Propagating temporal constraints for scheduling

Jean-Francois Rit
LIFIA, BP 68, 38402 Saint-Martin-d'Heres, FRANCE

ITMI, Chemin des pres, ZIRST, 38240 Meylan, FRANCE

Abstract

We give in this article a general frame for propagating temporal constraints over events, these events being exclusively considered as sets of possible occurrences (SOPOs). These SOPOs are the numerical expression of an uncertainty about the exact occurrence of an event, while this exact occurrence is constrained by the possible occurrence of other events.This key-problem of scheduling is an instance of the consistent labeling problem and is known to be NP-complete. We introduce a graphical representation of SOPOs which is a useful tool to understand and help solving the problem.We give a constraint propagation algorithm which is a Waltz-type filtering algorithm. Theoretically, it does not discard all the inconsistent occurrences; however, under a number of relatively weak assumptions, the problem can be transformed into a solvable one.

1 Introduction

When you want to define how a task must be done, you have to solve two different sub-problems:

1. First you have to *structure* the task.To perform this, you define the various elementary subtasks which allow the performance of the main task, and what resources will be needed. Furthermore, the logical relations which must be kept between the subtasks are stated. Some of them are *symbolic* temporal relations such as precedence relations.

2. Then you have to detail the *actual* execution of the task, that is when the subtasks will take place and the resources will be used. The main output of this activity is a schedule which keeps some symbolic information from the first step and gives some numerical information. In particular, the duration of the tasks is a key data in schedules.

The aim of this article is to provide a theoretical model for the definition and propagation of numerical temporal constraints for the *second* subproblem. Its purpose is to be a general framework for building a "temporal module" in a scheduling system. What the scheduling system should otherwise be is not in the scope of this paper. A general architecture where the place of a temporal module is clearly defined can be found in [3] (Delesalle Descotte, 86) for example.

The basic objects of our model are events linked with symbolic temporal relations. Events are characterized by *sets of possible occurrences*, where an occurrence is defined as the interval during which an event happens. The function of a temporal module for scheduling is to modify these sets of possible occurrences so that they become compatible with the symbolic relations which act as constraints upon them.

Suppose, for example, you plan to meet someone during your lunch-break. The two events `lunch-break` and `meeting` are linked by the relation `during`. Suppose your lunch-break lasts from half an hour to an hour, between 11:30 and 13:30, and your meeting lasts at least half an hour, ending before 12:30. This defines the initial sets of occurrences of `lunch-break` and `meeting`. In order to be consistent with the constraint `during`, the meeting must start after 11:30 and the lunch-break must start before 12:00. The sets of occurrences must be modified in order to build a coherent schedule. This operation consists in solving what we call a *constrained occurrences problem*. It must be done whenever some numerical scheduling is involved. Things may seem simple on this example, but the situation can become considerably more complex when more relations are considered.

A formal definition of the problem is given in Section 2. Section 3 introduces a graphical point of view which helps understanding how a constraint propagation algorithm can work.In Section 4 we discuss the validity of this model and show that it is not possible to discard every inconsistent occurrence unless some restrictions on the sets of occurrences are made and disjunctive relations are "eliminated" in some way. We also evaluate the computational complexity of the propagation. In Section 5 we relate our work to others', especially Allen's, who introduced a complete taxonomy of symbolic relations among intervals, and Vere's, who introduced the concept of *window* which is a particular kind of set of occurrences. Finally we conclude with suggestions on how the model could be strengthened.

2 A Formalization of the Constrained Occurrences Problem

2.1 What Events Are

Events are the basic objects of our time module. An event can be described as the association of a logic predicate and an *occurrence*. The semantics of the predicate is outside the scope of this temporal module. It can denote a fact, the existence of a process, the execution of a task, whatever the abstraction level. On the contrary, the occurrence is the significant data at this relatively low-level stage of reasoning. An occurrence is a one-dimensional interval, its intuitive meaning is "the moments when the predicate holds". The case of events happening

over multiple intervals, like having-lunch (which happens every day), is not considered here. At a higher level, such events can be easily decomposed into elementary subevents happening over single intervals.

Handling events with a uniquely defined occurrence is not very interesting. In most cases, the occurrence of an event is *numerically constrained* : it belongs to a domain where the duration, beginning and end of the occurrence are constrained. This domain is called the *set of possible occurrences* (SOPO). It should be now obvious that the meaning of the predicate is not relevant to the temporal module point of view.

