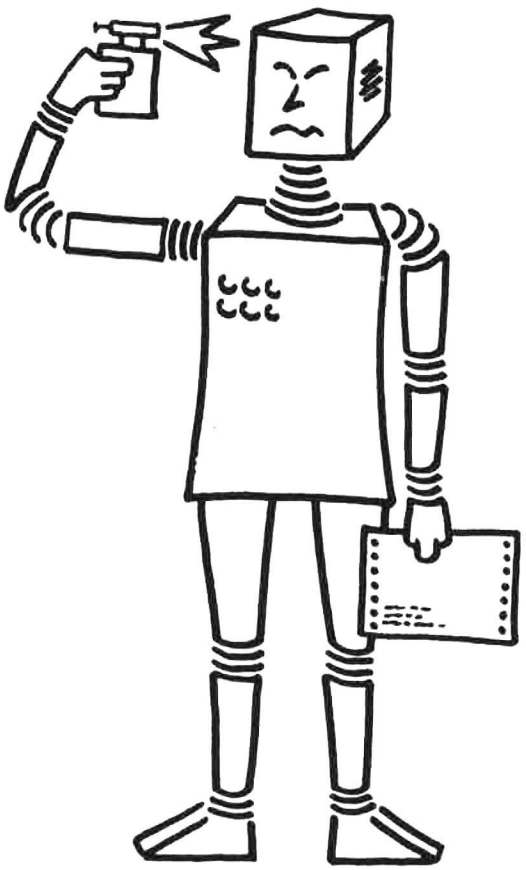
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany

SEKI-PROJEKT

SEKI MEMO



Distributed Constraint Propagation: A Case Study

Michael Reinfrank

SEKI Memo 84-07 September 1984

Interner Bericht

Distributed Constraint Propagation:
A Case Study

Michael Reinfrank

112/84 September 1984

Fachbereich Informatik

ersität Kaiserslautern · Postfach 3049 · D-6750 Kaiserslautern

Distributed Constraint Propagation:
A Case Study

Michael Reinfrank

112/84 September 1984

ABSTRACT

Constraint satisfaction techniques are now widely used in AI problem solving. Discrete constraint propagation algorithms were brought into prominence by David Waltz, who applied them to the problem of finding consistent labellings of line drawings. This paper presents the realization of a Waltz-like algorithm in terms of a society of cooperating agents. The architecture of SCENELAB is sketched, a computer system for labelling pictures drawn from scenes in the blocks world. SCENELAB is implemented in CSSA, a programming language for the realization of asynchronously concurrent processes. It is argued that this approach is not restricted to scene labelling algorithms. A class of constraints is identified which is suited to similar solutions and which SCENELAB is able to handle. Finally, some further points are more briefly addressed: a comparison of the present work to sequential and synchronized parallel versions of the Waltz-algorithm, and the problems raised by the occurrence of incomplete and erroneous picture data in real world applications.

0. Table of Contents

1. Introduction

Constraint satisfaction techniques are now widely used in AI problem solving. Discrete constraint propagation algorithms were brought into prominence by David Waltz (Wa72), who applied them to the problem of finding consistent labellings of line drawings. This paper presents the realization of a Waltz-like algorithm in terms of a society of cooperating agents. The architecture of SCENELAB (Re83), (Re84) is sketched, a computer system for labelling pictures drawn from scenes in the blocks world. SCENELAB is implemented in CSSA (BMS82), a programming language for the realization of asynchronously concurrent processes. It is argued that this approach is not restricted to scene labelling algorithms. A class of constraints is identified which is suited to similar solutions and which SCENELAB is able to handle. Finally, some further points are more briefly addressed: A comparison of the present work to sequential and synchronized parallel versions (Roetal76) of the Waltz-algorithm, and the problems raised by the occurrence of incomplete and erroneous picture data in real world applications. The problem of detecting distributed termination, which had to be solved for SCENELAB, too, is mentioned but not discussed in detail, as this is done elsewhere (BMR84).

2. The Waltz-Algorithm: Finding Labellings for Line Drawings

The purpose of scene analysis is to "understand" a two-dimensional drawing, that is to find an interpretation in terms of the scene it represents. A substantial part of this task is frequently accomplished by finding an attachment of labels to picture elements, a label standing for a certain property the corresponding scene element has. We will restrict our considerations to line drawings representing scenes from a so-called "trihedral" world. A trihedral world contains polyhedra, where at every vertex exactly three surfaces meet. The drawings then are composed of (straight-line)-segments, junctions, and regions, representing edges, vertexes, and surfaces, respectively.

2.1 The Huffman-Clowes Label Set

We are going to work with a very crude set of only four different segment labels, first introduced by Huffman (Hu71) and Clowes (Cl71). For a discussion of the expressive power of label sets see e.g. (Fr82). Figure 1 shows these labels and the meanings associated with them.

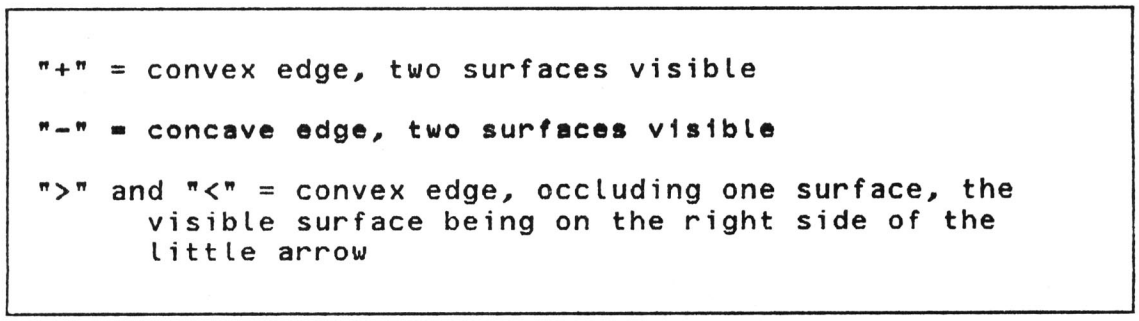


Fig. 1 : The Huffman/Clowes segment labels

The crucial point now is that trihedral junctions exhibit only four different prototypical geometric shapes - so-called junction figures - constraining the way in which their segments may be labelled in a specific manner. We call a list of potentially admissible composite junction labels a "label dictionary" (figure 2).

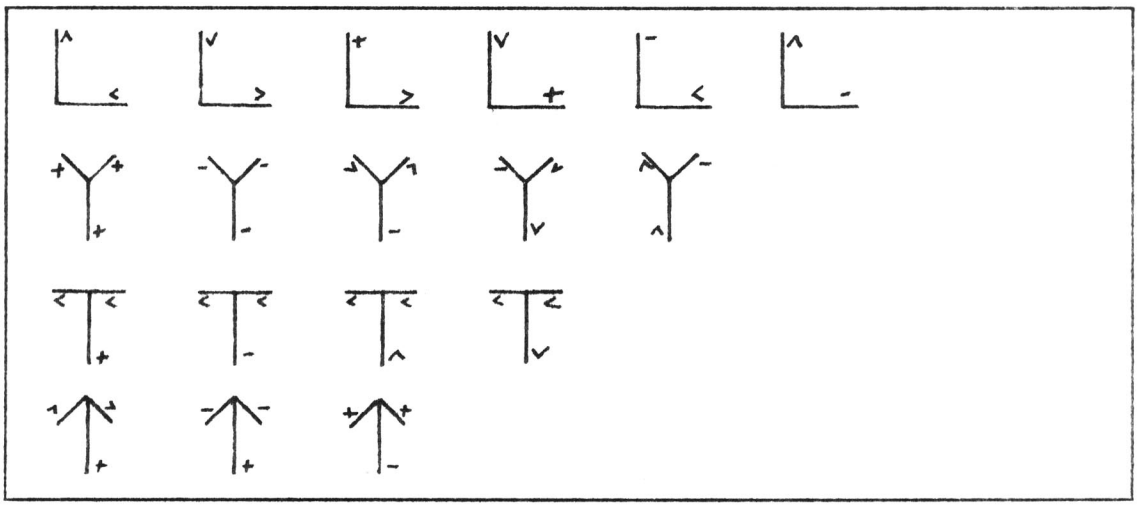


Fig. 2: The physically possible labellings of the trihedral junction figures ELL, FORK, TEE and ARROW form only a small subset of the 64 combinatorially possible labellings.

It is clear that, within the context of a line drawing, the composite labels of two adjacent junctions must induce two segment labels for the segment joining them which are compatible, i.e. which associate the same interpretation for that segment. A first step towards the interpretation of a line drawing is made by finding a "consistent" labelling, where the labels of any two neighboring junctions are mutually compatible. Figure 3 shows a cube consistently labelled with labels from figure 2.

As two adjacent junctions view their common segment from two complementary viewpoints, it should be clear that the label ">" is compatible with "<" and vice versa. In (Re84) we therefore used the labels "o>" and "o<", making the viewpoint of a junction explicit. For the sake of simplicity, we adopt here the notation most commonly used in the literature on scene labelling, although it might suggest that "<" and ">" are compatible with themselves.

