Risk and Decision Analysis 1 (2009) 85–101
DOI 10.3233/RDA-2009-0008
IOS Press

ETS as a structural language for decision modeling and analysis: Planning, anticipation and monitoring *

Lev Goldfarb, Ian Scrimger and B. Reuben Peter-Paul
Inductive Information Systems, 29 Pembroke Crescent, Fredericton, NB, Canada E3B 2V1
E-mail: lev.goldfarb@gmail.com

Abstract. Our main objective is to propose a recently-developed representational formalism, the Evolving Transformation System (ETS), as a general *structural* tool for decision and risk analysis, as opposed to the conventional numeric tools. We believe ETS to be the first formalism developed specifically with the goal of (properly understood) *structural representation* in mind.

We outline the use of ETS in the representation of an "insider's view" of a hypothetical terrorist plot. The example should be treated as that of an internal view of a *generic planning process*. We emphasize that the same tools can be used for modeling the external view of a decision process (and its execution).

The "atomic" representational unit of ETS is a *structured event*, and sequences of such events form *processes* that represent real human (including business) activities consisting of a series of various human actions. Each such action can be viewed structurally as an *event transforming* several "*incoming*" information processes into some "*outgoing*" information processes. Once the ETS concept of a class of (similar) processes is introduced, any process from that class can then be viewed as composed, in a class-specific manner, out of some simpler processes belonging to the constituent classes. Thus, a purchase process is, on the one hand, an element of the class of structurally similar purchase processes, while on the other hand, is a building-block of a larger business process, structurally fitting into it in a modular manner. Moreover, the concept of class in the formalism is introduced in such a way as to allow one – having constructed a class description, or, more formally, class representation – to *predict/anticipate* the overall structure of any process from that class.

It should become clear that the proposed framework could easily be adapted to address a broad range of needs, including decision modeling and analysis, anticipation of possible outcomes, various kinds of monitoring including surveillance, etc.

Keywords: Structural representation, decision modeling and analysis, planning, anticipation and monitoring

1. Introduction

The development of the Evolving Transformation System (ETS) [1] was motivated by various problems arising in pattern recognition, machine learning and artificial intelligence in general, and is the culmination of a quarter-century work. The radical novelty of this approach relates to the proposed idea of structural representation. The basis of this idea is a far-reaching generalization of the most basic mathematical concept: the Peano process for constructing natural numbers. In this generalization, the *single* and structurally very simple successor operation in that process (see Fig. 8(a)) is replaced by *several structurally more complex* operations, each one capturing the structure of a real event (see Section 3).

The central point to keep in mind is that within the ETS formalism, *objects are viewed and represented as structured processes*. This, *for the first time*, introduces into mathematics a representation that captures both temporal and structural features of real processes/objects. We want to emphasize the point that – despite considerable efforts expanded in many scientific areas over the second half of the last century – we are not aware of any formalism which would address the concept of structural representation at a comparable level of both generality (in the formal sense) and universal applicability.

Since the opportunity to present an application of

*The previous version of the paper was presented at 2007 Decision and Risk Analysis Conference, University of Texas at Dallas, May 21–22, 2007.

ETS in this conference has arisen, we decided to test the flexibility of the formalism as applied to this area, completely new to us. We mention this in order to emphasize that whatever insights emerge from this work, they should be attributed not to our expertise but rather to the structure of the formalism itself and its potential utility to this area. In particular, we believe that ETS provides a very convenient language for dealing with various decision and planning processes, including the monitoring of their execution. Again, that we, as novices, were able to generate the example we present in a very short time may testify to the convenience of ETS as an applied tool for this field.

We chose to model an "insider's view" of a hypothetical terrorist operation "Dirty Bomb", i.e., as it is viewed internally, from the perspective of the terrorists. The reasons for the choice are twofold: first, the operation itself can be treated as a generic business process (materials are purchased, labour is hired, management tracks progress, etc.), and second, this operation can also be approached from the surveillance point of view, when only some events can be observed while others should be inferred on the basis of a similar scenario constructed by the surveillance agency.

In the case of a generic business plan one can use the proposed structural representation as a tool to improve or even completely redesign the particular business model: various *modules in a business process are now treated as elements of a class, which has its own formal description*, with the implication that, for each of the above modules, many variations can now be *automatically* generated and plugged into the global structure to generate multiple versions of the entire process. These multiple versions can then, in particular, be risk-assessed. Again, we draw attention to the fundamental but typically overlooked fact that each processes is a member of a class of similar processes, i.e., processes of similar structure.

In the case of surveillance, our structural representation allows one to automate a more reliable recovery of the unobserved elements of a scenario: based on the observed principle events as well as some observed "surrounding" events, one can now tap into the previously constructed representations of classes of macro-events and optimally choose one of them that can stand for the fully recovered process. Obviously, such a structural tool should also allow one to make the process of information collection much less intrusive.

In general, it is important to note that the use of temporal object representation (as a process) carries with it predictive, or *anticipatory* [2], benefits: having observed only a part of a process and having the description of several classes of processes to one of which the object is expected to belong, one can now attempt to classify it and hence to anticipate what that object is. In connection with the *concept of class*, it is useful to keep in mind that that concept is pervasive at all levels of ETS formalism.

In summary, compared to numeric formalisms, the ETS – as the first classification-oriented formalism for structural representation – offers fundamentally different capabilities for dealing with classes of processes (and hence with processes themselves). Within the formalism, classes, *in contrast to numeric formalisms*, have generative descriptions, which implies that the unobserved class processes can now actually be constructed based on such descriptions.

2. Operation "Dirty Bomb"

In this section we present a high-level insider's view of a simplified scenario for a hypothetical terrorist operation "Dirty Bomb" or DB for short. As we mentioned in the introduction, the DB operation can also be viewed as an example of generic business process involving many other modular subprocesses (see Figs 1–4).

As far as the above four figures are concerned, we note first that each process in them is delineated as a closed shape, without showing the events comprising it: the events will be introduced in the next section and cannot be shown within such a confined space, besides their depiction would have enormously complicated the figures.

Second, the three modules shown in Fig. 1 should be viewed as elements of some *classes of processes*, as should each of the processes contained within these modules. Each module, and in general each process, may be substituted by another one belonging to the same class of processes, since each class is comprised of structurally similar processes. The differences between the modules and their various subprocesses relate simply to the organizational level at which these processes, or "class elements", are formed (see Section 4).

Third, as far as the depicted process overlaps are concerned, they contain common events that are necessary for the continuation of one or more of the processes involved. For example, in Fig. 2, the bottom key macro-event is comprised of several events shared among three processes: the prepared (recruited) spe-

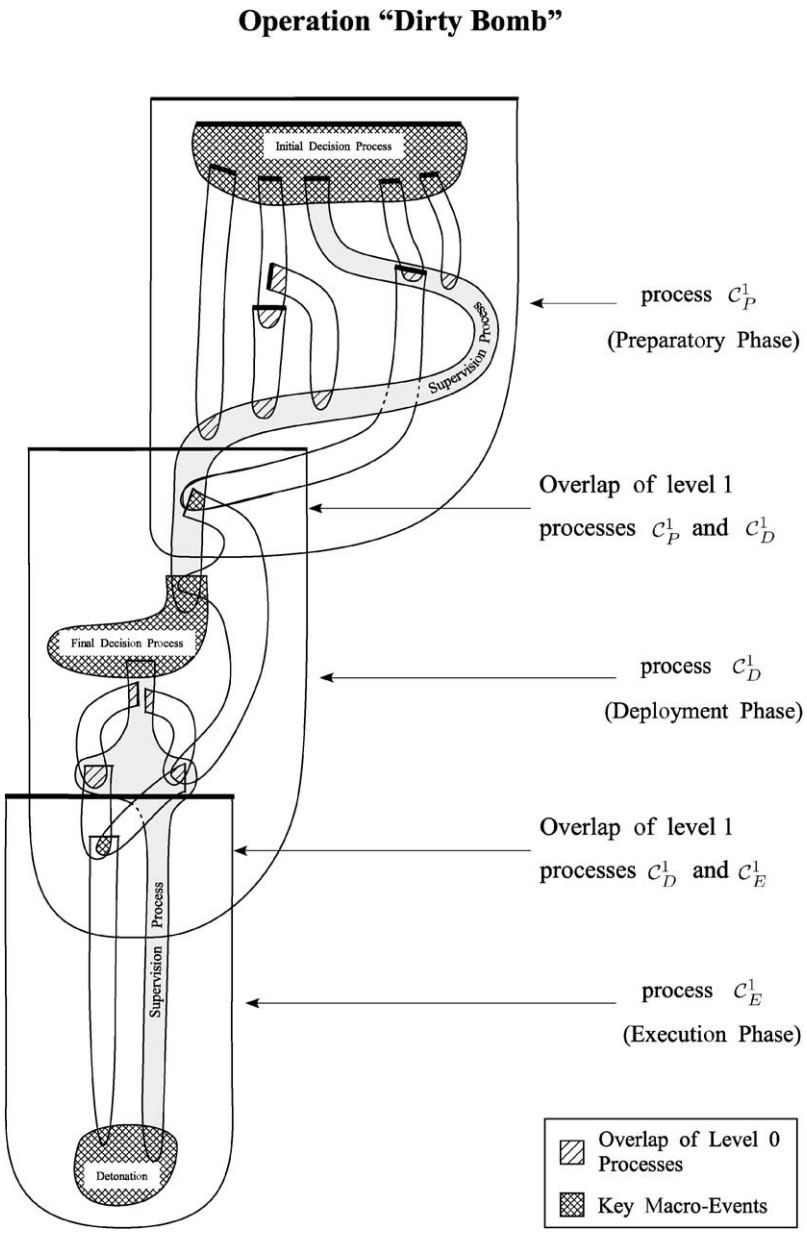

Fig. 1. High-level depiction of the entire process as composed of three pairwise overlapping subprocesses. Each subprocess (expanded in Figs 2–4), is identified on the right – including its symbolic notation c^1_K, $K = P, D, E$ – as an element of the corresponding level 1 class, to which the superscript refers, see Section 4.

cialists get their hands on the bomb-making ingredients/materials delivered by some representatives from the Supervision Process and can now begin the assembly, or bomb production, process (see Fig. 7).