2.2 Linking Events with Temporal Relations

The second type of input data are relational data. They are temporal links between events and are mainly derived from higher level descriptions of the world. For example, suppose you want to buy something, then your action of paying happens during buying because paying is hierarchically a *subtask* of buying. Moreover paying happens before getting the object because paying is a *condition* of getting the object.

For the temporal module, all the relations are translated into a low-level set of *primitives* which describe the relationship between two intervals. These primitives are before, during, overlap... A complete enumeration was given by Allen [1].

However, these primitives are used to link two intervals, and we want to link SOPOs. This requires some formalization. A temporal relation is a boolean function defined over the SOPOs of two events (we will also say that it is defined between these events). The arguments of such a relation are therefore two intervals :

$$\begin{array}{rccc} R_{12}: & O_1 \times O_2 & \rightarrow & \{0,1\} \\ & (o_1, o_2) & \mapsto & R_{12}(o_1, o_2) \end{array}$$

Where R_{12} is a temporal relation, O_1 and O_2 are SOPOs, and o_1, o_2 are occurrences.This relation express a constraint over the parameters of $o_1 and o_2$ (beginning, end and duration). For example, if R_{12} is the relation before, $b(o)$ the beginning of o and $e(o)$ the end of o:

$$R(o_1, o_2) = 1 \Leftrightarrow b(o_2) > e(o_1)$$

Let $O_1, \ldots, O_n$ be n SOPOs and $o_1, \ldots, o_n$ be n occurrences such that

$$\forall i \in [1, n], o_i \in O_i$$

The set $\{o_i, i \in [1, n]\}$ of occurrences is a *solution* of a set of relations $\{R_{jk}, j, k \in [1, n]\}$ iff :

$$\forall j, k \in [1, n],\ R_{jk}(o_j, o_k) = 1$$

Semantically, a solution of a set of temporal relations is a description of a world where each event has one occurrence and where all the temporal relations are satisfied.

2.3 The Constrained Occurrences Problem

Let $e_1, \ldots, e_n$ be events, $O_1, \ldots, O_n$ their SOPOs and $\mathcal{R}$ a set of relations between the events. The *constrained occurrences problem* is :
Discarding from $O_1, \ldots, O_n$ the occurrences which do not belong to any solution.

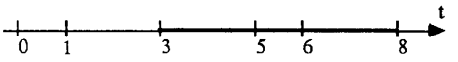

Figure 1: one-dimensional representation of occurrences

Of course, finding all the solutions of $\mathcal{R}$ would be more satisfying, but it is impossible to explicitly represent them since the SOPOs contain an infinite number of elements. It is then necessary to keep implicitly the solutions in the SOPO-based representation. In order to "extract" a solution, you must choose one occurrence o in a SOPO. If the constrained occurrences problem has been solved, o belongs to at least one solution. Choosing o is equivalent to defining a new SOPO, thus a *new* constrained occurrences problem. Solving this new problem will give all the solutions containing o.

Naturally, you can more generally choose a *subset* of a SOPO. As long as $\mathcal{R}$ is not changed, you know that the new constrained occurrences problem has some solutions.

3 A Graphical Representation of Constrained SOPOs

3.1 A 2-D representation of occurrences

An occurrence being a numerical interval, a one-dimensional line is an obvious graphical aid for representing occurrences and relations between them (fig 1). However, a 1-D representation is ambiguous because SOPOs have to be represented with intervals whereas they are *sets* of intervals. For example, the interval on fig 1 could represent the set of occurrences beginning after 2 and ending before 6, or the set of such occurrences with a duration equal to 3, or the set of occurrences whose beginning cannot belong to $[4, 5]$. This ambiguity comes from the 2-dimensional nature of an occurrence : it is completely determined with two parameters : beginning and end. This is why we use, as a graphical aid, the *plan of occurrences* with beginning and end axis (fig 2). As occurrences begin before they end, all possible ones belong to the dotted region of fig 2. The main diagonal has a particular meaning: it is the locus of occurrences whose beginning equals the end, thus the locus of dates. This links the 2-D representation with the 1-D one in the following manner : the beginning and the end of any occurrence o are dates and can then be "projected" onto the diagonal (fig 2). The segment defined by these two dates is the set of all the "instants" forming o, in other words, this segment is the 1-D representation of o. It is thus possible to switch easily from one point of view to the other : 2-D for sets of occurrences, 1-D for occurrences.