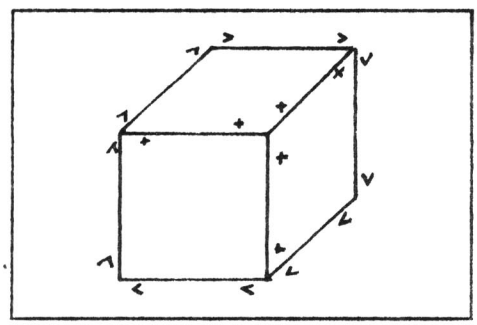


Figure 3: Consistent labelling of a cube. Note that the labels < and > may be interpreted as "ingoing" and "outgoing" segments when viewed from a junction.

Obviously, if the label dictionary is "sound" and "complete", representing exactly all physically possible combinations, a drawing of a real world scene must have at least one consistent labelling. (Procedures to obtain such a dictionary are given

e.g. in (Wa72) and (Wi77).) Unfortunately, the converse does not hold: Figure 4 shows the labelled drawing of an impossible object, together with the line drawing of another impossible object for which no consistent labelling can be found. The drawings are taken from Sugihara (Su82).

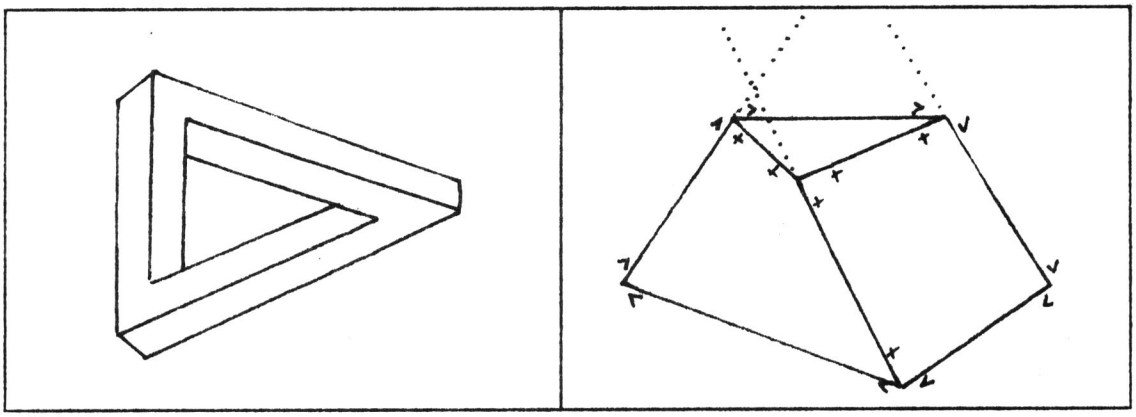


Fig. 4: Unlabellable drawing Labelled drawing of an impossible object

2.2 The Waltz-Algorithm

Exhaustive search methods are inefficient in finding consistent labellings (Wi77). This is true especially for drawings from less restrictive worlds, where a much more comprehensive label set is needed. Waltz (Wa72), using more than 3,000 different junction labels, suggested an algorithm to substantially prune the otherwise intractable search space by means of a locally restricted search. This solution is motivated by the following observation: A junction label from the original list of potentially admissible labels for a certain junction figure can only contribute to a consistent labelling if at least one compatible label can be found for every neighbor of the junction in consideration. Properly spoken, the Waltz-algorithm works as follows:

For every junction J do - in an arbitrary sequence -
 (1) Assign the list of all potentially admissible labels to J and mark J
 (2) Remove all the labels from J not having at least one compatible label at every marked neignbor N of J
 (3) For all marked neighbors N of J do
  Remove all labels from N having no compatible label at J
  Recursively repeat (3) for all marked neighbors N' of N until no more deletions occur

In (Re84) a more formal description of this algorithm is given and its correctness is proved. For a detailed discussion see (Wa72) or (Wi75).

2.3 The Consistent Labelling Problem

The problem of finding consistent labellings of a graph is not restricted to scene analysis applications but is an instance of the general so-called "Consistent Labelling Problem" CLP (HaSh79,80), defined as follows: Given a set $\{v_1,v_2,\ldots,v_n\}$ of variables which may take values from domains $D_1,D_2,\ldots,D_n$, and a number of constraint relations of the form

$R \subseteq D_{i1} \times D_{i2} \times \ldots \times D_{ik}$
where $k \in \{1,2,\ldots,n\}$
and $i_1,i_2,\ldots,i_k \in \{1,2,\ldots,n\}$,

find a tuple (or, all the tuples) $(d_1,d_2,\ldots,d_n)$ of values "satisfying" all the relations R in the sense that for every R, $(d_{i1},d_{i2},\ldots,d_{ik}) \in R$ holds. The scene labelling problem fits into that scheme either as a so-called "binary" CLP, with the junctions as variables, the junction labels as domains, and the binary relations constraining the labels of adjacent junctions, or as an "n-ary" CLP (n being the maximal edge degree in the line drawing), with the segments as variables, the segment labels as domains, and the relations constraining the labels of segments intersecting at a common junction. For more details see (Re84). An overview over CLP's and their algorithms is given e.g. in (Nu83).

Binary CLP's are often represented as networks, the nodes standing for variables and the edges standing for relations. The Waltz-algorithm captures the central idea of constraint propagation in such a network. Discrete, local constraints are evaluated at a single node (steps (1) and (2)). These constraints are then propagated to the environment of this node (step (3)), and are eventually distributed over the whole network (recursive repetition of step (3)). This basic idea has been generalized by Steele (G.St80) (among others) to a general purpose constraint language.

Constraints and constraint satisfaction techniques have been used in a wide range of AI-applications as, among others, in planning (M.St.81), scheduling (Foetal83), simulation (A.Bo79), graphics (Go83), natural language understanding (He82), deduction (McA80), temporal reasoning (AlKa84), and diagnosis (StSu77). It should be clear that the representation of constraints by means of simple tables as e.g. our label dictionary is inadequate for more complex constraint relations, especially when the domains are infinite. Moreover, constraint satisfaction techniques are not restricted to an unplanned, exhaustive propagation of local constraints (see e.g. (Go83) for more elaborate strategies). However, these issues lie beyond the scope of the present paper.

3. SCENELAB: Scene Labelling by a Society of Agents

For AI problem solving, the concept of systems of cooperating agents has become an important paradigm. An agent is an independent unit familiar with a part of the problem at hand, providing part of the expertise needed to develop a solution for the problem. Cooperating agents coordinate their efforts by the exchange of messages reporting partial results and delegating subtasks.

3.1 The Basic Idea

The central idea of the present work is to use a society of agents to perform a Waltz-like labelling. Given a line drawing, one distinct agent is created for every junction (see figure 5). The agent is initially supplied with a list of all potentially admissible labels for a junction of this type.

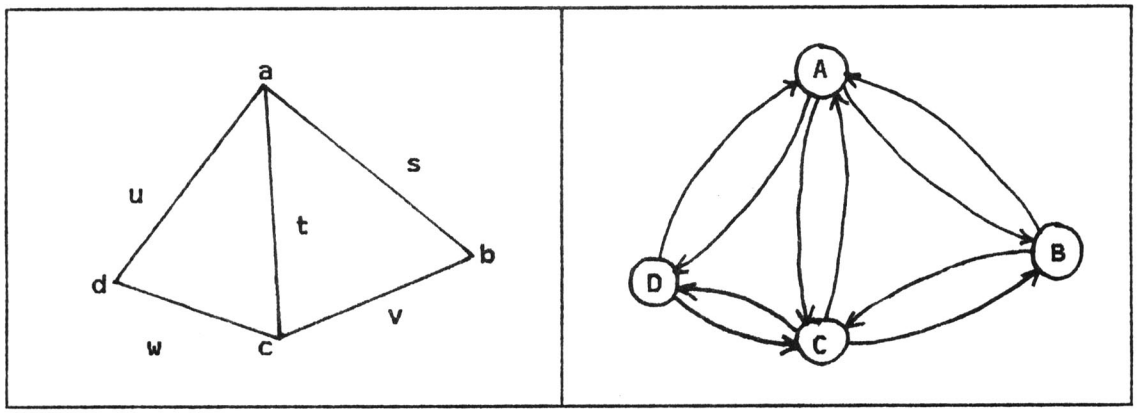


Figure 5: A society of agents working on the drawing of a tetraeder. Agents are represented by nodes, acquaintances by arcs.

The agents then start to communicate with their neighbors - two agents are acquainted to each other if the corresponding junctions are adjacent - telling each other what labels remain possible from their individual points of view. In order to keep their labels compatible, two agents must agree upon what labels they intend to attach to their junctions. This agreement is achieved by means of the following procedure: Every agent keeps on observing what kind of segment labels may be consistently assigned to the segments of his junction, with respect to his actual list of junction labels. Whenever a certain label is ruled out, he sends a message to his corresponding neighbor. Upon the receipt of such a message, an agent removes all the junction labels which would induce this specific segment label for the segment in consideration, and checks whether these deletions might give rise to further inconsistencies, i.e. if further segment labels have become impossible as a consequence of the deletions. Figure 6 shows the activities of the agent A from figure 5 after receiving a message from agent B excluding the label '+' for the segment joining A and B.

