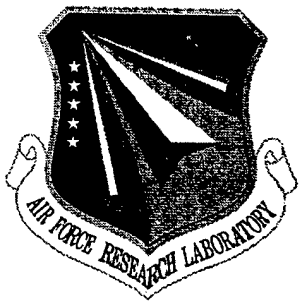
AFRL-IF-RS-TR-2001-273
Final Technical Report
January 2002



INTERACTIVE PLANNING AND MONITORING FOR THE ANCHOR DESK ASSOCIATE

Massachusetts Institute of Technology

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. B165

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-273 has been reviewed and is approved for publication.

APPROVED: 

JOHN SPINA
Project Engineer

FOR THE DIRECTOR: 

MICHAEL TALBERT, Maj., USAF, Technical Library
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTD, 25 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JANUARY 2002	Final Jun 94 - Feb 99

4. TITLE AND SUBTITLE
INTERACTIVE PLANNING AND MONITORING FOR THE ANCHOR DESK ASSOCIATE

5. FUNDING NUMBERS
C - F30602-94-C-0199
PE - 62301E
PR - B165
TA - 00
WU - 01

6. AUTHOR(S)
Gary Borchardt, Boris Katz, and Patrick Winston

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology
545 Technology Square
Cambridge Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

Air Force Research Laboratory/IFTD
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2001-273

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: John Spina/IFTD/(315) 330-4032

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes a technique for collaborative, human-computer accomplishment of planning and monitoring activity in a range of domains. The technique, called split-confirmation, provides for a partitioning of duties between the two participants, such that each vouches for the likelihood of occurrence for a distinct temporal portion of particular events or sets of events describing past, present or future activity. Assignment of duties is controlled by the human participant and can be altered at will in response to changes in the overall context for planning and monitoring activity.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Knowledge Bases, Planning, Scheduling, Knowledge Understanding	36
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

List of Figures

1 Background

Planning and monitoring occur in a wide range of contexts, from small-scale corporate purchasing efforts to major civil engineering projects. Planning and monitoring are also complex and difficult, requiring us to anticipate possible future situations, identify goals to be achieved, identify sequences of actions that might be attempted in particular situations to accomplish these goals, anticipate possible consequences and interactions for successfully and unsuccessfully completed actions, select particular sequences of actions to be performed, recognize occurrences of events as they transpire, identify discrepancies between planned and observed events, and more.

Given the current state of computer technology, only some of these subtasks can be adequately performed on the computer. In general, those subtasks having to do with exhaustive scanning and testing of alternatives appear to be well suited to performance by computer, while those subtasks having to do with insight and judgment appear to be best handled by humans.

Two general approaches have been taken in combining human and computer effort toward the tasks of planning and monitoring. The first approach issues largely from the artificial intelligence community and is based on symbolic or sometimes numerical computer representation of relevant knowledge from the domain in which planning and monitoring is to be performed. Examples of this approach include work in the areas of planning, monitoring, diagnosis and reasoning [11]. In this approach, the computer assumes the burden of responsibility for carrying out the various subtasks of planning and monitoring, with attendant humans contributing more or less peripherally by supplying definitional and factual knowledge and by specifying goals to be accomplished.

The second approach issues largely from the management science community and is based on a more coarse-granularity computer representation of various properties and relationships existing between the events in a plan, with the events themselves specified in a human- understandable form as text strings describing their nature. Examples of this approach include computerized tools based on the use of GANTT charts and PERT charts [4, 8, 9]. In this approach, humans assume the burden of responsibility for carrying out the various subtasks of planning and monitoring, with attendant computer processing playing a more or less peripheral role in performing such tasks as scheduling the execution of previously-planned actions in light of applicable resource constraints, computing critical paths for the completion of various actions, recording action completion times, and temporally shifting actions in response to delayed or early completion of other actions.

Both of these approaches fail to fully utilize one or other of the participants: human or computer. A preferable approach would be to construct a highly interactive system that allows these participants to contribute each according to their relative strengths, exchanging partial results as they go. Facilitating such interac-

tion is in part a representational problem: computer-oriented representations (e.g., the symbolic or numerical representations of the first approach) are difficult for humans to use, and human-oriented representations (e.g., the textual representations of the second approach) are difficult for computers to use.

A more recent approach to planning and monitoring offered by the artificial intelligence community is that of "mixed-initiative planning," in which a computer and human participant interact via manipulation of graphical diagrams, menu-oriented commands or an exchange of natural language utterances [6, 3, 12]. While these approaches do offer a degree of interactivity between the computer and human participants, the underlying symbolic representations still hinder human input and interaction at the more detailed task level of forming expectations about the desirability or likelihood of occurrence for particular events in particular circumstances.

Recent approaches from the management science community also support a degree of interaction between human and computer participants. Work in decision support systems (e.g., see [4]), including work in data warehousing and database mining, provides for such interaction by exploiting the computer manipulability and reasonable human understandability of relational databases (see [5]). However, these approaches typically do not encode sufficient background knowledge and detail in the modeling of events as to support extensive computer reasoning and interaction about event occurrences.

One representation that offers promise as a foundation for the design of interactive planning and monitoring systems is the *transition space* representation [1, 2]. This representation combines two important ingredients. First, it captures detailed specifications of what happens during the temporal unfolding of particular events. Thus, computer processing can be used to test particular specifications of events for consistency with the specifications of particular situations or other particular events. Second, individual assertions in the representation can be expressed in simple, stylized English. Thus, humans can easily understand the represented information, examine its role in particular computer-generated conclusions, and steer the course of computer processing.

The *split-confirmation* approach presented in this paper takes advantage of the computer manipulability and human understandability of an underlying representation such as the transition space representation. In this approach, either participant may assume the responsibility of vouching for the likelihood or desirability of particular portions of observed or anticipated events considered during planning and monitoring. The partitioning of duties is determined by the human participant, based on his or her knowledge of the surrounding context in which particular event occurrences fit into the planning and monitoring process. Furthermore, as this context may change during planning and monitoring, the partitioning of duties set by the human participant may require modification at any time.

