Representing tutoring interaction as structured goal satisfaction

Achilles D. Kameas

Educational Software Development Laboratory,
Department of Mathematics,
University of Patras,
P.O. Box 1399,
Patras 26500,
Hellas

e-mail: kameas@math.upatras.gr

Keywords: formal models, human-computer interaction, computer-based tutoring systems

Abstract

This paper presents T-IMFG (Tutoring Interactive Multi Flow Graph), a formal model for the specification and design of interactive tutoring dialogue. T-IMFG has been developed in the context of XGENITOR, an ITS generator. The model integrates the solutions proposed in the past for the representation of the various components of an intelligent tutoring system (ITS), and provides a holistic modeling framework that can be used for the learner-centered description, specification and design of ITS. T-IMFG is a process-based state transition model that uses the basic Petri Net formalism augmented with features of cognitive and user models. A tutoring application is represented as a network of dialogue threads (micro tutoring dialogues), each of which teaches a specific concept. Students traverse the network, forming their learning trajectories, and engage in the author-permitted tutoring dialogues. The system then records student performance as an integral part of the entire system state. This approach enables ITS designers to multiple perspectives during ITS development, while permits a dynamic ITS execution during tutoring.

1. Introduction

The tutoring dialogue can be regarded as a special class of interactive dialogue with the purpose to transfer a volume of knowledge, procedural or declarative, to the students. Knowledge can be represented as a topic network of Units of Learning Material (ULM), which form the domain base (data or knowledge) of the training application [8].

ITS authors must not only define the ULM of the application and the topic associations, but have to describe the space of allowed student trajectories that traverse the topic network. This space can be regarded as the instructional strategy. During tutoring, the student modeling and diagnosis mechanism keeps track of the students' trajectories and overall performance [2].

Then, the tutoring dialogue must support the implementation of tutoring strategies (which represent the tutoring plans of the authors) through the learning trajectories selected by the students (which represent the learning plans of the students) which are executed within the user interface of the training application.

The problem that arises is that of consistency between the tutoring plans of the authors and the learning plans of the students. In the ideal case, the learning plans of the students must be a subset of the authors' tutoring plans. In real cases, the aim is to have the learning plans converge to some of the tutoring plans [10]. On the other hand, authors can anticipate students behavior to a coarse level of detail only. This problem can be regarded as a special case of the problems that arise during the design of interactive applications, which are a result of the incompatibility between the model of the application functionality formed by its designers and its users [12].

This paper proposes T-IMFG (Tutoring-Interactive Multi Flow Graph), a model specially designed for the specification of tutoring dialogue. T-IMFG is an extension of IMFG [5], a model already being used for the representation of structured general-purpose interactive dialogue, which has also been applied to the design and specification of interactive authoring and tutoring systems [15]. This model represents equally the interacting sides, namely student and tutor, the medium used to carry on the dialogue, namely the training application and the underlying computer system, and at the same time takes into account the special nature of the tutoring process.

Other approaches towards the specification of tutoring dialogue are briefly discussed in the next section. The formalism of T-IMFG is presented in section 3. Then, an example application of the model is given, followed by the conclusions of the work.

2. Models of tutoring dialogue

In the context of intelligent tutoring applications, the tutoring dialogue modeling problem has been tackled as part of the tutoring discourse specification and as a side-effect of solutions proposed independently for the representation of the teaching domain, the instructional strategy or the student model [2].

Discourse management schemes are based on a semantic network of topics which are interrelated through skills, prerequisites, analogies and other relationships (the reader is referred to [9] for an excellent although brief survey). Most of these models include cognitive science principles, such as the formation of tutoring goal-subgoal hierarchies and the temporary stacking of tutoring objects. Others adopt a task-based approach and use a procedural network to model actions that achieve a hierarchy of goals and objects associated with these actions.

The most general of these approaches is the Discourse Management Network [13] (and its recent extensions), which supports a top-down refinement of tutorial goals through the strategies and tactics that implement them. On the other hand, the ECAL system is one of the few that use dialogue description [2]. The underlying mechanism records the dialogue history, decides on the current focus and keeps track of the goals that remain unsatisfied.

However, none of these solutions addresses specifically the tutoring dialogue problem, although it is generally agreed that the role of the tutoring interface is paramount to the quality of learning [11]. With the advent of learner-centered design the need for a tutoring dialogue representation will become imminent [7]. Such a proposal should model both the state and the dynamics of the dialogue, and should include features of all the dialogue participants (the student, the application and the author). Current modeling trends follow the inverse approach: features of the dialogue are integrated into the instructional strategy or the student model [14].

T-IMFG attempts to model tutoring dialogue as student-ITS interaction; it encompasses the following concepts:

- the final tutoring objective is teaching a module of knowledge;
- the tutoring dialogue is the principal and most important tutoring medium;
- this dialogue is "constrained" by the user interface capabilities and the "contribution" of the various ITS modules (i.e. teaching domain module, instructional strategy module, student model etc); therefore,
- a tutoring dialogue model should be usable both as a description and specification tool.