Fourth, as can be seen from all four figures, especially Figs 2 and 3, the supervisory process is in touch with practically all other processes, since it has to manage and coordinate all of them. For example, in Fig. 2, five early processes – Bomb Procurement (alternative), both Material Acquisition processes, and both Specialist Recruitment processes[1] – are terminated in the Supervision Process, for a number of reasons: Alternative Bomb Procurement was terminated by the Supervision

[1] By "the specialists" we mean the "engineers" that are responsible for the assembly of the bomb.

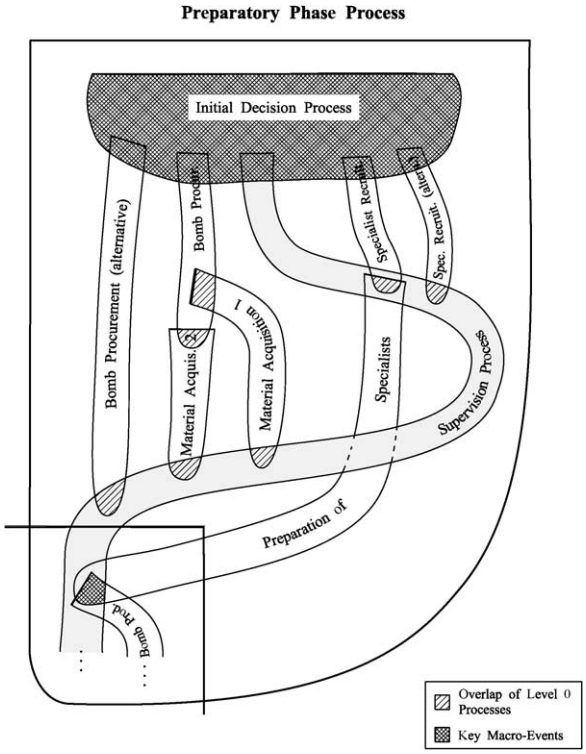

Fig. 2. High-level depiction of the Preparatory Phase of the operation. Note that two alternate Bomb Procurement and two alternate Specialist Recruitment processes are present in order to improve chances of success, including the mitigation of possible compromises. In the bottom left corner the overlap with the next, Deployment, phase is delineated.

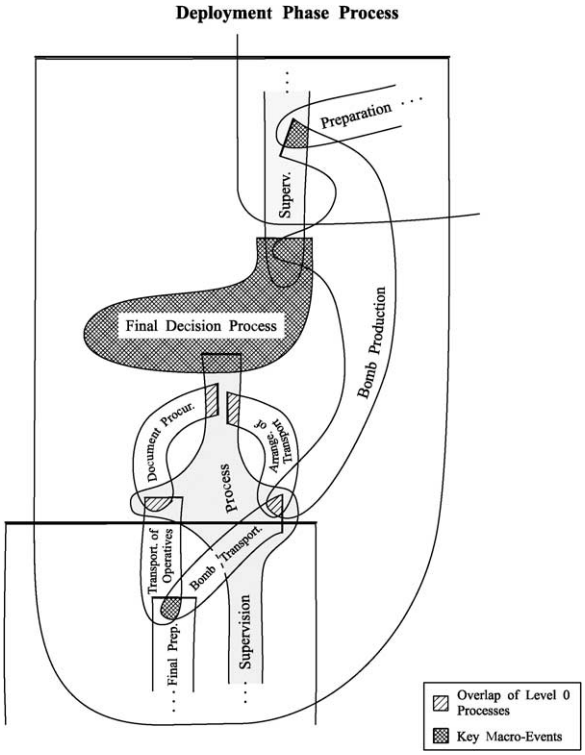

Fig. 3. High-level depiction of the Deployment Phase of the operation. In the top right corner the overlap with the previous, Preparatory, phase is delineated, while in the bottom left corner the overlap with the next, Execution, phase, is also shown.

Process after all Material Acquisition has succeeded; for the same reason, the Alternative Specialist Recruitment was terminated. The coordinating function of the Supervision Process is even more prominent in Fig. 3, where this process participates in all heavy overlaps except the very last one (at which point all remaining process have been set in motion and must maintain their covert nature). Finally, the very last overlap of the Supervision Process with the Detonation Process in Fig. 4 is meant to indicate that the informational ramifications of the detonation are being monitored.

3. A brief introduction to ETS in the context of the chosen scenario

As mentioned in the introduction, *all* applied mathematical formalisms rely on numeric representations, while ETS is based on a far-reaching, structural, generalization of the modern concept of natural number, in which the basic (Peano) block for constructing a number, shown in Fig. 8, is replaced by a more general structured event (see Figs 5–8).

It is useful to keep in mind that the structure of this formalism has absolutely no analogues to compare it with, despite the fact that its main entities, "structs", may have a superficial resemblance to some other known discrete objects such as, for example, graphs. Also, in view of a limited space, in what follows we will restrict ourselves to very brief informal descriptions of some of the basic concepts, so it is useful to consult the formal definitions in [1], Parts II, III and Appendix.

The most important point to keep in mind, however, is that, in ETS, *all objects are viewed and represented as (temporal) structural processes*, where the basic structural units are primitive events, each responsible for transforming the flow of several "regular" processes (see Figs 5 and 6). Also note that *the concept of class pervades all levels of consideration*.

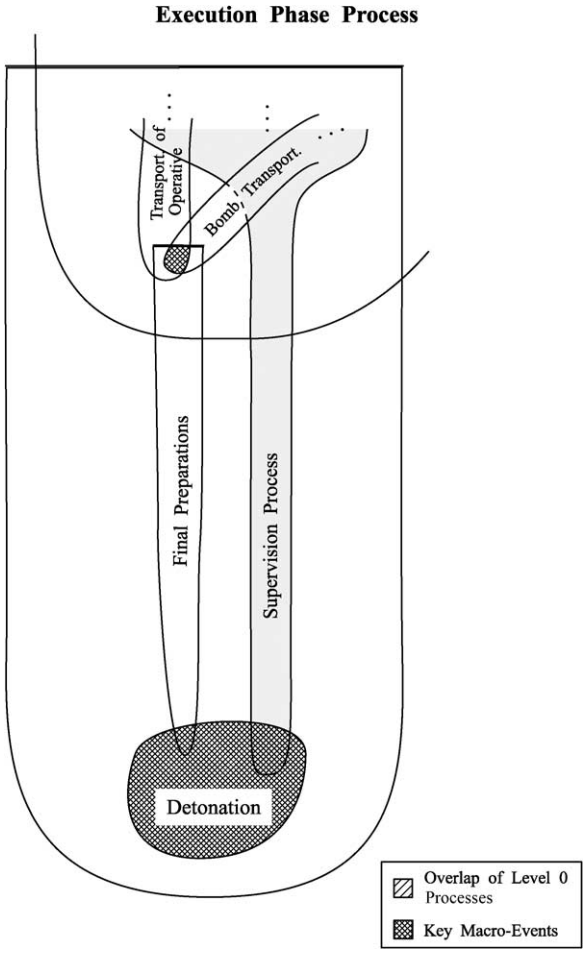

Fig. 4. High-level depiction of the Execution Phase of the operation. At the top the overlap with the previous, Deployment, phase is shown.

3.1. Primitive transformations

The first basic concept is that of a **primitive transformation** – or primitive event, or simply **primitive** – examples of which are depicted in Figs 5 and 6. The concept is relatively non-trivial but it is "atomic" to the ETS formalism. It stands for an event of fixed structure that is *responsible for transforming* one set of adjacent and presently interacting – hence the event itself – processes, called *initial processes*, into another set of processes, called *terminal processes*. Classes of initial and terminal processes, depicted in Fig. 5 as shapes and in later figures as different line-styles, are collectively called **primal classes**.

