Situated Nonmonotonic Inference with BABY-SIT

Erkan Tın and Varol Akman
Department of Computer Engineering and Information Science
Faculty of Engineering, Bilkent University
Bilkent, Ankara 06533, Turkey
E-mail: tin@cs.bilkent.edu.tr

1 Introduction

This paper presents a computational medium, called BABY-SIT, based on situation theory and demonstrates how problems requiring a nonmonotonic approach for their solution can be formalized and solved in this framework. It is not our aim to propose a solution to the deep problems of AI such as the frame problem; rather, we highlight the constructs available in BABY-SIT to tackle these problems with a simple approach. We consider a well-known AI puzzle, the Yale Shooting Problem, which seems to be a suitable test-bed for nonmonotonic theories.

2 BABY-SIT

BABY-SIT [12] is a computational medium based on situations and has been developed in KEE^{TM} (Knowledge Engineering Environment) [6]. The implementation language is Common Lisp and the BABY-SIT desktop is based on X Windows running on a SPARCStation. The primary motivation underlying BABY-SIT is to facilitate the development and testing of programs in domains ranging from linguistics to AI within a unified framework built upon situation-theoretic constructs [14, 18, 17, 12].

BABY-SIT accommodates the basic features of situation theory, and compared to the existing approaches [2, 3, 9], enhances these features [16, 19]. Devlin's fine book [4] has been extensively used in designing the formal skeleton of BABY-SIT.

Situation theory provides a collection of basic types that can be used for individuating or discriminating uniformities of the real world. There are nine basic types which are also employed in BABY-SIT: ~IND (individuals), ~TIM (times), ~LOC (places), ~REL (relations), ~POL (polarities), ~INF (infons), ~PAR (parameters), ~SIT (situations), and ~TYP (types). These are special objects of BABY-SIT that can be used in the same way as ordinary objects are. Moreover, they can be used to associate a type with a new object in the system.

Suppose `bob` is an individual and `sit1` is a situation. Then, these objects can be declared as:

```
I> bob:~IND
```

```
I> sit1:~SIT
```

Situations are viewed at an abstract level. This means that situations are sets of parametric infons, but they may be non-well-founded. All situations are required to cohere. Situations (and hence infons they support) may have spatio-temporal dimensions. A hierarchy of situations can be defined both statically and dynamically. A situation can have information about another which is a part of the former. This is made possible via structural links and 'infon inheritance' among situations. BABY-SIT has a minimal situation, called the 'background situation,' which is a part of every other situation.

Relations are categorized into two: infonic relations that can be used as major constituents of infons and non-infonic relations that can only fill the argument roles of infonic relations. Each infonic relation has 'appropriateness conditions' that determine the type of its arguments. Hence, each argument can be declared to be from one or more of the basic types. The number of arguments that an infonic relation can take define 'maximality and minimality conditions' for that relation. A well-formed infon then conforms to the specifications defined by appropriateness conditions, maximality and minimality conditions of its infonic relation. Consider the seeing relation. If we would like it to be an infonic relation with at most two arguments, the former being of type individual and the latter being of type either a situation or an individual, we can write:

```
I> <sees | ~IND, {~SIT, ~IND}> [1]
```

The number in square brackets indicates the minimum number of arguments that can be used with the relation. Hence, `≪sees, bob, 1≫`, for example, is considered to be a valid (*unsaturated*) infon in the system.

Infons are categorized as 'labeled' and 'unlabeled'. A labeled infon is associated with a label which is considered to be an infon object which can be used as arguments of infonic relations. For instance, the infon `≪sees, bob, sit1, 1≫` can be named `infon1` by the assertion:

```
I> infon1 = ≪sees, bob, sit1, 1≫
```

On the other hand, an unlabeled infon is not a usual BABY-SIT object. It must be expressed explicitly and hence is considered to be a distinguished object.

Parameters are place holders and can be anchored to unique individuals in an anchoring situation. A situation can be realized if its parameters are anchored, either partially or fully, by an anchoring situation. That is, only anchoring the parameters of an infon contributes a piece of information about the situation. 'Parameter restriction' allows parameters to be used to denote objects of more complex types, as exemplified below.