The next two sections present the split confirmation technique and the implemented IMPACT system in overview, focusing on capabilities enabled by each. Following these two sections, the next two sections describe the approach in greater detail, focusing on the underlying knowledge representation framework used in IMPACT and the split- confirmation algorithm used in IMPACT. The paper concludes with a brief discussion of current status and future work.

2 Overview of the Split-Confirmation Technique

Central to planning and monitoring is the idea of forming expectations about the occurrence of particular types of events in particular circumstances. The circumstances may be past, present or future, and the events themselves may be actions performed by those doing the planning, actions performed by others, and other events occurring independently or as a consequent or antecedent of other expected events. Typically, the formation of such expectations proceeds from postulating the occurrence of a particular type of event to determining suitable participants for that event occurrence. This can be abstracted to an operation of instantiating—that is, finding suitable values for variables occurring within—an event "pattern" in the context of a repository of static facts and other event occurrences.

As a simple example, suppose that the specification of such an event pattern includes information on what happens during various stages of the temporal unfolding of that event—assertions such as:

- Between "1998 01/01" and "1998 02/01", "The AEC341 Frame Buffer/Display" BECOMES manufactured at "a facility", and
- Between "1998 02/01" and "1998 03/01", the utilization of "a group of workers" INCREASES.

The process of instantiating that event pattern in the context of static facts and other event occurrences will produce an enumeration of acceptable values for variables such as "a facility" and "a group of workers"—values such as "The Denver Facility" and "Denver Testing Group #3". If the instantiation process yields no such values, this can be taken to imply that the event type in question cannot occur in the specified circumstances. If more than one acceptable value is identified for any variable, then this can be taken to imply that more than a single variation of the event type may be possible in the specified circumstances. Also, as a degenerative case, if there are no variables within the event pattern, then such an instantiation process can serve simply as a compatibility check, to test whether that particular event instance can or cannot occur in the specified circumstances.

However, computer and human participants in the planning and monitoring process may contribute in largely different ways to the instantiation process. While

it is natural for the computer participant to maintain the integrity of its knowledge base, or "repository," by continually testing for conflicts between newly considered events and previously accepted facts, changes and events, the act of ensuring an absence of conflicts for a newly considered event leaves open the question of whether or not there exists any *justification* for believing the facts and changes specified by that event. With respect to determining justification for these facts and changes, a natural partitioning of duties arises between the computer and human participants, such that they each vouch for a different temporal portion of the event. This partitioning of duties is illustrated below in the context of several types of events:

- For a "reported" event—that is, an event whose occurrence the human takes on faith from a reliable external source—the computer participant will simply accept the event's circumstances in their entirety on the basis of the human's voucher for the event.

- For a "planned" event—that is, an event for which some person or entity will see to it that the event occurs if its initial preconditions are met—the computer may check to see if there is independent justification for believing that the event's initial conditions will prevail, while it must accept on faith the human's voucher that the remainder of the event will be executed in such a case.

- For an "enabling" event—that is, an event for which some entity or process known to the human will ensure that a certain set of facts and changes will occur, leading up to a specified final set of conditions—the computer can test to see whether there is independent justification for believing that the final conditions do indeed hold, while it must accept on faith the human's voucher that the initial portion of the event occurs as specified.

- For a "consequent" event—that is, an event for which we have confidence that if an initial temporal portion of the event occurs (more than just the initial conditions), then the remainder of the event will occur—the computer can test to see if there is independent justification for believing that the initial portion of the event occurs, while it must accept on faith the human's voucher that the remainder of the event will occur.

- For an "antecedent" event—that is, an event for which we have confidence that if a final temporal portion of the event occurs, then the initial portion of the event has occurred before that—the computer can test to see if there is independent justification for believing that the final portion of the event has occurred, while it must accept on faith the human's voucher that the initial portion of the event has occurred.

- Finally, for an "inferred" event—that is, an event for which we cannot assume any part of its occurrence based on other information available to us—the computer must find independent justification for all temporal portions of the event, as the human has not offered a voucher for any portion of the event's occurrence.

It should be noted that humans are the best judges of which classifications to assign to particular events. Such judgments depend on knowledge of the reliability of external information sources, commitments of various parties, and likelihoods of particular occurrences continuing from or preceding particular sets of circumstances.

In all of the above cases, the partitioning of duties for finding independent justification for circumstances falls along *temporal* lines, with the computer vouching for one portion of an event's occurrence, while the human vouches for the remaining portion of the event's occurrence. One might imagine a temporal "dividing line" drawn at a human-designated point between the starting and ending time points for an event, such that the computer must vouch for that portion of the event falling on one side of the dividing line, and the human must vouch for that portion falling on the other side of the dividing line.

Of course, one might imagine dividing up events according to other criteria, too. For example, the partitioning of responsibility could be set according to where particular components of information might be found. In this paper, the partitioning is always along temporal lines; however, the extension of the split-confirmation technique to such other criteria follows in a fairly direct manner.

The following paragraphs describe a particular variant of split- confirmation that has been found to be both effective and conceptually simple. In this variant, the computer vouches for its portion of a considered event by first applying two types of contextual assumptions, described below, to estimate what might otherwise be expected to occur over the specified time interval, then applying constraint propagation (see, for example, [13]) to determine sets of acceptable values for variables appearing within the event pattern. The reader might note that other approaches could also be acceptably used by the computer participant to complete its reasoning duties, still within the spirit of the general split-confirmation method. Examples of such other approaches include generate-and-test, the use of heuristic evaluation functions, case- based or analogical reasoning, and the application of forward-chaining rule sets.