In other words, the concept of primitive encapsulates that of a "standard" interaction of several initial processes, each belonging to a particular class of

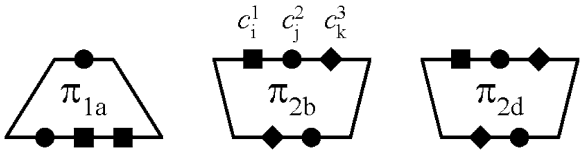

Fig. 5. Pictorial illustration of three *concrete* primitives. The first subscript of a primitive refers to the class of primitives all sharing the same structure, e.g., π_{2b} and π_{2d}. The initial classes are marked as various shapes on the top, while the terminal classes are shown as shapes on the bottom: each shape stands for a particular class of processes. The only *concrete* processes – i.e. the elements of these classes – identified in the figure are the initial processes of the primitive π_{2b}, with label $b = \langle c_i^1, c_j^2, c_k^3 \rangle$, where c_t^s is the tth process from primal class C_s, $s = 1, 2, 3$.

primal processes or a *primal class*. We can assume that these processes are "periodic" or "regular", while the event is responsible for their partial or complete modification into another set of periodic processes (some of which could coincide with some of the initial processes). The abstract structure of the event is such that it does not depend on the concrete initial (or terminal) process, as long as each of the processes involved belongs to the same (fixed) primal class of processes, depicted in Fig. 5 as a small solid shape at the top (or at the bottom) of the primitive.

In this paper we rely on the generalized concept of a primitive proposed in the Appendix of [1]. Our set of primal classes is comprised of the class of people, the class of "goods", the class of "bank accounts" and the class of various "sums of money". As one can see, at this, basic (or 0th), stage of representation – the only one considered in this paper – the structure of the initial and terminal processes is suppressed, as is the internal structure of the transforming event itself, and what is being captured by the formal structure of the primitives is the "external" *structure* of the event.

To ease reading of the following figures, one may want to keep in mind that, for primitives MP and GT, the *direction* of message passing and goods transfer between the people in those primitives is from the left person to the right one (as they are shown at the top of each primitive).

As to the general applicability of ETS, we note that according to ETS assumptions, nature is just a "collection" of various interacting temporal processes; and indeed, the examples of above *events* are all around us.

3.2. Structs

The second basic ETS concept is that of a **struct**, which is formed by a (temporal) sequence of connected

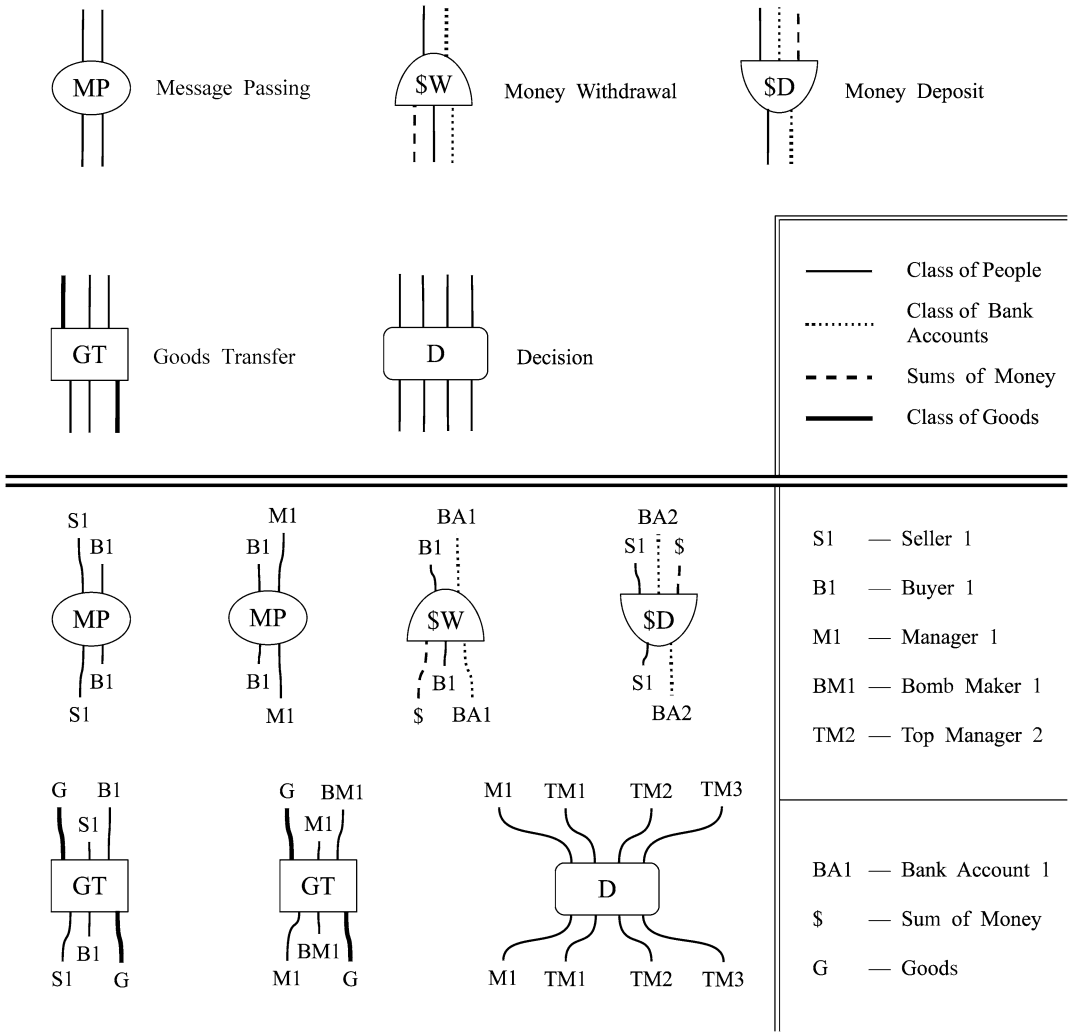

Fig. 6. Compared to the more precise *labelling* of primitives given in Fig. 5, we will use the simplified labelling shown here. (Top) Our five primitives together with the associated primal class, where the shapes in Fig. 5 are replaced by lines. MP stands for message passing – via any channel – from a person to another; $W stands for the withdrawal of a sum of money by a person from a bank account; $D stands for the deposit of a sum of money by a person into a bank account (to simplify labelling in this paper, we will not indicate a concrete sum of money that is being deposited or withdrawn); GT stands for the transfer of goods from one person to another; D stands for a meeting among an indicated number of people (number of initial processes) resulting in a decision. (Bottom) Examples of several primitives in action with the associated initial and terminal primal classes labelled.

primitive events observed by an agent, as shown in Fig. 7. This representation lends itself to a relatively simple implementation via a data structure that can be called "struct", which, as one can easily observe from the structure of our primitives, would maintain a relatively small width, thus ensuring low computationally complexity of the relevant (matching) algorithms.

Now, it is not difficult to see how the (temporal) classical Peano process of construction of natural numbers (see Fig. 8) was generalized to the construction of structs: the *single* "structureless" unit π_1 – out of which every natural number is built – was replaced by *multiple* structured ones. An immediate and critical consequence of the distinguishability (or multiplicity) of the units in the construction process is that we can now see *which unit was attached and when*. Hence, the resulting (object) representation for the first time embodies temporal structural information in the form of a formative, or generative, object history, recorded as a sequence of (structured) events. This was one of the basic driving motivations for the development of ETS, while the other main motivation was the development

