*Consiglio Nazionale delle Ricerche*

# ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

**PISA**



## Representation of Constraints in Extended Transition Network for System Modelling

D. Aquilino, P. Asirelli, K.G. Jeffery, J. Kalmus

Nota Interna B4-65
Dicembre 1992

# Representation of Constraints in Extended Transition Networks for System Modelling

*Keith G Jeffery*
*John Kalmus*
Systems Engineering Division
Rutherford Appleton Laboratory
Chilton, Didcot, UK
Tel. 44-235 446103
E-mail: kgj@ib.rl.ac.uk

*Patrizia Asirelli**
*Domenico Aquilino**
I. E. I.- C.N.R
via S Maria, 46
I - PISA
Tel. 39-50 593477
E-mail: asirelli@icnucevm.cnuce.cnr.it

***Abstract***

*We inspect the issues of using extended transition networks as a framework for representing systems development activity. In particular, joint work between the two teams authoring this paper is illustrated. The work on the extensions to the transition network as required for systems development (both theoretical and practical aspects) has progressed jointly with work on the further development of Gedblog; the deductive Data Base Management System with graphic features developed at IEI-CNR. Many aspects of systems development methods have been discussed; this paper concentrates on the representation of different kinds of constraints, as well as on the definition of a basic framework for semantics interpretation.*

## 1 Introduction

The problem we want to address is the definition of a suitable framework for handling the process of designing Information Systems, ranging from informal to formal specifications.

The aim is to overcome the limits and drawbacks of the techniques actually used in information systems modelling (see [Jeffery 92], [Olle 82] for a review on the subject) by setting up a formalism which could be effective at the different levels of abstraction, from enterprise modelling to systems implementation. The modelling technique we propose consists of a formalism to express the knowledge on data and process models of systems and an environment to handle such knowledge in order to support further processing.

The formalism should provide a set of mechanisms to represent each aspect of systems, that is: data, processes, their pre- and post-conditions, together with constraints on all aspects of the system model.

The environment should permit the application of further processing to this representation. In particular, it should support development, verification and validation-related activities, such as:

- verify the consistency of the model;
- identify incompleteness;
- exercise the model (animation);
- generate and test (a specification for) the required schemata and code.

*Authors partially supported by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo

The solution we are currently inspecting is based on an extension of the Transition Network formalism [Jeffery 92][Jeffery 89] to express knowledge about the system at various levels of abstraction, and on the use of the deductive environment *Gedblog* [Asirelli 90][Aquilino 92] that is capable to support information processing and system model development activities.

In the following we introduce the basic notions of Extended Transition Networks and sketch the formal notation for both syntax and semantics which is actually being developed. We are still exploring the theoretical properties of the technique, to ensure it covers all the requirements and possibly discover new features. Finally, we briefly report on the work in progress within the deductive engine *Gedblog*.

## 2 Extended Transition Networks: notions and representation issues

### 2.1 Basic Notions

Basically, Extended Transition Networks denote a System by direct graphs where:

**nodes** represent states;

**arcs** represent state-to-state transitions.

Each node can have one or more arcs arriving/departing, apart from the starting and the ending nodes.

1. arcs departing from (arriving at) a node denote alternative transitions (disjunction) in the net that depart from (arrive at) the given node;
2. sequences of arcs denote transition paths (conjunctions ) in the net.

An arc between states S1 and S2, represents the *transition* from S1 to S2 and is expressed by a tuple:

*<precondition, process, postcondition>*

an element of the tuple may be null (i.e. missing, optional).

Besides *transitions* a net is able to represent by arcs, other kinds of information. These are *constraints* and *transactions* . All above types of arcs may be further expanded for sub-transitions networks. Thus we have, in addition to transition arcs, the following:

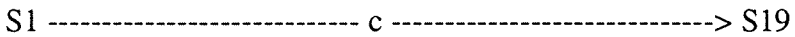
a.1) arcs representing constraints - that is conditions that span larger sections of the network at any level. Examples are non-functional requirements (eg elapsed time of traversal of an arc or arc-sequence) or constraints of the data model that must not be broken;

a.2) arcs representing transactions (i.e. delimiting the start and end of a logical unit of work) with the concept of locking and commit.