The model is a "holistic" approach which represents tutoring dialogue as a network of micro-dialogues (or dialogue threads). Each micro dialogue is strictly structured, supports the teaching of a consistent and coherent part of knowledge and helps the students form a closure of the notions contained in it. The dialogue network, however, is not structured, but dialogue threads are associated through attributes that describe tutoring features such as content associations, student performance etc. This dialogue structuring is in accordance with contemporary theories of educational psychology, which, in general, adopt the process of progressive differentiation in learning: the most inclusive material is presented first; then, these concepts are progressively differentiated in terms of detail and specificity [1]

Then, tutoring interaction is modeled as a tutoring-state-space, within which the learners are "free" to navigate along the state sequences pre-defined by the authors (which are the tutoring strategy). Each state is described by the state of the user interface, the condition of the tutoring system modules, and the tutoring actions available to the learner.

3. T-IMFG: model definition and semantics

T-IMFG regards tutoring dialogue as composed of passive and active components, which produce, consume or transport simple data or control structures (tokens). Passive components are called links; they are the model semantics. The status of all links at a given time represents the overall system state. Active components are called actors and they represent either the user interface operations or the actual processes performed by the tutoring application (i.e. the display of a ULM). Actors are the model structure; they correspond to integral student or tutoring plans which can be achieved by sequences of student- or system-initiated tutoring actions that take place in a well-defined tutoring context. An actor can be activated by events and its firing modifies the system state according to a set of actor behavior rules.

A T-IMFG can be reduced to a directed, bipartite graph (A, L, E) where:

- A is the set of actors
- L is the set of links
- $A \cap L = \varnothing$ and $A \cup L = \varnothing$
- $E \subseteq (A \times L) \cup (L \times A)$ is the set of directed arcs that connect actors to links or links to actors.

3.1 Actors

Formally, an actor $a \in A$ is defined as a 5-tuple *(SIL(a), SOL(a), R(a), FUN(a), TYPE(a))*, where:

- *SIL(a)* is the Set of Input Links $(\forall a \in A, SIL(a) \subset L)$
- *SOL(a)* is the Set of Output Links $(\forall a \in A, SOL(a) \subset L)$
- *R(a)* is the set of Rules that describe the behavior of the actor $(R(a) = r \mid r \equiv PRE(r) \rightarrow POST(r),$ *where* $\forall l \in PRE(r), l \in SIL(a)$ *and* $\forall l \in POST(r), l \in SOL(a)\}$
- *FUN(a)* is a function which represents the computation performed by the actor and
- *TYPE(a)* is a descriptor of the internal actor complexity and role *(TYPE(a) ∈{Context, Action, Library, Group})*

Each rule consists of a left-hand-side that describes the prerequisites for rule firing and of a right-hand-side that defines the result of that firing. Both rule sides are subsets of the actor links. Then $SIL(a) = \{ l \mid l \in \cup PRE(r), \forall r \in R(a)\}$ *and* $SOL(a) = \{ l \mid l \in \cup POST(r), \forall r \in R(a)\}$

T-IMFG represents explicitly the tasks students may execute within a training application, as well as their strategies in order to achieve tutoring goals. In addition, it gives a clear description of the training context into which student actions take place.

The semantics of actor types are as following:

- *context* T-IMFG actors roughly correspond to the pedagogic states of other models, and from students' perspective represent the tutoring goals that must be achieved. Each context actor is refined, according to the tutoring plan, into sub-contexts, which in turn represent tutoring subgoals that must be reached by the students. As students reach successive tutoring subgoals, they form their learning plan
- *action* T-IMFG actors (also called closure actors) represent the "closure points" of tutoring goals. They signify the end of a tutoring plan, help students form a closure of the knowledge taught with the specific plan, and provide an exit point of the tutoring dialogue thread
- *library* T-IMFG actors are used for refinement or synthesis of tutoring plans. Although only AND and OR refinement is explicitly represented, all other synthesis procedures (sequence, loop, decision etc) can be represented with the correct usage of links
- *group:* they form a graphical grouping of actors, without any computational or cognitive significance

3.2 Links

Formally, a link l can be defined as a 5-tuple *(SPA(l), SCA(l), CLASS(l), USE(l), CHECK(l))*, where:

- *SPA(l)* is the Set of Producing Actors; these actors supply the link with tokens $(SPA(l) = \{a \mid l \in POST(a), \forall a \in A$ *and* $SPA(l) \subset A)$
- *SCA(l)* is the Set of Consuming Actors $(SCA(l) = \{a \mid l \in PRE(a), \forall a \in A$ *and* $SCA(l) \subset A)$
- *CLASS(l)* is an attribute that describes the class (type) of the link which defines the type of tokens accepted by the link *(CLASS(l) ∈ {Context, Event, Condition, Data})*
- *USE(l)* is an attribute that describes the usage that the link permits for its tokens *(USE(l) ∈ {Normal, OK, Debit, Read-only})*
- *CHECK(l)* is a condition that describes the limits of acceptable values for the tokens used by the link