Suppose we want to have a parameter `E1` denoting any individual and `E2` denoting any individual that sees situation `sit1`. This can be done by asserting:

```
I> E1 = ~IND
```

I> E2 = IND1 ^ $\ll$sees, IND1, sit1, 1$\gg$

Then, E1 is a parameter which can be anchored to any object of type ~IND. IND1 is a default system parameter of type ~IND. E2 is considered to be an object of type ~PAR such that if it is anchored to an object, say obj1, then obj1 must be of type ~IND and the background situation must support the infon $\ll$sees, obj1, sit1, 1$\gg$.

Defining more complex types is made possible by 'type abstraction.' Type abstraction can be done either on the object-level or situation-level by employing a certain parameter and a 'grounding situation.' They are of the form [P | $s \models I$] where P is a parameter, s is a situation (i.e., a *grounding* situation), and I is a set of infons. The type of all situations that Bob sees can be defined in BABY-SIT as follows:

I> ~SITALL = [SIT1 | w $\models$ $\ll$sees, bob, SIT1, 1$\gg$]

Hence, ~SITALL is seen as an object of type ~TYP in BABY-SIT and can be used as a type specifier for declaration of new objects in the environment. An object of type ~SITALL, say obj2, is an object of basic type ~SIT such that the background situation supports the infon $\ll$sees, bob, obj2, 1$\gg$.

In BABY-SIT, a *situation browser* enables one to create situations, browse them graphically, add or delete infons, and establish hierarchies among situations. For example, the following sequence of assertions creates a situation sit2 and then adds the infon $\ll$sees, bob, sit1, 0$\gg$ into it:

I> sit2:~SIT

I> sit2 $\models$ $\ll$sees, bob, sit1, 0$\gg$

Variables in BABY-SIT are used in constraints and query expressions, and have scope only within the constraint or the query expression they appear. A variable can match any object appropriate for the place or the argument role it appears in. For example, given the relation above, variables ?S and ?X in the proposition ?S$\models$$\ll$sees, ?X, sit1, 0$\gg$ can only match objects of type ~SIT and ~IND, respectively.

BABY-SIT allows the use contextual information which plays a critical role in all forms of behavior and communication. Situations and constraints can be grouped to form a whole (i.e., a perspectivity constraint set) which provides a computational context. Moreover, the partial nature of situations facilitates computation with incomplete information. Constraints in BABY-SIT come in three flavors: forward chaining constraints, backward chaining constraints, and bidirectional chaining constraints. They enable one situation to provide information about another and serve as links between representations and the information they represent. A BABY-SIT constraint is of the form:

$$\textit{antecedent}_1, \ldots, \textit{antecedent}_n \ \{\texttt{<=}, \texttt{=>}, \texttt{<=>}\} \ \textit{consequent}_1, \ldots, \textit{consequent}_m.$$

Each $\textit{antecedent}_i$, $1 \leq i \leq n$, and each $\textit{consequent}_j$, $1 \leq j \leq m$, is of the form $\textit{sit}$ $\{\models, \not\models\}$ $\ll$$\textit{rel}$, $\textit{arg}_1$, $\ldots$, $\textit{arg}_l$, $\textit{pol}$$\gg$ such that *rel* and each $\textit{arg}_k$, $1 \leq k \leq l$, can either be an object of appropriate type or a variable.

Each constraint has an identifier associated with it and must belong to a group of constraints. For example, the following is a forward chaining constraint named FALLING-BLOCK under the perspectivity constraint set NATURAL-LAW-PERSPECTIVE, which states that blocks drop if not supported:

NATURAL-LAW-PERSPECTIVE:
  FALLING-BLOCK:
    ?S1 $\models$ $\ll$`block, ?B, 1`$\gg$,
    ?S1 $\not\models$ $\ll$`supports, ?C, ?B, 1`$\gg$ `=>` ?S2 $\models$ $\ll$`drops, ?B, 1`$\gg$
  BACKGROUND-CONDITIONS:
    `w:`$\ll$`exists, gravity, 1`$\gg$.

where `?S1`, `?S2`, `?B`, and `?C` are variables. `?S1` and `?S2` can only be assigned objects of type ~SIT while `?B` and `?C` can have values of some type appropriate for the argument roles of the block, drops and supports infonic relations, respectively. This constraint can apply in any situation. Hence, BABY-SIT constraints can be global. Constraints can also be situated.

