaning of the sentence. *The Prague School of Structural and Functional Linguistics*, 41:223, 1994.

[9] Bernd Heine, Heiko Narrog, Vilmos Ágel, and Klaus Fischer. Dependency grammar and valency theory. *The Oxford Handbook of Linguistic Analysis*, 2015.

[10] Thomas Herbst, David Heath, Ian F. Roe, and Dieter Götz. *A Valency Dictionary of English: A CorpusBased Analysis of the Complementation Patterns of English Verbs, Nouns and Adjectives*. De Gruyter Mouton, 2013.

[11] Marek Menšík, Marie Duží, Adam Albert, Vojtěch Patschka, and Miroslav Pajr. Machine learning using TIL. In *Frontiers in Artificial Intelligence and Applications*, pages 344 – 362, Amsterdam, 2019. IOS Press.