The two types of contextual assumptions applied by the computer participant are as follows:

- a persistence assumption—that is, an assumption that circumstances pertaining to particular points in time continue to hold at future time points, up to the point where different circumstances are known to prevail (see, for example, [7]), and

- a closed-world assumption—that is, an assumption that all attributes and relationships are "absent" or "false" for all objects not explicitly known to possess those attributes and relationships (see, for example, [10]).

For reported events, these assumptions are not applied, as the computer has not been directed to vouch for any portion of the event's occurrence. For planned, consequent and inferred events, the assumptions are applied in tandem from all points prior to the event pattern's period of occurrence up through some initial portion of the event pattern's period of occurrence. For enabling and antecedent events, the assumptions are applied in the reverse direction, in tandem from all points after the event pattern's period of occurrence back through some final portion of the event pattern's period of occurrence.

In all cases, the persistence and closed-world assumptions have the effect of augmenting the surrounding repository of static facts and other event occurrences to include new assertions that may possibly conflict with particular instantiations—particular sets of variable assignments—for the target event pattern, thereby restricting the range of acceptable instantiations for that event pattern. However, the extent to which the contextual assumptions provide additional constraint to the instantiation process is limited by the fact that these assumptions provide only *default* information that can be overridden by other, explicitly posted assertions in the repository. The net effect of applying the contextual assumptions, then, is to force acceptance of only those instantiations of a target event pattern whose *non-default* facts and changes in the period of coverage of the assumptions meet with independent substantiation in the form of explicit repository assertions. This is precisely what is needed to support differential treatment of the six classifications of events as listed above. In particular:

- For a reported event, the contextual assumptions are not applied. Thus, constraint propagation ensures merely that instantiation of the reported event generates no direct conflicts with other facts and changes known to have occurred at the same time.

- For a planned event, the contextual assumptions are applied from prior time points up to the starting time point for the event. Thus, constraint propagation ensures that, in addition, all non-default initial conditions of the event instance are independently substantiated in the repository.

- For an enabling event, the contextual assumptions are applied from successive time points back to the ending time point for the event. Thus, constraint propagation ensures that, in addition to the absence of direct conflicts with repository information, all non-default final conditions of the event instance are independently substantiated in the repository.

- For a consequent event, the contextual assumptions are applied from prior time points up through the end of the first time interval of the event. Thus, constraint propagation ensures that not only non- default initial conditions, but also all *changes* occurring within the initial time interval of the event instance are independently substantiated by assertions in the repository.

- For an antecedent event, the contextual assumptions are applied from successive time points back to the beginning of the last time interval of the event. Thus, constraint propagation ensures that not only non-default final conditions, but also all *changes* occurring within the final time interval of the event instance are independently substantiated by assertions in the repository.

- Finally, for an inferred event, the contextual assumptions are applied from prior time points through the end of the entire event. Thus, constraint propagation ensures that non-default initial conditions plus all changes occurring throughout the event instance are independently substantiated by assertions in the repository.

In addition to controlling the degree to which the contextual assumptions are applied during constraint propagation, the human participant interacts with the computer participant in three supporting ways—in all, yielding a considerable degree of interactivity between the human and computer participants. First, the human participant initiates the constraint propagation process itself by requesting processing steps such as the verification of externally supplied information, the enumeration of possible effects of a particular event, and so forth. Second, the human participant can explore different hypothetical scenarios by alternatively "activating" and "deactivating" particular events with respect to the constraint propagation process. Third, in those cases where constraint propagation comes to a halt with more than a single value remaining for one or more of the variables, the human participant may explore the range of alternate instantiations for the event by choosing specific values for specific variables, then allowing the constraint propagation process to continue.

In the context of planning and monitoring, it is indeed a frequent occurrence for planned, enabling, consequent and antecedent events to admit multiple possible instantiations. Thus, quite often, the constraint propagation process will come to a halt with more than a single value remaining for one or more of an event pattern's variables, and the human participant will be needed to steer the processing toward particular instantiations and away from others, based on the human participant's knowledge, experience and insight. Consequently, it is implementationally advantageous to employ a faster, less complete method for constraint propagation than, for example, the Waltz method, thereby minimizing the amount of time spent on alternative instantiations that may never be explored by the human participant. This streamlined method simply enumerates for each variable those values that

appear in at least one acceptable complete instantiation for each of the assertions within the specification of the temporal unfolding of the event pattern. The current invention also generalizes to use with the Waltz method (see, for example, [13]) or any other method for constraint propagation, if this is so desired.

3 Overview of the IMPACT System

The split-confirmation technique is embodied in the design and functionality of IMPACT, a domain-independent, interactive planning and monitoring system that enables its users to incrementally develop, modify and track the execution of plans on the basis of real-world observables. IMPACT comprises about 20,000 lines of code written in C and uses an X/Motif interface and stylized English to interact with its users. Figure 1 presents a sample screen shot of the IMPACT system.