This constraint is associated with a set of *background conditions*. Background conditions are, in fact, assumptions which are required to hold for constraints to be eligible for activation. FALLING-BLOCK constraint can become a candidate for activation only if it is the case that $w \not\models$ $\ll$`exists, gravity, 0`$\gg$, i.e., if there is no information as to the absence of gravity in the background situation.

Declarations of objects and assertions of situation propositions can be made in the assertion mode. Assertions may activate the forward chaining mechanism of BABY-SIT which lets one derive new information over the forward chaining constraints (or bidirectional chaining constraints) of a given perspectivity constraint set. A candidate forward chaining constraint is activated whenever its antecedent part is satisfied. All the consequences are asserted if they do not yield a contradiction in the situation into which they are asserted. New assertions may in turn activate other candidate forward chaining constraints. Candidate backward chaining constraints are activated either when a query is entered explicitly or is issued by the forward chaining mechanism.

BABY-SIT provides a useful query mechanism. It is possible to issue situated or unsituated queries in its query mode. Queries can be proved by using the backward chaining constraints (or bidirectional chaining constraints) of a given perspectivity set. In addition to querying which situation supports what, it is also possible to ask which (particular) situation does not support an infon or a set of infons.

The query mechanism also enables the user to issue oracle queries [4]. These are queries that collect information about a single object in the system. By associating an issue set with the oracle query, it is possible to retrieve information about particular aspects of the oracle object. Rather than measuring the degree of relevance of an infon for the oracle of an object, we apply a restricted form of the notion of 'contextual effect' and accordingly determine if an infon is relevant or not to the current context [1, 12].

Objects in the environment and the attainment of information flow in BABY-SIT are compatible with the ontology of situation theory. Computation is context-sensitive and type-theoretic. The basic computation regime is unification. The mode of computation is built upon appropriateness conditions, conveyance and inheritance of information, consistency of situations, anchoring, and constraint satisfaction.

BABY-SIT provides an interactive environment for the development and testing of programs and examining the behavior of the system as the computation proceeds. User interfaces and menu-driven operations ease the tasks of the programmer. Various types of model-based reasoning such as data-driven reasoning, goal-driven reasoning, and nonmonotonic reasoning are possible in BABY-SIT.

3 The Yale Shooting Problem

Various nonmonotonic formal systems have been proposed to facilitate commonsense reasoning. *Situation calculus* [8] has initially been used to reason about the effects of actions. Hanks and McDermott [5] describe what they call *temporal projection* in the framework of situation calculus as follows. Given a description of the current situation, some descriptions of the effects of possible actions, and a sequence of actions to be performed, how do we predict the properties of the world in the resulting situation?

Hanks and McDermott [5] applied some of the existing logics (e.g., Reiter's default logic [10]) to scenarios to see whether the expected results are indeed produced. The *Yale Shooting Problem* (YSP) is one of these scenarios posed by Hanks and McDermott [5] as a paradigm to show how the temporal projection problem arises in logical frameworks. At some point in time, a person (Fred) is alive. A loaded gun, after waiting for a while, is fired at Fred. What are the results of this action? One expects that Fred would die and the gun would be unloaded after the firing of the gun. But Hanks and McDermott [5] demonstrate, in the framework of circumscription [7], that unintended minimal models are obtained; the gun gets unloaded during the waiting stage and firing the gun does not kill Fred.

Shoham's causal theories and preference criteria [11] provides a satisfactory solution to the temporal projection problem.[1] Causal theories contain axioms to reason about the effects of actions. Proceeding in time, knowledge about the future is obtained from what is known and what is not known about the past. This forms the core of the causal inference mechanism.