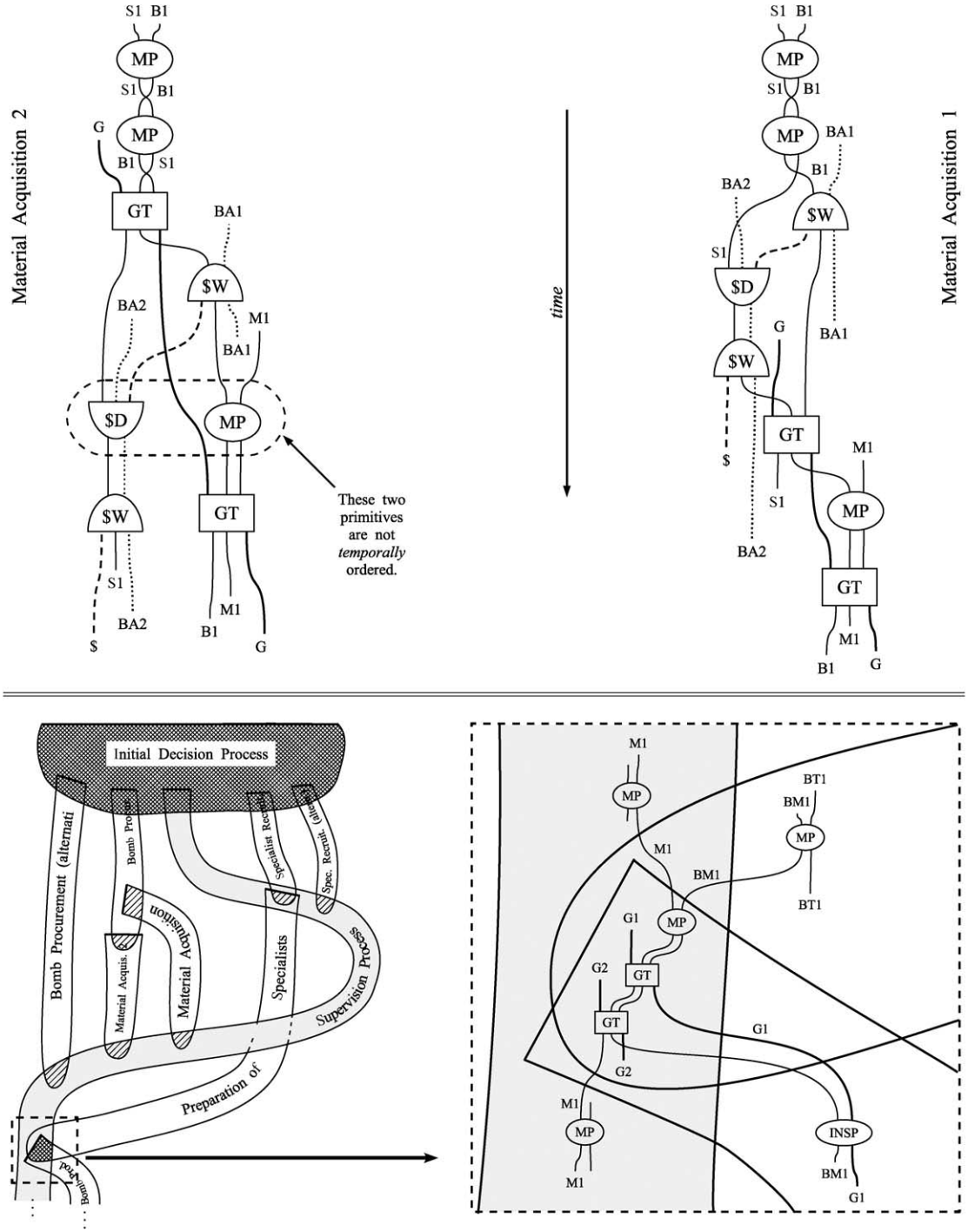

Fig. 7. (Top) Example of two structs showing the details of the two identified processes from the Preparatory Phase (see Fig. 2). (Bottom left) Preparatory Phase with an inset delineated by a dashed line. (Bottom right) The struct corresponding to an expanded key macro-event with the cross-hatching omitted. Note the new primitive INSP for Inspecting Goods, and the new primal process BT1 for the Bomb Maker's Teacher (used just in this figure).

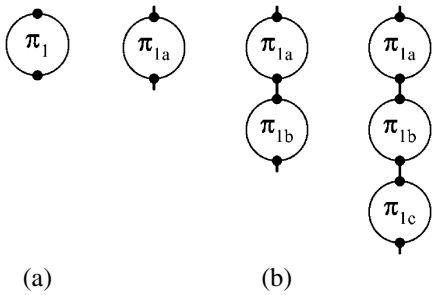

Fig. 8. The single primitive involved in the ETS representation of natural numbers (left) and three structs representing the numbers 1, 2 and 3.

of a unified *class-oriented representational formalism*. The two motivations should ensure that the needs of a large variety of information-processing areas as well as natural sciences are met.

3.3. Struct assembly

One of the basic operations on structs is struct assembly (Fig. 9(a) and (b)), which relies on the events shared by the structs involved. This operation allows one to combine in one struct several individually observed, but typically overlapping, structs. In the depicted example, a single buyer participates in both structs even though each struct focuses on one particular transaction involving that buyer, so that the structs overlap on the basic activities of the buyer.

In general, *the overlap* of several structs, representing several related "perceptual" processes, captures a *"non-interfering" interaction of these processes*.

The main difference between the proposed and the conventional forms of representation has to do with the temporal nature of ETS representation, which allows one to "compare" objects based on their "formative history". The latter *considerable additional information is simply not available in any conventional form of representation.*

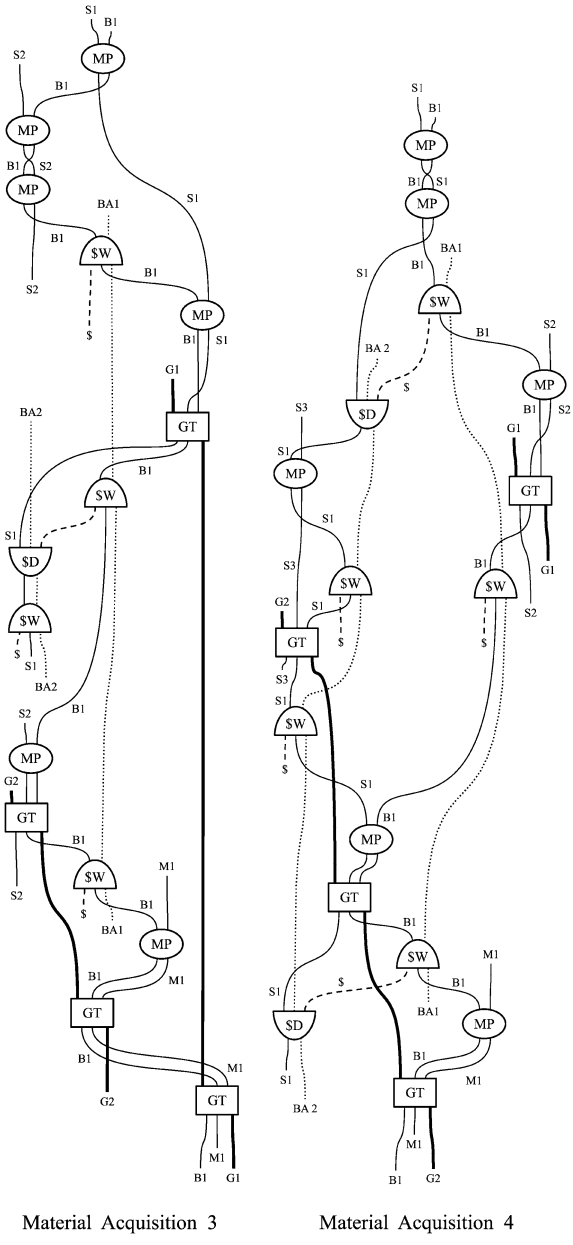

Fig. 9. (a) Example of two structs from the class Material Acquisition, which are *not* involved in Operation DB.

4. ETS classes and their representations

4.1. Level 0 classes and their dependence on the environment

The third basic concept is that of a **class**, which is defined by its **class representation**. *All* classes are assumed to be specified in this manner, except in the case of primal classes (Section 3.1) where such information is simply undisclosed. Moreover, even within the basic representational *stage* – the only one discussed here – each class is also viewed as possibly multi-leveled.

A **single-level class representation** is specified by means of a **single-level, or level 0, class generating system**, which details the *stepwise mode of construction* of the class elements.[2] *Each step* in such a system

[2]Note that since we began with level 0, it has no lower levels; level 1 has one lower level (level 0); level 2 has two lower levels (lev-

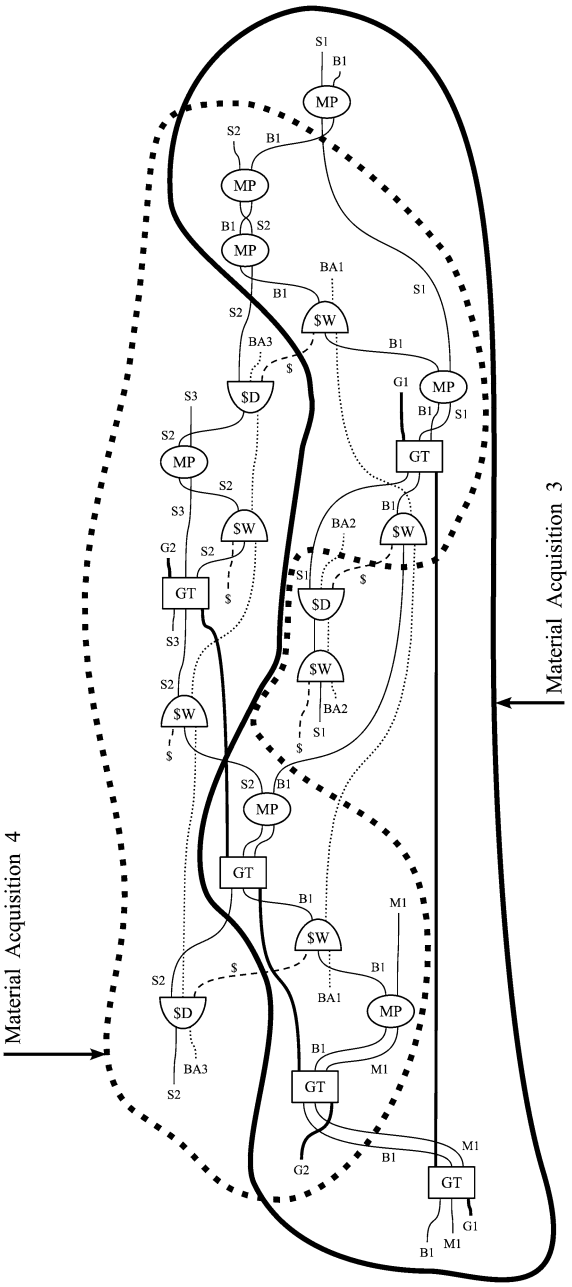

Fig. 9. (b) The struct which is the assembly of the two structs shown in Fig. 9(a).

is specified by *a set of* (level 0) **active constraints**, see Figs 11 and 13.