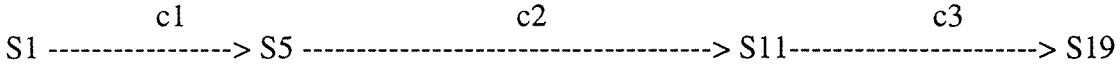
As an exeption to 1. above these additional types of arcs are to be considered logically ANDed with the other arcs departing from (arriving at) the involved nodes. The need for this extention of the formalism is discussed in Section 2.2.

Thus a net can be represented as follows:

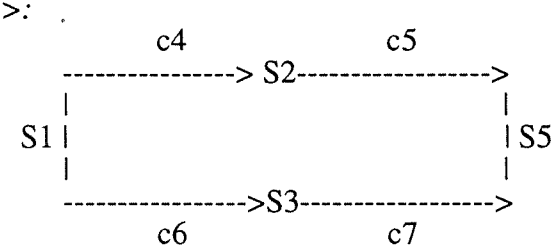
*Highest-level network:*




*Next detailed level:*



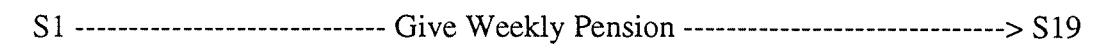

*Details of arc <S1,S5>:*



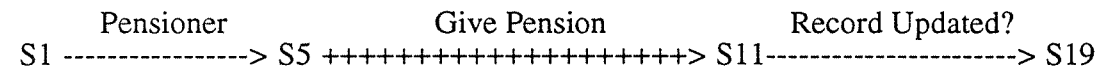
where $S_i$ denote states and $c_i$ are either constraints, transition or transactions.

Here, an example is introduced to show an application of the tecnique. We depict a net that is intended to model a system for managing pensions delivering at the office of Abingdon. In the graphic representation we use, transitions are drawn as --->, constraints are double-dashed arrows(===>) and transactions are represented as +++>.

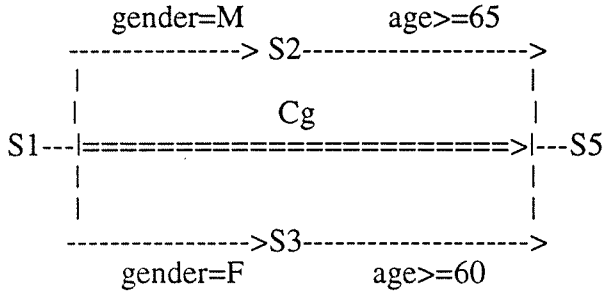
*Highest-level network:*



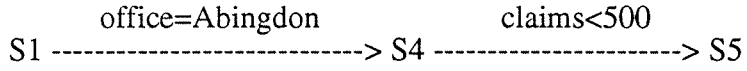

*Next detailed level:*




*Details of transition <S1,S5>:*




*Details of constraint <S1,S5>:*




**2.2 Considerations on the expressiveness of the extended formalism:**

The essential idea behind the extended transition network approach is that an arc between two nodes represents a transition at a given level of abstraction. This implies that the arc, representing all the information systems technology to perform a state change, also is able to represent an abstraction of much detail; the detail being arcs and nodes (i.e. detailed transition networks or sub-networks).

The main feature of this kind of representation is that constraints are defined along with the relevant process that acts on the data, at the suitable level of refinement. They can be expressed at the appropriate level of abstraction: more global constraints at a higher level in the transition network hierarchy with respect to transitions and transactions to which they refer to.

In this perspective, the relations among the identified kinds of transitions play a central role in the process of expressing a system model by extended transition networks. This section motivates the extensions and consider some alternatives to the choosen representation.

The problem is the representation of three semantically distinct concepts:
(a) state transitions involving <precondition>,<process>,<postcondition>
(b) constraints affecting one or more of (a)
(c) transactions affecting one or more of (a)

The situation is complicated by the fact that preconditions and postconditions are, themselves constraints. We overcome this confusion, by referring to this type of constraint as 'local' and a constraint which affects more than one transition as 'global'.

The next problem is the interpretation of the arcs; (b) and (c) type arcs (as defined above) affect >=1 (a)-type arc, and thus are not truly congruent transitions. We have made various attempts to map (b) and (c) type arcs into (a) type arcs, for example by explicitly mapping (b) and (c) type arcs into preconditions and postconditions. However, this is an expensive expansion and would also add computational complexity at runtime. An alternative is to have a arc with 'global scope' before a sequence of transition ((a)-type) arcs; however, this suffers from the disadvantages that

(1) the context of evaluation is not always clear in the succeeding (a)-type arcs
(2) non-monotonic behaviour (eg deletion) which changes the overall system state would lead to either undefined contexts or expensive repeated evaluation.
(3) the implication is procedural rather than declarative style, which was not the design intention.