Fig 3 is a unambiguous representation of the SOPOs that fig 1 cannot represent.

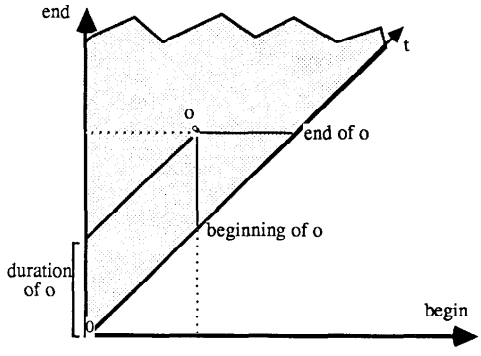

Figure 2: two-dimensional representation of occurrences

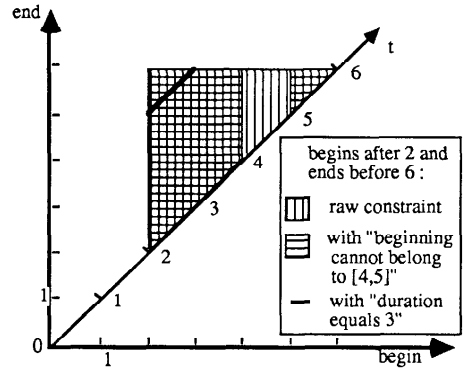

Figure 3: Fig 1 revisited with a 2-D representation

3.2 2-D Representation of Relations and SOPOs

Let R be a temporal relation, and o an occurrence, the set of all the occurrences verifying the relation R with o, $\{x/R(x,o) = 1\}$, is a generally easily representable region in the plan. We will call it the ***region allowed by o and R.***

Fig 4 is a "complete" mapping of the plan with the set of temporal relations given by Allen [1]. Any occurrence on it can be temporally related to o. Of course, there are obviously other ways of mapping, with more or less primitives, depending on how complex the temporal module should be. We will keep this one in the following.

Disjunctive relations can be represented as the union of the regions allowed by the parts of the disjunction (fig 5). In the same way, conjunctions are represented through the graphical intersection of regions (fig 6).

A SOPO is also a region of the plan. The simplest SOPO (except the trivial occurrence set) is the ***window***, using Vere's (one-dimensionnal) [7] terminology. Occurrences in a window must begin after an earliest start time and end before a latest finish time. If their duration is not constrained, the window has a triangular shape, if their duration is given, the window

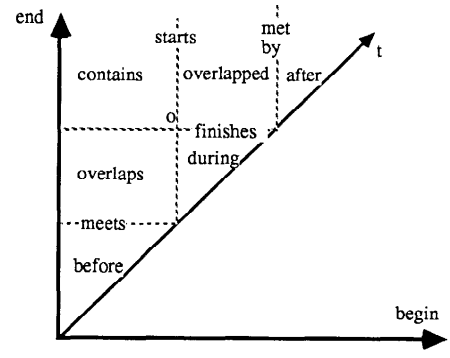

Figure 4: A mapping of the plan using Allen's primitives

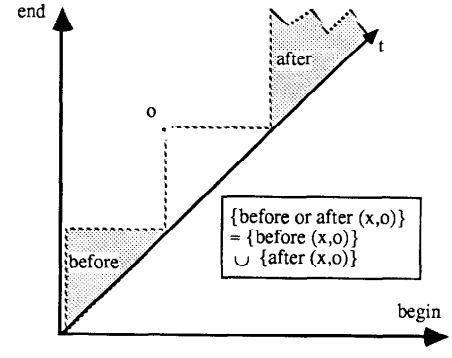

Figure 5: An example of a disjunctive constraint

is a segment (fig 7). The SOPO of a given event can theoretically have any shape. However, in the following, we will speak only of one sort of SOPO, unless otherwise stated, namely the ***generalized window*** where the beginning, end and duration of an occurrence are independantly bounded. This simple representation covers a fair number of cases. The shape of such a

