Inference and Visualization of Spatial Relations*

Sylvia Wiebrock, Lars Wittenburg, Ute Schmid, and Fritz Wysotzki

Methods of Artificial Intelligence, Technical University of Berlin
FR 5–8, Franklinstraße 28/29, D 10587 Berlin
sppraum@cs.tu-berlin.de
http://ki.cs.tu-berlin.de/~sppraum

Abstract. We present an approach to spatial inference which is based on the procedural semantics of spatial relations. In contrast to qualitative reasoning, we do not use discrete symbolic models. Instead, relations between pairs of objects are represented by parameterized homogeneous transformation matrices with numerical constraints. A textual description of a spatial scene is transformed into a graph with objects and annotated local reference systems as nodes and relations as arcs. Inference is realized by multiplication of transformation matrices, constraint propagation and verification. Constraints consisting of equations and inequations containing trigonometric functions can be solved using machine learning techniques. By assigning values to the parameters and using heuristics for the placement of objects, a visualization of the described spatial layout can be generated from the graph.

1 Introduction

Understanding descriptions of spatial layouts implies that questions about spatial relations which were not included in the description can be answered, that the described scene can be visualized, or that it can be judged whether a depiction corresponds to the described layout. We propose an AI approach which realizes the first two claims: inference of spatial relations and construction of visualizations. From an application perspective, understanding of linguistic descriptions is a step towards a more natural human-machine communication, for example in the domains of robot instructions (*take the box from the* **right** *table*) and graphical user interfaces (*put the mailbox* **next to** *the file-manager window*). Furthermore, it might lead to a tool for visualization of textual descriptions (Jörding and Wachsmuth 1996). From a cognitive science perspective, our approach might provide a formal backbone to the theory of mental models in text understanding (Johnson-Laird 1983; Claus et al. 1998). Mental models are proposed to be the representation of a described situation that is constructed in addition to a propositional representation.

* This research was supported by the Deutsche Forschungsgemeinschaft (DFG) in the project "Modelling Inferences in Mental Models" (Wy 20/2–2) within the priority program on spatial cognition ("Raumkognition").

Ch. Freksa et al. (Eds.): Spatial Cognition II, LNAI 1849, pp. 212–224, 2000.

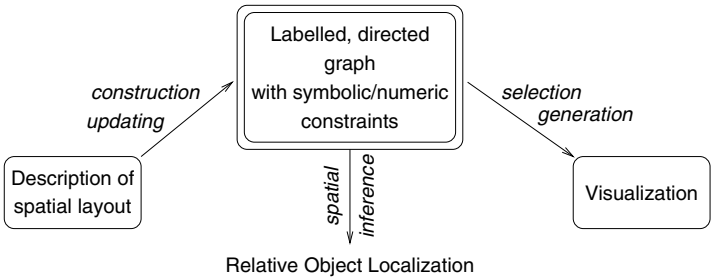


Fig. 1. Overview of the approach

In contrast to qualitative approaches to spatial reasoning, our approach is not based on discrete, symbolic models. Instead, we propose a metric approach with an underlying procedural semantics. While Clementini et al. (1997) demonstrate the advantages of qualitative reasoning, there are some critical issues. Finding the granularity appropriate for the problem and representing object orientations necessary to handle *intrinsic* relations are some of them. Besides, results presented in Musto et al. (2000) indicate that information loss during relation composition can be unacceptably high for long chains of qualitative inferences. In their paper, they use a mixed approach: measured metric values are abstracted to qualitative ones. In the example given, they delay the discretization to overcome this problem, and first gather the metric values for some steps before switching to qualitative values.

An overview of our quantitative approach is given in Fig. 1 (see also Claus et al. (1998); Schmid et al. (1999) for a more detailed description). Input into the system is the linguistic description of a spatial layout as a sequence of propositions. From this description, a graph is constructed incrementally: each object introduced in the text is represented as a node which is associated with a canonical coordinate system. The origin of the coordinate system represents the geometric center of the object. For objects with intrinsic axes, the positive x-axis represents the intrinsic right, the positive y-axis the intrinsic front and the positive z-axis the intrinsic top. Extensions of objects are represented by variables for their **h**eight, and their **w**idth and **d**epth, or **r**adius, respectively. Currently, objects are restricted to cuboids and cylinders.