In this section, we will demonstrate how similar theories of commonsense reasoning can be modeled in BABY-SIT. For this purpose, we consider the following variant of YSP. Mary loads a gun loaded at some point in time and fires it at Fred at a later time. We would like to reason about the effect of firing the gun. We provide a possible axiomatization in causal theories of Shoham where the gun is loaded at time 0 and fired at 2:

1. $\Box$(0,loads,mary,gun)

[1] While Shoham's causal theories have computationally simple model-theoretic properties, they are not very efficient [13] and have some technical limitations [15]. These issues are not relevant to our discussion.

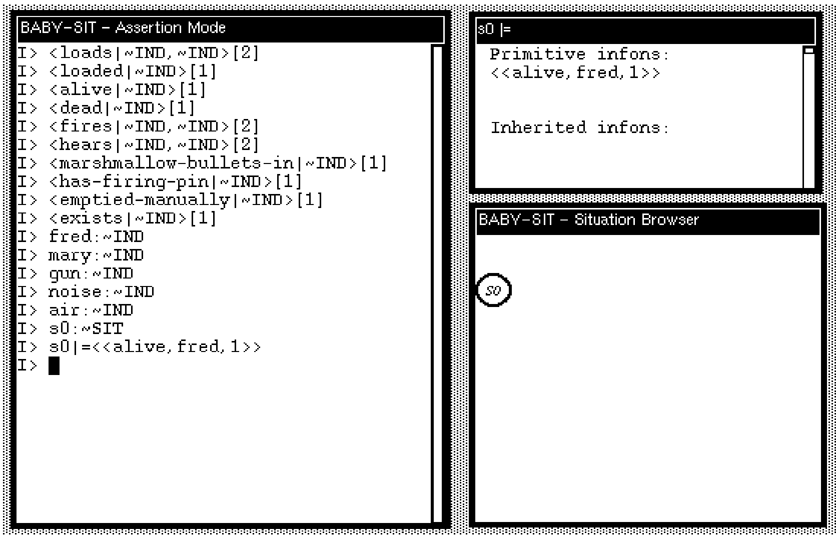

Figure 1: Object declarations for YSP.

2. $\Box$(0,alive,fred)
3. $\Box$(2,fires,mary,gun)
4. $\Box$(t,loads,mary,gun) $\supset$ $\Box$(t+1,loaded,gun), $\forall$t
5. $\Box$(t,alive,fred) $\wedge$ $\Diamond$(t,fires,mary,gun)
   $\wedge$ $\Diamond$(t,exists,air) $\supset$ $\Box$(t+1,alive,fred), $\forall$t
6. $\Box$(t,loaded,gun) $\wedge$ $\Diamond$(t,$\neg$emptied-manually,gun)
   $\wedge$ $\Diamond$(t,$\neg$fires,mary,gun) $\supset$ $\Box$(t+1,loaded,gun), $\forall$t.
7. $\Box$(t,alive,fred) $\wedge$ $\Box$(t,loaded,gun) $\wedge$ $\Box$(t,fires,mary,gun)
   $\wedge$ $\Diamond$(t,$\neg$marshmallow-bullets-in,gun)
   $\wedge$ $\Diamond$(t,has-firing-pin,gun) $\supset$ $\Box$(t+1,$\neg$alive,fred), $\forall$t.
8. $\Box$(t,loaded,gun) $\wedge$ $\Box$(t,fires,mary,gun) $\wedge$ $\Diamond$(t,has-firing-pin,gun)
   $\wedge$ $\Diamond$(t,$\neg$marshmallow-bullets-in,gun)
   $\wedge$ $\Diamond$(t,exists,air) $\supset$ $\Box$(t+1,hears,mary,noise), $\forall$t.

Axioms (1–3) are the boundary conditions. The fourth one is an axiom schema saying that loading a gun makes it loaded. The fifth and the sixth ones are axiom schemata needed for persistence. For instance, the fifth one says that Fred remains alive unless certain conditions obtain. The last three ones are causal axiom schemata. The eighth one, for example, states that Mary's firing a loaded gun causes her to hear a noise unless certain conditions obtain.

