PLANNING AND EXECUTING OFFICE PROCEDURES IN PROJECT ASPERA

M. Cristina Bena, Giorgio Montini
Laboratorio Intelligenza Artificiale
CSI-Piemonte
C.so U.Sovietica 216 - 10134 TORINO (ITALY)

Franco Sirovich
Dipartimento di Informatica
University di Torino
C.so Svizzera 185 - 10149 TORINO (ITALY)

ABSTRACT

An approach based on planning techniques is proposed in order to support the complex activity of a multi-office organization. The Italian Public Administration has been taken as an example and analyzed. The synthesis and the execution of procedures are investigated in an automated office based on knowledge about the organization structure, the documents and the rules that regulate the activity. A system is described whose aims are the formation of a plan that represents an executable specification of a procedure and the effective execution of the required operations. When an exception is raised, the planner is automatically invoked to update the plan according to the current state of the office system. A prototype has been built and tested on some applications within the domain of the Piedmont administrative procedures.

I INTRODUCTION

The integration of the Artificial Intelligence (AI) concepts and methods with the current Office Information Systems (OIS) facilities is a good way of improving the quality of the latter, and of increasing their functionalities and efficacy. It is plain that the OIS'a capabilities cannot merely consist in supporting a few simple tasks such as electronic mail, information retrieval, document preparation and so on, which are common to most offices. More than a uniform data representation and a system language for programming office procedures is required to effectively solve the Office Automation (0A) issue. To this respect, it is by now widely accepted (Barber 1982, Crof-; 1984, Aiello 1984) that AI techniques can be useful in many aspects of the office work, e.g. the utilization of problem solvers to perform specific activities in specialized domain.

Indeed, an effective automated office system should help users also in performing their higher level unstructured tasks, which are related to the goals and functions of offices. Knowledge based systems are useful in this activity, too. Such systems can make use of knowledge about the structure and the organization of the environment, of knowledge coded within the laws and rules that regulate the activities, and of more informal knowledge on current practices, derived from the gained experience. These kinds of knowledge can help in establishing the behaviour of the user in an organization consisting of many specialized offices, by means of a problem-solving activity whose aim is to arrange complex procedures that achieve the user goals in a cooperative way.

These requirements are particularly emphasized in the Italian Public Administration (PA), where we have very complex and multi-layered laws (e.g. state vs. local laws) and a particular work organization. Even though the various agents have some margin of authonomy and a particular degree of responsibility and accountability for their portion of work, their activity is subject to a significant control, either from a technical, or from a decisional point of view. The organizational structure of the PA is shaped according to the control structure which is required by law and such a kind of cooperation is needed in order to accomplish the work. Because of this fact, the problem-solving activity in the PA must be very sensitive to the organization itself. The domain of the PA proved to be quite an interesting field to experiment with, verify the applicability of problem-solving techniques to the synthesis of office procedures and improve the OIS features.

The paper is organized in five sections: Section II outlines our view of multi-office organizations; Section III determines the boundaries of the problem and analyzes how problem-solving techniques are currently used in the office environment; in Section IV we propose an approach, based on planning techniques, to the synthesis and the execution of procedures in an automated office; finally, the last two sections of the paper describe a prototype we built at Csi-Piemonte and some examples of its functionalities.

II THE DOMAIN CHARACTERISTICS AND NEEDS

In the PA environment, most of the office work consists in tackling procedures, which are neither strictly coded nor exactly repeated. Each case must

be first handled by doing some problem-solving activity in order to identify the rules and the competences which are relevant to it.

The procedures concern many matters and categories of laws and rules. Some of these are stable, some are evolving, some leave a wide room to the discretion of the administrative structure. This discretion is frequently exerted by bureaucratic regulations, e.g. circulars or deliberations.

The knowledge on the specific procedures must be combined with the general rules and the organization: The problem-solving activity consists in building a strategy for facing the current case, and then applying it. This results in a planning process which produces a sequence of activities, whose main goal is to operationally solve the new task. In this domain, it is worth noting that most undertaken steps are involved and must be certified by documents which record, possibly in a synthetic form, that all relevant regulations have been complied with and all activities have taken place in a lawful way. For this reason, the planning process is linked to the structure of the necessary documents on the one hand and defines the different structures of these documents on the other.