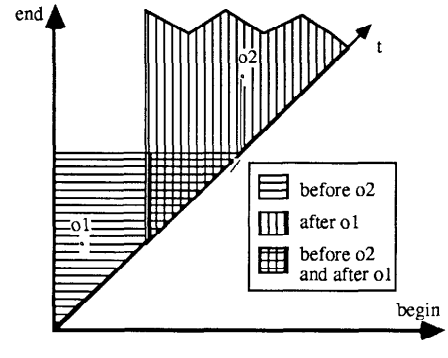

Figure 6: An example of a conjunctive relation

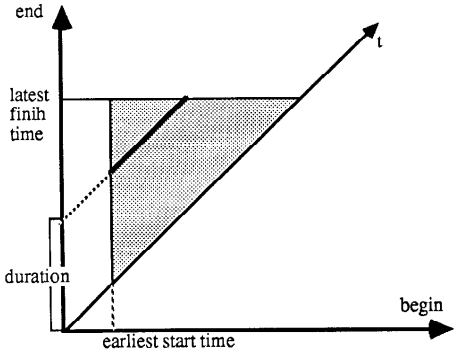

Figure 7: windows

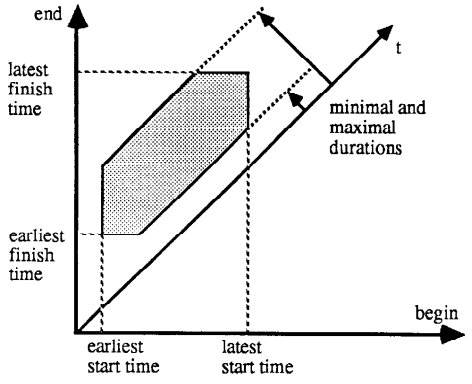

Figure 8: a generalized window

SOPO is on fig 8.

3.3 Propagation of Temporal Constraints over SOPOs

Returning to the problem of temporal relations, a SOPO can be considered as a disjunctive clause where each part of the clause is a single occurrence. In this respect, given a relation R and a SOPO O_1, we can define a *region allowed by O_1 and R* as the union of the regions allowed by each occurrence in O_1 and R (formally, this allowed region is $\{x \,/\, \exists o \in O_1 \,/\, R(x,o) = 1\}$). If R is defined between O_1 and a SOPO O_2, this means that any occurrence of O_2 that does not belong to this region does not belong to any solution, whereas the other occurrences may do so. On a dual point of view, the intersection of all such regions($\{x \,/ \forall o \in O_1,\ R(x,o) = 1\}$) defines a "certain" region : all the elements of O_2 in this region belong to a solution, whatever the choice of an element in O_1, whereas the elements which do not belong to this certain region might not belong to any solution (fig 9).

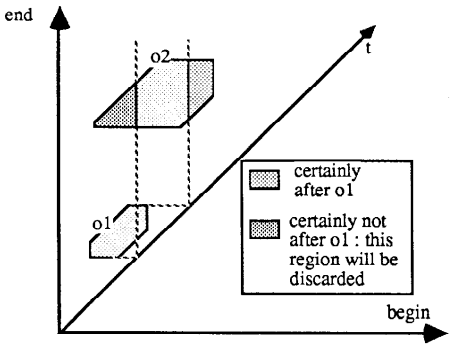

Figure 9: Propagation of a constraint on a SOPO

This is the basis of a temporal constraint propagation. Each time a SOPO O_i is constricted, the regions allowed by O_i and any relation R_{ij} involving O_i are constricted. If the SOPOs O_j such that $O_i\ R_{ij}\ O_j$ are therefore constricted, the propagation must be furthered. The general structure of the algorithm is thus :

```
begin
    while S ≠ ∅
        for i = 1,n
            if O_i ∈ S
                for j = 1,n
                    if O_j ∩ A(O_i, R_ij) ≠ ∅
                        O_j ← O_j ∩ A(O_i, R_ij)
                        S ← S ∪ {O_j}
                    endif
                endfor
                S ← S − {O_i}
            endif
        endfor
    endwhile
end
```

where S is the set of modified SOPOs to be checked and $\mathcal{A}(O_i, R_{ij})$ is the region allowed by O_i and R_{ij}.

An interesting feature of this method is that disjunctive constraints can be propagated. Disjunctive constraints can create "holes" in a SOPO. In fig 10, the SOPO O_1 creates a hole in the window O_2 which therefore loses its connexity. However, constraints involving O_2 can still be propagated, furthering a kind of common information about the two parts of O_2.