We first define individuals such as `fred`, `mary`, etc. in BABY-SIT. Then, we introduce relations such as `alive`, `loads`, `fires`, etc. that hold among these individuals. Figure 1 shows the declaration of these objects in the Assertion Mode.

We consider each snapshot in time as a situation. For instance, Mary's loading the gun and Fred's being alive at time 0 form a state. Any inferred information about future will be

```
xterm
GUNFIRE R7
?S1|={<<precedes,?S1,?S2,1>>,<<precedes,?S1,?S3,1>>} =>
?S3|=<<make-part-of,?S3,?S2,1>>
-

GUNFIRE R6
?S1|={<<part-of,?S2,?S1,1>>,<<precedes,?S1,?S3,1>>}, ?S2|=<<precedes,?S2,?S4,1>>
=>
?S1|=<<make-part-of,?S4,?S3,1>>
-

GUNFIRE R5
?S1|=<<loads,?M,?G,1>> => ?S1|=<<loaded,?G,1>>
-

GUNFIRE R4
?S1|=<<alive,?F,1>> => ?S2|=<<alive,?F,1>>, ?S1|=<<precedes,?S1,?S2,1>>
-
  w: <<exists,air,1>>,
?S1: {<<fires,?M,gun,0>>,<<precedes,?S1,?S3,0>>}

GUNFIRE R3
?S1|=<<loaded,?G,1>> => ?S2|=<<loaded,?G,1>>, ?S1|=<<precedes,?S1,?S2,1>>
-
?S1: {<<fires,?M,?G,0>>,<<emptied-manually,?G,0>>,<<precedes,?S1,?S3,0>>}

GUNFIRE R2
?S1|={<<alive,?F,1>>,<<loaded,?G,1>>,<<fires,?M,?G,1>>} =>
?S2|=<<dead,?F,1>>, ?S1|=<<precedes,?S1,?S2,1>>
-
?S1: {<<has-firing-pin,?G,1>>,<<marshmallow-bullets-in,?G,0>>}

GUNFIRE R1
?S1|={<<loaded,?G,1>>,<<fires,?M,?G,1>>} =>
?S2|=<<hears,?M,noise,1>>, ?S1|=<<precedes,?S1,?S2,1>>
-
  w: <<exists,air,1>>,
?S1: {<<has-firing-pin,?G,1>>,<<marshmallow-bullets-in,?G,0>>}
```

Figure 2: The constraints for YSP.

collected in a new state. For this reason, we create a situation `s0` as the initial state and assert the facts that Mary loads the gun and Fred is alive in `s0`.

For axiom schemata (4–8), we can write equivalent forward chaining constraints in BABY-SIT. $\Box$-conditions of these axiom schemata can be represented by positive schematic infonic propositions while their $\Diamond$-conditions are assumptions and hence can be represented by background conditions in constraints. These constraints are shown in Figure 2. Constraints (R1–R5) correspond to axiom schemata (4–8).

Since manipulation of mathematical expressions is not possible in BABY-SIT, we cannot assign incremental temporal values to the infons during reasoning. Instead, we use the relation *precedes* to represent the fact that a situation immediately precedes another, and hence the infons they support. It may be the case that there are two different situations that immediately succeed a situation. These two situations, however, must represent a unique snapshot in time. For this reason, they must be unified under a single situation. We achieve this by introducing new constraints, R6 and R7, which make one of such situations a part of the other.