Relations between pairs of objects A and B are represented by constraints for their transformation matrices (Ambler and Popplestone 1975). A homogeneous 4×4 transformation matrix contains a vector necessary to "translate" the origin of an object A to the origin of an object B, and a rotational part that aligns A's coordinate system with B's. The constraints restrict parameters of the transformation. For example, to represent *right(A, B)*, "B is intrinsically right of A", the x-value of the translation vector has to be at least half the width of A plus half the width of B to ensure nonoverlapping. Further constraints might restrict the admissible deviation on the front-back axis, for example with respect to a

defined sector for *right* (Hernández 1994, see Fig. 4(a)). Note that constraints are equations and inequations defined on *parameterized* relational representations. Each given relation is introduced as a labelled directed arc into the graph. Formally, the resulting graph represents the class of all models satisfying the given constraints. That is, our formalization of a mental model, i.e. the graph as a relational description together with the constraints on its parameters, describes *sets* of real spatial situations. This corresponds to the fact that language is under-determined with respect to depictions and to the claim that mental models represent classes of *possible* situations (Johnson-Laird 1983). Note that as a consequence the inference mechanism manipulates implicitly sets of possible real situations satisfying the constraints (i.e. the given relations)!

To infer relations between objects which were not given in the description, a path between these objects has to be found and the matrices along this path have to be multiplied. The resulting matrix with the constraints on its elements can then be compared with the predefined spatial relations. If it satisfies the constraints for such a relation, the relative position of the objects can be verbalized, otherwise it can at least be visualized. As is well known in robotics, transformations involve trigonometric functions in general so that known methods for constraint solving cannot be applied for inference. For this general case we propose to use a machine learning technique to induce the resulting relations from training examples.

Generating a visualization of the described scene involves new problems, because we have to find a globally consistent instantiation of all variables in the graph that also satisfies the general constraint (not included in the graph) that objects must be pairwise nonoverlapping. In the following, we will present our approach to spatial inference, machine learning, and visualization in more detail.

2 Inference

Inferring the spatial relation between two objects consists in propagating the constraints with respect to the positions of those objects and then checking which (if any) of the defined relations hold. The most important questions are *which* relations have to be inferred and *when* to infer them. This depends largely on the assumptions we make of the mental model. Some possibilities are discussed in Claus et al. (1998), Hörnig et al. (2000), and Hörnig et al. (1999).

While these questions are open, we organize the graph according to efficiency considerations. The graph contains all the relations explicitly given plus the arcs for the background knowledge that every object is in the room (thus ensuring that the graph is always connected), and those arcs that have been inferred after a query from the user. Inference is done when the relation between A and B is explicitly queried by the user and there is no arc between them in the graph, or when a relation between A and B is explicitly given and both objects are already represented in the graph. In the latter case, we start the inference process to ensure *that the constraints on the (new) arc contain all the information available about the relative positions of* A *and* B. As the graph is

connected, we can find a path between A and B, compute the transformation matrix for it, and equate with the matrix for the given relation.

The problems concerning constraint handling are due to the fact that the constraints consist of equations and inequations with parameters and trigonometric functions. Furthermore, we cannot always compare two constraints for the same variable. Therefore, the inference process often consists in gathering constraints for the variables rather than computing the intersection of those constraints and simplifying the problem. The occurrence of trigonometric functions means that for *unknown* rotations, algebraic methods cannot in general be applied to decide the satisfaction of the constraint inequations.

In our first approach described in Claus et al. (1998), sizes of objects were only constrained by the relations, e.g. in Fig. 3, we know that $2C.d + S.r \leq \Delta_x^{(S,W1)}$. Most constraints were purely symbolic. Considering that we need to find "realistic" values for the visualization (cf. section 3), we have augmented the object definitions to include default values and admissible intervals for the extensions. If we also use default values for object orientations, checking constraint satisfaction can often be done by case analysis using instantiations of the variables. This is shown in the example below. For the general case we have experimented with machine learning programs. Some results are presented in Sec. 2.2.