Each active constraint $\mathrm{ACon}(\overline{\Pi}, \mathcal{UT}, \overline{\Pi}^{\mathrm{anc}}, \overline{\Pi}^{\mathrm{opn}})$[3] specifies a family of "admissible" structs sharing the same structural "components" (see Fig. 10, left part): common **pivot primitives**, $\overline{\Pi}$, constrained structs connecting the pivot primitives, $\mathcal{UT}$, and anchor and open markings for some of the pivot primitives, $\overline{\Pi}^{\mathrm{anc}}$ and $\overline{\Pi}^{\mathrm{opn}}$. The pivot primitives are the primary primitives, i.e., those which must be present in all admissible structs. The tuple $\mathcal{UT}$ specifies, essentially, sets of (non-pivot, e.g., those that might be contributed by the environment, noise, etc.) primitives that *can* participate in the formation of various substructs, each "connecting" two pivot primitives within an admissible struct. The **anchor** primitives are those special primitives that a struct from the above family must share with another struct before it can be assembled with the latter. The **open** primitives are marked as such to indicate which pivot primitives in a struct (from the above family) are available for future "interfacing", i.e., where anchor primitives of *other* structs are allowed to overlap the struct.

Thus, the active constraint in Fig. 10 – as any of our active constraints – is satisfied by structs that may include various (presently unspecified) primitives appearing between the pivot primitives. In this paper, we do not specify various sets of primitives that are allowed to appear "between" the pivot primitives, i.e., those specified in $\mathcal{UT}$, because we did not want to introduce a large number of extra *concrete* primitives – dealing with the events non-essential to the main processes we are modeling – into the paper and figures.

During a *particular* step in the construction process, the struct that is being attached to the part of the class element that has been assembled so far must satisfy one of the *active constraints specified for that step*. Of course, one has to check whether a particular active constraint is currently "*applicable*", i.e., whether there exists a struct that satisfies the active constraint *and* that shares its anchor primitives with the struct that has been assembled so far.

If, within a step, several active constraints are applicable, one can then non-deterministically or probabilistically choose which one to apply. The presence of a null constraint (Θ) in a set of constraints allows for the *possibility of no constructive action at this step, even though some other constraint for this step might be applicable*.

To be more accurate, in the definition, it is assumed that *each* such step *can* be preceded by a step executed by the **environment**, i.e. *by some other class generating system "intervening" or "participating" in the*

els 0, 1); etc. Also note the difference between a level and a stage: levels refer to those of classes and appear within a single representational stage.

[3] For details, see Section 5 in [1].

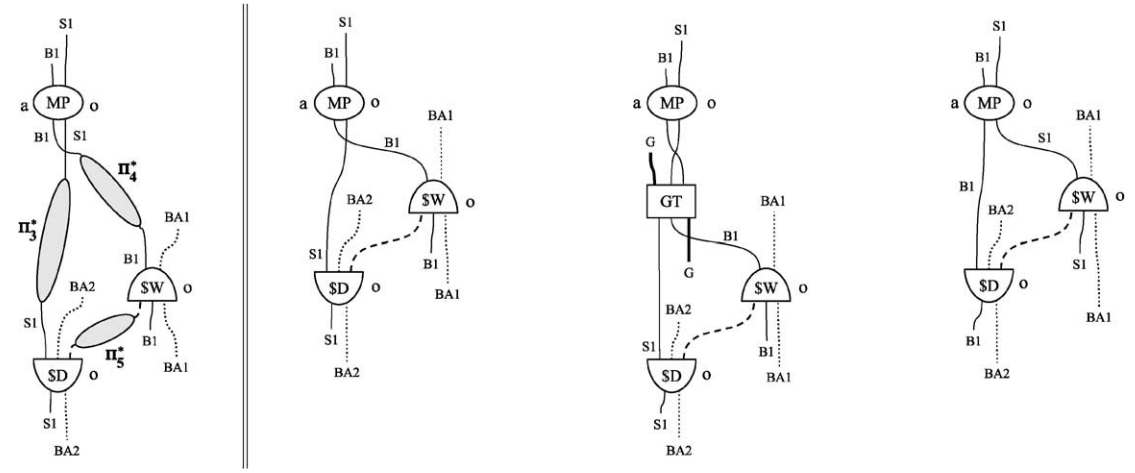

Fig. 10. (Left) A pictorial illustration of a (payment) active constraint $\mathrm{ACon}(\overline{\Pi}, \mathcal{UT}, \overline{\Pi}^{\mathrm{anc}}, \overline{\Pi}^{\mathrm{opn}})$, where the tuple $\mathcal{UT}$ (including its primitives) is not specified (see also Fig. 11). Letter 'a' next to a primitive denotes an anchor pivot primitive, while letter 'o' next to a primitive denotes an open pivot primitive. (Right) Three active structs, only the first two of which satisfy the constraint. The only non-pivot primitive is GT.

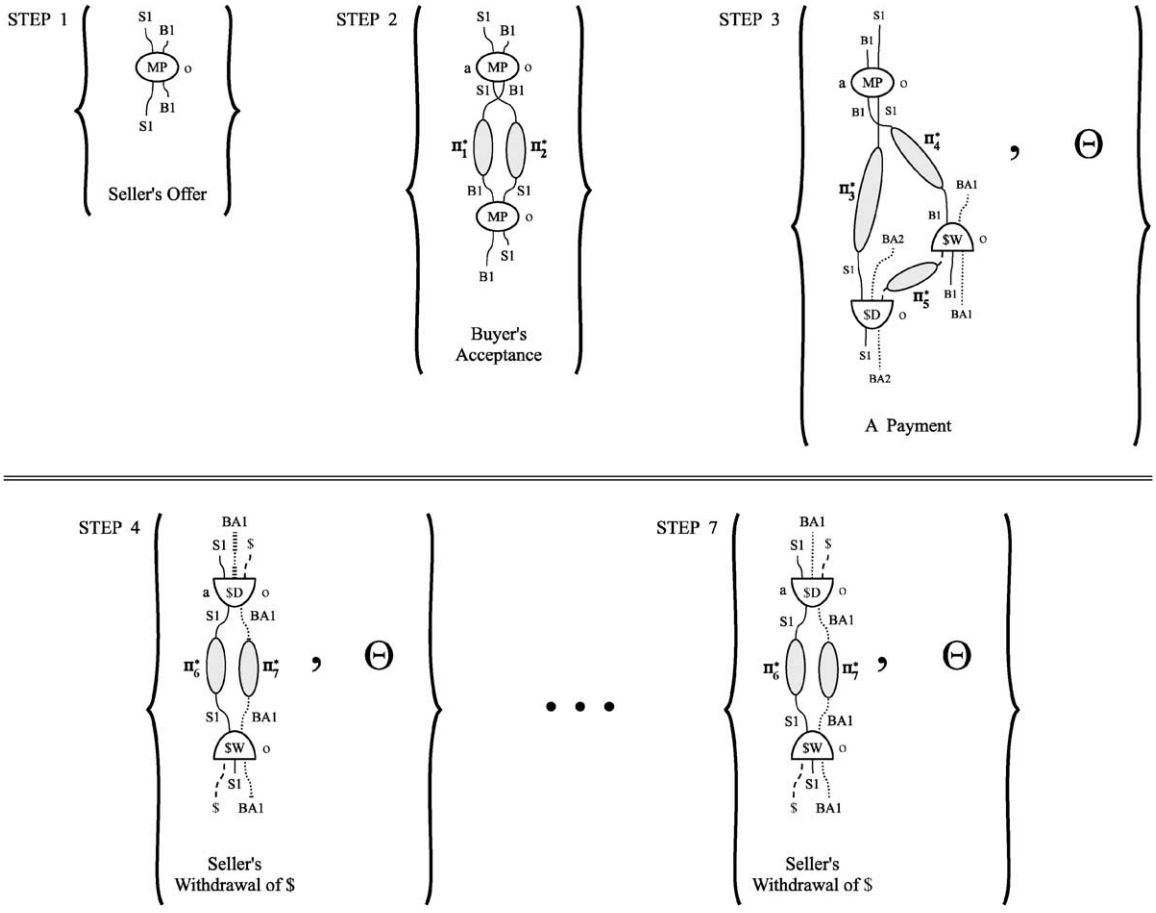

Fig. 11. Stepwise specification of a single-level (level 0) class representation for the class of structs Material Acquisition (see Fig. 2). Each step is specified by its own set of active constraints: each set is either a singleton or contains two elements, one of which is the null constraint. Note that the constraints for some steps are identical.