In the PA, the resolution of the problems we analyzed as testbed is entrusted to the level of the officials responsible for the offices (alias Services in this environment). It is plain that this kind of office work requires capacities pertaining to a different level than, for example, the secretary's one. Rather, it concentrates on the tasks of the other levels, and integrates them within procedures. The officials have some kind of discretion and accountability, both in establishing the practices which make concrete the general content of the laws, and in taking decisions involving technical aspects. In fact, the duties these officials carry out do not include issues which require political decisions. In the PA, this type of issues is of concern for the elective political organs only.

These officials never work alone: The operative part of their activity is strictly bound to the interaction with the other agents in the system. They receive incitements from the environment, and themselves produce incitements. Their activity is subject to control, and they themselves exert some control over the other agents activity. Moreover the operations they solicit (visas, signatures, production of documents, ...) not always yield the expected results, just because of the technical discretion the other officials exert and because of the indeterminacy and of the indefiniteness of the decisions the political organs take.

The complexity of the environment and the characteristics of the tasks these persons perform cause their prevailing activity to be the (quasi interactive) construction and execution of procedures. Reasoning ability and knowledge is required:

- to build new procedures;
- to face unusual cases and exceptions to the normal course of a procedure;
- to adapt a procedure to the changes in the environment, for example because of new laws or dispositions.

III PROBLEM DELIMITATION

The challenging and difficult problem of interpreting legal texts and solving the contradictions among the rules (Cook 1980, McCarty 1980, Goldman 1985, Rissland 1985) has been avoided in order to deal with the complexity of the domain. A knowledge-base, which provides relevant formal rules and combines them with the informal rules gained from the experience has been developed with the help of domain experts.

In particular, our attention has been focused on making and managing contracts. In this case the government regulations and laws on the one hand and the routine procedures and the officials' experience on the other, state:

- how the PA must choose a contracting party;
- that the PA's contracts depend on certain requirements, formalities and others documents;
- how these contracts must be managed by the PA;
- who carries out which functions in the organizational structure of the PA.

These rules are at the bottom of the reasoning for "constructing" and "executing" suitable procedures. Then, our work is in a complementary position in relation to the complex legal decision making problem, although it falls within one of application area for expert system in law: the legal procedure and document generation (Waterman 1986).

The addressed problem is the following: In the field of the administrative procedures, as we pointed out above, and in many other offices, as we feel, the problem-solving activity involves both the procedure oriented and the data oriented approach. The reasoning activity aims at developping plans (i.e. procedures) which provide them both. Procedures and documents are strictly related and interdependent for a series of reasons. Firstly, the properties of a document to be produced often determine or require specific steps in the procedure, or superimpose a particular execution order. Secondly, the actions require the existence of several documents as a precondition. Thirdly, the production of documents is itself an operation. Finally, the documents record and certify the most important parts of the procedures. Therefore, we cannot cope with all these requirements by modeling

the problem-solving activity in the office from only one of these two points of view.

In fact, in the interpretation of problem-solving as a data oriented activity, the solution of a goal is achieved by picking out which documents are necessary and which characteristics are required for them. Documents are described by means of a set of properties, connected by various relationships, usually expressed as rules or constraints. As an example, consider the definition of the documents and their relationships in the OMEGA system by means of "descriptions" (Attardi 1980). From this point of view, part of the problem-solving activity consists in building documents. Informations are gathered by interacting with the user and by conveniently querying data bases.

On the other hand, the use of problem-solving techniques in the procedure oriented approach aims at stating which steps must be done in order to perform the task. In this case, the rules of a knowledge base define the logical decomposition of the task in a set of subtasks and the corresponding precedence relations among them, down to the level of the application of single tools.