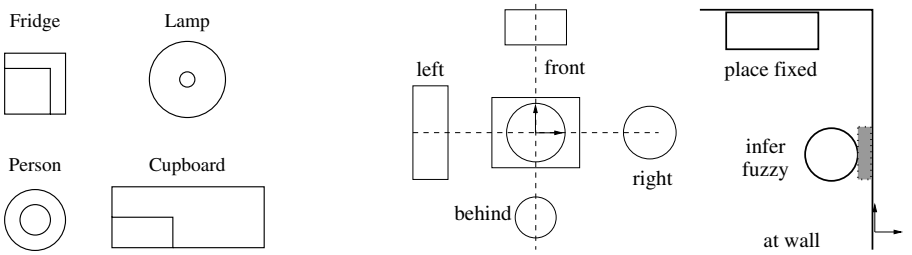
2.1 Example for the Inference of Spatial Relations

To keep things simple, consider the objects and relation definitions for the 2D case shown in Fig. 2: persons are represented by circles with a radius $r \in [0.2, 0.4]$, fridges are rectangles with a width $w \in [0.3, 0.4]$ and a depth $d \in [0.3, 0.4]$, cupboards are rectangles with a width $w \in [0.4, 1.0]$ and a depth $d \in [0.2, 0.5]$, lamps are circles with a radius $r \in [0.1, 0.5]$, and the room (not shown in the figure) is a rectangle with a width $w \in [1, 5]$ and a depth $d \in [1.5, 5]$. Persons, fridges, and cupboards have intrinsic front and right sides[1], while the orientation of the lamp's coordinate system is arbitrary.

For this example, the only defined relations are those for the four basic directions, where the center points of the located object must be on the corresponding axis in the coordinate system of the relatum, and the additional relation *at_wall*, where the default is to place an object directly against the wall, but inference of the relation is also possible when the distance is smaller than 0.5. The default orientation for objects standing at the wall is to orient their front sides (if they have an intrinsic front) away from it. The coordinate system for the walls are chosen such that the y-axes are oriented counterclockwise along the walls, while the positive x-axes always point outside the room (see Fig. 2).

Suppose we are given the propositions
(1) *right(S,C)* ("The cupboard C stands right of Stefanie S.")
(2) *at_wall(W1,C)* ("The cupboard stands at the wall $W1$.", (1) and (2) are the linearization of "The cupboard stands at the wall to the right of Stefanie.")

[1] The question of handedness is left out in this example.



(a) Shapes and sizes (b) A simple set of relations

Fig. 2. Example for objects and relations

(3) *front(S,F)* ("The fridge F stands in front of Stefanie.")
(4) *at_wall(W2,F)* ("The fridge stands at the wall $W2$.", (3) and (4) are the linearization of "The fridge stands at the wall in front of Stefanie." Thus we know that $W1 \neq W2$.)
(5) *left(F,L)* ("The lamp L stands left of the fridge.")
(6) *right(C,L)* ("The lamp stands right of the cupboard.")

Each of the first five propositions introduces a new object. Thus there cannot already be a path in the graph between the two objects, and no inference is necessary. The constraints for the first five propositions are

$$\Delta_x^{(S,C)} \geq S.r + C.d \qquad \Delta_y^{(S,C)} = 0 \qquad \theta^{(S,C)} = 90°$$

for (1) with the default orientation: front side away from the wall.

$$\Delta_x^{(W1,C)} = -C.d \qquad \text{default placement directly at the wall for (2)}$$

$$\Delta_y^{(S,F)} \geq S.r + F.d \qquad \Delta_x^{(S,F)} = 0 \qquad \theta^{(S,F)} = 180°$$

for (3) with the default orientation: front side away from the wall.

$$\Delta_x^{(W2,F)} = -F.d \qquad \text{default placement directly at the wall for (4),}$$

and $$\Delta_x^{(F,L)} \leq -F.w - L.r \qquad \Delta_y^{(F,L)} = 0 \qquad \theta^{(F,L)} = 0°$$

default orientation aligned with the fridge as the relatum, because the lamp has no intrinsic sides, for (5). When adding proposition (6), however, there is already a path in the graph (see Fig. 3).[2] We must now compute the transformation matrix for the path $L \rightarrow F \rightarrow S \rightarrow C$.[3] Equating the transformation matrix for the relation *right(C,L)* with the computed matrix we get

[2] The node for the room and the arcs to the room node are not shown.
[3] In the graph there is only an arc C → S, but arcs can easily be inverted.