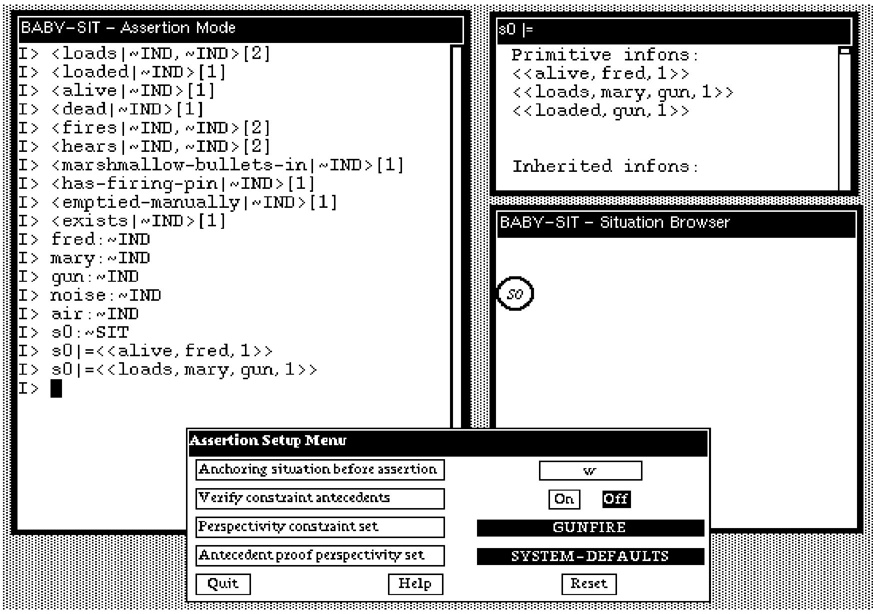

Figure 3: After Mary's loading the gun.

Setting the current forward chaining perspectivity set to GUNFIRE and then asserting the fact that Mary loads the gun causes constraint R1 to be fired and the gun gets loaded in `s0` (Figure 3). Forward chaining over the existing information creates, via firing of constraint R3, a new situation `R3-1` where the gun remains loaded and then creates, via firing of R4, another situation `R4-2` where Fred is alive (Figure 4). Constraint R6 unifies these situations into one in the next activation of forward chaining (Figure 4). Activation of forward chaining once more creates situations `R3-3` and `R4-4` by reasoning over `R3-1` and `R4-2`, respectively (Figure 5).

Subsequent activations of forward chaining unify new states and then create new states by forming new situations in a similar manner. Assume that we iterate forward chaining twice and assert the fact that Mary fires the gun. Then two new situations `R2-7` and `R1-8` are created (by R2 and R1), where Fred ceases to be alive and Mary hears a noise, respectively (Figure 5).

However, if we had asserted the proposition `w` $\models$ `<<exists, air, 0>>` prior to Mary's firing the gun, a new situation, `R2-9`, would be created where Fred would be dead, but Mary would not be able to hear a noise (Figure 6).

4 Conclusion

We have demonstrated how AI problems that require a nonmonotonic approach can be handled in a situation-theoretic computational framework, viz. BABY-SIT. In this direction, we