In the OA literature, a similar process can be found • in the automatic generation of Augmented Petri Nets (APN) from procedure descriptions expressed in a non-procedural language (Zisman 1977, Zisman 1978). In this non-procedural language we can distinguish two different procedure representation levels. One level defines successive tool applications as "macros" of the system (Activity detail), while the other level combines the macros introducing precedence relations (Activity initiation). An alternative way of viewing the problem-solving methods in the office procedures is to define procedure hierarchies (Croft 1984), where the terminal nodes represent single tool invocations. The planner examines the state of the objects contained in the semantic database and the procedure descriptions in order to automatically carry out as much of that procedure as possible.

IV OUR PROPOSAL

In offices like those we described in the previous sections, the goal of the reasoning activity is to build a "plan" and then to control its execution by modifying it when an unexpected situation occurs. This activity aims at identifying the various operations to be done and the precedence relations among them, at establishing which agents must perform the most "routine" and mechanical activities, as well as the less structured ones, that involve choices and decisions, and at defining all the documents involved in the procedure.

A planning system can take the role of a consultation system in an effective and useful way for this kind of users, because its running mimics the activity they perform, that is the construction of procedures and documents. In addition, an integrated approach to plan generation and plan execution (Sacerdoti 1977, Wilkins 1985) can be profitable in an office environment where exist a few tools, however simple, such as electronic mail, text processors, archives and so on. In this case, the process of monitoring plan execution can automatically carry out as much of the step specified in a plan as possible, help the users to coordinate the less structured activities and, when a failure or a surprise occurs, shift to the planning process to modify the plan.

We propose a planner which uses both properties of documents and procedural steps, and therefore can introduce new informations in documents and/or particular steps in procedures, in order to reach its goals. A uniform representation for the knowledge on both procedures and documents is required because they share the same "rationale" and find reciprocal justification and explanation.

According to the laws in the domain, the plan resulting of the reasoning phase could be interpreted as the way the overall organization acts. To make the result of the reasoning process an executable procedure we could examine the plan from the point of view of an individual official and apply a mapping between the predicates that describe primitive operations for the planning mechanism and "macros" of an office language. The office language could provide primitives for synchronizing the activities and a macro definition facility. Moreover, its interpreter could invoke a series of integrated tools and execute steps in parallel.

The planner, on the other hand, should be not only a procedure generator, but also a document generator. In this case, the documents could be described in the plan by means of a set of properties. In analogy to the procedural aspect, we could define a mapping between properties, on one side, and structure and contents on the other. Applying this mapping, the document part of a plan can be translated in a form definition language (Tsichritzis 1982) or in a constraint language (Ferrans 1982). Even tools for the automatic composition of free text documents (Zisman 1977, Sprowl 1980) can avail themselves of the informations produced by the reasoning program. The generic text of a document can be described by means of a production rule language, where the antecedent part matches the informations generated by the planner, and the consequent part corresponds to portions of text containing variables that are bound during the pattern matching phase.

Note that with a homogeneous representation of documents and procedures, we can express the semantic consistence of a document also by relating it to the context of the procedure it pertains to, being this context known during the reasoning phase. In this way, our planner can acquire, integrate and organize the knowledge needed to produce documents, determine the conceptual structure of the documents and get nearer to an effective document generator, improving the features of already existing tools. This fact paves the way for effectively enriching the OIS environment, by controlling several OIS tools from a higher level, that is from the cognitive level.

V SYSTEM DESCRIPTION

The ideas suggested above have been implemented within the project ASPERA* of the Csi-Piemonte in CProlog Vers.1.1 on AT&T 3B2 microcomputer under UNIX** System V as operating system. The overall architecture of ASPERA consists of two processes communicating by means of two named-pipes. The first process is a UNIX shell which simulates an OIS system, the second is a Prolog environment where a planner and an execution-monitor act. Fig.l shows the ASPERA modules whose main functionalities are described below. The simple unbroken arrows show which modules call which others. The double unbroken arrows show where the informations produced by the various modules are loaded and the broken ones the flow of data through the system.