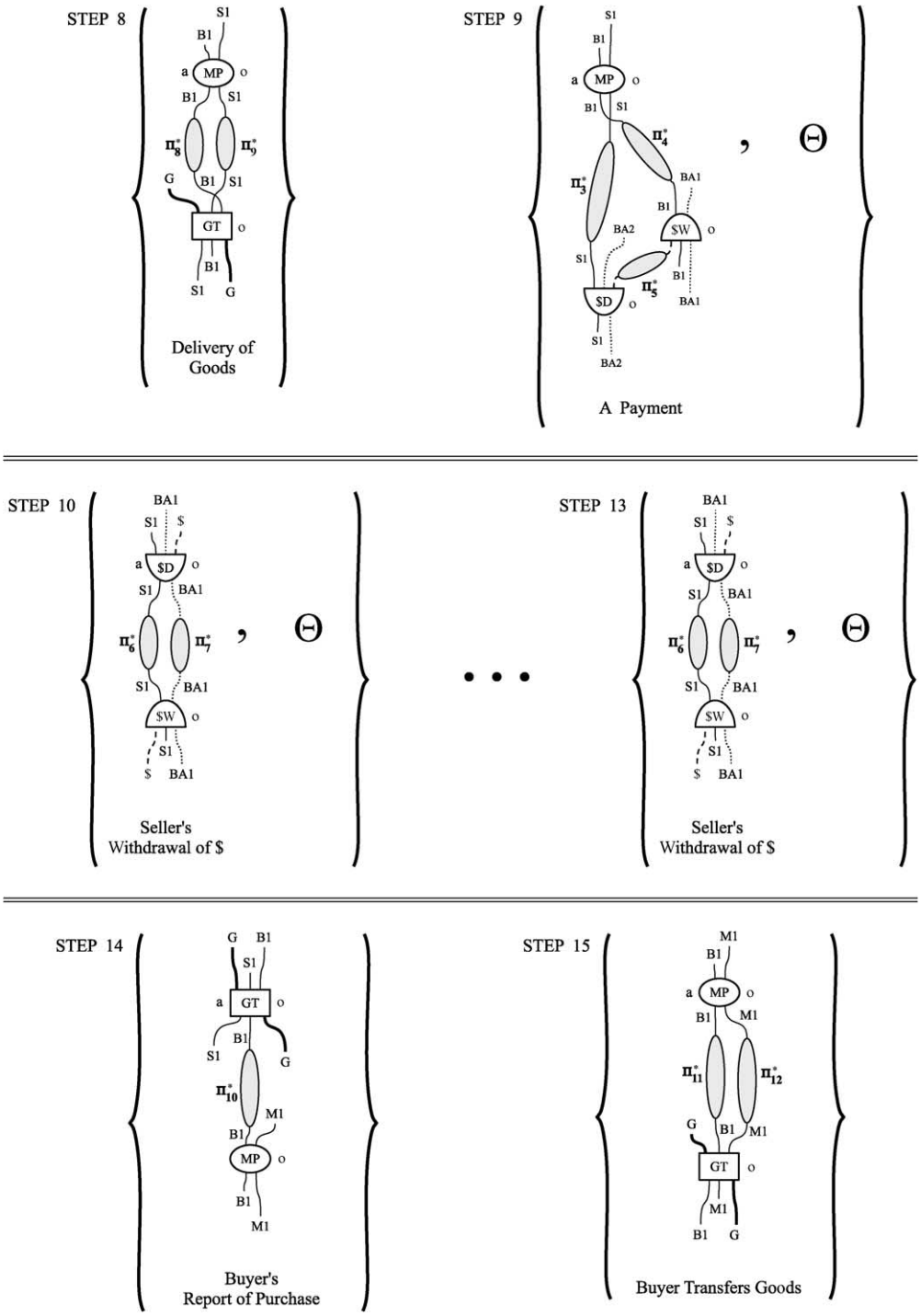

Fig. 11. (Continued).

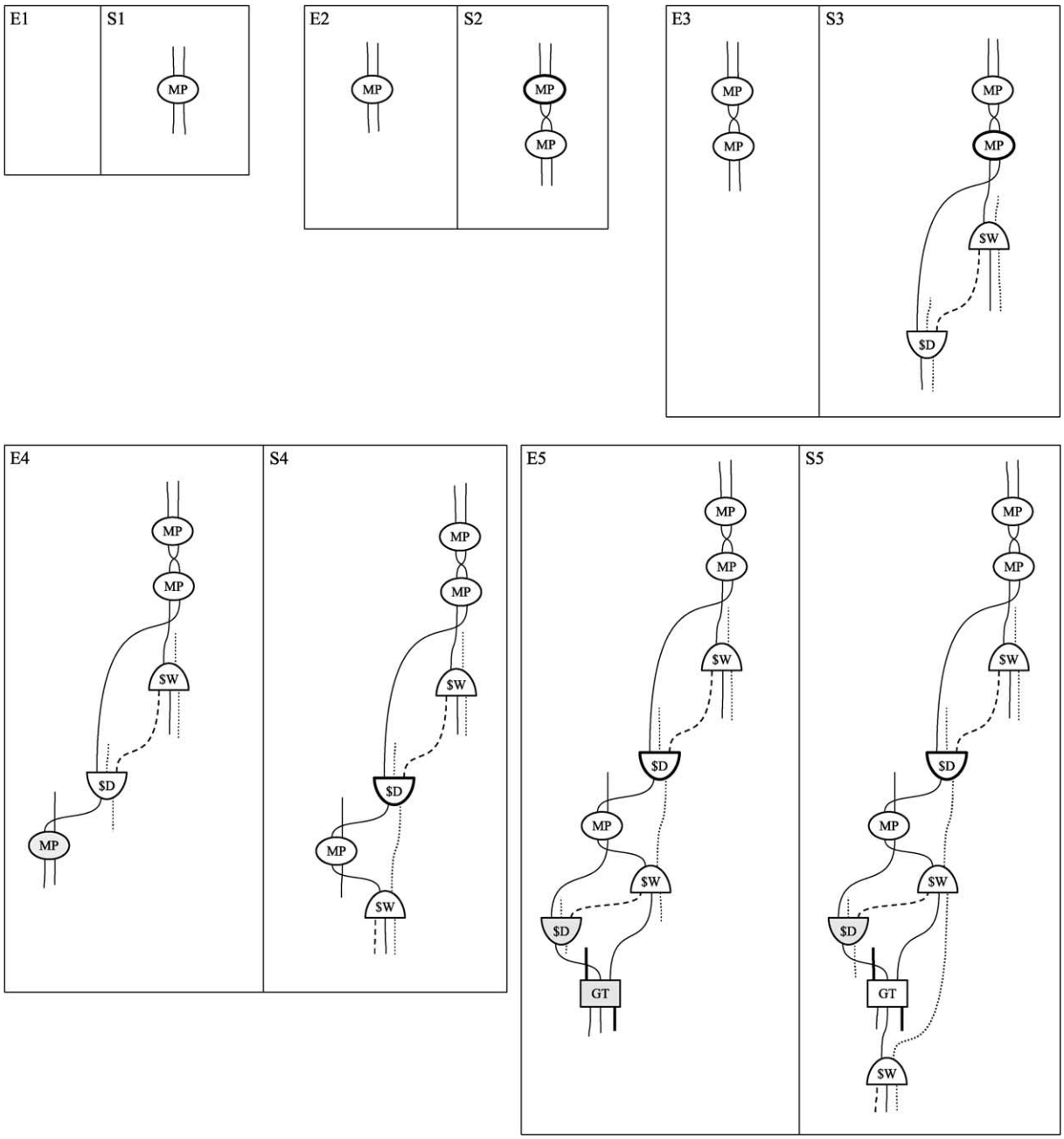

Fig. 12. Stepwise construction process of (level 0) class element Material Acquisition 4 shown in Fig. 9(a). The construction process always proceeds in two-step units, where a step by the class generating system, labelled S, *must be* preceded by an environment (often the null) step, labelled E. For each two-step unit, the anchor primitives are shown in bold while the primitives added by the environment are shaded in dark gray, until/if they are absorbed in some step by the generating system, in which case they lose their shading. For simplicity, the open markings are suppressed, since, in our case, *all* pivot primitives are open. The steps in which the null constraint was applied are not shown.

construction process (see Fig. 12, in which the description of the environment is not shown).