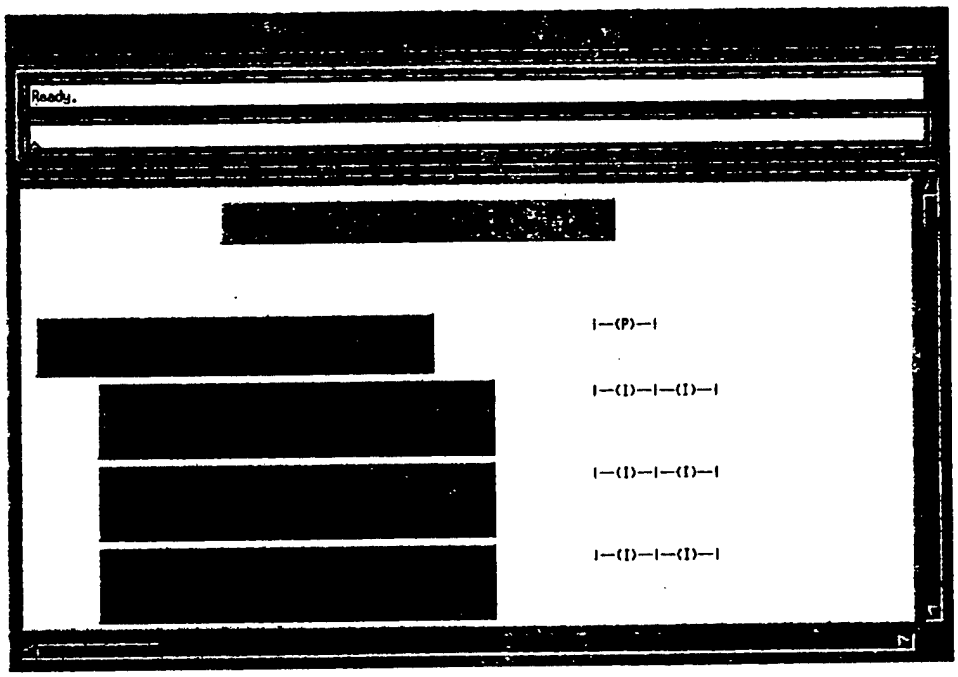


Figure 1: A screen shot of the IMPACT system displaying four events—a proposed action and three anticipated consequences—occurring as part of a scenario labeled "Scenario #1".

Abstractly, IMPACT behaves much like a "spreadsheet for events." Whereas a standard spreadsheet allows users to gather numerical information, reconcile accounts received from different sources, graphically organize the information for inspection, propagate numerical constraints, and examine alternative scenarios by

inserting hypothetical values, IMPACT allows users to gather human- or machine-generated field reports regarding observed activities, reconcile reports from different sources, graphically organize this event-oriented information, propagate constraints among the represented events, and perform "what-if" reasoning by exploring possible consequences of hypothetical or planned events.

At present, IMPACT handles reported, planned, consequent and inferred events. Current plans call for enabling and antecedent events to be included in the near future.

Figure 2 presents a block diagram of the IMPACT system. The IMPACT system contains three main components: an interface component that communicates with humans and data files, a reasoning engine that executes the split-confirmation algorithm, and a repository of information.

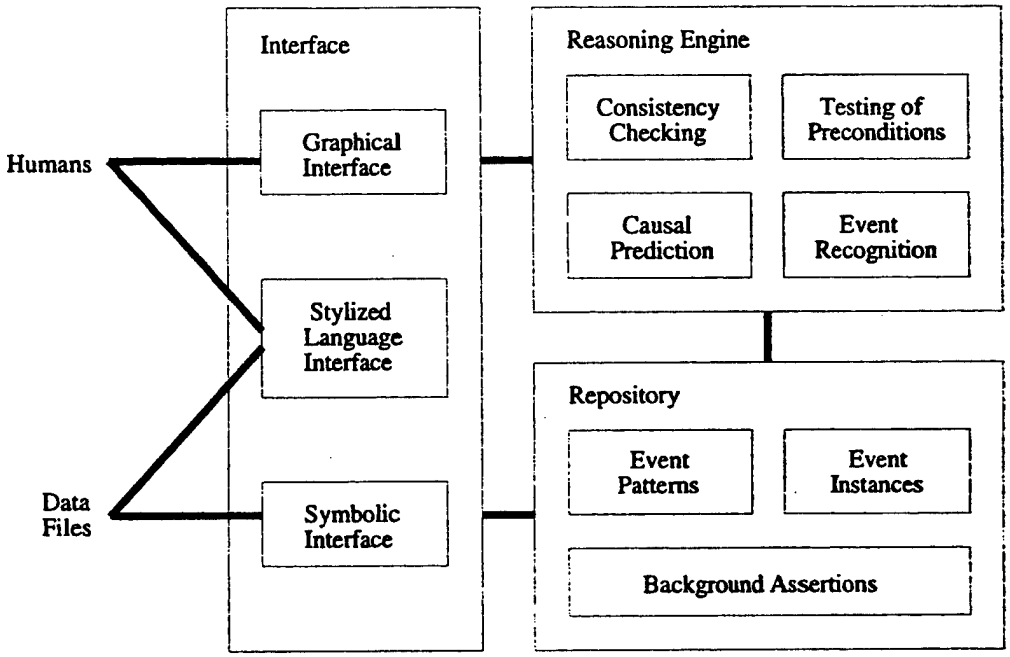


Figure 2: A block diagram of the IMPACT system, indicating interfaces to humans and data files, functional capabilities, and maintained information types.

The repository contains three varieties of information: (1) background assertions, which are individual statements of facts or changes, (2) event instances, which are sets of assertions that collectively describe what happens to selected properties of participating objects over particular intervals of time, and (3) event patterns, which are like event instances, except that their assertions may contain variables in place of one or more time points or participating objects.

IMPACT can perform four principal reasoning operations, including verifying that reported events are consistent with other known occurrences, checking preconditions of planned events, performing causal prediction and explanation, and recognizing occurrences of events from low-level change information. Supporting these operations is a single underlying technique in which IMPACT and a human operator interactively instantiate event patterns by replacing their variables with constants, thus producing new event instances.

Within the process of event pattern instantiation, the task of the reasoning engine is to characterize the set of suitable values that might be used in place of each variable within an event pattern, such that the event pattern remains consistent with a currently "active" subset of the event instances and background assertions. Event pattern instantiation is carried out through an iterative process in which the reasoning engine is engaged repeatedly on the assertions of the event pattern, interspersed with possible guidance from the human operator in the form of specific, chosen values for particular variables.