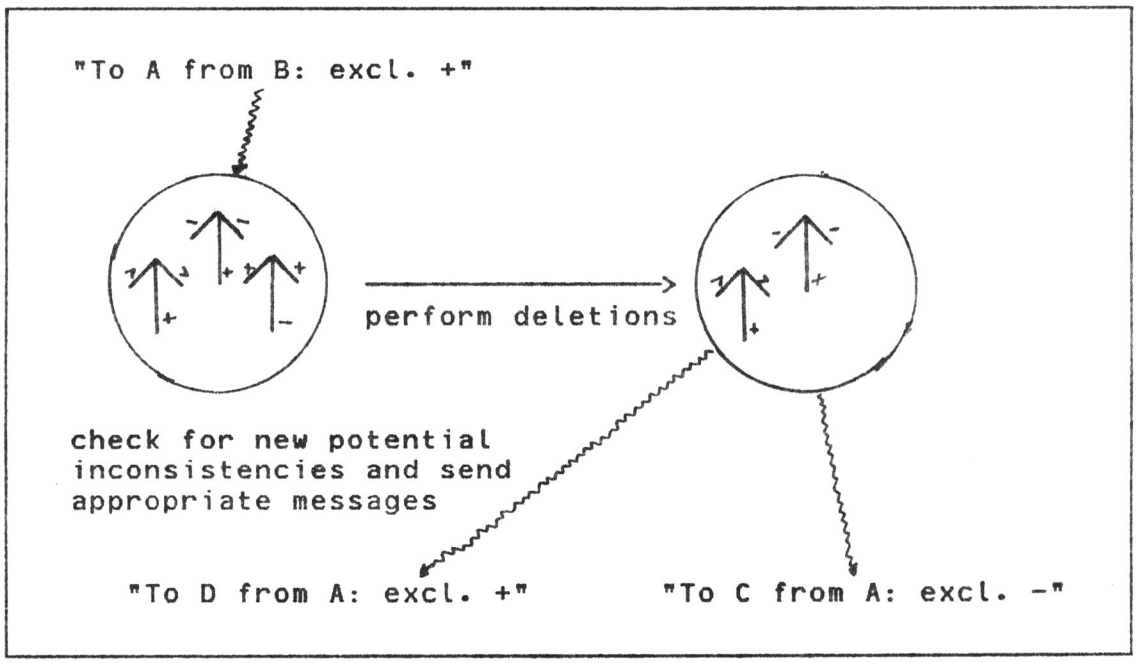


Fig. 6: An agent evaluates and propagates constraints

3.2 SCENELAB

In figure 7, the architecture of SCENELAB is sketched, a fully implemented computer system for scene labelling based on the principles outlined above. The kernel of the system is constituted by a constraint network of cooperating agents called Local Analysts.

The user specifies a picture description and a label dictionary in a special language PDL (for picture description language). He is completely free in defining junction figures and labels of his own. This makes SCENELAB no general purpose constraint system as e.g. those reported in (G.St80) or (L.St82), but it enables the user to define and handle arbitrary constraint relations which exhibit the same structure as those given for scene labelling purposes, i.e. finite constraint relations where the compatibility between composite node labels reduces to a one-to-one compatibility between edge labels. Here are the PDL-definitions of the junction figure ARROW and of a "figure" MOD-2-ADDER:

```
def-lab +;-;<;>.                    The basic segment labels.
spec-com +:+;-:-;<:>;>:<.           Compatible segment labels.
def-fig ARROW.                      A named figure.
spec-deg ARROW: 3.                  An ARROW is of degree 3.
spec-lab ARROW: +/-/+,+/+/+,>/+/<.  Admissible ARROW-labellings.
```

```
def-lab 0;1.
spec-com 0:0;1:1.
def-fig MOD-2-ADDER.
spec-deg MOD-2-ADDER: 3.
spec-lab MOD-2-ADDER: 0/0/0,0/1/1,1/0/1,1/1/0.
```

The following PDL-statements specify a part of the tetraeder in figure 5, the segments s,t,u joining the junction a with b,c, and d, respectively.

```
def-seg s;t;u;v;w.          Named segments.
def-jun a;b;c;d.            Named junctions.
spec-fig a: ARROW.          A is an ARROW.
spec-gra a: s/b,t/c,u/d.    Structure of the picture graph at a.
```

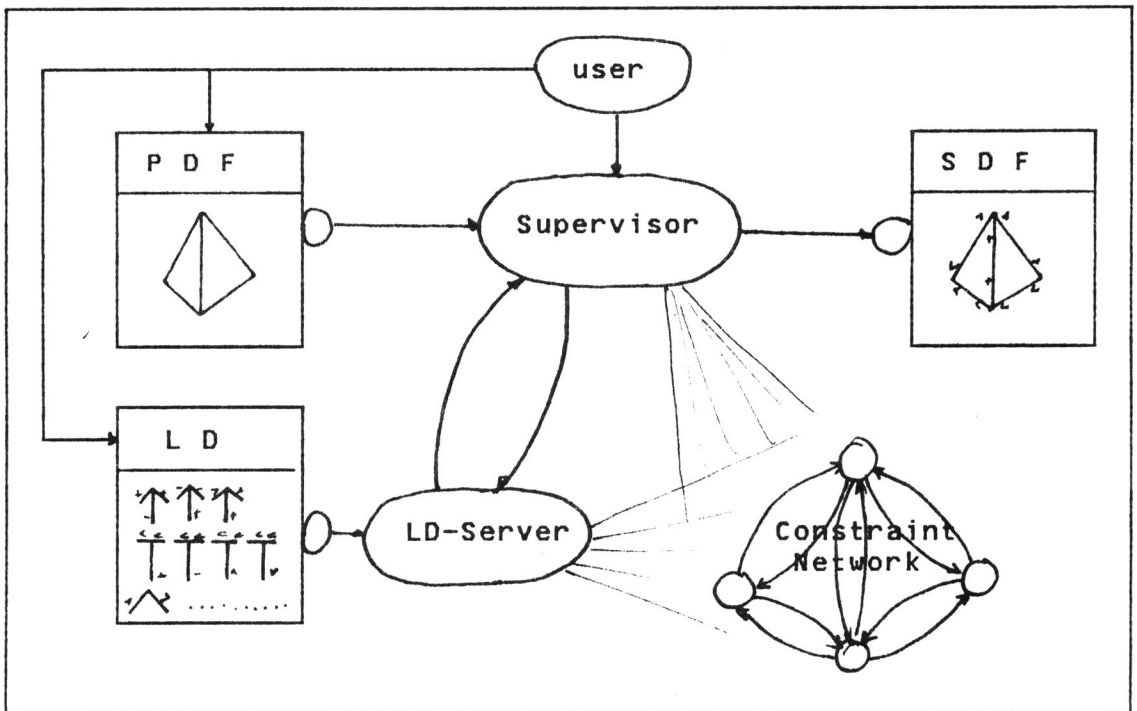


Fig. 7 : The architecture of SCENELAB

The files LD and PDF are logical files which can be linked to predefined files - as e.g. a standard label dictionary - or to interactive input devices. They contain the PDL-representations of the picture description and of a label dictionary and are incrementally translated into an internal representation. PDL as well as the internal data structures are realizations of the model presented in chapter 3 of this paper. The user is integrated into the system as a special agent and communicates with the Supervisor and others. He controls the system by the sending of messages. The constraint net is dynamically generated by the Supervisor. The following sequence of commands would lead to the construction of a constraint net to label a drawing being

described on a PDF linked to a special agent PDF-reader with labels from a LD linked to a LD-reader.

```
send INTERPRETE(PDF-READER) to SUPERVISOR;
send INTERPRETE(LD-READER) to LD-SERVER;
     (Tell the Supervisor and the LD-Server where to get
      their input data)
send INFORM-LD-SERVER to SUPERVISOR;
     (Initiate compatibility-check between the PDL-
      descriptions in the PDF and in the LD)
send INIT-NET to SUPERVISOR;
     (Create net and supply the Local Analysts with
      their initial information, as e.g. acquaintances
      to others, initially admissible labels, ....)
```

The user may impose additional constraints by means of messages during the constraint process excluding certain labellings or trying to enforce a specific label for a given segment. The following two commands start the constraint propagation and impose an additional "external" constraint on the segment s.

```
send START-NET to SUPERVISOR;
     (The Supervisor will send a START-message to
      every Local Analyst)
send EXCLUDE ('+',s) to SUPERVISOR;
     (Rule out label '+' for the segment s. The Supervisor
      will send appropriate messages to the Local Analysts
      working on the junctions joined by s)
```

For a detailed description of SCENELAB see (Re84).