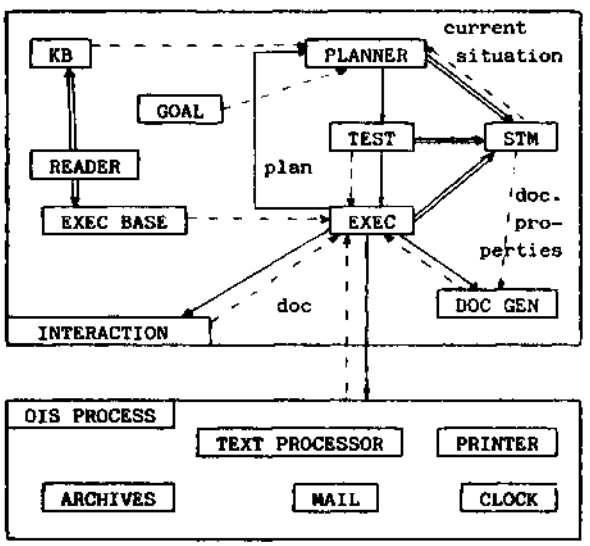

FIGURE 1: THE ASPERA SYSTEM

The knowledge is described by a knowledge representation language oriented to the office domain and based on logic programming. With respect to most planners described in the AI literature (Sacerdoti 1977, wilkins 1984) we do not only represent the knowledge by means of "operators", but we employ the logical decomposition of the Horn clauses formalism to define the relations expressed both in the laws and regulations and in the informal rules the officials use to cope with the problems. We combine this "qualitative" reasoning with an "operative" one concerning which primitive actions*** are to achieve and which conditions are necessary before an action can be performed. The result of the reasoning is a network of instantiated primitive actions (nodes) at just one level of detail with partial ordering relation on them.

Note that low cost hardware and general purpose computers, which the PA can utilize today, are the target machines for this application. Thus we chose to adapt the expressiveness of Prolog to the needs of the addressed problem instead of employing more powerful tools on Lisp machines.

A. The language

The knowledge representation language extends the Horn clauses formalism (Clocksin 1981), by supplying suitable mechanisms for the office domain. The main extensions are:

. classification of predicates according to their role within the office system;
. definition of a soundness range for the variables according to a taxonomy of concepts in use in the domain;
. precedence relations among operations;
. definition of the functional dependencies in the predicates;
. identification of the document types and of the document versions.

The knowledge base (the KB of Figure 1) consists of definitions of predicates, taxonomic concepts and logical rules. The predicates are classified in four categories:

• operations, whose instances correspond to office system macros;
• document predicates, whose instances define properties and contents of the documents;
. interaction predicates, which describe informations to be found outside the system (users, data bases, ...);
. logical predicates, which can represent different kinds of relationships.

*ASPERA is the acronym of Automatic Support for Planning & Executing Regional Administrative procedures. This project has been carried out with the collaboration of the Department of Computer Science of the University of Turin and of the Regional Shareholdings Service of Regione Piemonte.

**UNIX is a Trademark of Bell Laboratories.

***That is the leaves in the deduction tree.

The taxonomy of concepts represents the hierarchy of semantic categories* existing in the domain and is a finite graph without cycles. The arcs which interconnect the categories specify subset relations (Figure 2).

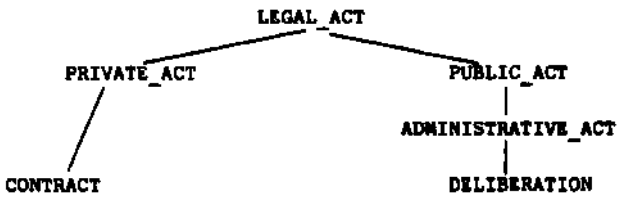

FIGURE 2: A TAXONOMY

The variables and the constants (objects) which appear in the rules (Figure 3) can be restricted according to the categories specified in the taxonomy by means of the Prolog operator "isa".

```
registered(Contract) :-
        >> signed(Contract),
        reference_number(Contract,N),
        entered_in_the_register(Contract,N)
    :   Contract isa contract.
```

FIGURE 3: A RULE

The precedence relations among operations represent some procedural knowledge of the domain that contributes to the success of a plan. These relations are expressed by preconditions and marked with the symbol » . For example, the rule of Figure 3 shows** that people must sign a contract before it can be registered. A general rule may contain both preconditions and subgoals in which the rule consequent is decomposed.