The core of the problem is that in a transition network, arcs in sequence are in 'AND' condition, while two alternate arcs (or sequences of arcs) are intended in 'OR'. While retaining this structure for state transitions, we wish to be able to 'AND' arcs of types (b) and (c) with single or multiple sequences of arcs of type (a).

Supporting declarativeness brings us to prefer to spread the constraints over all arcs explicitly. Implementation problems may require that the constraints are not really handled locally but only denoted on the arcs, and then handled by setting up nested evaluation environments, e.g. by means of a stack of contexts.

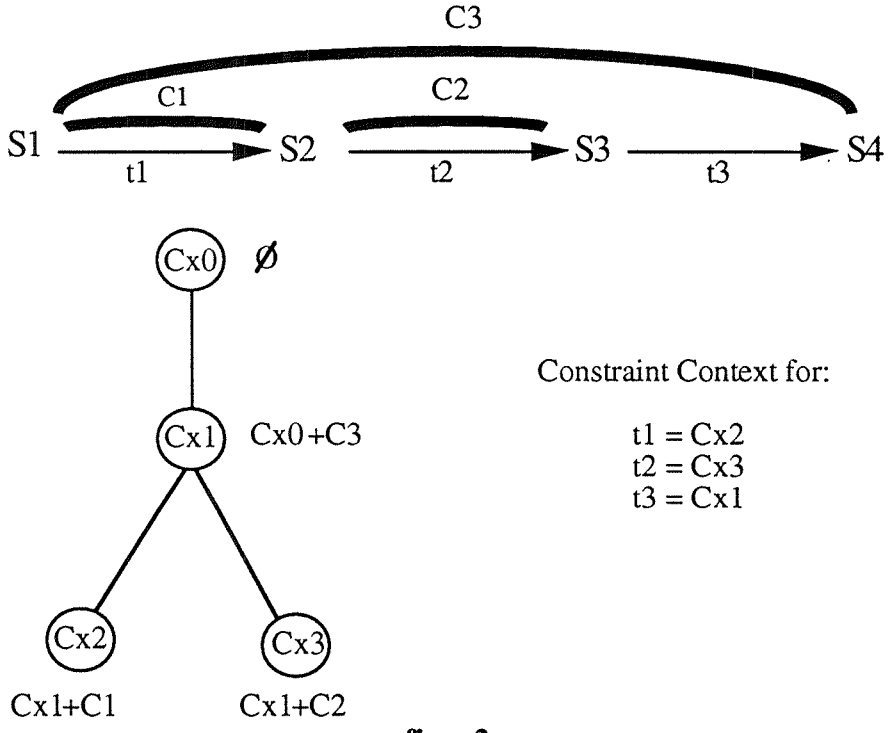



**fig. 2**

The representation is thus only a syntactic sugar: representing the global constraints at a higher level by means of a different type of arc seems to be a suitable way to solve the problem, while the semantics remain the same.

As concerns arcs representing a transaction, let us note that they denote compound operations that, locally, can change the state inconsistently with respect to constraints. The concept of transactions, seen as state-transitions that are 'closed' with respect to global constraints, is expecially needed when modelling the system at deeper levels of details. Thus, transactions need to be represented by a different kind of arc, to denote a semantics which is completely different from the other kind of arcs.

It would seems that the simplicity of the Transition Network formalism is doubted. The process we have followed would lead us to say: do we need a different representation arc every time we have to express arcs with a different semantics?

Indeed the Transition Network formalism only should say that arcs have a semantic function attached to them, and the evaluation procedure for such functions is defined in the environment. To this respect, we can consider that the intuition that underlies the transition network concept we are dealing with is a particular application of the formalism and we are denoting with different arcs what really are different semantic functions.

Furthermore, there are other considerations on the nature of this interpretation that can lead to some confusion in the sense that it can be obscure whether the constraints are more privitive elements than processes and transactions. We came to the conclusion that the net has two primitive notions one is the "transition" the other one is the "transaction": they both denote state-to-state transitions, but the former is considered "open" and the latter "closed" with respect to constraints. Thus, with this meaning the constraints semantic function "flows down" and applies only onto open sub-nets.