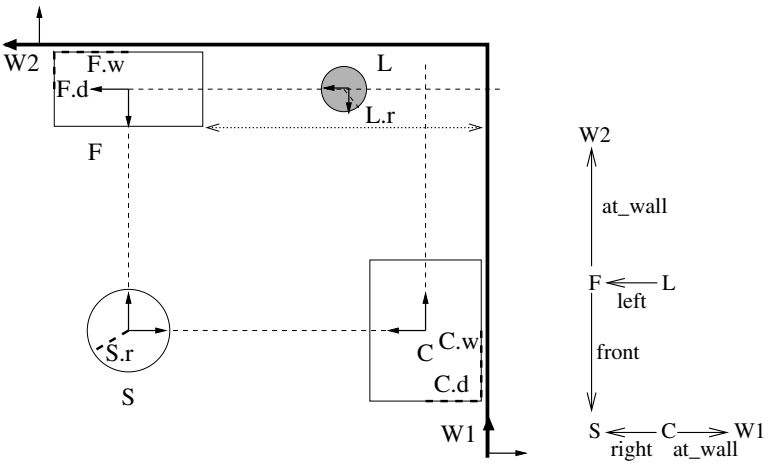


Fig. 3. Situation after propositions (1) to (5)

$$\begin{pmatrix} \cos\theta & -\sin\theta & \Delta_x^{(C,L)} \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & \Delta_y^{(S,F)} \\ 1 & 0 & \Delta_x^{(F,L)} + \Delta_x^{(S,C)} \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore we know that $\theta = 90°$, $\Delta_x^{(C,L)} = \Delta_y^{(S,F)}$, and $\Delta_x^{(F,L)} = -\Delta_x^{(S,C)}$ and can substitute for the variables on the left sides. These constraints express the fact that the lamp must be positioned on the point where the left/right axis of C's coordinate system crosses the left/right axis of F's coordinate system.

Another situation arises, when a query is given to the system. For instance, we might ask the question *at_wall(W?,L)*. This means we have to check for every wall, whether the lamp is standing at it. This is done by finding a path between the lamp and the corresponding wall, computing the transformation matrix and then checking whether the constraints for the relation *at_wall* are satisfied.

For the wall $W1$ we can compute $T^{(W1,L)} = T^{(W1,C)} \times T^{(C,L)}$.
This matrix consists of a rotation matrix for 180°,

$$\Delta_x^{(W1,L)} = -C.d \qquad \text{and} \qquad \Delta_y^{(W1,L)} = \Delta_y^{(S,F)} + \Delta_y^{(W1,C)}$$

As shown in Fig. 2(b), the only constraint is $\Delta_x^{(W1,L)} \geq -L.r - 0.5$,
i.e. the distance of the lamp w.r.t. the wall must be at most 0.5.
Substituting for $\Delta_x^{(W1,L)}$ gives

$$-C.d \geq -L.r - 0.5 \qquad \Leftrightarrow \qquad L.r \geq C.d - 0.5.$$

If we assign $C.d$ its maximal value, we get $L.r \geq 0.5 - 0.5 = 0$
and from $L.r > 0$, we can conclude that the relation holds for all possible values of $C.d$ and $L.r$. Thus we know that *at_wall(W1,L)* is true.
Analogously, we can prove that *at_wall(W2,L)* holds.

Now consider the case *at_wall(W3,L)*. Using the background knowledge (a room has 4 walls, $W1$ is opposite $W3$), we can compute

$$T^{(W3,L)} = T^{(W3,W1)} \times T^{(W1,L)} \qquad \text{(one possible path)}.$$

We get a rotation matrix for 0°,

$$\Delta_x^{(W3,L)} = C.d - 2R.w, \qquad \text{and} \qquad \Delta_y^{(W3,L)} = -\Delta_y^{(S,F)} - \Delta_y^{(W1,C)},$$

where $R.w$ is the width of the room. Again, we have to check whether

$$\Delta_x^{(W3,L)} \geq -L.r - 0.5. \qquad \text{Substituting for } \Delta_x^{(W3,L)} \text{ gives}$$

$$C.d - 2R.w \geq -L.r - 0.5 \quad \Leftrightarrow \quad L.r \geq 2R.w - C.d - 0.5.$$

Checking for the upper bound of the right expression, we get

$$L.r \geq 10 - 0.2 - 0.5 = 9.3.$$

This cannot be true, therefore we know that the lamp *need not* stand at $W3$. To complete, we check for the lower bound of the right expression and get

$$L.r \geq 2 - 0.5 - 0.5 = 1.$$

This is false, too. Thus we know that the lamp *cannot* stand at wall $W3$. The computation for $W4$ is analogous.