4 Evaluating the Algorithm

4.1 Consistency

The constrained occurrences problem belongs to a class of problems which received much attention in Artificial Intelligence and Operational Research, namely the *consistent labeling problem*. In our case, the domain of variables is not finite because it is continuous.

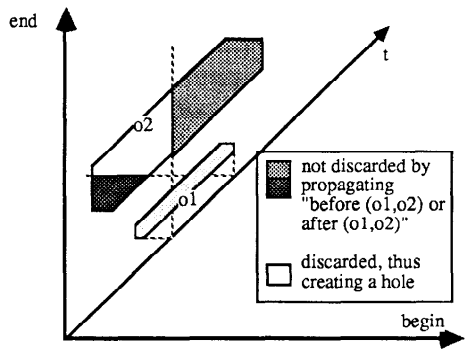

Figure 10: Making a hole in a SOPO

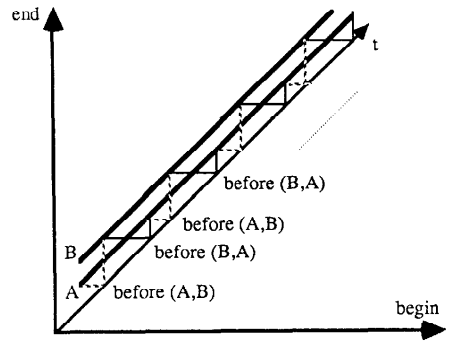

Figure 11: A bad case for constraint propagation

The algorithm we gave for temporal constraint propagation in section 3.3 is the same than Waltz's algorithm [8] used for scene analysis problems. According to Mackworth's terminology [4] it achieves only arc-consistency because it only checks consistency between pairs of SOPOs. As defined in section 2.2, belonging to a solution is a *global* property of an occurrence. A local check between pairs of SOPOs is necessary but not sufficient. This is why such an algorithm is only a filtering algorithm, it "lets go by" inconsistent occurrences.

In fact the consistent labeling problem is NP-complete [4]. However, if the SOPOs are generalized windows, and if the disjunctive relations are eliminated by enumerating the different terms, the constrained occurrences problem can be solved with an arc-consistency algorithm (see [5] for hypotheses a bit weaker than generalized windows). Such an elimination should come after the arc-consistency elimination, otherwise the interest in our formalism of propagating disjunctions would be greatly spoiled.

4.2 Complexity

We will consider here only the complexity of the arc-consistency algorithm since it is the heart of the constraint propagation algorithm. We saw in section 3.3 that each time a SOPO is modified, a consistency check must be made. The complexity of the algorithm is then :
number of constrictions × cost of a consistency check
≃ (number of constricted SOPOs × number of constrictions per SOPO) × (number of relations per SOPO × cost of a consistency check between two SOPOs)

$$\simeq n \times e \times r \times c$$

The evaluation of this expression is in fact quite difficult :

n number of events or SOPOs, is precisely defined and cannot be influenced.

e measures the efficiency of the constrictions, $e = 1$ means that each SOPO is modified *once* with a "one-shot", definitive constriction. Unfortunately, things can go very bad in the case of contradictory relations. For example, on fig 11, $e = 4$ for A and B : you must apply four times the constraints `before (A,B)` and `before (B,A)` in order to remove this inconsistent conjunction.

r is the average number of relations involving a given event. If r is big, the algorithm will be more costly, having more constraints to propagate. But the "information" of a constriction will be propagated more efficiently, thus lowering *e*.

c depends on the "shape" of the SOPOs, and not on their number. If the shape is very complicated (for example with a lot of holes) the global efficiency will be lowered. In fact, *c* is directly bound to the number of disjunctions, since *they* make holes into SOPOs, thus complicating their shape.

As an example, if *r* is biggest, *e* lowest, the complexity of the algorithm is cn^2. It is analogous to the complexity of arc-consistency algorithms : Nn^2, where N would be the number of elements in a SOPO. But we must repeat here that the true complexity can be highly variable.

5 Related Works

Many systems were designed to be temporal modules. They all suppose that occurrences are intervals (or their degenerated form : points). But they differ in the objects they describe best and thus belong to two families : numerical and symbolic systems.