The following paragraphs describe capabilities of the IMPACT system that are directly attributable to the split-confirmation technique. Given a surrounding repository of facts and events that constitute a particular set of circumstances, a core set of supported capabilities is as follows:

- IMPACT can identify which of many conceivable events might account for a given change and be consistent with the repository.
- IMPACT and a human user can interactively explore the range of specific instantiations of an event that are consistent with the repository.
- IMPACT can verify that specific, observed facts and events are consistent with the repository.
- IMPACT can identify which of many conceivable actions can be taken.
- IMPACT and a human user can interactively explore the range of specific instantiations of an action that can be taken.
- IMPACT can verify that a particular intended action or sequence of actions is possible.
- IMPACT can identify which of many conceivable cause-effect occurrences can take place.
- IMPACT and a human user can interactively explore the range of specific instantiations of a possible cause-effect occurrence.
- IMPACT can verify that a particular cause-effect occurrence can happen.

- IMPACT can identify which of many conceivable events has occurred on the basis of what is known in the repository.

- IMPACT and a human user can interactively explore the range of specific instantiations of an event that have occurred.

- IMPACT can verify that a particular event has occurred.

In addition, a human user of IMPACT can perform a number of useful operations by requesting a *reclassification* of particular events in IMPACT's repository. Such reclassifications take particular advantage of human abilities to provide insight and judgment in determining the appropriate context for processing particular components of information in light of evolving external circumstances and the progression of activity in the planning and monitoring process:

- A human user can request that an intended goal event be initially classified as "reported" so that IMPACT may verify that it does not conflict with other expected occurrences. Later, the user can request a reclassification of this event as "consequent" or "inferred" so that IMPACT may verify that the event follows from planned activity or is otherwise directly substantiated by observed activity.

- A human user can request that a "reported" event be reclassified as "planned" so that IMPACT may determine whether or not the event could conceivably have been carried out intentionally by some party, and if so, which prior events were responsible for ensuring that the initial conditions of the event were established.

- A human user can request that a "reported" event be reclassified as "consequent" so that IMPACT may determine whether or not the event could conceivably have occurred as an effect of one or more other events. In this manner, explanations for events can be constructed, possibly extending to entire sequences of events.

- A human user can request that a "reported" event be reclassified as "consequent" or "inferred" if the human has lost confidence in the external source of information for that event. IMPACT will then accept the event into the repository only when it has established that the event could be anticipated from other known occurrences or that it is directly known to have occurred on the basis of other known occurrences.

- A human user can request that a "planned" event be reclassified as "inferred" if the human has lost confidence that a responsible party will ensure that the event will occur if its initial conditions are met or simply wishes to have

an after-the-fact confirmation of the occurrence of the event. IMPACT will then accept the event into the repository only when it can substantiate all components of the event's occurrence from other known occurrences.

- A human user can request that a "consequent" event be reclassified as "inferred" if the human feels that the event is an unlikely occurrence given the occurrence of the initial portion of that event or wishes to have an after-the-fact confirmation of the occurrence of the event. IMPACT will then accept the event into the repository only when it can substantiate the remaining components of the event's occurrence.

- A human user can request that an "inferred" event be reclassified as "consequent" or "planned" if the human has independent knowledge that the event is a consequence of other events or an action carried out by some party. Such action may allow other events to be removed from the repository without affecting the status of the event in question.

- A human user can request that an "inferred," "consequent" or "planned" event be reclassified as "reported" if the human has independent knowledge that the event has been observed to have occurred. Such action may allow other events to be removed from the repository without affecting the status of the event in question.

- A human user can request that an observed event with one or more variables in its description be initially classified as "consequent" or "planned," so that the added contextual constraints provided by these classifications may help in determining suitable values for the variables. Later, if the human user feels that no other instantiation could reasonably account for the observed event, the user can request that the generated event instance be reclassified as "reported."

Taken together, these capabilities enable the human and computer participants to interactively engage in a wide range of subtasks to the planning and monitoring process, including: projecting future circumstances, setting goals, identifying alternative actions, exploring possible effects of those actions, comparing alternative plans, merging independently developed plans, accommodating information regarding recent occurrences, identifying discrepancies between planned and observed occurrences, and developing updated plans to accommodate unexpected occurrences. Computer processing is brought to bear in the advantageous role of exhaustively scanning and testing data representations. In turn, human effort is brought to bear in the advantageous role of providing insight and judgment, not only in selecting events for consideration in various capacities, but also in controlling the reclassification of these events in light of evolving external circumstances and the progression of activity in the planning and monitoring process.

4 IMPACT's Representational Framework

In order to facilitate implementation of the split-confirmation technique, IMPACT provides a supporting framework of representational and processing capabilities. This section outlines these capabilities, while the next section details the algorithm itself.

Serving as a foundation for IMPACT's supporting framework of capabilities is its use of the transition space representation to encode and reason about time, events and causality [1, 2]. The Transition Space representation is designed to be both computer-manipulable—consisting of assertions in a constrained form of predicate logic—and human-understandable—being easily converted to and from a simple, stylized English rendering of individual statements about circumstances and events.

Figure 3 illustrates how the transition space representation is used to encode information in IMPACT's repository. The background assertions, event instances and event patterns in the repository can be viewed as distributed along two dimensions: (1) their time of occurrence (e.g., January 1, 1998), and (2) which particular attributes of which particular objects they concern (e.g., the operational status of a particular mainframe computer).

There are two varieties of background assertions: (1) facts, which hold at individual time points (e.g., "On January 1, 1998, the Corporate Headquarters mainframe system is operational.") and (2) changes, which hold across two time points (e.g., "Between January 1, 1998 and January 2, 1998, the Corporate Headquarters mainframe system ceases to be operational."). Taken collectively, a particular set of background assertions can be viewed as populating scattered points and stretches of the two-dimensional diagram, but leaving large areas unspecified. For these unspecified areas, we simply have no information concerning the status or changing status of the indicated attributes of the indicated objects.