2.2 Machine Learning

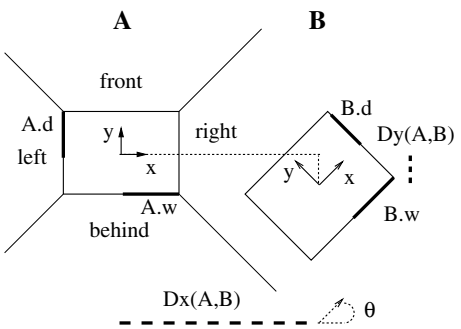
In the general case, the constraints for an intrinsic relation between two objects contain the extensions of both objects, the elements of the translation vector, and the elements of the rotation matrix. Even in the restricted case, where all objects stand upright, and rotation is only allowed around the z-axis, the constraints for a relation involve seven variables. For example, the binary relation *right(x,y)* is transformed into a 7-dimensional constraint, represented by a region in the 7-dimensional parameter space. What we are really interested in is the border in this space between the region(s) where the relation holds and the inadmissible regions where it does not hold. An approximation of this border can be found by machine learning programs.

In our case, such programs are fed a set of training data each consisting of a 7-dimensional parameter vector, which uniquely describes a spatial constellation of the two objects, and a class value (e.g. R for right and N for not right, see Fig. 4(a)). From these training samples, a classifier is constructed that can decide for arbitrary constellations whether an object falls into the *right* region of another object or not. Using this classifier is in general faster than checking the (in)equations for the parameter values for high dimensional parameter spaces (in the simple example below, there are 12 inequations). For the learning task, in principle any standard algorithm for classification learning, which separates class regions in a continuous parameter space, can be applied. As high accuracy in approximating the class boundaries is wanted in our case, an artificial neuronal net of the Perceptron type was preferred. We have used *Dipol92* (Schulmeister and Wysotzki 1997), a hybrid (statistical/neural) algorithm that computes for a given training set and given numbers of clusters for every class, e.g. R and N (the numbers of clusters are the only parameters to be chosen by the user), the discriminating hyperplanes for each pair of clusters.

By decomposing each class into a set of clusters, a piecewise linear approximation of the class boundaries is computed, which can be converted into a decision function for the constraint satisfaction problem, which could then be used in the inference process.

A critical issue is the distribution and number of training data to be generated to obtain satisfactorily low error rates. The task was to try whether we can get smaller errors while using fewer data for the training set when we *selectively* generate data near the boundary to be learned. For the sample problem shown below, we had admissible intervals for all seven parameters, and also preferred intervals where more data had to be generated. This was motivated by the assumption that medium sized objects occur more often (i.e. have higher probability) than very small or very big ones, and the distance values $Dx(A, B)$ can only reach their maximum when both objects are positioned at opposite walls. The seven parameter values were then generated independently (i.e. a small $A.w$ value can be paired with a large $A.d$ value), and the class value (R or N) was computed using the (in)equations defining the constraint (relation) *right*. As *right* is defined such that B must be completely included in the *right* region of A, only $\approx 20\%$ of the data were in class R when data were generated randomly, but respecting the given admissible and preferred intervals for each parameter. This method was called REF. But since our task is learning class boundaries by exploration ("active learning"), i.e. by *generating* training sets, it is desirable to have equal a priori probabilities for both classes to get optimal decision functions (classifiers).

This problem is discussed in Stolp et al. (1999). Among other methods we tried out, EPS2 was best. Here, data are generated according to the REF method and then filtered. This means, only those data vectors $(A.w, A.d, B.w, B.d, Dx(A, B), Dy(A, B), \theta)$, where changing any one parameter value by $+\epsilon$ or $-\epsilon$



(a) *Right* for two rectangles

Data gen.	ϵ	train error	test error
1000 points for training			
REF	/	0.50%	6.64%
EPS2	1.5	1.20%	3.24%
2500 points for training			
REF	/	1.36%	5.20%
EPS2	1.5	5.28%	2.66%
25000 points for training			
REF	/	3.04%	3.75%
EPS2	1.5	8.10%	1.93%