The definition of links distinguishes among the type of tokens they store and the way links are used by the model. This permits system design from different, complementary perspectives. T-IMFG links are typed so that the different information flows that occur in an interactive tutoring application are distinguished. The following link types are supported:

- *context*, which represent the context within which the training interaction occurs. It is decomposed into:
  - *tutoring context*, which represent the hierarchy of the tutoring goals set by the authors of the training application, and
  - *interaction context*, which represent the context of user actions that have no tutoring significance,
- *event*, which cause a system state transition and represent a learning or a user interface action by the student, or a tutoring action at the initiative of the training application. Usually, an event will initiate a tutoring dialogue thread and eventually cause the presentation of a block of knowledge, either selected by the student, or by the tutoring system (in the case of an Intelligent Tutoring System),
- *condition*, which represent global system attributes (i.e. represent availability of actors, priority of execution, etc). Conditions describe whether the system is ready to process a user action that will lead to the achievement of a subgoal. In T-IMFG, four classes of condition links make up this type:
  - *student*, which represent those attributes that relate to student performance only
  - *domain*, which represent characteristics of the content of the training subject
  - *instructional*, which represent the tutoring strategy
  - *general*, which contain all other links
- *data*, which represent the data exchanged among system components and component modules

3.3 Refinement and abstraction

Since T-IMFG is context-oriented, context-based refinement of actors is supported. The meaning of actor refinement is equivalent to the decomposition of user goals into a plan of ordered simpler goals. The refinement of a context actor may initiate a new tutoring dialogue thread. T-IMFG supports two kinds of plan refinement:

- *AND refinement:* a higher level goal is refined into a set of lower level goals, all of which have to be achieved, in order for the plan to be complete. Such a refinement scheme starts with library actor AND-S and ends with AND-E.
- *OR refinement:* a higher level goal is refined into a set of lower level goals, at least one of which has to be achieved in order for the plan to be complete. Such a refinement scheme starts with library actor OR-S and ends with OR-E.

AND refinement does not place, in principle, any satisfaction priorities on the lower level plans. If designers wish to place such priorities, they will have to use conditions. Similarly, OR refinement does not specify any particular lower level goal that has to be satisfied; instead, the achievement of any lower level goal (that is, the successful firing of any lower level actor) is enough to complete the plan. The designers can again use priorities to describe any preference in goal satisfaction.

A set of links of the same type can be abstracted into a typed complex link, for graphical purpose.

3.4 Representation of state

The state of the dialogue thread modeled by each T-IMFG is represented by the current marking *M*, which is an ordered tuple that assigns to each link a number (possibly zero) of tokens. A marking is changed as a consequence of actor execution (firing): each firing changes the distribution of tokens and produces a new marking *M'*. The Petri Net firing rule is incorporated in T-IMFG, as follows:

1. an actor a is said to be enabled if there exists a rule $R_I \in R(a)$ which has an input context link $g_I \in PRE(r_I)$ that contains a token (that is, if one of the actor rules belongs to the currently active tutoring context)
2. an enabled actor may fire only if the input event link $e_I \in PRE(r_I)$ of this rule contains a token (that is, if the triggering event takes place) and if the other input condition and data links of the left-hand-side of the rule (if any) contain tokens
3. an actor firing removes tokens from all the links l, $l \in SIL(a)$ that belong in the $PRE(r_I)$ and places tokens to all the links l', $l' \in SOL(a)$ that belong in the $POST(r_I)$

This firing rule implies that all actors belong to some tutoring or interaction context (that is, no student learning goal is tutoring context-independent), and that all state transitions are caused by a system or student event.

Consequently, each student action takes place in a known context and leads to the achievement of a defined tutoring goal; the students' actions make up the student learning plan, which takes them from the beginning to the exit point of a micro dialogue. The availability of a user action is determined by the tutoring dialogue network and by the previous student actions within the dialogue threads.

The use of this rule, however, has proved too strict for the modeling of interactive tutoring systems. Thus, the following modifications have been introduced:

1. a token in the appropriate input event link of a ready-to-fire actor is enough to trigger actor firing. That input link of an actor, which in a way "owes" a token is called a "debit link". Since this is a link usage property, a link may at times appear as of normal or debit usage. The link is responsible for eventually getting the missing tokens
2. in some cases, not all tokens may be consumed by actor firing. A link that preserves its tokens after firing is marked as "read-only" and is equivalent with a normal link that appears both as input and output to the same actor.
3. each actor has at least two output links of type context and event and usage OK. Link context(OK) marks the achievement of the goal represented by the actor, while link event(OK) marks the successful processing of the event that caused actor firing. The appearance of tokens in *both* these links marks successful actor firing. However, actor execution terminates if a token appears at least in one of these links.

A complete tutoring system can be represented as a set of T-IMFGs. Then, global system state can be regarded as the union of all local states (it is an ordered tuple of tuples, where each inner tuple is the marking of a T-IMFG that represents a dialogue thread:

$M \ = (M_1, M_2, \ldots, M_n)$.