Operation, document and interaction predicates allow to define functional dependencies of their arguments (Attardi 1984). For example, the predicate definition guaranty (#offer, type_of_guaranty) expresses that an offer must provide for a particular type of guaranty. The symbol#means that every offer must fix univocally the guaranty type.

At last, a section of KB allows to declare the types of the documents involved in the domain. For each document a version sequence can be defined (draft, text to be voted). The planner examines these informations to enforce some precedence relations among the nodes automatically. For example, the system assumes a particular version of a document must be always drawn up before it can be used.

*A semantic category is a name given to a collection of known elements in the domain (McSkimin 1979): e.g. subject, administrative act, contract.
**We follow Prolog convention: an initial upper case character means a variable and a lower case denotes constants, i.e. objects and categories.

The Reader module loads in the Prolog database the knowledge base, builds a few internal representations for it, initializes the short-term memory (STM) of the system and loads also the rules that define how an operation has to be executed (these "macros" are loaded into the EXEC BASE of Figure 1).

B. Reasoning and Plan Formation

Several completeness levels of a plan could be considered according to the number of cases the plan is able to cope with. The simplest level supposes "optimistically" that each operator returns a successful (and expected) result. In this case, a coherent plan is established by considering as reference the procedures that generally are used. A mechanism for revising the plan is triggered when an exception occurs. An exception can be defined as the mismatch between the situation expected in the plan and the actual situation. A most complex level consists of a program which provides for all the alternatives on which there exists some knowledge. In other words, we can say that the second type of plan contains as much informations as we are able to infer from the knowledge.

In our domain, for the complex, dynamic and indefinite nature of the external world described above and for the inherent incompleteness of the knowledge base, it is useless that the planning system investigates all the possibities that can happen in the world: This 1S not always practical or desirable. It is more useful to accept that things in the real world not always proceed as planned and to aim at an integrated approach to plan formation and plan execution with capabilities to replan.

In the current experimentation, the interpreter (Planner of Figure 1) produces a plan by proving a theorem which expresses a goal. The proof occurs according to the usual method of the LUSH-resolution. A set of nodes is associated with each proof and represents operations that will produce the effective realization of the goal, if they are executed with success. The nodes are globally stored in the STM. During the reasoning process both the precedence relations among the nodes and the properties of the documents involved in the procedure are also inserted in the STM.

During the proof, the variables can be restricted to more specific categories and the unification algorithm considers these restrictions to unify variables and objects (Montini 1987). This simplified constraint approach improve the efficiency and the generality of the planner, reduces the amount of backtracking necessary during a proof and allows a more general symbolic reasoning.

The system applies rules of the knowledge base and acts in different ways in accord with the type of the predicate it is examining. A logical goal is achieved by selecting a rule and by proving in its turn each subgoal which appears in it. A set of nodes is associated with the proof of each subgoal; the union of these sets is associated with the proof of the goal. Precedence relations among nodes are introduced by examining the preconditions that appear in the rule: we introduce in the global description of the plan a precedence between the nodes associated with the preconditions and the nodes associated with the goal. In short, in the plan the nodes which correspond to preconditions precede the ones which correspond to the goal.

The rules that describe the next three kinds of predicates, i.e. the operations, the predicates about the documents and the interactive predicates, can only contain preconditions, and not subgoals. From the point of view of the logical decomposition in subgoals, these predicates, therefore, must be considered as "primitive". In the course of the reasoning process, a goal whose type is operation or interaction is considered proved, if the operation or interaction has already been done. Otherwise, the preconditions are examined, a new node is built, associated with the goal and inserted in the plan, in STM, together with the precedence relations with regard to the preconditions. The new node is thus the result of the proof and is passed to the higher level goal.