4. Modelling the Problem Domain

In this chapter, I will try to answer the following questions: (1) Given a line drawing, what information do we need as an input to a labelling algorithm? (2) A Waltz-like labelling does not necessarily find a consistent labelling (if there is any), what else, then, does it find?

4.1 A Formal Description of the Problem Domain

First, we observe that much of the information we need concerns the topological structure of the drawing. The segments S and the junctions J of a line drawing form an undirected graph. This "junction/segment-graph" does not capture all the significant features we need for modelling a drawing, as demonstrated in figure 8.

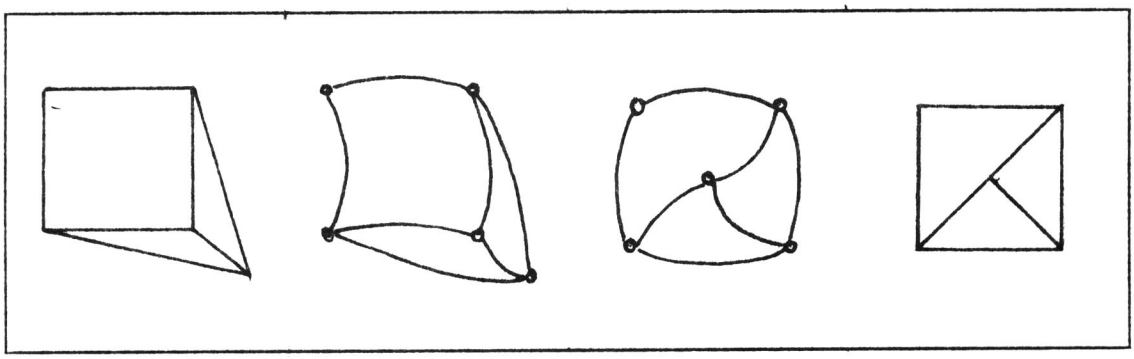


Fig. 8: Two line drawings with isomorphic junction/segment graphs

We must also know something about the geometric shape of the junctions. Therefore, one figure from a given set of prototypical junction figures is associated with every junction. For a junction x, let f(x) denote its figure.

Next, there is a collection Σ of segment labels and a list of composite junction labels. There are two points to be made: (1) Different junction figures allow for different combinations of segment labels, and (2) the order of the segments is sometimes important. (See figure 9).

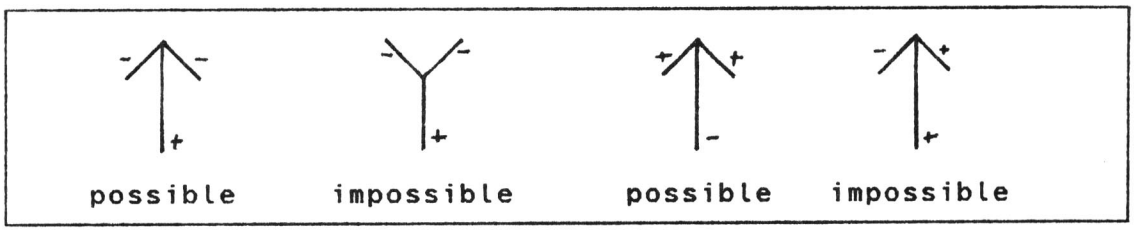


Fig. 9: Possible vs impossible labels, depending on the figure of a junction and on the order of the segments

To represent these points, we define a label dictionary as being

composed of a number of pages - one for each figure in consideration - a page comprising a list of ordered tuples of segment labels. Thus, a page is of the form

$L(y) \subseteq \Sigma^n$, n being the degree of the figure y

To perform a proper assignment of labels to junctions, the standard graph representation of the junction/segment-graph is augmented by an ordering information. Let a function g assign to every junction an ordered tuple of all of its neighbors and of its corresponding segments.

$g(x) = ((s_1,a_1),\ldots,(s_n,a_n))$
where $Inc(x) := \{s_1,\ldots,s_n\} \subseteq S$
$Adj(x) := \{a_1,\ldots,a_n\} \subseteq J$

We call the pairs (x,g(x)) for all junctions $x \in J$ the ordered graph G of the line drawing. For our trihedral junctions, we define this order as shown in figure 10.

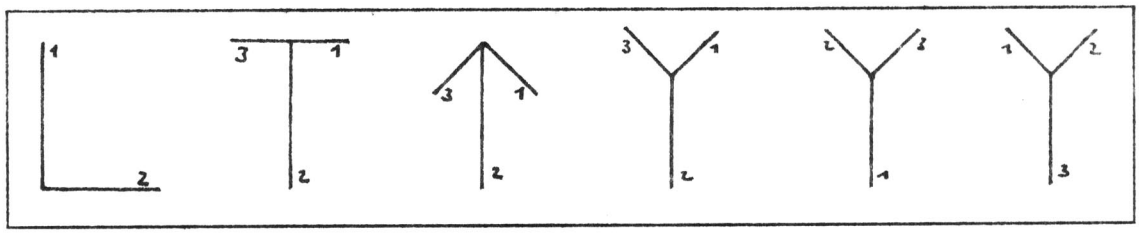


Fig 10: Unique order defined for ELL, TEE and ARROW;
Arbitrary but clockwise order for a FORK

A labelling I assigns to every junction x a possibly empty set of labels from the corresponding dictionary page.

$I: x \mapsto PS(L(f(x)))$ (Power set PS)

We say that a labelling problem is a tuple LP=(J,S,G,f,F,n,Σ,L), where the symbols have the following meanings:

J	junctions
S	segments
F	figures
$n : F \rightarrow N$	degree of a figure
$f : J \rightarrow F$	figure of a junction
$G : J \rightarrow (S \times J)^n$	ordered graph of the drawing
Σ	segment labels
$L : F \rightarrow (\Sigma^n)$	label dictionary

(The notations for G and L are simplified, since the upper index n is not fixed)

4.2 Consistent Labellings

To capture the notion of consistency, we proceed as follows: First, we postulate a symmetric 1-1 relation $\equiv \subseteq \Sigma \times \Sigma$, standing for mutual compatibility between segment labels. In our example, $\equiv$ is not reflexive, as '<' $\equiv$ '>' holds and vice versa. Clearly, for two adjacent junctions a and b, there are indices i and j such that

$$g(a) = (\ldots,(u_{i-1},x_{i-1}),(s,b),\ldots,(u_m,x_m))$$
$$g(b) = (\ldots,(v_{j-1},y_{j-1}),(s,a),\ldots,(v_n,y_n))$$

For an arbitrary labelling I, we say that a label $\underline{\sigma} \in I(a)$ is compatible with a label $\underline{\rho} \in I(b)$ if and only if the following holds:

$$\sigma_i \equiv \rho_j, \text{ where}$$
$$\underline{\sigma} = (\sigma_1,\ldots,\sigma_m)$$
$$\underline{\rho} = (\rho_1,\ldots,\rho_n)$$

(We write small greek letters for segment labels and underline the letters for junction labels.)

As mentioned above, Waltz-like algorithms search for labellings where the labels of any two adjacent junctions are compatible. We call a labelling I locally consistent at a junction a if and only if for every label $\underline{\sigma} \in I(a)$ and for every neighbor x of a there is at least one compatible label $\underline{\rho} \in I(x)$. We call I completely locally consistent - clc for short - if the condition above holds for every junction. The overall goal we are aiming at is a labelling with the following properties:

(1) $\forall x \in J : |I(x)| = 1$
(2) I is clc

A labelling which fulfills (1) and (2) is called legal. To specify the result of a Waltz-like algorithm, we need two additional definitions. We call I' a sublabelling of I if

$$\forall x \in J : I'(x) \subseteq I(x)$$

and write $I' \subseteq I$

Given a labelling problem LP and a labelling I, we call I^* the maximal clc labelling of LP with respect to I if and only if

(1) $I^* \subseteq I$
(2) I^* is clc
(3) $\forall I' \subseteq I : I' \text{ clc} \Rightarrow I' \subseteq I^*$

Rosenfeld (Roetal76) showed that for every labelling problem there is a labelling I^* which is maximally clc with respect to an initial labelling I_1 defined as

$$\forall x \in J: \quad I_1(x) = L(f(x))$$

4.3 The Waltz-Algorithm as a Filter

Now, finally, we are able to specify the result and the relevance of Waltz-like algorithms:

- Given a labelling problem LP, a Waltz-like algorithm evaluates its maximally clc labelling I^*

- Every legal sublabelling $I \subseteq I_1$ is contained in I^*. Especially, if I^* is empty, then the drawing has no legal labelling at all. Such a drawing depicts an impossible

object, provided that the label dictionary in use is sound and complete.

- Unfortunately, the converse does not hold (see fig. 11). There are labelling problems without any legal labelling where I^* is not empty.

- In general, I^* is very small in comparison to I_1.

Thus, Waltz-like algorithms are usually suitable as "filters", substantially scaling down the number of possibilities which must be considered by a subsequent exhaustive search procedure to find a legal labelling.