## 3 A formal model of Extended Networks

In this section we introduce a formal characterization of the Extended Transition Networks. The major objectives are to overcome the ambiguities of the graphical representation and to provide a mathematical framework for discussing semantics issues.

The flavour of the notation is that of transition diagrams, like that used for modelling Petri nets [Reising 85]. There are, however, noticeable differences; in particular the different kinds of arcs need to be distinguished.

We define a Transition Network as a triple

$$TNet = <S, T, L>$$

where S is the set of states, T the set of state transitions and L a set of transition labels. We define T as follows

$$T \subseteq <c,tr,t>;$$

where c,tr and t are of the same type :

$$c \subseteq SxLxS,\ tr \subseteq SxLxS,\ t \subseteq SxLxS;$$

and

$$c \cap tr = \{\},\ t \cap tr = \{\},\ c \cap t = \{\}.$$

The set of transitions labels L is defined as

$$L \subseteq <Prec, Process, Post>,$$

such that:

$$Prec, Post \in \textit{Embed}FORMULA) \mid TNet$$

$$Process \in \textit{Embed}(PROCESS) \mid TNet$$

where the 'embed' function denotes terms of the host-languages used to express constraint formulas and control processing features.

To provide a semantic specification for Extended Transition Networks we are concentrating on the interpretation of the intuitive concept of performing a net traverse, in order to distinguish the meaning of the different kinds of arcs and their relationships. The semantics we deal with is independent with respect to the language embedded in the previous definitions to denote constraints formula and processes. A functional language, like the relational calculus proposed by [Jeffery 92][Orlowska 91] (the language used in the examples of this paper) or a logic-based programming language (or compromises of this two frameworks [Bertino 92][Saraswat 90]) could support the technique as well. Thus, we want to introduce a framework for the specification of transition networks semantics interpretation. This framework could be further detailed by providing the interpretation of the particular embedded language. In particular, dealing with logic formulas, a notion of *entailment* should be introduced in

order to express the 'satisfaction' relation between constraints and states of nets. However, a considerable work can be carried out even regarding these features as external with respect the basic notions of networks. We will come back on these topics in the concluding remarks of the paper.

To specify the intended meaning of a network with respect its sub-networks we use the metaphor of the "net traveller shadows". A net traveller is a token that travels over the arcs of the network from initial state S to final state F. While crossing the net, the token dinamically "executes" the transitions placed on arcs, where "execute" can be interpreted as 'verifies a constraint', 'proves a condition' or, in general, 'performs an action'. In order to traverse arcs which are refined at lower levels, the traveller projects down its shadows to walk low-level nets and so on, till covering all refinement levels. Shadows are projected also on the global constraint arcs. In this case the travellers of transition arcs are 'under the control' of their shadows running on constraints.

By this analogy, the semantic interpretation of a transition network can be defined by this kind of 'traveller game' (in the flavour of the token-games usually associated to Petri nets). Here, we prefer to give an interpretation of TNs that is abstract with respect the computational details of embedded languages, which can be left unspecified. Nteworks are interpreted on expressions, built by some simple operators, which represent the sequences of transitions a net traveller "executes" to walk from the initial state to the final one. "Execution" means something different for each kind of arc; however, while for transition and transactions (which involve processing) it can be easly understood as a state change, for constraints its interpretation is much highly intuitive.

We are actually working on the formalization of a set of rules for two possible operational interpretations: one relies on the fact that a constraint is always active and an event in fired when a violation is detected along execution of transitions in its scope (we refer to this as a "true concurrent" interpretation); in the other one, the constraints verification is performed after each traverse of an arc in its scope ("interleaving" interpretation). We are investigating to provide an operational semantics of constraint traversing which abstracts these interpretations to allow both behaviours in the system. In the following, we introduce a transition sequences algebra to specify with expressions the transition paths a net traveller and its shadows perform, and demonstrate how transition networks can be interpreted onto the terms of this algebra, by means of an example.

**Definition 1**

Given a finite set $T$ of transition symbols, images of state-to-state transitions, the *transition sequence* terms on $T$, with typical elements $Q$, $Q1$, $Q2$ are generated by the following context-free production rules:

$$
\begin{aligned}
Q &= nil \\
Q &= t \,.\, Q \\
Q &= Q1 \,\Delta\, Q2 \\
Q &= Q1 + Q2 \\
Q &= \{Q\}
\end{aligned}
$$

where $t$ is a transition (meta-)variable ranking on $T$.