(b) Results with Dipol

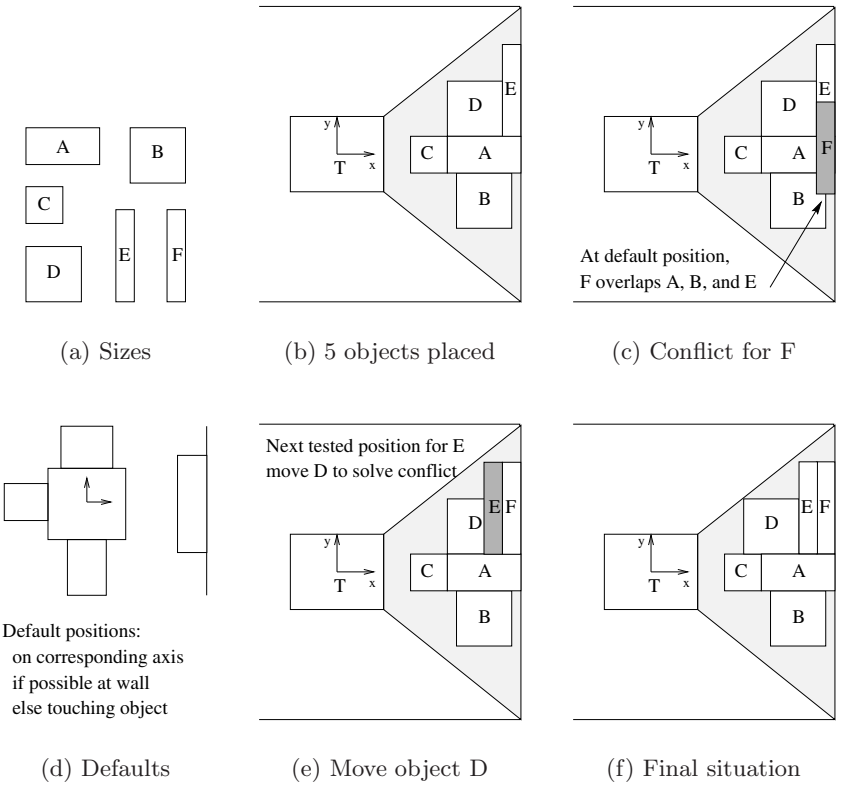
Fig. 4. Learning *right* for two rectangles with Dipol

changed the class (from N to R or vice versa) were kept. This has the advantage that all values are near the critical border, both classes get the same number of training samples, and still the preferred intervals for every variable are respected. An appropriate value for ϵ was found in some training runs. The number of clusters for each class was set to 50 for both R and N. The table in Fig. 4(b) shows the train errors (errors on the data set used for training) and test errors (errors on a different test set of data for the learned classifier). For testing, a set with 100 000 points generated with REF was used throughout. As can be seen in the table, the generalization is quite good for both methods. For reasonably large sets of training data, the errors on the training set and on the test set are nearly equal for REF. Due to the fact that all data points are near the border, EPS2 has higher errors on the training set, but surprisingly small errors on the test set. Our main goal, to achieve good error rates with a small number of training data, was fulfilled. Using EPS2 for generating data, we got a lower test error with 1000 training vectors than by using REF with 25.000 training vectors. A more detailed account is given in Stolp et al. (1999) and Wiebrock and Wysotzki (1999). Currently, we are working on the generation of depictions using the learned regions where the relations hold. Depending on the efficiency of this method, we may use it in the future as an alternative to the algorithm described below.

3 Generating Depictions

When generating depictions, we are faced with several problems: all variables in the graph must be instantiated, in addition to the explicitly represented constraints we must ensure that two objects do not overlap, and we must choose a perspective for the visual representation. The last point leads back to the organization of the mental model (see Sec. 2). As we are using VRML to visualize the scenes, we have to compute positions in a global coordinate system for every object. The perspective is set afterwards and can be changed interactively. We generate the depiction parallel to the graph, i.e. immediately update the depiction for every new proposition. For every object, default sizes are given, and for every relation, default positions are provided.

Classical constraint propagation algorithms like Waltz (1975) work for a finite number of possible instantiations for each variable. The algorithm for Parcon, a program for handling objects in GUIs described in Uhr et al. (1997), is a variant for intervals over real numbers. The algorithm handles disjunctive constraints by copying the constraint network and working in parallel on all copies. For our domain, where we have lots of disjunctive constraints (i.e. nonoverlapping), this would not be efficient, because the number of models to check grows exponentially in the number of objects. Therefore we propose a simpler variant involving backtracking. We can divide the variables occuring in the constraints into three classes: 1) object sizes, 2) relative object rotations, and 3) distance parameters. The order is meant to indicate the "variability" of the values: object sizes are least likely to be changed, and distance values are most variable. In the im-



(a) Sizes (b) 5 objects placed (c) Conflict for F

(d) Defaults (e) Move object D (f) Final situation

Fig. 5. Example problem with six objects

plemented algorithm outlined in Tab. 1, only distance values are manipulated. Rotations are restricted to multiples of 90° and are fixed.