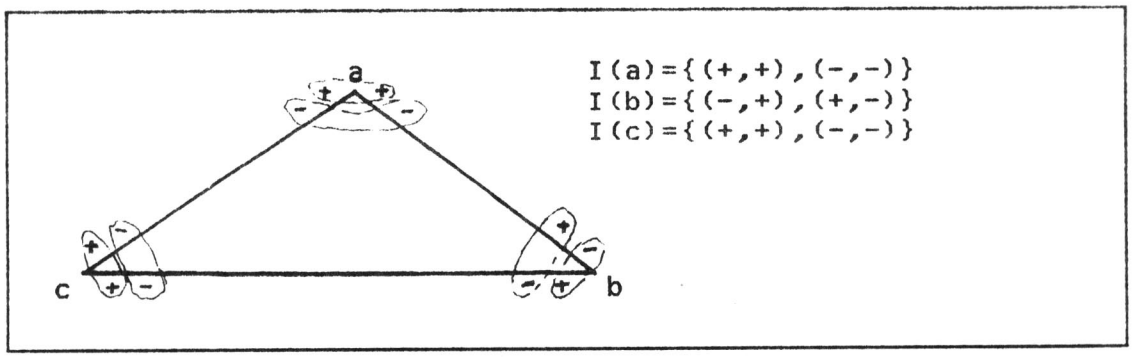


Fig. 11: A clc labelling without any legal sublabelling.

5. SCENELAB Revisited

In this chapter, the realization of SCENELAB's constraint network is discussed. Some arguments are given to claim that this net correctly determines the maximal clc labelling of a given labelling problem. The solution is outlined in a hypothetical "transaction-oriented" programming language.

5.1 On How to Realize a Society of Agents

The programming language realizes a society of cooperating agents as follows: A script is the prototypical specification of the behavior of an agent, given by (1) a universe of states and (2) a number of state transformations to transform a given state into a successive one. An agent is a process which is generated according to a script. At any time, an agent is either in a stable state taken from its universe or in a transaction state, performing a state transformation. The agents communicate with each other by sending messages to acquainted agents. A transaction is initiated by the receipt of a message. Once started, transactions are indivisible, i.e. no messages may be accepted during a transaction. Within a transaction, an agent is free to send messages to acquainted agents. The communication protocol is asynchronous, message transmission times are unknown but non-negligible. As agents performing a transaction are inable to receive a message, there must be a buffering mechanism. We consider this buffering mechanism as being part of the underlying communication system and assume that messages are delayed until the corresponding addressee is in a stable state. The only requirement we impose is that any message sent will eventually arrive at its addressee. A message is composed of the name of the transaction to be performed by the receiver and by a possibly empty list of values, enabling the sender to transmit a part of its internal state to the receiver.

SCENELAB is implemented in CSSA (Computer System for Societies of Agents) which provides the facilities to realize the concepts outlined above (and much more). For a detailed description of CSSA see (BMS82).

5.2 The Basic Operations of SCENELAB

The constraint net used by SCENELAB is composed of individual agents defined by a script called Local Analyst, one for each junction to be labelled. Let analyst(a) denote the agent working on the junction a. Analyst(a) has internal state representations of Σ, $\equiv$, Adj(a), f(a), and n(f(a)). Furthermore, there is a representation of a list of labels $I(a) \subseteq \Sigma^n$. In the following, we will not distinguish between the representations of objects and the objects themselves. We assume that the internal states are "correctly" initialized by means of INIT-transactions, such that e.g. every agent is acquainted to all of the agents working on adjacent junctions and that I(a) initially comprises just the

corresponding dictionary page L(f(a)). In SCENELAB, this is done by the Supervisor and by the LD-Sever sending INIT-messages with the correct parameters to the Local Analysts. Besides such an INIT-transaction, a Local Analyst is able to perform two transactions, called START and PROPAGATE, as described beyond. For a junction a, a segment label σ, a neighbor x of a, and a set I(a) of labels, let M(I(a),σ,x) denote the set of all labels in I(a) which assign σ to the segment joining a and x. For a given state, say S, (or a part of S), of a Local Analyst, and a transcation performed by that agent, let succ(S) be the state resulting after the transaction. Let is-sent NAME(PARAMETERS) to AGENT denote the fact that the corresponding message has been sent to the agent AGENT during this transaction. We now specify the effects of the transactions START and PROPAGATE as follows:

Transaction START executed by analyst(a):

$\forall\ x \in Adj(a)\ \forall\ \sigma \in \Sigma$:
M(I(a),σ,x) = {} if and only if
is-sent PROPAGATE(a,σ') to analyst(x),
where $\sigma' \equiv \sigma$

Furthermore, I itself remains unchanged, i.e.
succ(I(a))=I(a)

Transaction PROPAGATE(x,σ) executed by analyst(a):

succ(I(a))= I(a) $\setminus$ M(I(a),σ,x)

$\forall\ y \in Adj(a)\ \forall\ \rho \in \Sigma$:
M(I(a),ρ,y) $\neq$ {} and
M(succ(I(a)),ρ,y) = {} if and only if
is-sent PROPAGATE(a,ρ') to analyst(y), where
$\rho' \equiv \rho$

The message send PROPAGATE(a,ρ') to analyst(x), where x was the sender of the PROPAGATE-message having initiated this transaction is not necessary and is suppressed in the real implementation.

5.3 The Implementation of SCENELAB Is Correct

In order to show the correctness of a constraint net working with these transactions, we assume that

- The transactions performed by a Local Analyst satisfy the path-condition INIT;START;(PROPAGATE)*, where INIT stands for the correct initialization of the Local Analysts.

- The initial messages INIT and START are issued by the Supervisor; there are no PROPAGATE-messages besides those sent by the Local Analysts. ("External" PROPAGATE-messages occuring as a consequence of an EXCLUDE-message issued by the user may change the labelling problem.)

Based on these assumptions and on the specifications of START and PROPAGATE, we now proceed to show the correctness of the constraint net. To get things right, we postulate for every Local Analyst analyst(x) an original state, comprising the universe of all combinatorially possible labels for x, i.e. $I_0(x)=\Sigma^n$, n being the degree of x. This allows us to consider the combination of the two transactions INIT;START as a unit, removing the initially inadmissible labels from every junction and propagating the corresponding constraints. We then can visualize the activities of an agent as shown in figure 12: For every agent, there is a sequence of stable states, starting with the hypothetical original state. An agent is either in such a state or performs a transaction, evaluating a successive state. In its n-th stable state, an agent represents an actual (local) labelling $I_n(x)$. During a transaction, some labels (or none) are removed, leading to a labelling $I_{n+1}(x)$. At the same time, every newly derived constraint is immediately propagated by the sending of messages.

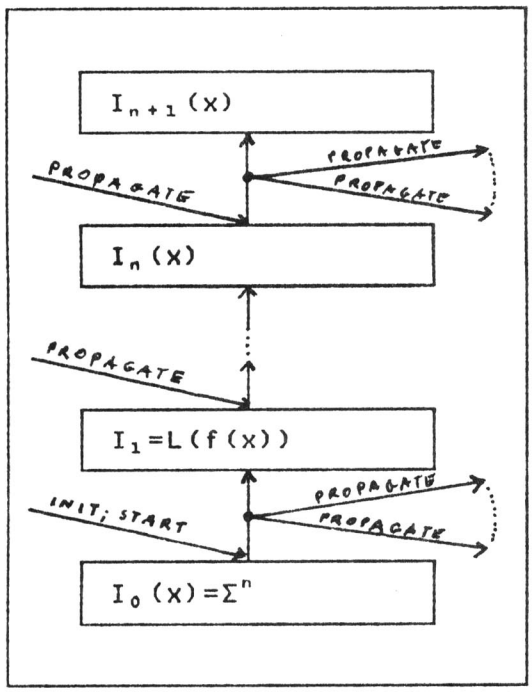


Fig. 12: The sequence of stable states of an agent x defines a sequence of local labellings.

For an entire line drawing, we get a constellation like that one in figure 13.