The set $T$ of transitions and the *nil* constant are the generators of the transitions algebra, while the operators (., $\Delta$, +, { }) introduced by the production rules are transition sequences terms constructors with the following informal meaning:

| | | |
|---|---|---|
| t . Q | (*Sequencing*): | first state-transition t1 then transitions in Q are performed; |
| t1 $\Delta$ Q | (*Context* ): | t1 is a constraint context for states of the |

transitions in Q;

Q1 + Q2 (*Alternative*): transition sequences in Q1 or those in Q2 are performed;

{Q} (*Closure*): transition sequences in Q are closed with respect to the surrounding constraints context;

**Definition 2**

The interpretation function **I** maps a net onto a transition sequence term;

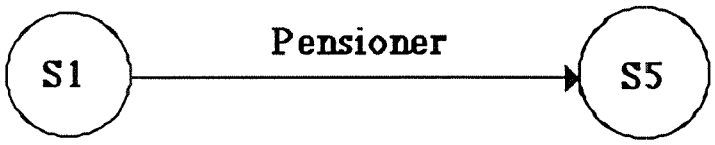
I: NET --> Q.

Constraints, transitions and transactions of the network NET are mapped into the set T of transition symbols of Q, while their different meaning is retained by the operators Δ and { }.

In the context of this paper, we give a hint about how the interpretation function **I** works using an example, which relies on the net introduced in section 2. We use the auxiliary relation E to associate to an arc its transition or, if any, the interpretation of its refining net.

Transition sequences terms are given avoiding the *nil* terminator and using brackets to ensure the correctness in operators precedence

*LEVEL 1*




Net1:

S1 ---t---> S5

At this level of abstraction, a traveller of the net can be expressed as one who crosses the arc S1-S5 identified by the transition t=pensioner which is refined into Net2. Thus:

I[Net1] = E[t]=E[Net2].

*LEVEL 2*

The previous constraint t is refined into the following net:

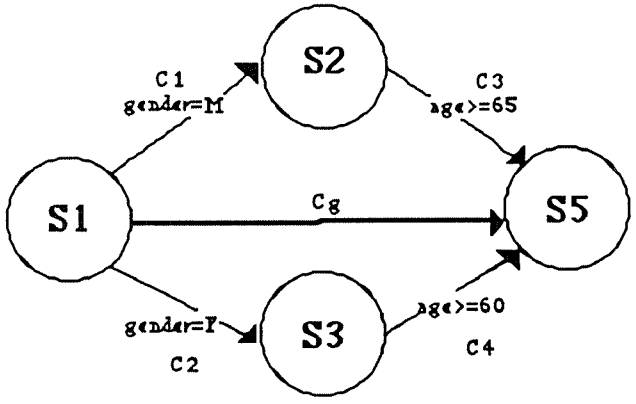



**fig. 3**

Net2:

S1 ---t1---> S2 ---t3---> S5
S1 ---t2---> S3 ---t4---> S5
S1 =======cg======> S5

The constraint sequences expression for this net is the following:

$$E[t] = (E[cg]\Delta(t1.t3)) + (E[cg]\Delta(t2.t4))$$

*LEVEL 3*
The constraint arc cg is refined by the following net :

Net3:

officename = Abingdon  count of claims =< 500
S1----------c5----------S4----------c6----------->S5

(i.e., there may be no more than 500 pension claims at a given office)

$$E[cg] = c5.c6$$

So, we can define

$$I[Net1]=((c5.c6)\ \Delta\ (t1.t3)) + ((c5.c6)\ \Delta\ (t2.t4)).$$

## 4 Concluding remarks and further work

We have presented an approach to system modelling which is based on the use of Transition Networks extended in order to represent transactions and constraints besides the usual state-transition concepts. We have also discussed the difference among the above concepts that we regard as important for system modelling activities, providing some motivations for the needs of extension of the basic formalism.

One criticism that can be made to the approach is concerned with the data modelling phase, that is totally missing in our presentation. The same nets of the example used in this paper looks much more like requirement specifications; no data model has been fixed to represent concepts such as pensioners and offices.

Indeed, in the Transition Network approach, data modelling is spread over the entire development. So, our approach to the use of transition networks for information system development is different with respect classical techniques. It is mainly concerned with modelling a body of resources necessary to specify, by successive refinements, the set of transformations allowed on them, together with local and global conditions on these transformations. This way, data refinement is viewed throught levels of details of transitions and constraints . An approach that reveals some analogies with this can be found in researchs on applying high-level Petri-Nets to system requirements specification [Peterson81][Ghezzi92].