During plan formation, as the nodes are globally stored, two equal instances of the same node are not allowed*. A new node is not introduced in the plan, if its description corresponds to the one of an already inserted node: in this case the two nodes are unified and regarded as one. Great care is also taken in backtracking, when we have to delete some nodes and some precedence relations from the STM. Consequently, we only delete a node from the plan when the backtracking arrives at the point in which the node was really inserted into the net.

The global informations are also kept consistent by examining the functional dependencies (for example, nodes with contradictory informations are not inserted into a plan) and by checking the absence of cycles in the plan (i.e. the precedence relations among the nodes of the plan cannot involve cycles). This last task is performed by the TEST module of Figure 1.

Likewise, a document property goal is satisfied by examining the description of the document, if it already exists and has been filled. Otherwise, the preconditions are examined. If the proof is successful. a new specific node is built. This node represents the operation of drawing up the document. The property is then inserted into the document description in the STM, and the consistency is checked with respect to functional dependencies.

C. Plan Execution

The link between the planning system and the interpreter of the office system programming language is performed by the translator (EXEC module of Figure 1), whose task is to map the net of nodes and the precedence relations into an executable procedure. In the plan, each node represents an operation. Some of its arguments are interpreted by the translator as the agents who must carry out the operation, according to the syntactic conventions in the mapping. Thus, the translator can interpret the plan as the way the overall organization acts and can identify where the roles and the competences of the agents within the area of the procedure fall.

During the translation phase, a sequence of steps is built by examining the plan from the point of view of an individual agent. Translation of an operation into a macro is achieved taking into consideration the point of view the translator assumed and the arguments of the operation. If an operation of the plan is forwarded(agentA, agentB, documentD), when the translator has the point of view of the agentA, the operation is mapped into the macro send_mail(documentD, agentB); otherwise, when the translator acts as the agentB, the operation corresponds to a statement as receive_mail(documentD, agentA), which predisposes agentB to receive documentD from agentA; again, if the plan is examined from the point of view of a third agent, the agentC, the operation corresponds to no action.

During the execution, the available tools can be the usual ones in an office system. Among them, it is worth mentioning expert systems to solve specific problems in the domain (Barber 1983). The role of the preparation document tools has been emphasized in the implementation. These tools make use of the results of the reasoning phase both to automatically fill form and to put parts of documents together in order to produce free text documents. In order to achieve these results, the system uses a particular document description, in which parts of text and predications are mixed, in a language that substantially remembers the precondition and action rule paradigm. The produced documents are, of course, objects of the archiving system, so next steps in the procedure, or even other procedures, can use them again.

At last, when an operation produces a different

♦This choise is closely related to the heuristic "Use existing object" used by NOAH (Sacerdoti '77).

result than the one expected by the plan, the execution of the procedure is interrupted and the planning system is invoked to extend the plan, according to the current situation, i.e. the operations already performed and the results obtained. The already performed operations, the drawn up documents, the query already asked do not appear in the new plan which only provides the actions which have still to be done. Then the control is returned with the modified plan to the execution monitor and this cycle can continue until the whole procedure has been completed.

In this regard, a problem of incompleteness owing to the characteristics of the office operations exists. The execution of an office operation cannot be undone, once happened, unless we explicitly provide for the execution of the "inverse" operation in the plan, if this is possible. The incompleteness is due to the fact that such an inverse operation does not always exist or can be applied (for example, a signature usually cannot be withdrawn). In such cases the execution of an operation could prevent the possibility of going through alternative ways, and, thus, of achieving the overall goal. Likewise, there is a problem of minimality in the plan formation: if an undo operation exists, the total cost of the plan is made higher by two operations which cancel out.

This problem can take advantage of a replanning approach like that of (Sacerdoti 1977, Wilkins 1985) and in part solved with plans which provide alternatives. Our research is continuing in this direction, by taking into consideration both the possibilities of symbolic execution involved in the use of restricted variables and the cost of such plans in relation to the dynamic nature of the external world.