Each actor firing modifies only one $M_i \ \in M$ to produce M'. Each M_i represents the "locus of control" of the corresponding T-IMFG. Thus, multiple loci of control are maintained by the system, enabling the students to learn within multiple active tutoring contexts.

3.5 The actor-ready list

An actor becomes ready-to-fire only when one of its rules belongs to an active tutoring or interaction context (that is, its input context link contains a token). All such actors are placed in the actor-ready list. Only actors in this list can eventually fire. Placement of actors in the list is context-based. When an event occurs, at first the actors of the current tutoring and interaction contexts are checked. If no one among them recognizes the event, the rest of the list is checked. Furthermore, the actor-ready list contains actors of all the active T-IMFGs of the system on a most-recently used order.

The actor-ready list represents tutoring dialogue control. At any moment, it contains all the tutoring goals that can be achieved with the next event. When an actor fires, the list contents change in a way that depends on the type of the actor. Firing of a context actor usually adds its constituent actors to the list, with or without removing the initial actor. This is a result of the fact that in order to reach a tutoring a goal, a student must first achieve the subgoal structure of the goal. On the contrary, firing of an action actor usually removes a set of actors from the list, since execution of such an actor usually marks the closure of a tutoring dialogue thread, and thus, the entire structure is removed from the list. In this way, termination errors are avoided.

The actor-ready list corresponds to the user goal stack: most cognitive models assume that users stack their immediate goals in the Short Term Memory. In the case of T-IMFG, however, a list instead of a stack is maintained, in order to better represent the asynchronous nature of the learning process.

Furthermore, the actor-ready list is used for model synchronization. Each T-IMFG may manage its own actors, but the existence of a common list ensures that all events will be processed sequentially. The actor-ready list is managed by the library actors, which add to or remove from it actors depending on the goal structure they represent.

This list is an alternative representation of system state: current state can be represented with all the learning goals that students can achieve (cognitive-oriented definition). Each goal can be achieved with a student action. Consequently, this list represents the actions that are available to the students in order to achieve their goals (learning task-oriented definition). The combination of these two definitions gives a causality perspective: at any moment, it can be checked which are the achievable tutoring goals and by using which learning plan they can be achieved. Of course, reachability of goals or availability of actions will change with the next student action that will cause an actor firing.

4. T-IMFG as a design tool

T-IMFG is the modeling platform upon which the XGENITOR system is being designed. XGENITOR is the upgrade of GENITOR [6], an authoring platform designed and developed at the ESD*Lab (IMFG, the model upon which T-IMFG is based, had been used in GENITOR [15]). The main axis of system evolution are:

- tutoring interaction is considered as a set of tutoring services offered by tutoring agents (called tutoring agencies)
- tutoring agents are, in effect, tutoring micro-tools, which behave and interact autonomously, but exhibit social properties, such as cooperation, competence, etc

Then, T-IMFG is used for the representation of tutoring agencies (as a synthesis of tutoring dialogue threads), while a model with compatible formalism is used for the representation of agent behavior (their plans). Then, a pattern matching process is used to trace the possible dialogue paths and to interpret the learners' actions.

4.1 A dialogue modeling example

According to the state of the tutoring system, the students may engage in one of the available tutoring micro dialogues. The availability of a micro dialogue is determined by the training application. When the students enter a micro dialogue, they have to reach its exit point, in order to obtain the related knowledge. A micro-dialogue can or cannot be pre-emptive, depending on the authors' tutoring principles.

In order to describe an interactive tutoring application, authors have to build the micro dialogue network, which is isomorphic to a high-level network of the domain topics. Then each node of the network is refined into a sequence of T-IMFG levels; each level is the refinement of a context actor and represents the tutoring goals that must be reached by the students. At the end of a refined network, authors place the action actors, which define the topic closure points. Authors have to describe both the structure of each tutoring micro-dialogue by using the T-IMFG graphical notation, and the semantics of each model component (actor or link), by filling the slots of the corresponding record.

The graphical formalism of the model contains thick-line rectangles that represent context actors, plain-line rectangles for action actors, round-edged boxes for library actors, dotted rectangles for group actors, thick line circles for action links, plain line circles for context links, diamonds for condition links and pentagons for data links; the horizontal dotted line on a link means that the link is read-only, and the V sign means that this is an OK link. The ANDS-ANDE refinement structure means that *all* constituent actors must execute successfully, in order for the represented micro-dialogue to terminate successfully.

In this section, an example dialogue thread is represented using T-IMFG. This dialogue is part of a tutoring application called "TeachReal" [4], which teaches the Yourdon Structured Methodology for Real-Time Systems Development (YSM). TeachReal has been used as a testbed during T-IMFG development.