One should note that, for example, the two classes associated with an acquisition, Material Acquisition (shown in Fig. 2) and Document Procurement (shown in Fig. 3), most likely have a different structure – despite possibly some shared constraints – mainly due to the nature of the two acquisition processes. Also,

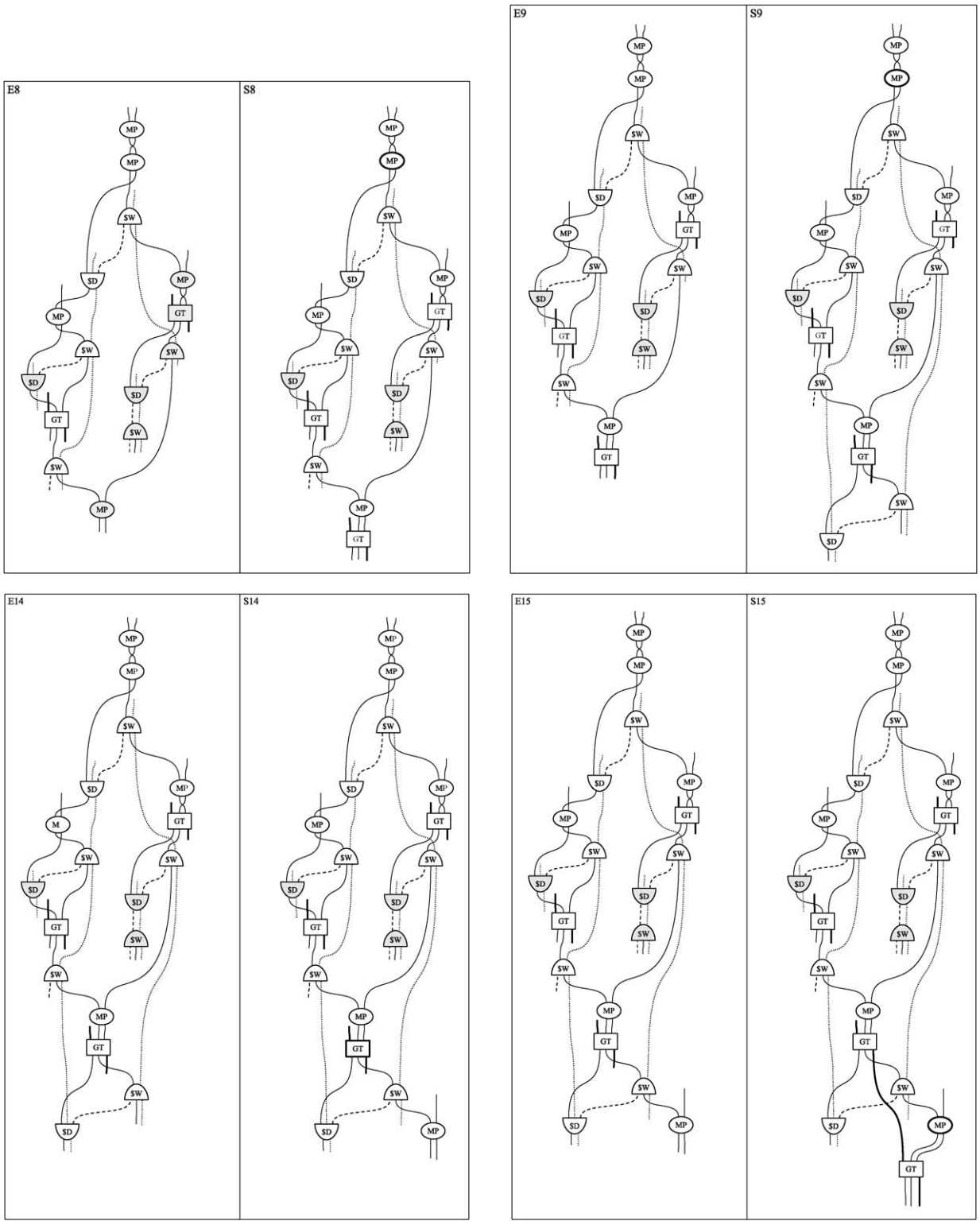

Fig. 12. (Continued).

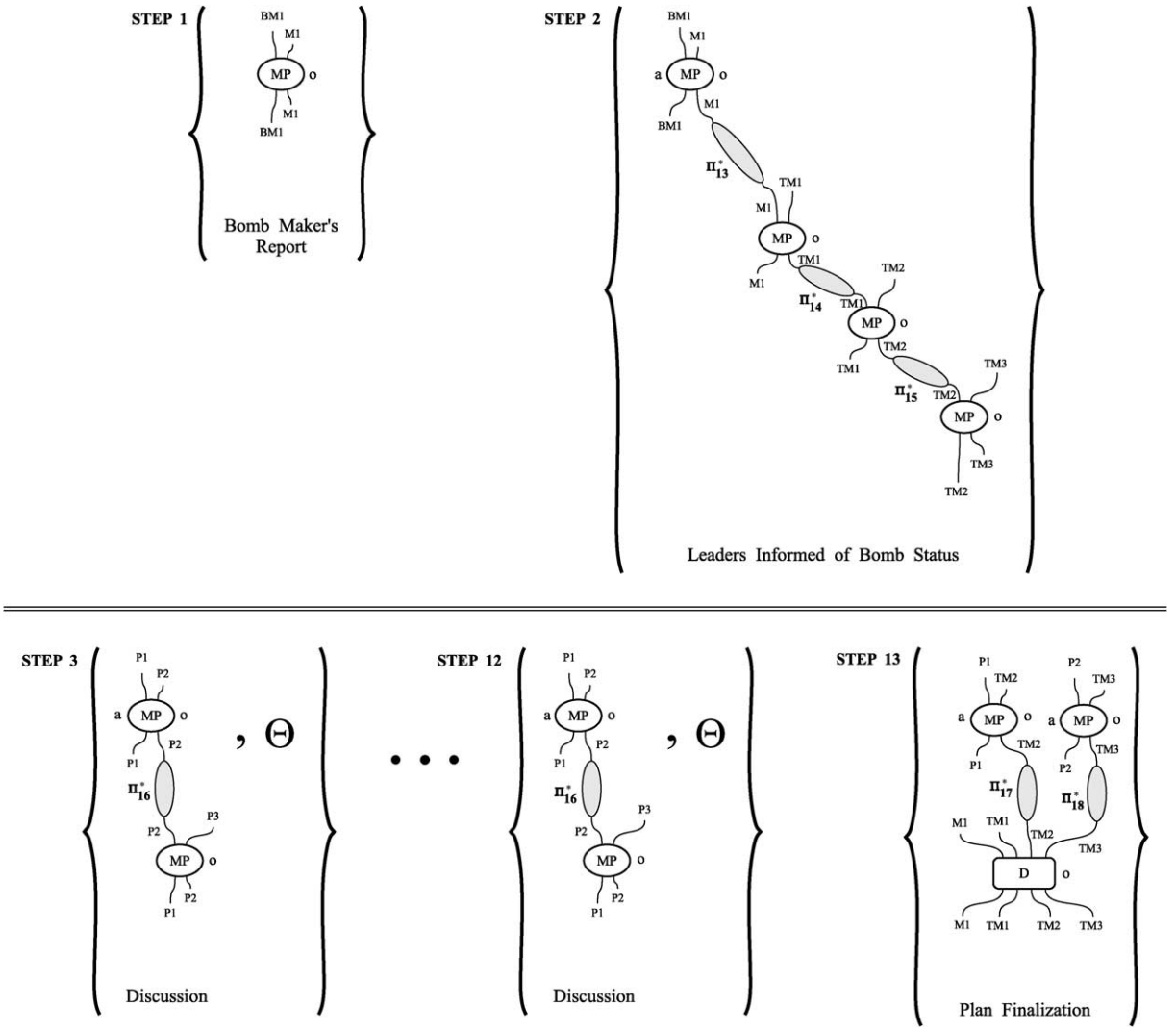

Fig. 13. Stepwise specification of another single-level (level 0) class representation for class Final Decision (see Fig. 3). Note that the form of the active constraint for step 2 may vary depending on the organization's communication protocol. Also note that the anchor primitives in the last (terminating) constraint are the concluding messages originating from the very top decision makers.

comparing the Material Acquisition and Bomb Procurement classes (see Fig. 2), one should note that Bomb Procurement processes are typically more complex processes involving extensive search and negotiations, while Material Acquisition processes, as we treat them, are considerably shorter processes associated more immediately with the purchases of the bomb materials themselves.

With regard to Fig. 14 depicting two elements from the class Final Decision (see also Fig. 3), it is easy to see that these elements capture a hypothetical structure of the discussions among the top managers about the deployment of the dirty bomb. Typically, when engaging in surveillance, one cannot expect to have such detailed information, since this information is a function of the group members' complex interpersonal dynamics, unless a "mole" is present at the meeting. In the latter case, the obtained structural information can help one to improve the representation of future high-level decision meetings.