However, beside proposing a revised formalism for information system modelling, our goal is to provide a suitable tool for the manipulation of this information. The technique relies on an information base where requirements, specifications and other systems engineering objects could be stored. That is, it assumes the existence of a kind of powerfull repository management system.

The nature of processing expected to be performed upon the representation (system models development, supported by verification and validation) and the success of a number of experiences in similar application domains [Bordiga 89][Hitchcock 91], is in favour of the employement of a logic-based repository. Furthermore, an expected feature of the technique is the congruent representation of state-transitions at any level of abstraction (or conversely, detail). Thus, a further advantage is gained if manipulation is supported by a deductive Database system.

Evaluation of constraints is a foundamental requirement for our system. One possiblity is that a constraint validation mechanism is given to the system together with the *evaluation* interpreter for processes, depending on the represented world (what are states, what are Constraints formuals etc). On the other hand, if this information is a formal specification (for example, a logic theory) and constraints are logic formulas that the same system can handle [Asirelli 85, Asirelli 88], than we have for granted an environment where the intended behaviour is performed.

The features that the system should offer and that we are nvestigating better are:

1) model development support: the use of a graphic interface to define transition networks, to allow for a direct and simple interaction with end-users (at the level of enterprise system modelling) and for model simulation purposes;

2) validation of system models: an approach which revealed fruitfull for systems validation is that of animation of a graphic representation (such approach is an usual practice in Petri-Nets communitiy [Tsalgatidou 90] but it is being supported in a number of new proposals).

3) verification: the arc representation (preconditions, process, postconditions) forms a tuple that can be stored in the database system. It is then possible to retrieve all references to an attribute in any constraint (and thus obtain all constraints affecting the attribute), or to walk networks (at an appropriate level of abstraction) with example input states for the purpose of verifying completeness of the expressed model.

The *Gedblog* system [Aquilino 92][Asirelli 92] is a deductive database management system with declarative graphic capabilities which matches some of the previous requirements and can be profitably used as a supporting tool for networks processing. Several activities are going on in parallel at present. The *Gedblog* graphic user interface is being used to handle the graphic primitives needed for the transition network representation and the structures needed to display the multiple levels of abstraction.

To start with, we have produced with *Gedblog* a first prototype of the graphic interface for Transition Networks construction . This can be used to define a Network graph at different levels of abstraction by direct manipulation of graphic objects such as nodes, arcs and sub-views. The following figure shows a snapshot of a session to define the net we used in the examples of this paper.