YSM, as taught by TeachReal, is a structured methodology that consists of four sequential phases: Structured Analysis, Processor Environment Modeling, Software Environment Modeling and Code Organization Modeling. Each phase contains activities, which must be carried out in a certain order in order to complete the phase. In order to help the trainees best acquire YSM, TeachReal makes them execute the steps of the methodology within a simulated environment; an expert system is then used to check and correct their actions. The trainees may access a library of topics that relate to the phases of the methodology. At the topmost level, the micro-dialogues that support access to these topics are structured using 6 context actors (figure 1): CA1 represents a short Introduction to the subject, CA2-CA5 represent the four phases of YSM, respectively, and CA6 represents a set of tests on the entire YSM, used to verify beyond any doubt the trainees' competence.

All the six actors belong to context C1 (denoted with an appropriate input context link), which represents the TeachReal application. When the trainees execute the application, input link C1 receives a token, and all these six context actors are placed in the actor-ready list, meaning that they may visit any set of topics they wish. Note that the trainees have to access the topics of all six phases, before TeachReal can declare their training as "completed" (this is represented by using a AND refinement scheme). The trainees may initiate each micro-dialogue an unlimited number of times, as denoted by the read-only input context link in actors CA1-CA5. However, once they enter the micro-dialogue represented by actor CA6, they cannot return (this dialogue is pre-emptive).

In order to initiate a micro-dialogue, the trainees have to cause the appropriate event (denoted by input user event links Ua1-Ua6). For example, if they wanted to access the topics that relate to the Structured Analysis phase, they would have to cause event Ua2. Then a token would appear on the input event link Ua2 of actor CA2, causing this actor to fire according to its template (figure 3). The token of link Ua2 would be consumed, but the token of link C1 of this actor would remain intact, because this link is read-only. Note that at this design phase, the notion of user event is used

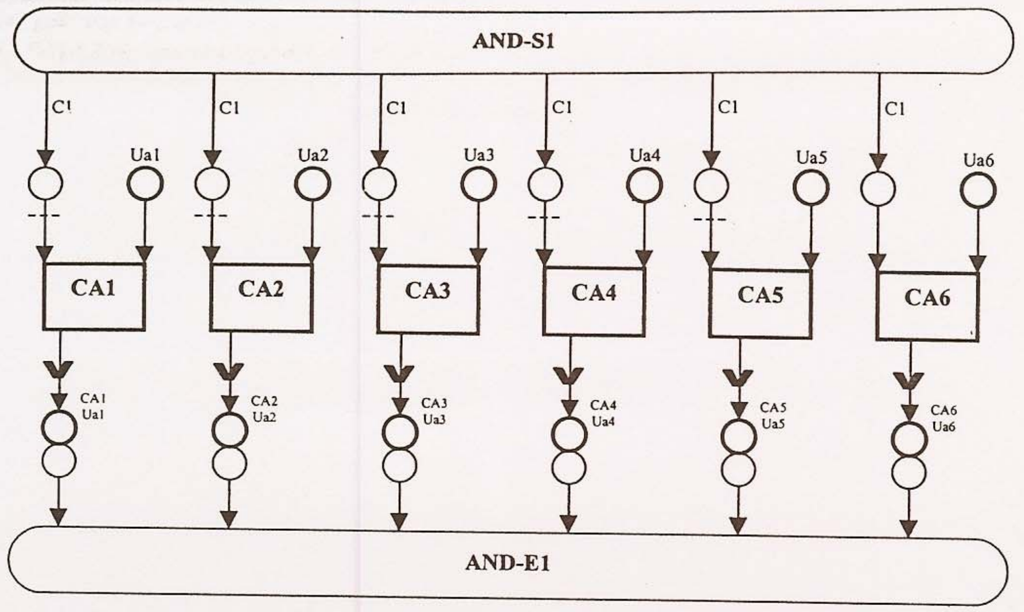

Figure 1. The T-IMFG diagram of the topmost level of TeachReal

in an abstract way; in later phases, events may become concrete (for example, event Ua2 could be "click on item Structured Analysis of the Phase menu", if the application user interface was to contain a menu with all YSM phase-related topics).

When a micro-dialogue (or one of its threads) is terminated, a token is produced in the output context and event links (these links are of OK use, as denoted by the V mark): a token in the output context link means that the corresponding tutoring context has closed successfully, while a token in the output event link means that the corresponding event has been processed successfully.

When CA2 fires, the current context changes from C1 to C1;CA1 and the constituent actors of CA1 (CA7-CA10) are placed in the beginning of the actor-ready list. While inside the phase of Structured Analysis (figure 2), the trainees must learn how to Appreciate System Requirements (context actor CA7), how to Build the Essential Environmental Model (context actor CA8), how to Generate the Essential Behavioral Model (context actor CA9) and how to Complete the Project Dictionary (context actor CA10). Each of these context actors represents a thread of the tutoring micro-dialogue; none of these threads is pre-emptive, as represented by the read-only input context link. However, in order to learn how to Generate the Essential Behavioral Model or how to Complete the Project Dictionary, the trainees must already have learned about Appreciating System Requirements and Building the Essential Environmental Model. Technically, actors CA9 and CA10 cannot fire unless domain condition links DCon1 and DCon2 contain tokens; this may happen only after actors CA7 and CA8, respectively, execute successfully. Again, the trainees may choose to fire any of these context actors by causing the appropriate user event (Ua7 – Ua10, for actors CA7 – CA10, respectively). In this case however, the system itself may choose to present to the trainees a set of topics by causing an actor firing via system event links Sa1 – Sa4; such a system initiative could well be based on the student condition links SCon1 – SCon4.