For every object, we know its size and coordinates, the constraints for its position, and the objects the position depends on. We also keep a list *posl* of possible positions. If an object is to be placed, we first try the default (= preferred) position. If there is a collision, we move the object front/left/back/right by the amount necessary to avoid the collision and add those positions to *posl*. For the actual position, we store the name(s) of the colliding object(s) and mark the position as visited. For the colliding objects, too, we try wether by moving them we find a new position to be tested. To avoid cycles, object positions are stored. We consider positions equal if they differ by less than a given ϵ value. This limits the number of possible positions and ensures termination while for extreme examples we don't find all possible solutions.

As an example for the algorithm, consider the following problem: there are six objects A to F. Each object is right of T, i.e. must be placed in the grey area.

Table 1. Overview of the placing algorithm

```
type Obj = {name, extensions, LINE list *bounds, Obj list *dep_obs}
                /* bounds: list of lines bordering the admissible area
                   dep_obs: objects with parameters mentioned in bounds */
type Pos = {x, y, Obj list *coll} /* coll: objects colliding at position (x,y) */
type Scene = {Obj *obj, Pos *act_pos, Pos *pref_pos,
              Obj list *coll, Pos list *posl } /* posl is initialized to pref_pos */
              /* colliding objects at act_pos, and alternative positions posl */
type Config = {Scene list *placed, Scene list *add}
bool place_obj (Scene list *placed, Scene *curr)
/* try to add new object without repositioning objects already placed */
{ repeat collision := false;
    pos := first unvisited position in curr→posl;
    forall obj in placed overlapping curr at pos do
    { move curr front/left/back/right to avoid conflict and add positions to curr→posl;
      move obj  front/left/back/right to avoid conflict and add positions to obj→posl;
      /* positions are only added if they are not already on the list*/
      add obj to curr→coll; collision := true; }
    mark pos as visited;
  until not collision or no more unvisited positions in curr→posl;
  return collision; }
bool place_list (Scene list *placed, Scene list *add)
/* try to place each object in add without repositioning objects already placed */
{ curr := first object in add;
  repeat success := true;
    if place_obj (placed, curr)
    { move curr from add to placed;
      if not place_list (placed, add)
      { mark act_pos in curr as visited; success := false
        move curr from placed to add; } }
    else success := false;
  until success or no more unvisited positions in curr→posl;
  return success; }
bool place_all (Scene list *placed, Scene list *add, Config list *all)
/* try to place each object in add; if necessary objects are removed from placed list;
   all tested configurations are stored to avoid loops */
{ conf = {placed, add};
  repeat success := true;
    if not place_list(conf.placed, conf.add)
    { unplaced := first element of conf.add; success := false;
      forall coll in unplaced→posl do
      { new_conf := conf with all scenes for objects in coll moved from placed to add;
        insert new_conf into all such that all is sorted by expected effort; }
        /* least number of remaining objects, shortest collision list */
      conf := next element of all; }
  until success or not conf; /* no more untested configurations */
  if success { placed := conf.placed; add := conf.add; }
  return success; }
```

Additional constraints are: *behind(A,B)* (B is behind A) and *left(A,C)* (C is left of A). Suppose that all objects have the same orientation as T. Object sizes are shown in Fig. 5(a) and default positions in Fig. 5(d). When objects A to E are placed, we have the situation shown in Fig. 5(b). Now trying to place F at the default position gives the conflict shown in Fig. 5(c). To resolve the conflict, object E is removed, because A and B are interdependent. F is positioned at E's place. Now we have to re-place E. The next position tried is to the left of its former position, see Fig. 5(e). This gives a conflict with D. D is moved by the width of E, and E is included, see Fig. 5(f). To compute the scene depicted here, there were between 29 and 73 positions stored for the objects (314 for the six objects together), and 174 constellations were tested.