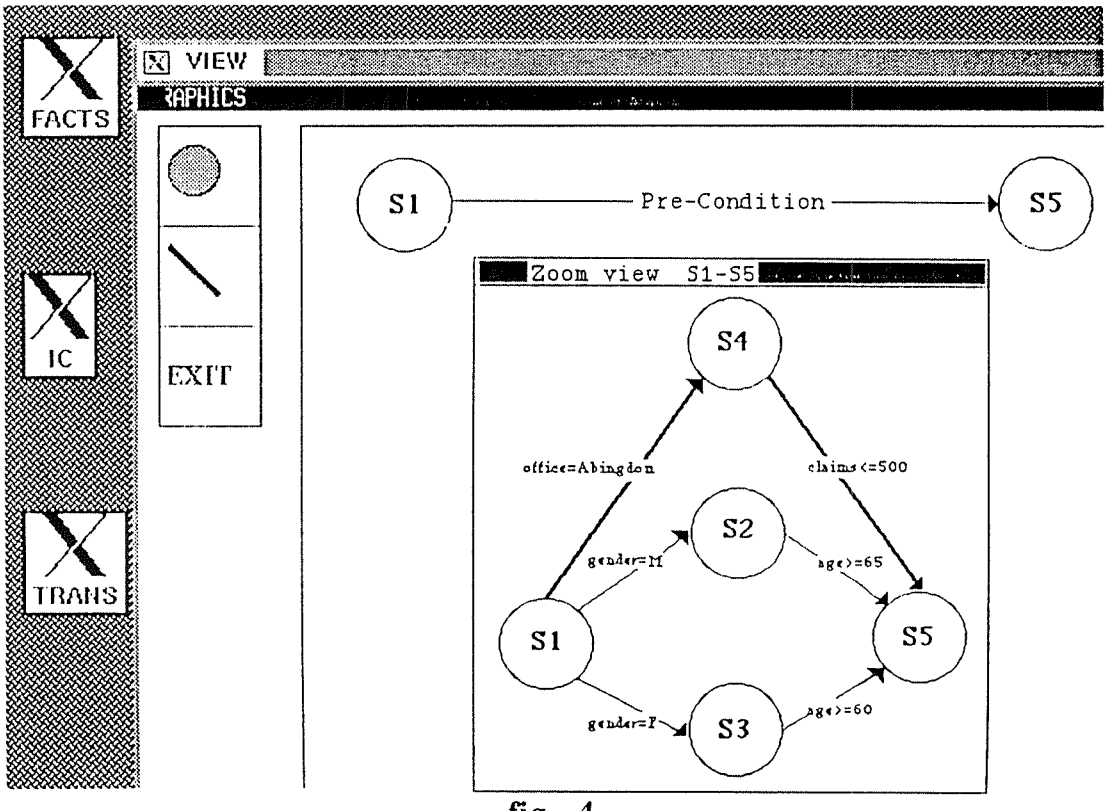



**fig. 4**

An important feature that the *Gedblog* system supports is the possibility to link (by using deductive rules) the graphic representation of visualized objects to relations of a different data model (see 3rd rule of the RULES window in fig. 5 below, which shows a snapshot of the logic database theory of transition networks). This can enable also unexpert users to define the theory corresponding to a network (possibly at a large-grained level of refinement) in a simple manner, by directly drawing arcs, nodes and short-sized textual information on the screen.