As described previously, in order to "fill" the unspecified areas of the conceptual diagram, two commonly-used assumptions can be applied. The first is a persistence assumption. This assumption states that whatever is true at a particular point in time continues to be true at future time points, up to some point determined by the particular version of the persistence assumption applied (e.g., continuing for 7 days, or, continuing indefinitely). The second is a closed-world assumption. This assumption states that in the absence of explicit information that something is true at a particular time point, that thing is assumed to be false. The persistence assumption is depicted graphically by a rightward arrow in the diagram, while the closed-world assumption is depicted graphically by a stretch of X's where no facts or changes appear. Applied together, the two assumptions have the effect of specifying a status, one way or another, for each and every point in the two-dimensional diagram.

Event instances can be viewed as larger rectangles in the two-dimensional space,

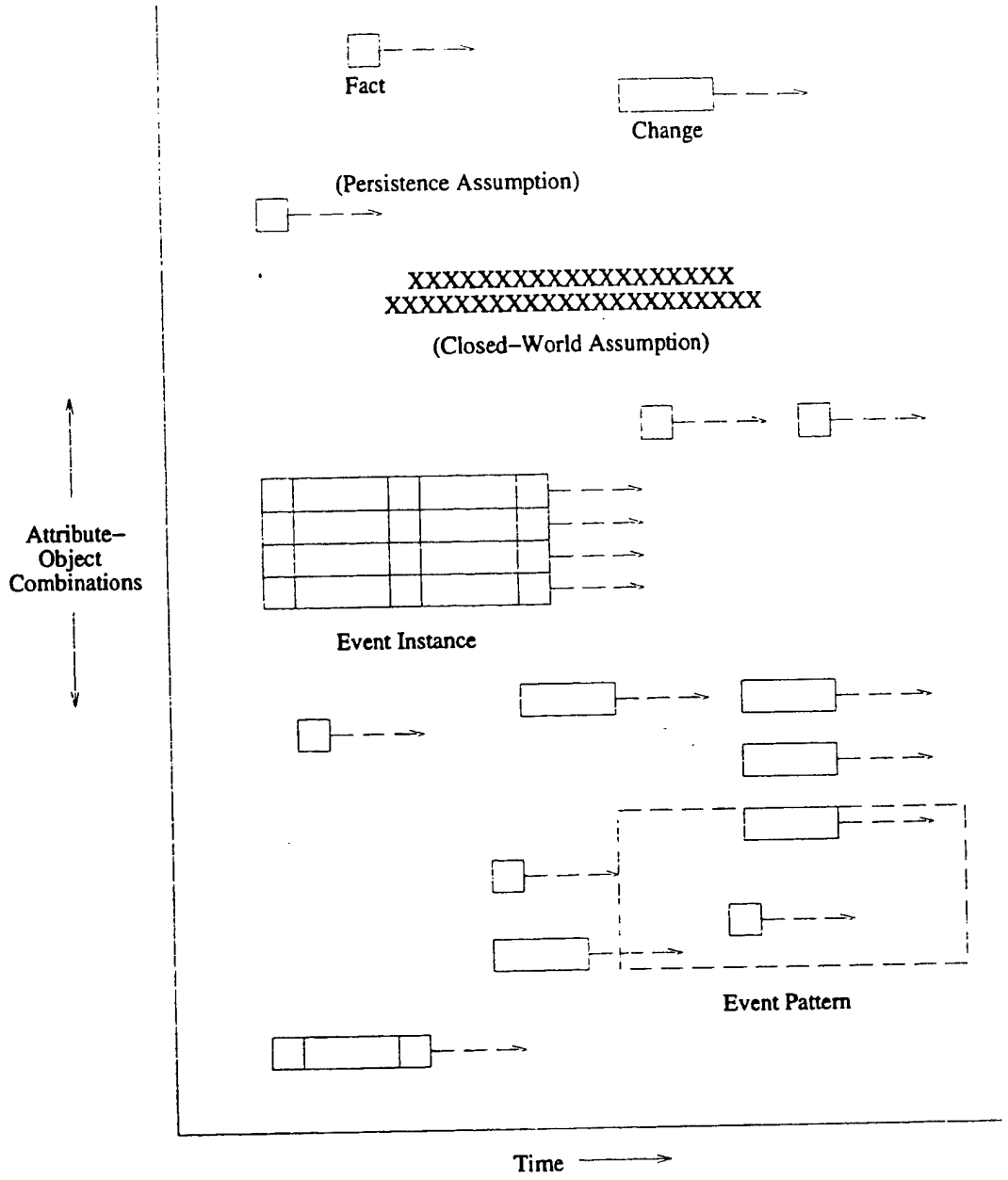


Figure 3: A conceptual illustration of IMPACT's repository, its contents, and operations that are performed on these contents.

being organized collections of facts and changes concerning particular attributes of particular objects and covering particular, consecutive intervals of time. Event patterns, because they can contain variables in place of objects, can be viewed as even

larger rectangles, extending over entire ranges of possible attribute-object combinations and time points. Each event pattern can be transformed into an event instance by replacing its variables with constants. To test whether a particular instantiated version of an event pattern is consistent with the repository, one conceptually constructs the corresponding event instance, then checks to see whether the repository contains conflicting information at the same points and stretches addressed by the event instance. Constraint propagation can then be employed to simultaneously rule out many unsuitable values for all of the variables within an event pattern, based on conflicts that appear with assertions in the repository. This goes a long way toward determining which particular instantiated versions of the event pattern are consistent with the repository.

Borrowing the notational language of Figure 3, Figure 4 illustrates how the six types of events can be handled differentially by specifying a temporal dividing point between the starting and ending time points for the event, such that the computer participant must vouch for the occurrence of all activity either proceeding or following, and possibly including, the temporal dividing point, and the human participant must vouch for the occurrence of all remaining activity in the event. The computer participant accomplishes its assigned duties by applying the persistence and closed-world assumptions throughout its assigned segment of the event. As described previously, the assumptions are not applied for reported events, are applied from prior time points up through part or all of the event for planned, consequent and inferred events, and are applied from subsequent time points back through part of the event for enabling and antecedent events.