4 Conclusions

In the text above, we have considered three different representations of a situation: the propositional description of the spatial constellation, the graph as the structural representation constructed from it, and a visual representation of the situation. Models of spatial representation based on the work of Kosslyn, e.g. Schlieder and Behrendt (1998), postulate that beside the mental *model* a mental *image* is constructed that can be used for inspection. Unlike the mental model, this image is necessarily bound to a fixed perspective. This can make updating the model harder when the texts induce a change of perspective. Some results discussed in Hörnig, Eyferth, and Gärtner (2000) indicate that the visual representation is only available as long as the perspective does not change. Possibly the cognitive effort for updating both representations simultaneously would be too high, otherwise.

For our AI model, the question of a global reference frame is relevant for the updating and inference process where we have to decide which relations are represented explicitly in the graph and available without inference. Several possibilities are discussed in Hörnig et al. (1999), and Hörnig et al. (2000). Choosing a perspective for the generated depiction is a related problem. Another one concerns finding good heuristics for the instantiation of variables, especially with respect to the plausibility of the resulting depiction. While there is some evidence for the admissible regions and default positions for simple spatial relations, it is still unclear how to generalize those results for complex scenes.

References

Ambler, A. P. and R. J. Popplestone (1975). Inferring the Positions of Bodies from Specified Spatial Relationships. *Artificial Intelligence 6*, 157–174. 213

Claus, B., K. Eyferth, C. Gips, R. Hörnig, U. Schmid, S. Wiebrock, and F. Wysotzki (1998). Reference Frames for Spatial Inferences in Text Comprehension. In C. Freksa, C. Habel, and K. F. Wender (Eds.), *Spatial Cognition - An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, Springer. 212

Clementini, E., P. D. Felice, and D. Hernández (1997). Qualitative representation of positional information. *Artificial Intelligence 95*, 317–356.

Hernández, D. (1994). *Qualitative Representation of Spatial Knowledge*. Springer. 214

Hörnig, R., B. Claus, and K. Eyferth (1999). In Search for an Overall Organizing Principle in Spatial Mental Models: A Question of Inference. In S. O'Nuallain and M. Hagerty (Eds.), *Spatial Cognition; Foundations and Applications*. John Benjamins. Forthcoming.

Hörnig, R., K. Eyferth, and H. Gärtner (2000). Orienting and Reorienting in Egocentric Mental Models. This volume.

Johnson-Laird, P. N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference and Consciousness*. Cambridge: Cambridge University Press. 212, 214

Jörding, T. and I. Wachsmuth (1996). An Antropomorphic Agent for the Use of Spatial Language. In *Proceedings of ECAI'96-Workshop on Representation and Processing of Spatial Expressions*, S. 41–53. 212

Musto, A., K. Stein, A. Eisenkolb, T. Röfer, W. Brauer, and K. Schill (2000). From Motion Observation to Qualitative Motion Representation. This volume.

Schlieder, C. and B. Behrendt (1998). Mental model construction in spatial reasoning: a comparison of two computational theories. In U. Schmid, J. Krems, and F. Wysotzki (Eds.), *Mind modelling: a cognitive science approach to reasoning, learning, and discovery*, Lengerich, S. 133–162. Pabst Science Publishers.

Schmid, U., S. Wiebrock, and F. Wysotzki (1999). Modelling Spatial Inferences in Text Understanding. In S. O'Nuallain and M. Hagerty (Eds.), *Spatial Cognition; Foundations and Applications*. John Benjamins. Forthcoming.

Schulmeister, B. and F. Wysotzki (1997). DIPOL – A Hybrid Piecewise Linear Classifier. In G. Nakhaeizadeh and C. C. Taylor (Eds.), *Machine Learning and Statistics - The Interface*, S. 133–152. Wiley. 218

Stolp, R., B. Weber, M. Müller, S. Wiebrock, and F. Wysotzki (1999). Zielgerichtete Trainingsmethoden des Maschinellen Lernens am Beispiel von DIPOL. In P. Perner (Ed.), *Maschinelles Lernen, FGML'99*. IBaI Report.

Uhr, H., P. Griebel, M. Pöpping, and G. Szwillus (1997). Parcon: ein schneller Solver für grafische Constraints. *KI 97*(1), 40–45.

Waltz, D. (1975). Understanding Line Drawings of Scenes with Shadows. In P. H. Winston (Ed.), *Psychology of Computer Vision*. New York: McGraw-Hill.

Wiebrock, S. and F. Wysotzki (1999). Lernen von räumlichen Relationen mit CAL5 und DIPOL. Fachberichte des Fachbereichs Informatik No. 99-17, TU Berlin.