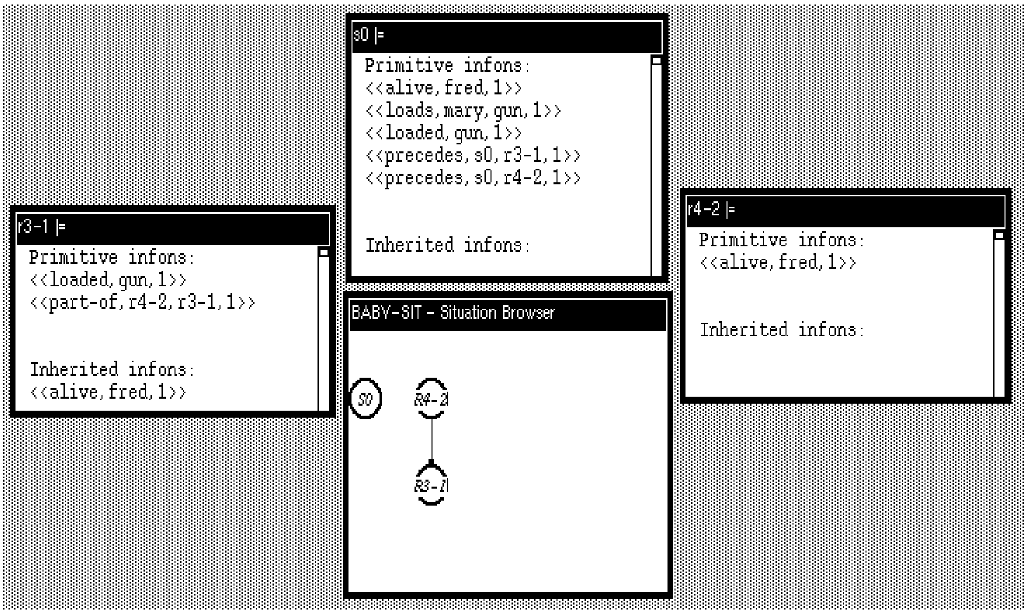

Figure 4: After the second iteration of forward chaining.

have considered the Yale Shooting Problem and taken a simplified approach to axiomatize it in BABY-SIT. By virtue of background conditions and the constraint antecedents that enable us to inquire if an infon is not support by a situation, it is made possible to infer the affects of actions and the facts that persist in time. We are planning to include other examples in the full version of this paper.

References

[1] V. Akman and M. Surav. Contexts, Oracles, and Relevance. In S. Buvaç and R. Fikes, editors, *Working Notes of the AAAI-95 Fall Symposium on "Formalizing Context"*, pages 23–30. AAAI Press, Cambridge, MA, 1995.

[2] A. W. Black. *An Approach to Computational Situation Semantics*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, U.K., 1993.

[3] J. Borota, M. Frank, J. Fry, A. Ito, H. Nakashima, S. Peters, M. Reilly, and H. Schütze. The PROSIT Language, version v1.0. Manuscript, Center for the Study of Language and Information, Stanford, CA, 1994.

[4] K. Devlin. *Logic and Information*. Cambridge University Press, Cambridge, U.K., 1991.

[5] S. Hanks and D. V. McDermott. Nonmonotonic Logic and Temporal Projection. *Artificial Intelligence*, 33:379–412, 1987.

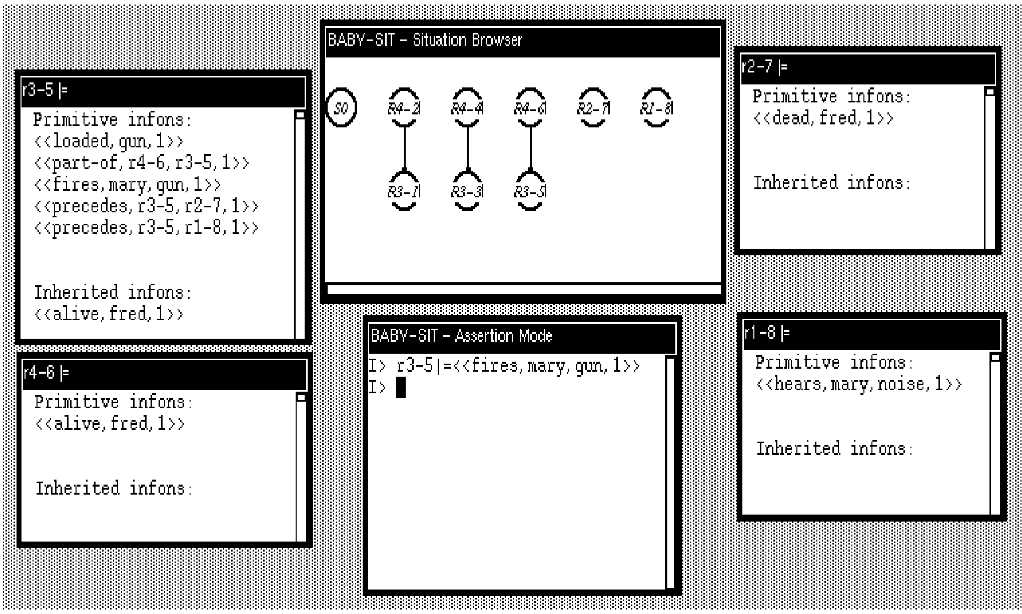

Figure 5: After Mary's firing the gun.