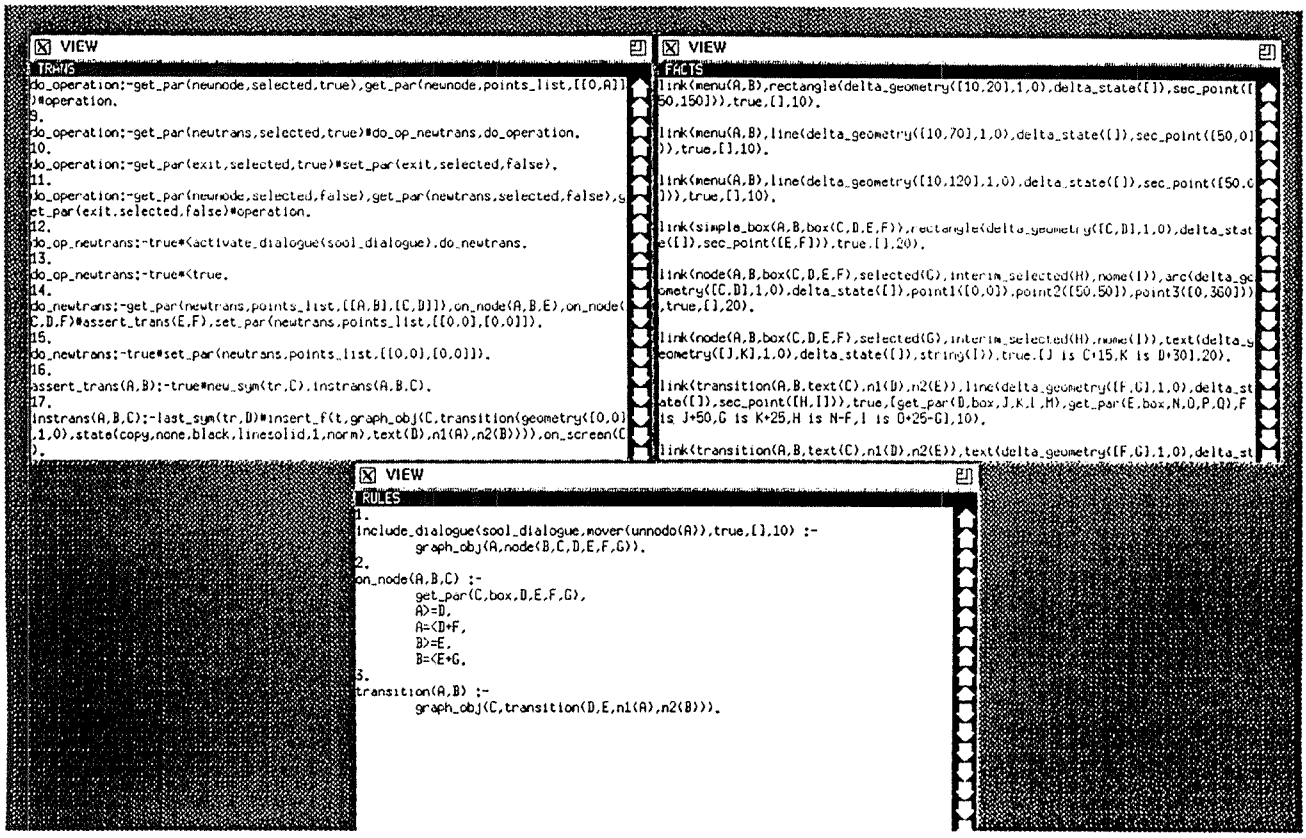



**fig. 5**

The next step we are actually taking concerns the possibility of simulating the traverse of a Network. This not only means providing suggestive visualization of the dynamics of graphic objects, but also to deal with "execution" of the operational semantics of arcs. The approach we are currently working on relies on the use of the deductive framework on which the *Gedblog* system is based to define a meta-interpreter for simulating the execution of constraints and transition of the networks. This job is semplified by the fact that *Gedblog* already includes support for transactions and constraints handling.

## References


[Aquilino 92] Aquilino, D , Asirelli, P., Inverardi, P.: 'Prototyping in the GEDBLOG System'; 4th International Conference on Software Engineering and Knowledge Engineering, June 1992.

[Asirelli 85]
P. Asirelli, M. De Santis, M. Martelli, Integrity Constraints in Logic Data Bases, Journal of Logic Programming, Vol. 2, No. 3, Ottobre 1985.

[Asirelli 88]
P. Asirelli, P. Inverardi, A. Mustaro, Improving Integrity Constraint Checking in Deductive Databases, 2nd International Conference on Database Theory, Bruges, Belgium, August/September 1988, Lecture Notes in Computer Science n. 326. Springer-Verlag, 1988.

[Asirelli 90] P. Asirelli, D. Di Grande, P. Inverardi, F. Nicodemi: Graphics by a Logic Data Base Management System, Progetto finalizzato Sistemi Informatici e Calcolo Parallelo, Internal Report.R/6/16 July 1990.

[Bertino 92] Bertino, E., Martelli, M., Montesi, D.: 'A Constraint logic rule based database'; 3th Ercim Database Research Group workshop, Pisa, September 1992.

[Bordiga 89] Borgida,A,Jarke,M,Mylopoulos,J, Schmidt,J W and Vassiliou,Y.: 'The Software Development Environment as a Knowledge Base Management System in Schmidt,J W and Thanos, C (Eds) 'Foundations of Knowledge Base Management' Springer-Verlag 1989.

[Ghezzi92] Ghezzi,C., Pezzè, M.: 'Cabernet: an Environment for the Specification and Verification of Real-Time System', Proceedings of 1992 DECUS Europe Symposium, Cannes (France), 1992.

[Hitchcock 91] Hitchcock,P 'Linkages Between Databases and Software Engineering' Proceedings BNCOD9 Butterworth Heinemann 1991.

[Jeffery 92] Jeffery, K.G.: 'Intelligent Support for Systems Development : The Notion and the Issues' Journal of Intelligent Information Systems (to appear 1992).

[Olle 82] Olle,T W,Sol,H G and Verrijn Stuart,A A (Eds): 'Information System Design Methodologies - A comparitive review' North-Holland, 1982.

[Orlowska 91] Orlowska,M E and Jeffery,K G.: 'A Functional Method of Data Processing Based on the Relational Algebra' Proceedings CAiSE91 Lecture Notes in Computer Science 498 Springer-Verlag 1991.

[Peterson81] Reisig,W.: 'Petri Nets', EATCS Monographs on Theoretical Computer Science Springer Verlag 1985.

[Reising 85] Peterson,J.: 'Petri Net Theory and the Modelling of Systems', Prentice-Hall, 1981.

[Saraswat 90] Saraswat, V.A. and Rinard, M.: 'Concurrent Constraint Programming'; Proceedings of 17th ACM Symphosium on Principles of programming Languages, San Fransisco, Jan 1990.

[Tsalgatidou 90] Tsalgatidou,A,Karakostas,V and Loucopoulos,P: 'Rule-Based Requirements Specification and Validation' Proceedings CAiSE90 Conference, Lecture Notes in Computer Science Number 436, Springer-Verlag, 1990.