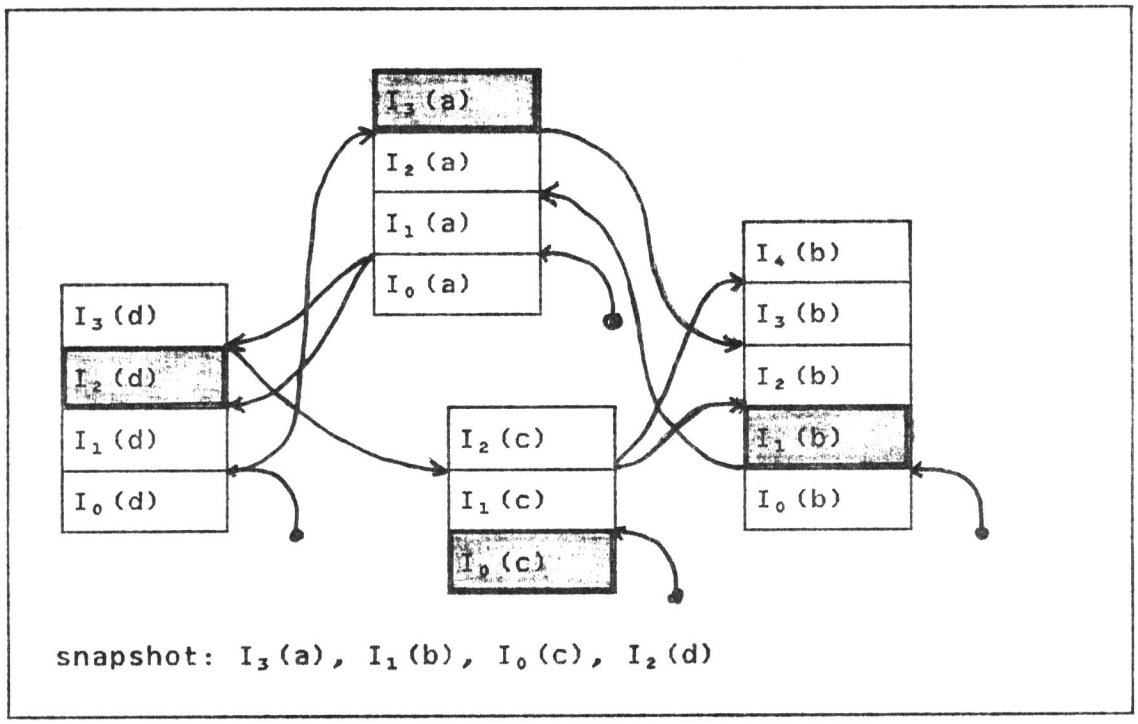


Fig. 13: The "growing" towers of subsequent stable states of the agents in a constraint net, and a "snapshot"

The crucial point now is, that only one agent is observable at a time. No simultaneous global state is observable in a society of asynchronously cooperating agents. To inform us about the progress the labelling procedure has made, we must ask every agent about his actual labelling and thus get a global but possibly inconsistent view. We call such a view a snapshot. Nevertheless, a snapshot defines a global labelling I, in spite of the fact that its components have been evaluated at different times, namely at the times when the corresponding agents have received and answered our queries. (This is done by a transaction ANSWER-QUERY sending the actual local labelling to the sender of the query by means of a message IS-ACT-LABELLING(I(a)).) We will show that the constraint net will eventually come to a halt, and that a snapshot taken after this time represents the maximal clc labelling of the labelling problem at hand.

First, we state (Lemma 1) : Given an arbitrary actual local labelling $I_n(a)$, exactly one PROPAGATE-message has been issued by analyst(a) for every empty set $M(I(a),\leq,x)$ to the corresponding neighbors of a. Obviously, the specifications of START and PROPAGATE guarantee that Lemma 1 holds.

Next, we claim that only necessary deletions are made (Lemma 2): If a PROPAGATE-transaction of an analyst(a) removes a label belonging to $I^*(a)$ from I(a), then the sender, say analyst(b), of this message has removed a label belonging to $I^*(b)$ from its own label set during the transaction in which the PROPAGATE-message has been issued. To prove this lemma, we argue

as follows: If a label, say $\underline{\leq}$, may be removed from $I^*(a)$, then I^* is not empty. Furthermore, $I^* \subseteq I_0$ holds. Clearly, at least one label $\underline{\varphi} \in I_0(b)$ must be compatible with $\underline{\leq}$. The specifications of PROPAGATE and START assure that the PROPAGATE-message leading to the removal of $\underline{\leq}$ is issued exactly once, namely in a transaction of analyst(b) removing $\underline{\varphi}$.

As the set J of junctions as well as the dictionary pages L(f(x)) are finite, we conclude by Lemma 1 that only a finite number of PROPAGATE-messages may be issued within a constraint net. We call a snapshot a <u>final</u> one, if it is taken after all of these messages have been processed. Such a final snapshot will eventually be achieved, since every message sent must arrive after a finite delay.

We now state that the labelling defined by a final snapshot is clc (Lemma 3): Let the labelling defined by a snapshot be not clc. Then this snapshot is no final one. We outline a proof of Lemma 3 as follows: If a labelling is not clc, then there are at least two adjacent junctions a and b that for two segment labels $\leq$ and φ with $\leq \equiv \varphi$ the following holds
$M(I_i(a),\leq,b) = \{\}$ and $M(I_j(b),\varphi,a) \neq \{\}$
where the snapshot comprises the i-th stable state of analyst(a) and the j-th stable state of analyst(b), respectively. The specifications of the transactions assure, that analyst(a) must have sent the message PROPAGATE(a,φ) to analyst(b). They also assure that this message has not yet been received by analyst(b). Thus, the snapshot is not final.

We now show, that a final snapshot must comprise I^* (Lemma 4): Every snapshot comprises I^*. This fact follows by iteration of Lemma 2. For every PROPAGATE-transaction P there is a finite sequence $T_1,\ldots,T_n$ of transactions such that the message initiating T_{i+1} has been issued during T_i and where $P=T_n$. Furthermore, $T_2,\ldots,T_n$ must be PROPAGATE-transactions, whereas T_1 is a combined INIT;START-transaction, transforming $I_0(a)$ to $I_1(a)$ for a junction a. Thus, if P removes a label from I^*, then, by view of Lemma 2, an INIT;START-transaction must have removed another label from I^*, and this is obviously impossible, as $I^* \subseteq I_1$.

We are now ready to claim that the constraint net eventually reaches a final state representing the maximal clc labelling of the given labelling problem. To proof this, we argue that a final snapshot is eventually reached(Lemma 1 and finiteness of J, L). This final snapshot represents a labelling I which is completely locally consistent(Lemma 3). Furthermore, I comprises I^*(Lemma 4). As I^* is maximal, I <u>is</u> I^*.

6. Discussion

6.1 Final vs Satisfactory Snapshots

As we have shown in the previous chapter, the constraint net of SCENELAB eventually comes to a stable state, having evaluated the desired result. We did not address the problem of how to detect that the propagation of constraints is terminated. In the absence of a consistent global view, this "distributed termination detection" is a non-trivial problem. We call a snapshot a satisfactory one, if the labelling represented is the maximal clc labelling I^*. Lemmata 3 and 4 state that a final snapshot is necessarily a satisfactory snapshot. The converse does not hold, as PROPAGATE-messages do not always lead to deletions of labels. In the present implementation, we replaced the problem of detecting a final snapshot by that one of detecting a satisfactory snapshot and used a brute-force solution : the constraint-net is simply "frozen" to enforce a simultaneous global view and then a snapshot taken in a frozen state is tested for consistency. For a detailed discussion of the definition of distributed termination and some approaches to a solution of this problem see (BMR84).

6.2 The Perfectness-Assumption and Reality

When discussing the structure of line drawings, we implicitely made the assumption that a drawing be "perfect" in the sense that its line segments represent exactly all the edges of the corresponding three-dimensional objects. For real world applications, however, this is a rather restrictive assumption which is generally not guaranteed to hold. The construction of a line drawing is normally based on a matrix of grey-values. A line is recognized as the border line between two areas with significantly different grey-values. Thus, lack of contrast may prohibit the recognition of a line corresponding to an edge, whereas e.g. shadows may give rise to a substantial contrast - and thus to the recognition of a line - where there is no real edge. Thus, an interpretation system for line drawings should be able to handle both incomplete and potentially erroneous picture data. If such a system is integrated in an application system as e.g. a roboter, the need for fast decision making may arise, preventing the system from getting more and better data in time. This means that conclusions must be drawn on the base of partial evidence, and that in the light of further evidence, those conclusions may turn out to be wrong and must be withdrawn, i.e. the system reasons non-monotonically. The need for non-monotonically reasoning systems is not restricted to computer vision, but seems to be one of the most challenging bottlenecks for many AI-applications. For a introduction to this area see e.g. (D.Bo80).

6.3 Sequential vs Parallel Labelling Algorithms

It seems that an asynchronous approach to Waltz-like labelling procedures is "in some sense" a generalization of sequential and synchronized parallel versions. This "sense" deserves further explanation, as these approaches are not directly comparable. A basic operation which is common to all these solutions, is the removal of all the labels at a junction assigning a segment label to a segment which is no longer matched by the labels of a neighboring junction. (In chapter 5, we called this set $M(I(a),\lessdot,b)$.) The approaches differ mainly in how restrictively the sequencing of these operations is controlled. Translating the sequential and synchronous versions into our model of nodes and links, these sequencing modes result in a behavior as sketched in figure 14. To do this, we clearly must abstract from the realization of such a system. A rectangle in a "tower" of subsequent local labellings at a node is just a set of labels, and links represent causal relationships of the adjustments made between two label sets by means of the basic operation mentioned above. In the original sequential version, the operations are executed one after another, in a predefined fashion, the FOR-loop of the Waltz-procedure (see chapter 2) ranging over an arbitrary enumeration of the set of junctions, and the recursively repeated deletions being made, say, in a breadth-first order. We call a labelling I <u>partially locally consistent</u> with respect to a set $K \subseteq J$ of junctions, whenever its <u>completion</u> I' is clc, where

$\forall x \in K: I'(x)=I(x)$

$\forall y \in J\backslash K: I'(y)=\Sigma^n,\ n=n(f(y))$

After every iteration of the FOR-loop, the Waltz-procedure results in a well-defined snapshot, representing a partially locally consistent labelling of the junctions visited so far.