If they chose to learn how to Build the Essential Behavioral Model (that is cause the firing of actor CA8), then they could continue (figure 4) with choosing to learn about The Event List (context actor CA11), or The Context Diagram (context actor CA12 – the one shown as an OR refinement scheme). In this case, actors CA11 and CA12 would also be added to the actor-ready list, while CA8 would become the active context. Then, if the trainees wanted to learn about the Context Diagram, actor CA12 would fire.

Learning about the Context Diagram means to learn about its Components (action actor A1), learn how to Build and Check it (action actor A2); examples could be shown if required (action actor A3); self-evaluation tests could also

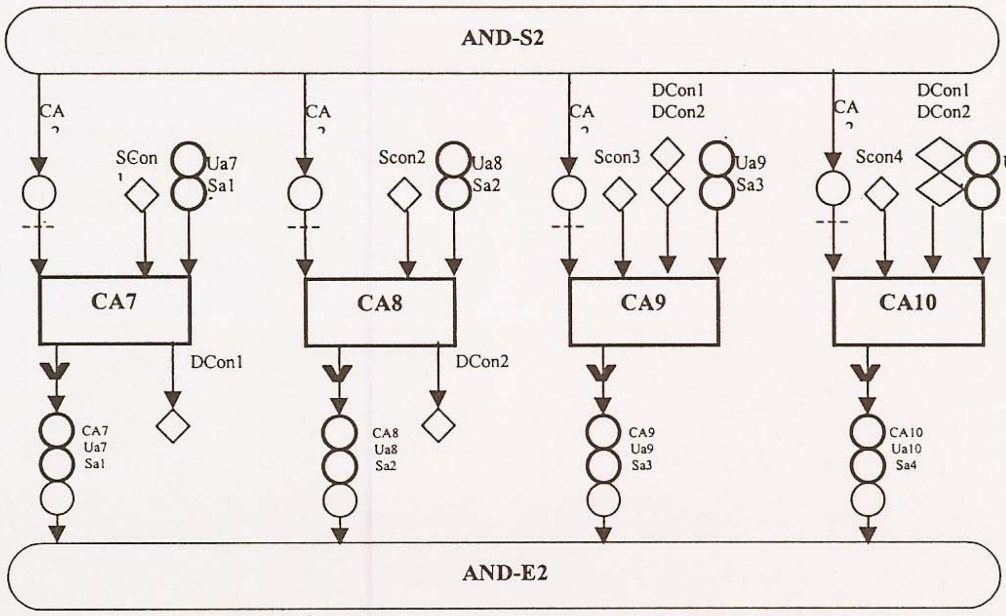

Figure 2. The T-IMFG diagram of the topmost level of the micro-dialogue that teaches the Structured Analysis phase of YSM

be taken (action actor A4), if one dared to; all these actors would be added to the actor-ready list. The trainees may choose any of these topics; the system may present examples or ask them to take tests (system action links Sa7 and Sa8), depending on the degree of satisfaction of the tutoring goals (tutoring context links T1 and T2). Input student condition links could also be added if the trainees' level of competence was important, as well.

Suppose that the trainees choose to view some examples (that is, cause event Ua15 within context CA12). Then, actor A3 would fire; the examples would be presented; tokens would be produced at the actor output links (A3(OK) and Ua15(OK)). This would cause the firing of actor OR-E2, which would remove actors A1-A4 from the actor-ready list and produce tokens in the output links of CA12, making context CA8 active again. If the trainees had executed actor CA11 as well (note, however, that no precedence constraints are placed on the relative execution of these actors), then actor AND-E3 would fire. Then, actors CA11 and CA12 would be removed from the actor-ready list, tokens would be produced in the output links of actor CA8 and CA2 would become the active context.

In the same manner, when actors CA7-CA10 have all fired, the dialogue thread that teaches the Structured Analysis phase of YSM terminates, as represented by the AND refinement scheme used.

NAME	CA2
DESCRIPTION	Micro-dialogue for teaching Structured Analysis phase
INPUT LINKS	C1, Ua2
OUTPUT LINKS	CA2, Ua2
RULES	C1(RO), Ua2 $\rightarrow$ CA2(OK), Ua2(OK)
FUNCTION	{call the code segment that displays the actor window}
TYPE	CONTEXT

Figure 3. The record that describes actor CA2

4.2 Design aspects

Authors can use T-IMFG as a design tool that enables them to plan and design a tutoring application by adopting a learner-centered approach, since the model represents equally well the learning and the tutoring plans. In fact, the incompatibility between these two classes of plans is resolved by forcing authors design a tutoring application through the learning plans that will be made available to its users.