Symbolic systems put the emphasis on the description of temporal relations. In [1] (Allen 83) a relation is a disjunction of primitives which are very cleanly built in [2] (Allen Hayes 85) from the "meet" relation between two intervals. This representation allows a very easy computation of conjunctive relations ($Ar_1B \wedge Ar_2B \rightarrow Ar_3B$) which is in some way an intersection of combinations. Moreover, Allen gives a table for computing the composition of relations ($Ar_1B \wedge Br_2C \rightarrow Ar_3C$). This allows the computation of path-consistency (according to Mackworth's terminology [4]) which is more complete than an

arc-consistency, hence a $O(n^3)$ complexity, where n is the number of events. However, the system only deals with symbols, so numerical data are not efficiently handled, especially duration.

On the other hand, numerical systems put the emphasis on the description of occurrences. The beginning and the end of an occurrence are explicitly represented with two numbers. DEVISER [7] (Vere 83) is a planning system where the windows were first introduced. They are compressed in the same way that we constrict our SOPOs. The hypotheses of DEVISER ensure that the arc-consistency is sufficient to solve the constrained occurrences problem. However, all Allen's primitives are not representable (some are with an astute use of the formalism) and disjunctions cannot be propagated. Moreover, the temporal relations are inside the system since they are implicitly deduced from the plan structure, so the user has no access to them. Smith [6] proposes a temporal module for the ISIS job-shop scheduling system. ISIS builds a hierarchical task net with resources and Smith's temporal module propagates the reservations of these resources through the net, also using window compressing. Here the plan structure is also the only way of specifying temporal relations, and disjunctions are not propagated.

6 Conclusion

We gave in this article a general frame for propagating temporal constraints over events, these events being exclusively considered as sets of possible occurrences (SOPOs). These SOPOs are the numerical expression of an uncertainty about the exact occurrence of an event, while this exact occurrence is constrained by the possible occurrence of other events. Making the SOPOs compatible with the temporal constraints is what we call a *constrained occurrences problem.* This key-problem for scheduling, is an instance of the consistent labeling problem and is known to be NP-complete. We introduce a graphical representation of SOPOs which enables the visualization of a constraint propagation algorithm. This algorithm is a Waltz-type filtering algorithm [8], it does not remove every inconsistency, unless the SOPOs are of the "generalized window" type and all the disjunctions are removed.

We are developping our current reflections along two axis :

1. Making the algorithm more efficient. The complexity can be variable, depending on how effectively the constraints are propagated. In an ideal network of constraints, every SOPO is linked to the other so that only one constriction is sufficient (i.e. the whole information is propagated during the first shot along the links). If there is a number of constrictions over one SOPO, this means that the temporal relations involving the SOPO are too weak in comparison with the implicit constraint of the net. For a greater efficiency, the topology of the net should be changed, making explicit the actual constraints. Such a symbolic inference can be done using Allen's [1] symbolic propagation.

2. Using the temporal module for higher level scheduling concepts. All the solutions of a constrained occurrence problem are not good ones. Qualifying solutions involves high level concepts such as robustness. A robust schedule should require minor adjustments when a perturbation occurs. In this respect, a graphical point of view might give visual "patterns" useful for detecting the presence of good solutions.

Acknowledgements

This paper benefited greatly from discussions with Y. Descotte and from the useful comments of J-C Latombe and J. Crowley.

References

[1] J. F. Allen, *Maintaining knowledge about temporal intervals* Communications of the ACM, November 83, Volume 26, Number 11, pp 832-843.

[2] J. F. Allen and P. J. Hayes *A common sense theory of time*, in proc. IJCAI 85, Los Angeles, USA, August 85, pp 528-531.

[3] H. Delesalle Y.Descotte *Une architecture de systeme expert pour la planification d'activite* (in french), in proc. 6th International Workshop on Expert Systems and their Applications, Avignon, France, April 86, pp 903-916.

[4] A. K. Mackworth *Consistency in networks of relations*, Artificial Intelligence 8 (1977), pp 99-118.

[5] J-F Rit *Vers une representation du temps pour la planification* (in french) memoire de DEA, Institut National Polytechnique de Grenoble, June 85.

[6] S. Smith *Exploiting temporal knowledge to organize constraints* Technical report CMU-RI-TR-83-12, Carnegie Mellon University, July 83.

[7] S.Vere *Planning in time : Windows and durations for activities and goals*, IEEE trans. on PAMI, Vol. PAMI-5:3, May 1983, pp 246-267

[8] D. L. Waltz. *Generating semantic descriptions from drawing of scenes with shadows* MAC AI-TR-271 MIT 1972