5 Split-Confirmation Algorithm and Example

Figures 5, 6, 7 and 8 present the split-confirmation technique's event instantiation algorithm and its subparts. The algorithms are structured in such a way as to distinguish logical components that can be replaced by alternate methods. In the implementation of these algorithms, it is possible to restructure them slightly for increased efficiency. For example, if access time for the repository is a concern, the algorithm for generating related assertions (Figure 7) can be extracted to precede the second step in the instantiation algorithm (Figure 5), with additional steps inserted to prune the lists of related assertions as variables are replaced with constant values. Or, if access time for the repository is not a concern, but temporary memory usage is, the algorithm for generating related assertions can be merged with the first inner loop in the algorithm for completing the determination of candidate variable values (Figure 8) in such a way as to permit immediate disposal of generated assertions.

The algorithms in Figures 5, 6, 7 and 8 are expressed in pseudo-code, which is further elaborated here by means of a skeleton example. For simplicity, the ex-

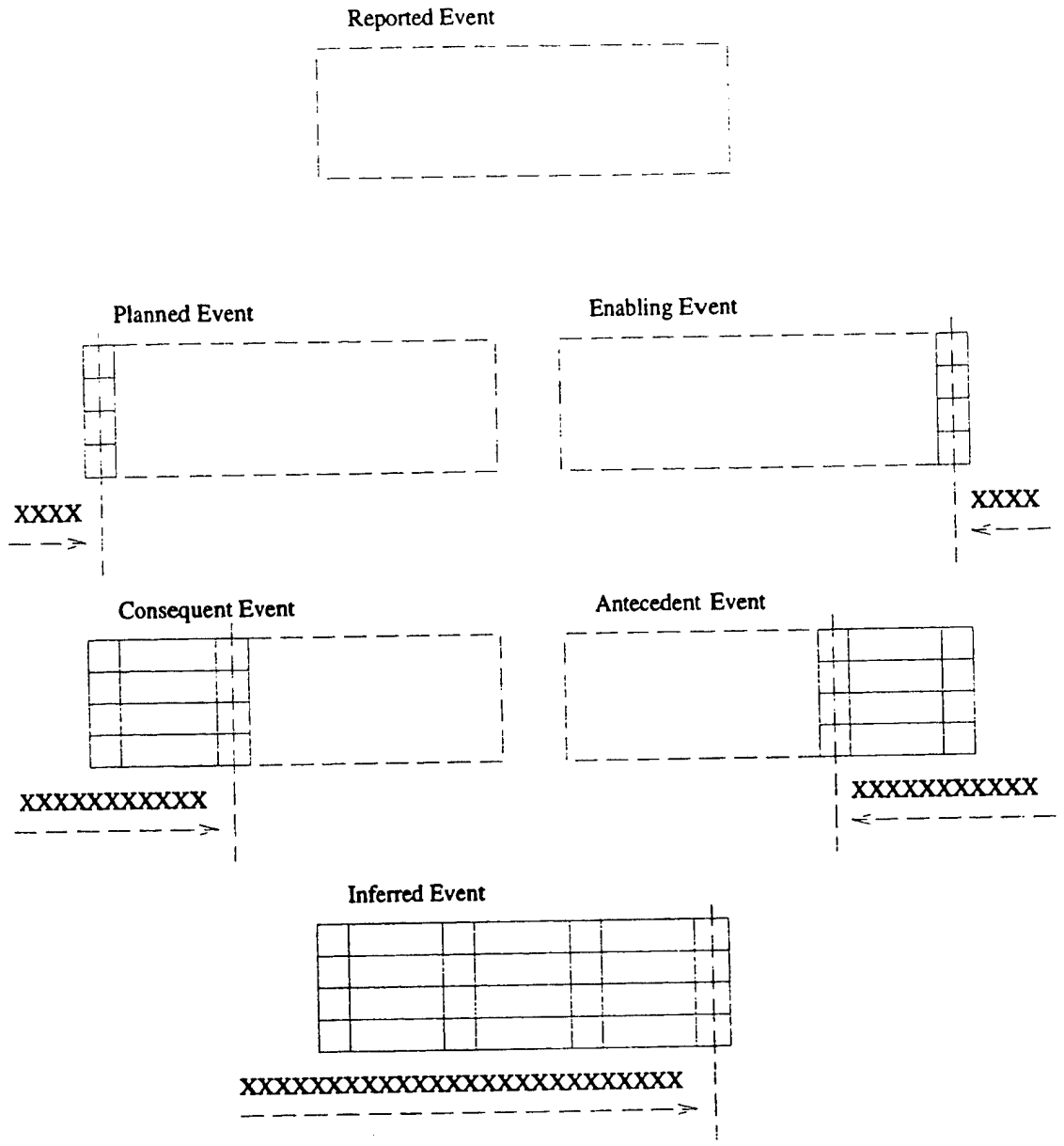


Figure 4: Varied application of contextual assumptions in processing the six types of events.

ample involves only transition space assertions describing changes, not assertions describing instantaneous circumstances. To include such assertions describing instantaneous circumstances follows in a straightforward manner from the discussion presented here.