Since T-IMFG treats each link type in a different way, as indicated by the various types of usage assigned to each link type, authors can address several perspectives of a tutoring application, such as:

- *tutoring planning:* represented as the actor decomposition scheme based on tutoring context links
- *student goal-subgoal structure:* this perspective is a result of the cognitive features included in the model, and uses context actors and goal links to represent the structure of goals and subgoals that students may achieve
- *learning task analysis:* complements cognitive user representation by using user action links to represent all the operations available to students
- *causation:* combines the previous perspectives (that is, it uses context actors, context and event links) to represent the causes of system state transitions
- *event flow:* a necessary perspective when asynchronous processes, as is the case with human-computer dialogue, have to be represented; it uses user and system action links to represent all the events that may take place in the system.
- *student model update:* by using student condition links, in combination with event and context links and the actor refinement scheme, students' performance can be recorded and used as input to an overlay or bug-catalogue diagnosis module
- *instructional strategy:* by correctly placing instructional condition links, authors may describe the instructional strategy. During tutoring, the system can dynamically apply it in combination with context and event links
- *data flow:* data links are used to represent data items that are exchanged among actors

T-IMFG is a graphical model that can be used not only for the description of a tutoring dialogue, but also for the specification of a tutoring system through this description. Then, this learner-centered design approach can lead to tutoring-dialogue-based prototyping of tutoring applications, as is the case with XGENITOR (for example, T-IMFG is an interface-independent model, since it uses the abstract notion of user event; thus, authors can develop and test different tutoring interfaces for the same tutoring dialogue).

T-IMFG is a formal model that can, in the same way with IMFG, be refined into a Petri Net model [5]. Then, designers can use the formal analysis methods that apply to Petri Net models, in order to formally verify their design. These methods could be used for reachability analysis (which tutoring goals are achievable from a certain system state), liveness testing (whether a tutoring goal can ever be achieved), etc.

5. Conclusions

The T-IMFG model presented in this paper may be used both for the specification of tutoring interaction and for the prototyping of an interactive tutoring application. In addition, tutoring functions are distinguished from each other (i.e. domain, instructional, student modeling) while they are separated from pure computation functions. Finally, the model supports actor design at multiple, hierarchical levels of abstraction.

The tutoring dialogue takes place within a tutoring system, which consists of the tutors as they are represented in the training application, its user interface and the students. This holistic view permits the definition of a closed model, where all events are internal and state transitions are caused by the model components in a deterministic way. Although some model components (authors, students) are non-deterministic by nature, they are constrained in exhibiting deterministic behavior, since they have to interact through the user interface of the application, which is deterministic.

The dual nature of T-IMFG makes it capable of supporting the entire life cycle of the development of an intelligent tutoring application, leading to the development of learner-centered ITS; T-IMFG can be used equally well for the design and analysis of an ITS. However, due to the heavy semantics used in the model, the resulting Petri Net model will be considerably extended; thus, a manual analysis of the net could be tedious. Thus, in the context of XGENITOR, an integrated tool that will automate the entire life-cycle of ITS development based on various adaptations of the T-IMFG model will be developed. Then, a straightforward ITS prototyping mechanism could be implemented.