VI AN EXAMPLE

The prototype we described in the previous section has been tested on some applications within the domain of the Piedmont regional administrative procedures. In order to exemplify the concepts we present in this paper, some relevant rules follow. The following rule, that is the transcription of the Regional Piedmontese Law 13/78, art. 4

```
chosen_contractor(Contract, csi,rp):-
   object_of_contract(Contract,Work)
   able_and_willing(Contract,csi,rp),
   write_legitimation_contract(Contract,'law 13/78')
: Contract isa contract, Work isa computer_science.
```

ensures Csi-Piemonte (referred to as csi) has right of priority in entering into contracts with Regione Piemonte (referred to as rp) when the contract concerns the field of computer_science. In the rule, the mixture of procedural steps (e.g. able_and__willing) and document properties (as write legitimation__contract) is evident. Moreover, the rule contains elements of procedure recording. In fact, if in achieving its goal the planner makes use of this rule, the condition write_legitimation contract must be satisfied. The planner inserts this property in the STM, and, during the execution phase, the free text document generator uses it to insert into the contract a reference to the pursuance of the regional law as a justification.

The fact that people (X) express their will to enter into a contract with someboby (Y) by forwarding him an offer is established by the rule:

```
ab1e_and_willing(Contract, X, Y): -
      >> valid_offer(Offer,X,Contract),
      forwarded(X,Y,Offer)
  :   Contract isa contract, Offer isa offer,
      X, Y isa juridical_person.
```

The validity of the offer is a precondition. This property is defined in turn by other rules that specify further properties the offer must satify, and that, during the planning phase, introduce more nodes and precedence relations into the net.

Observe that the "forwarded" predicate is declared as an operation, and that, hence, during the planning phase its instances correspond to nodes (see Figure 4). Taking into consideration the "valid_offer" precondition, the plan contains suitable precedence relations between the nodes deriving from the precondition and the one which corresponds to the forwarding. At execution time all the operations relative to the validity of the offer must be completed before the actual forwarding of the document starts. Among these operations there is the drawing up of the offer itself.

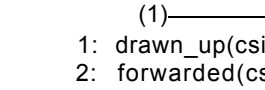

FIGURE 4: A PRECEDENCE NETWORK

The translation of the network in Figure 4 can be done from two points of view, that is the agent csi and the agent rp. The procedure for csi expects that two macros are successively invoked. The first macro calls the free text document generation tool in order to produce an offer based on the description contained in the plan. The other macro, as we previously saw (V.3), invokes the electronic mail facility, and sends Region the offer. In the procedure for rp the node (1) will be translated into the empty macro, because it requires no operations on behalf of that agent, while the node (2) predisposes the agent to receive the document. Remark that the macro can also contain some timeout mechanisms, in order to send a solicitation in the case of delays in forwarding the offer. The

receivejnail macro, lastly, takes care of invoking the tool, which checks that the contents of the documents coincide with the expectations in the plan. If such a check fails, the conveniently invoked planner, making use of the new informations, can eventually modify the plan.

VII CONCLUSIONS

The approach to the office activity presented in this paper is centered on planning techniques, which are based on knowledge about the organization structure, the documents and the procedures of the office. This approach can be considered, according to the opinion of (Hammer I960), as providing tools for supporting the office system analysis. The task of the office analyst is to describe all the necessary knowledge, and, by using the planning capability, to arrive at an executable specification of procedures in the automated office system.

In a highly dynamic and ever changing environment, like the PA's world, this analysis is itself one of the system tasks (indeed, one of the main tasks). Unlike other kinds of office, this task is not a prerogative of an individual worker, but is constantly performed by the officials responsible for Services, to face both the very great number of execeptions, and the changes in the laws and regulations. Moreover, we feel this activity must be supported in an automated office, and included in an overall office model, because it is not closely related to the domain we considered, but is involved in the general management of the activities in a firm.

ACKNOWLEDGEMENTS

The authors wish to thank F.Massacesi and D.Formento, of the Regional Shareholdings Service of Regione Piemonte, for their work as domain experts. We are grateful to our colleagues at Laboratorio di Intelligenza Artificiale, with which we had useful exchanges of opinions, and P.Tribaudino for her helpful editorial assistance.