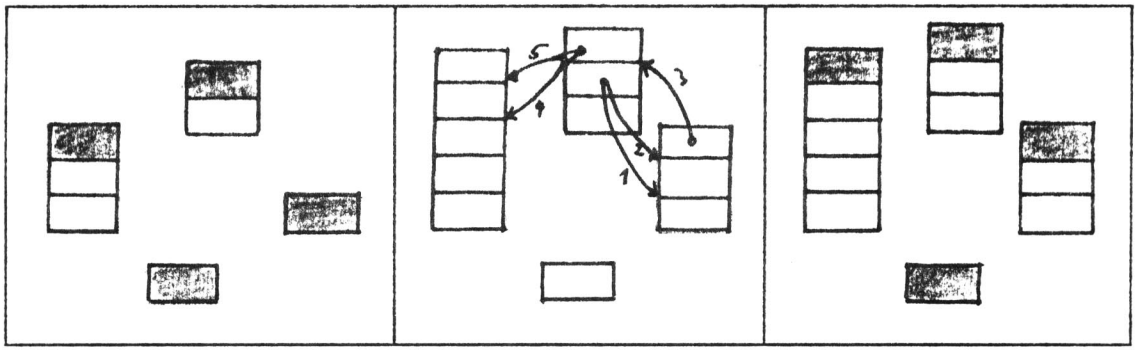


Fig. 14a : Sequential "growth"

In a synchronized parallel version, the propagation of newly derived constraints is retained until the labelling of each junction is adjusted with respect to all of its neighbors. Thus, the "towers" grow from one global level to the next. The local labellings at one level are locally consistent with respect to the most recent global level below. Once two subsequent global levels are identical, the labelling procedure may halt.

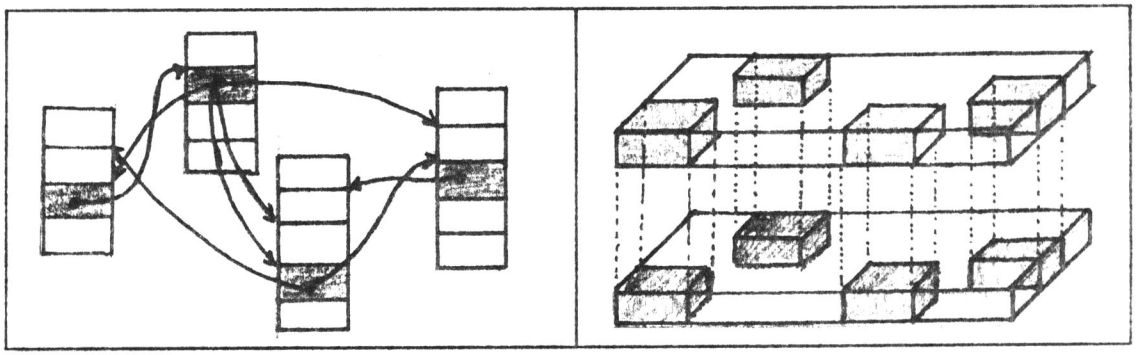


Fig 14b : Synchronized parallel "growth"

In the asynchronous case, there are no restrictions at all. As every labelling problem has a unique solution (i.e. its maximal clc labelling I^*), we draw the following conclusion from Lemma 1 of the previous section: For every labelling problem there is a fixed number n of PROPAGATE-messages (where n is the number of sets $M(I^*(a),\leq,x)$ being empty in the final labelling I^*) $P_1,P_2,\ldots,P_n$, being issued in a non-deterministic order by an asynchronous constraint net when evaluating the solution of the labelling problem. Thus, for every snapshot taken from the constraint net, there are numbers k and m, where $0<=k<=m<=n$, such that m PROPAGATE-messages have been sent while k of them have been received. In general, there will be no well-defined intermediate stages as those reached by sequential or synchronized parallel constraint nets.

Labelling problems seem to favor parallel solutions for at least two reasons:

- A labelling problem divides into small, well-defined subproblems. Moreover, these subproblems interact in a specific way supporting highly cooperative solutions. Information derived in one region of a drawing may be immediately used to scale down the number of possible labellings to be considered in other regions of the drawing. Unlike brute-force parallel algorithms for mutually independent subproblems as e.g. occuring in sorting algorithms, constraint nets allow for an early pruning of the search space.

- By their very nature, the representation of labelling problems in terms of parallel processes is straightforward. As complete local consistency is defined as the conjunction of pointwise local consistencies, it is more convenient to think of locally restricted reasoning units trying to keep the labellings locally consistent than to think of a globally reasoning unit iteratively inspecting one point after another.

One major problem of the approach taken here is the usage of such powerful computational units as CSSA-Agents to realize simple, finite constraints (see next section). Depending upon the implementation of such agents, the communication overhead may

become large in comparison to the amount of local reasoning.

6.4 SCENELAB Works with Very Simple Constraints

In this last section, I would like to discuss the generality of this approach to the realization of constraint nets. We claim that the basic idea of the present work applies to general constraint problems, as e.g. those discussed in (G.St80). An abstract constraint relation is defined within a script, and instances of these relations are realized by agents which are created according to those scripts. The interrelationships between single constraints are represented by means of acquaintances between agents. Basically, all the constraints expressible in the particular language in use may be realized this way.

SCENELAB, however, uses specific techniques to represent and propagate constraints restricting its applications to constraint problems which exhibit "the same structure" as the constraints used in the Waltz-Algorithm. This "same structure" is characterized by two points. First, the constraint relations are finite, they are even small enough to use representations in form of tables. A dictionary page merely lists all the tuples of a relation constraining the values of segment labels. Second, the relations constraining the combinations of the composite junction labels are defined in terms of a one-to-one correspondence between segment labels, therefore enjoying a certain property which I would like to call "pseudo-transitivity". (Note that, in our application, these constraint relations depend upon the topologic structure of the line drawing and are implicitely represented in the constraint net.)
Pseudo-transitivity is defined as follows:
Let A,B be two disjoint sets. A relation $R \subseteq A \times B$
is pseudo-transitive, if the following holds:

$$\forall a_1, a_2 \in A \quad \forall b_1, b_2 \in B :$$
$$(a_1, b_1), (a_2, b_2), (a_2, b_1) \in R \Rightarrow (a_1, b_2) \in R$$

The graph of the relation R is composed of the nodes $A \cup B$ and the links (a,b) for every $(a,b) \in R$. If R is pseudo-transitive, then this graph is partitioned into completely bipartite connected subgraphs. The structure of such a graph may be very simply represented by replacing all the links in such a subgraph by a single link between the two disjoint node sets of this bipartite subgraph. That is, R is simplified to a relation SR on the powersets of A and B. Fig. 15 shows such a graph and its simplification.

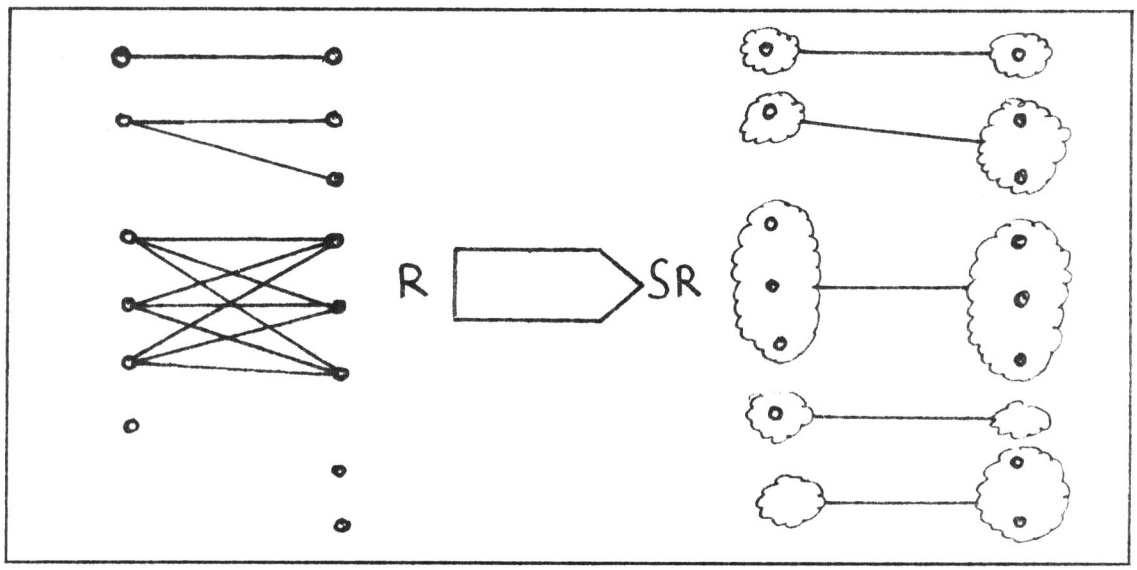


Fig. 15 : The completely bipartite connected subgraphs of the relation graph of a pseudo-transitive relation allows for a significantly simplier representation of the graph

This specific structure enables SCENELAB to use its simple propagation technique. As long as there is at least one member in such an equivalence class of nodes(here: labels), no new constraint must be forwarded. Once such a class becomes empty, all the members of the corresponding matching class may be deleted. This is in fact what SCENELAB does. Figure 16 shows the constraint relation between the labels of an ELL and an ARROW in a specific context.