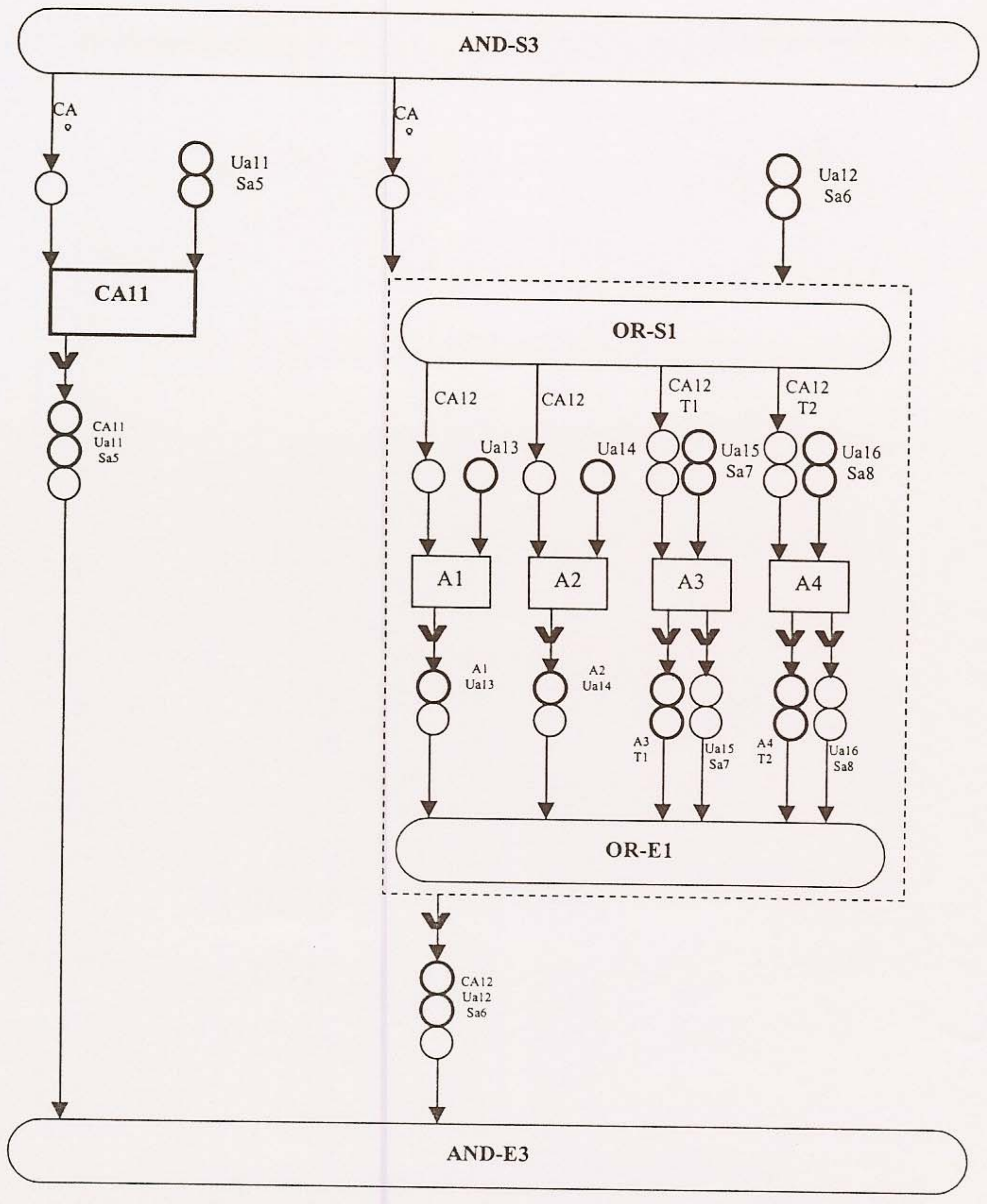

Figure 4. The T-IMFG diagram of an inner level of the micro-dialogue that teaches the Structured Analysis phase of YSM (that of teaching how to Build the Essential Behavioral Model)

References

[1] D. P. Ausubel, "Educational psychology: a cognitive view". Holt, Rinehart & Winston, New York, 1978.

[2] M. T. Elsom-Cook, C. O'Malley, "ECAL: bridging the Gap between CAL and ITS". Computers in education, 15(1-3), pp 69-81, 1990.

[3] V. Gerogiannis, D. Giakovis, A. Kameas and P. Pintelas, "Intelligent systems in education". Proceedings of the 1st Hellenic Conference on the Didactics of Mathematics and on Informatics in Education, October 20-23, Ioannina, Hellas, pp 231-252, 1993.

[4] V. Gerogiannis, A. Kameas and P. Pintelas, "An intelligent tutoring system for the Yourdon Real-Time Systems development methodology". Proceedings of the 1st International Conference on Computers and Advanced Technology in Education (CATE96), Cairo, Egypt, March 18-20, 1996, pp 287-299.

[5] A. Kameas, "A formal model for the specification of interactive dialogue and the design of interactive applications". PhD Thesis, Dept. of Computer Engineering & Informatics, Univ. of Patras, 1995.

[6] A. Kameas and P. Pintelas, "The functional architecture and interaction model of a GENerator of Intelligent TutORing applications". Journal of Systems and Software, to appear in 1997.

[7] D. Norman and J. Spohrer (eds), "Learner-centered Education". Communications of the ACM-special issue, 39(4), 1996.

[8] C. Olimpo, A. Choiccariello, M. Tavella and G. Trentin, "On the concept of reusability in educational design". In Learning Technology in the European Communities (A. Cerri and J. Whiting, eds), pp 535-549, 1992.

[9] J. W. Rickel, "Intelligent Computer-Aided Instruction: a survey organized around system components". IEEE trans. on Systems, Man and Cybernetics, 19(1), pp 40-57, 1989.

[10] R. Schank and A. Kaas, "A goal-based scenario for high-school students". Communications of the ACM, 39(4), pp28-29, 1996.

[11] E. Soloway, M. Guzdial and K. H. Hay, "Learner-centered design: the challenge of HCI in the 21st century". ACM Interactions, 1(2), pp 36-48, 1994.

[12] H. Thimbleby, "User Interface Design". ACM Press, 1990.

[13] B. Woolf, "Theoretical frontiers in building a machine tutor". In Artificial Intelligence and Instruction (G. Kearsley, ed), Addison-Wesley, pp.229-267, 1987.

[14] B. Woolf, "Intelligent Multimedia Tutoring Systems". Communications of the ACM, 39(4), pp30-31, 1996.

[15] I. Zaharakis, C. Diplas, A. Kameas and P. Pintelas, "Specifying the interaction with an Expert System Shell using IMFG". Proceedings of the 5th Hellenic Conference on Informatics, Athens, December 7-9, 1995, pp 601-613.