REFERENCES

1 Aiello,L., D.Nardi and M.Panti "Modeling the Office Structure: A First Step Towards the Office Expert System." In Proc. 2nd ACM-SIGOA Conf. Office Inf. Syst. Toronto Canada June 1984, 25-32.

2 Attardi,G., G.Barber and M.Simi "Towards An Integrated Office Work Station." (in Italian) Automazione e Strumentazione, 28:3 (1980) 223-235.

3 Attardi,G. and M.Simi "Metalanguage and Reasoning Across Viewpoints." In Proc. 6th European Conf. Artificial Intelligence, Pisa, Italy, Sept. 1984, 315-324.

4 Barber,G. and C.Hewitt "Foundations for Office Semantics." In Naffah N. (ed.), Office Information Systems, Amsterdam 1982, 363-382.

5 Barber,G. "Supporting Organizational Problem Solving with a Work Station." ACM Trans. Office Inf. Syst., 1:1 (1983) 45-67.

6 Clocksin,W.F. and C.S.Mellish Programming in Prolog. Springer-Verlag, 1981.

7 Cook.S. and R.Stamper "LEGOL as a Tool for the Study of Bureaucracy.", In Lucas H. (ed.), The Information System Environment, Amsterdam, North Holland, 1980.

8 Croft,W.B. and L.S.Lefkowitz "Task Support in an Office System." ACM Trans. Office Inf. Syst., 2:3 (1984) 197-212.

9 Ferrans,J.C. "SEDL: A Language for Specifying Integrity Contraints on Office Forms." In Proc. 1st SIGOA Conf. Office Inf. Syst., Toronto, Canada, June 1982, 123-130.

10 Goldman,S.R., M.G.Dyer and M.Flowers "Learning to Understand Contractual Situation." In Proc. IJCAI-85, Los Angeles, California 1985, 291-293.

11 Hammer,M. and J.S.Kunin "Design Principles of an Office Specification Language." In Proc. AFIPS Nat. Comp. Conf., May 1980, 541-547.

12 McCarty,L.T. "The TAXMAN Project: Towards a Cognitive Theory of Legal Argument." In Niblett B. (ed.), Computer Science and Law, Cambridge, Cambridge University Press, England 1980, 23-43.

13 McSkimin,J.R. and J.Minker "A Predicate Calculus Based Semantic Network for Deductive Searching." In Findler, N.V.(ed.), Associative Network, London, Academic Press, 1979, 205-238.

14 Montini,G. "Efficiency Considerations on Built-in Taxonomic Reasoning in Prolog." In Proc. IJCAI-87, Milano, Italy, 1987.

Ib Rissland,E.L. "AI and Legal Reasoning." In Proc. IJCAI-85, Los Angeles, 1985, 1254-1260.

16 Sacerdoti,E.D. A Structure for Plans and Behavior Elsevier North-Holland, New York, 1977.

17 Sprowl,J. "Automated Assembly of Legal Documents." In Niblett, B. (ed.). Computer Science and Law, Cambridge, UK, 1980, 195-205.

18 Tsichritzis.D. "Form Management." Commun. ACM 25:7 (1982) 453-478.

19 Waterman,D.A. and J.Peterson "Expert System for Legal Decision Making." Expert Syst. 3:4 (1986).

20 Wilkins.D.E. "Domain-indipendent Planning: Representation and Plan Generation." Artificial Intelligence, 22:3 (1984) 269-301.

21 Wilkins,D.E. "Recovering From Execution Errors in SIPE." Technical Note 346, SRI International Menlo Park, California, Jan. 1985.

22 Zisman,M.D. "Representation, Specification and Automation of Office Procedures." Unpublished Ph.D. Dissertation, Working Paper 77-09-04, Warthon School, Univ. of Pennsylvania, 1977.

23 Zisman,M.D. "Use of Production Systems for Modeling Asynchronous, Concurrent Processes." In Waterman and Hayes-Roth, Eds., Pattern Directed Inference Systems, New York Academic Press 1978.