[6] KEE^{TM}. (Knowledge Engineering Environment) Software Development System, Version 4.1. IntelliCorp, Inc., Mountain View, CA, 1993.

[7] J. McCarthy. Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

[8] J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, pages 463–502. Edinburgh University Press, Edinburgh, U.K., 1969.

[9] H. Nakashima, H. Suzuki, P.-K. Halvorsen, and S. Peters. Towards a Computational Interpretation of Situation Theory. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 489–498, Institute for New Generation Computer Technology, Tokyo, Japan, 1988.

[10] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[11] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press, Cambridge, MA, 1988.

[12] E. Tın. *Computational Situation Theory with BABY-SIT.* PhD thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1995.

[13] E. Tın and V. Akman. Computing with Causal Theories. *International Journal of Pattern Recognition and Artificial Intelligence*, 6(4):699–730, 1992.

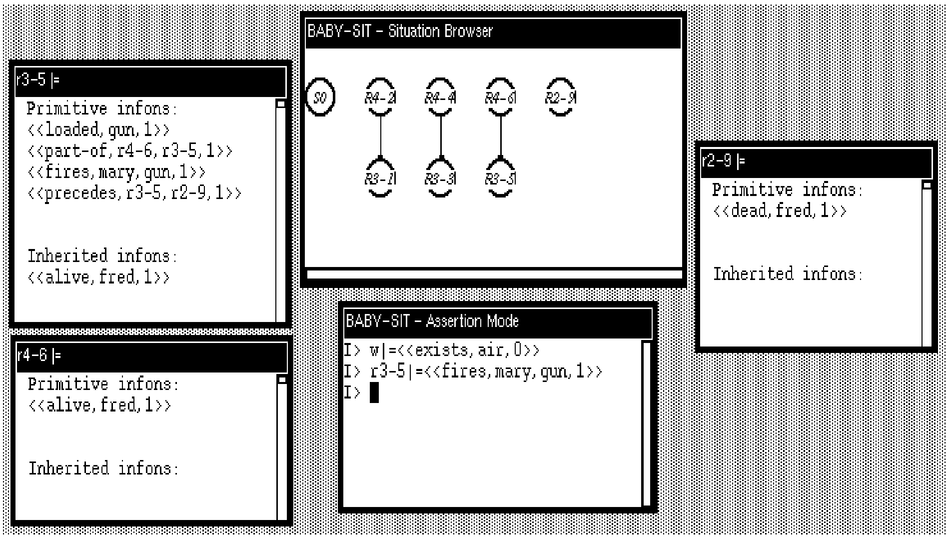

Figure 6: After Mary's firing the gun in vacuum condition.

[14] E. Tın and V. Akman. BABY-SIT: A Computational Medium Based on Situations. In P. Dekker and M. Stokhof, editors, *Proceedings of the 9th Amsterdam Colloquium*, pages 665–681. Institute for Logic, Language, and Computation, University of Amsterdam, Amsterdam, Holland, 1993.

[15] E. Tın and V. Akman. The Logic of Counteraction. *Elektrik: Turkish Journal of Electrical Engineering and Computer Sciences*, 1(3):167–181, 1993.

[16] E. Tın and V. Akman. Computational Situation Theory. *ACM Sigart Bulletin*, 5(4):4–17, 1994.

[17] E. Tın and V. Akman. Situated Processing of Pronominal Anaphora. In H. Trost, editor, *KONVENS '94, 2. Konferenz, Verarbeitung natürlicher Sprache*, Informatik Xpress 6, pages 369–378. Springer Produktions-Gesellschaft, Berlin, Germany, 1994.

[18] E. Tın and V. Akman. Information-Oriented Computation with BABY-SIT. In D. Westerståhl and J. Seligman, editors, *Language, Logic, and Computation: The 1994 Moraga Proceedings*, CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, CA, 1996. to appear.

[19] E. Tın, V. Akman, and M. Ersan. Towards Situation-Oriented Programming Languages. *ACM Sigplan Notices*, 30(1):27–36, 1995.