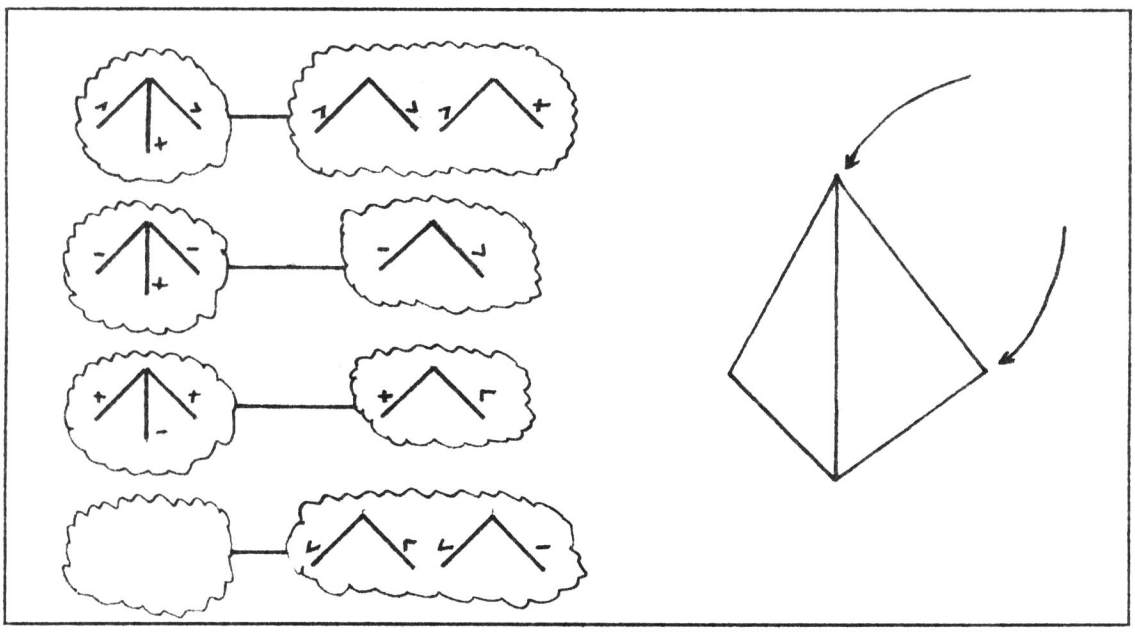


Fig. 16: Simplification of the graph of the relation constraining the labels of an ELL and an ARROW being connected in a specific way.

7. Current Status and Future Work

This paper is a short version of my Diplomarbeit which will be available, I hope, at the end of this year. An implementation of SCENELAB has been running since July 1983, with some slight modifications.

Actually, CSSA is running on a multiprocessor simulation system (BMS82) implemented in SIMULA on a SIEMENS BS2000 single-CPU machine. Christian Beilken and Friedemann Mattern are currently working on a really distributed realization on a network of 32-bit processors, together with their colleagues of the SFB 'VLSI und Parallelitaet'.

Several students of Prof. Peter Raulefs are investigating and - as I understand - developing a general purpose constraint system. This paper puts a (preliminary?) end to my own work in this area, in spite of an ongoing interest in distributed constraint systems.

Acknowledgements

I thank Hans Voss, who asked me in spring 1983 whether I would like to implement the Waltz-algorithm in CSSA, thats how this all began. Since then, he has been contributing to my work in many discussions, providing valuable insights. Thanks also to Christian Beilken and Friedemann Mattern, who helped to resolve some problems with the SIMULA system. I am also grateful to Prof. Peter Raulefs for commenting on an earlier draft of this paper. No need to say that any flaws in the work reported herein are to my fault.

8. Bibliography

(This bibliography contains only papers the present work refers to. A more comprehensive bibliography is given in the extented version (Re84)).

(AlKa84) Allen, James F.; Kautz, Henry A.: A Model of Naive Temporal Reasoning. Computer Science Research Review 1983-84, University of Rochester, Rochester 1984.

(BMR84) Beilken, Christian; Mattern, Friedemann; Reinfrank, Michael: Verteilte Terminierung - ein wesentlicher Aspekt der Kontrolle in verteilten Systemen. (In German), SFB VLSI und Parallelitaet, Universitaet Kaiserslautern, Kaiserslautern 1984 (forthcoming report).

(BMS82) Beilken, Christian; Mattern, Friedemann; Spenke, Michael: Entwurf und Implementierung von CSSA - Beschreibung der Sprache, des Compilers und eines Mehrrechnersimulationssystems.(In German), Memo-Seki 82-03, Universitaet Kaiserslautern, Kaiserslautern 1982.

(A.Bo79) Borning, Alan: THINGLAB - A Constraint Oriented Simulation Laboratory. PhD dissertation, Stanford 1979.

(D.Bo80) Bobrow, Daniel G.(edt.): Special Issue on Non-Monotonic Logic. ARTIFICIAL INTELLIGENCE Vol. 13 No. 1,2, 1980.

(Cl71) Clowes, M.B.: On Seeing Things. ARTIFICIAL INTELLIGENCE Vol. 2, 1971.

(Foetal83) Fox, Mark S.; Allen, Bradley P.; Smith, Stephen F.; Strohm, Gay A.: ISIS - A Constraint Directed Reasoning Approach to Job Shop Scheduling. CMU-RI-TR-83-3, Carnegie-Mellon-University, Pittsburgh 1983.

(Fr82) Freuder, Eugene C.: On the Knowledge Required to Label a Picture Graph. ARTIFICIAL INTELLIGENCE Vol. 15, 1982.

(Go83) Gosling, James: Algebraic Constraints. CMU-CS-TR-83-132, Carnegie-Mellon University, Pittsburgh 1983.

(HaSh79) Haralick, R.M.; Shapiro L.G.: The Consistent Labelling Problem.
Part I : IEEE Transactions on PAMI Vol. 1 No. 2, 1979
Part II : IEEE Transactions on PAMI Vol. 2 No. 3, 1980

(He82) Hein, Uwe: Constraints and Event Sequences. LITH-MAT-R-82-02, Linkoeping University, Linkoeping 1982.

(Hu71) Huffman, D.A.: Impossible Objects as Nonsense Sentences. MACHINE INTELLIGENCE 6, Edinburgh 1971.

(McA80) McAllester, David A.: An Outlook on Truth Maintenance. AI-LAB Memo 551, MIT, Cambridge 1980.

(Nu83) Nudel, B.: Consistent Labelling Problems and their Algorithms: Expected Complexities and Theory-Based Heuristics. ARTIFICIAL INTELLIGENCE Vol. 21, 1983.

(Re83) Reinfrank, Michael: The Implementation of SCENELAB. (An annotated CSSA-Program), Projektarbeit, Universitaet Kaiserslautern, Kaiserslautern 1983.

(Re84) Reinfrank, Michael: Scene Labelling by a Society of Agents - A Distributed Constraint Propagation Algorithm. Diplomarbeit, Universitaet Kaiserslautern, Kaiserslautern 1984 (in preparation).

(Roetal76) Rosenfeld, A.; Hummel, R.A.; Zucker, S.W.: Scene Labelling by Relaxation Operations. IEEE Transactions on SMC Vol.6 No. 6, 1976.

(G.St80) Steele, Guy L.: The Definition and Implementation of a Computer Language Based on Constraints. MIT AI-LAB TR 595, 1980.

(L.St82) Steels, Luc: Constraints as Consultants. Proc. ECAI 82, pp. 75-78, 1982.

(M.St81) Stefik, Mark: Planning with Constraints (MOLGEN: Part 1). ARTIFICIAL INTELLIGENCE Vol. 16, 1981.

(StSu77) Stallman, R.M.; Sussman G.J.: Forward Reasoning and Dependency-Directed Backtracking in a System for Computer Aided Analysis. ARTIFICIAL INTELLIGENCE Vol. 9 ,1977.

(Su82) Sugihara, K.: Mathamatical Structures of Line Drawings of Polyhedrons - Towards Man-Machine Communication by Means of Line Drawings. IEEE Transactions on PAMI Vol. 4 No. 5, 1982.

(Wa72) Waltz, David: Generating Semantic Descriptions from Drawings of Scenes with Shadows. MIT MAC-TR 271, 1972 (reprinted in Wi75).

(Wi75) Winston, Patrick H.(edt.): The Psychology of Computer Vision. Mc Graw Hill, New York 1975

(Wi77) Winston, Patrick H.: Artificial Intelligence. Addison-Wesley, Reading, 1977.