4.2. Level 1 structs

Suppose that an agent has already learned several level 0 classes, which together form the current *level 0*

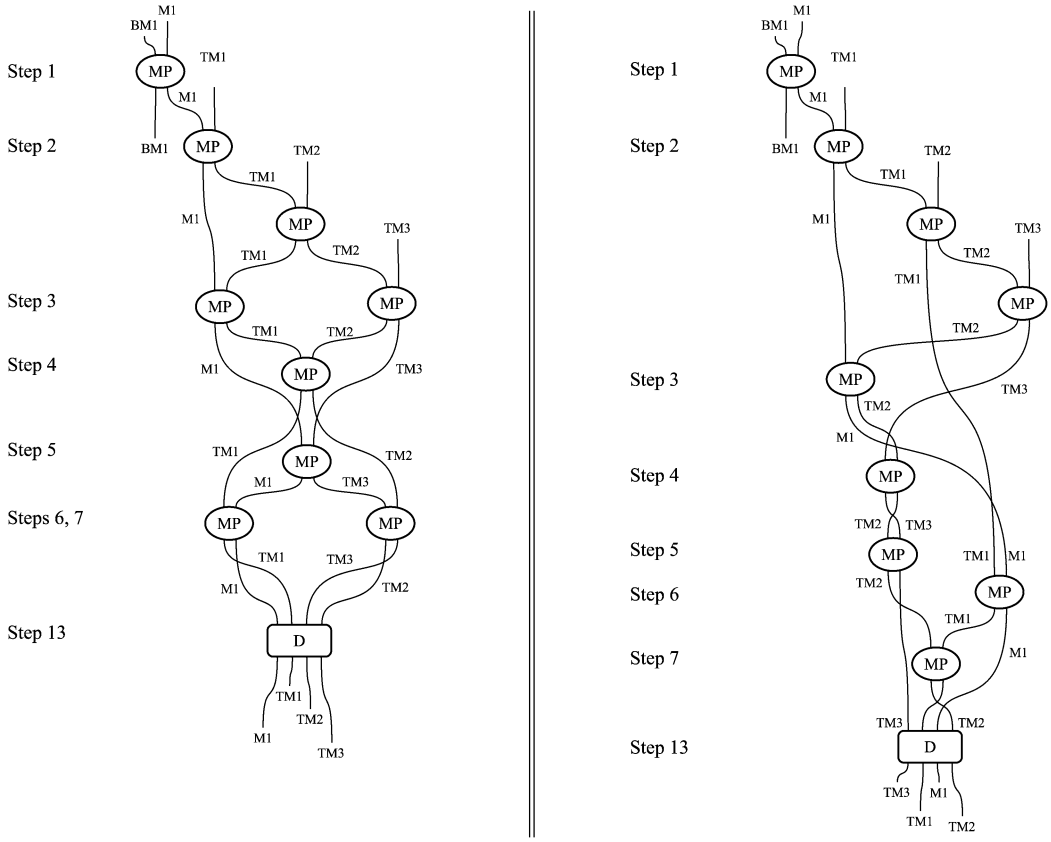

Fig. 14. Two elements belonging to the class Final Decision whose specification is given in Fig. 13. Again, the steps in which the null constraint was applied are not shown.

class setting. Then, when representing objects, the agent has an access to a more refined/powerful form of object representation than a plain level 0 struct: it can now see if the observed struct is in fact composed of several level 0 class elements (each belongs to one of the classes in level 0 class setting). For example, if a computer program recorded events depicted in Fig. 15 – without the benefits of the partitioning into the subprocesses depicted in that figure – the corresponding struct would be considered as a level 0 struct. However, once these subprocesses are identified as level 0 class elements, and the information reflected in the level 0 struct, the resulting struct, partitioned accordingly, is called a **level 1 struct** (Fig. 15). Thus, a next level (level 1) struct, provides extra representational information – as compared to the underlying level 0 struct – in the form of the appropriate partition of the previous level (level 0) struct.

4.3. Higher level classes and their dependence on the environment

In a **2-level** (also referred to as "level 1") version of the **class representation**, each step is associated with a set of *level 1 active constraints* (not introduced here). However, during the construction process, the level 1 struct that is being attached (at this step) to the previously constructed part *must now be composed only out of level 0 admissible class elements* [4] in a manner satisfying one of the active constraints for this step. Figure 16 illustrates this construction process for level 1 class element, c_P^1, the Preparatory Phase (see Fig. 2).

For a level 2 class element – this element is an output of a level 2 (or three-levels) class generating system –

[4] Each of those must come from a class belonging to a (previously learned or given) set of level 0 classes, *comprising level 0 class setting*.

at each step of its construction, the relevant part is assembled out of several level 1 class elements in accordance with one of the active constraints specified for this step. The entire Operation "Dirty Bomb" should be considered as a level 2 class element, constructed out of three level 1 class elements (c_P^1, c_D^1, c_E^1), as shown in Fig. 1.

5. Conclusion

First, we should draw attention to the following point: our Dirty Bomb example could have been easily replaced by some typical business scenario, so that the methodology is obviously applicable to any human interaction process, by simple modification of the corresponding primitives and class representations.

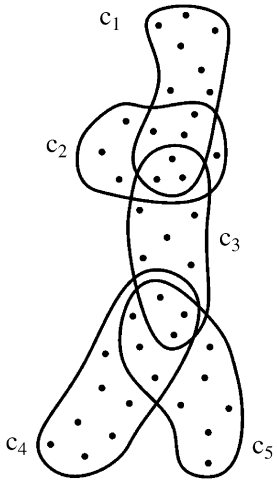

Fig. 15. Simplified pictorial representation of a level 1 struct in the contracted form: dots stand for primitives and solid lines delineate level 0 class elements c_i's.

The title of our paper – "ETS as a Tool for Decision Modeling and Analysis: Planning, Anticipation, and Monitoring" – should be interpreted in that general manner. Thus, as was mentioned in Introduction, structural modules, once acquired, could be effectively used as "plug-in" modules in various tasks related to decision modeling and analysis: one can use ETS as a systematic tool for creating and analyzing various relevant scenarios whose details are otherwise not easy to handle. Moreover, the process of selection on the basis of risk assessment can now be carried out in a more systematic manner by assessing the risks of interference for each scenario on the basis of both the *structures* of interfering *classes* as well as the structure of classes participating in the scenario.

As far as monitoring is concerned, the ability to predict, or anticipate, future actions – on the basis of the multilevel class structure – allows more effective management, by revealing situations where the influence of management processes are more critical as compared to other situations (see Section 2 and Fig. 7). In the case of surveillance, the same considerations apply, and moreover, the capability to anticipate larger process modules allows one to focus the surveillance resources on the currently most critical parts of processes. More specifically, since events always occur within some structural context, ETS provides a systematic tool for inferring the occurrence of such key events, even when

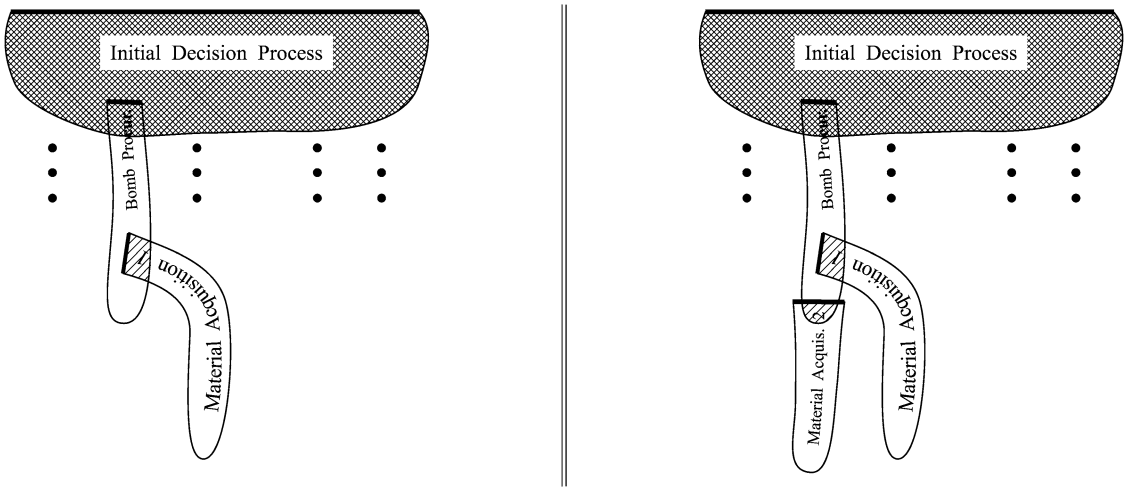

Fig. 16. Simplified depiction of a step in a *level 1* generating process, involving the addition of both Material Acquisitions (see Fig. 2) in the construction of a level 1 class element c_P^1, the Preparatory Phase.

they cannot be observed directly (e.g., due to the covert nature of the activities under surveillance). In the face of enormous amounts of data that needs to be monitored, the latter capability becomes paramount.

To summarize, our basic message is that temporal structural representations, when realized properly (although they require some initial investment) – in sharp contrast to the currently pervasive numeric formalisms – should allow one to adequately capture the ubiquitous reality of various evolving *interacting* processes and to capitalize on the resulting representation.

Acknowledgments

We would like to thank Mihai Nadin for inviting the paper to the session on Anticipation and Risk Assessment organized by him at the 2007 Decision and Risk Analysis Conference.

References

[1] L. Goldfarb, D. Gay, O. Golubitsky, D. Korkin and I. Scrimger, What is a structural representation? A proposal for an event-based representational formalism, Sixth Variation, Technical Report TR07-187, Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, 2007; http://www.cs.unb.ca/~goldfarb/ETSbook/ETS6.pdf.

[2] M. Nadin, *Anticipation: The End Is Where We Start from*, Lars Müller Publishers, Baden, Switzerland, 2003.