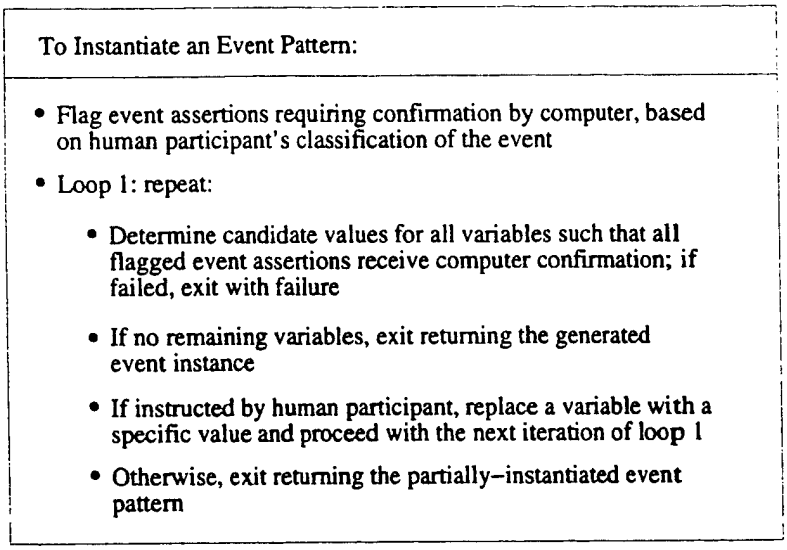


Figure 5: Split-confirmation algorithm, part 1.

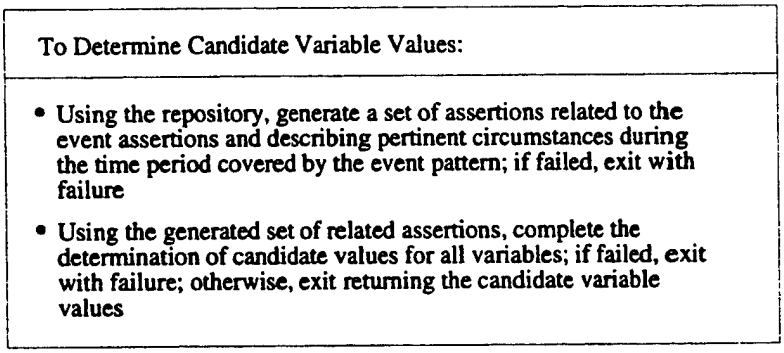


Figure 6: Split-confirmation algorithm, part 2.

The example concerns a manufacturing context, in which an event pattern containing four assertions is to be instantiated. The event pattern describes a causal expectation that when a machine stops being used for production of a particular product, the production time for that product may subsequently increase. The event pattern covers two intervals of time and has been mapped by the human paticipant so that the referenced time points are the specific dates "1998 01/02", "1998 01/03" and "1998 01/04".

The four assertions of the event pattern are as follows:

1. Between "1998 01/02" and "1998 01/03", "a machine" CEASES TO BE

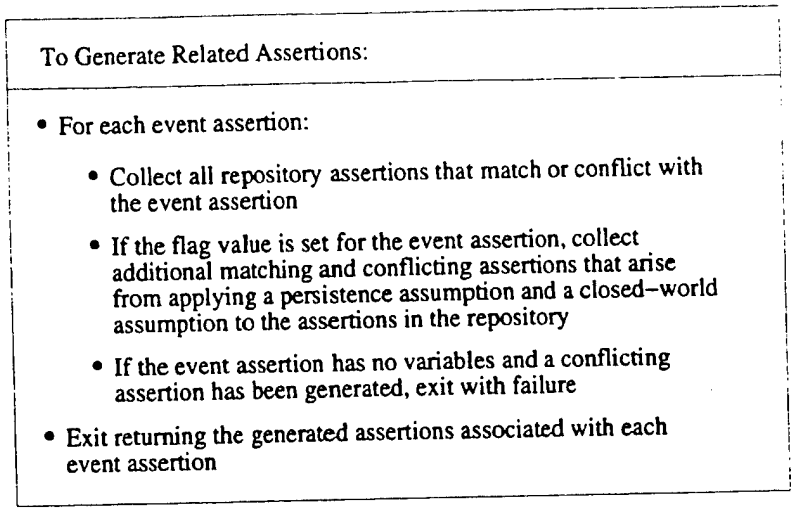


Figure 7: Split-confirmation algorithm, part 3.

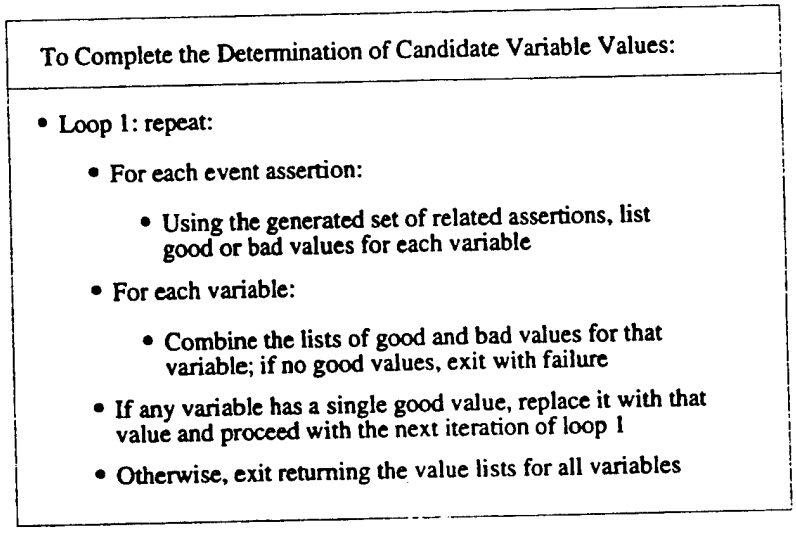


Figure 8: Split-confirmation algorithm, part 4.

utilized for "a product".

2. Between "1998 01/02" and "1998 01/03", the production time for "a product" DOES NOT CHANGE.

3. Between "1998 01/03" and "1998 01/04", "a machine" DOES NOT BECOME utilized for "a product".

4. Between "1998 01/03" and "1998 01/04", the production time for "a product" INCREASES.

In addition, we assume the presence of nine background assertions in the repository, as follows:

1. Between "1998 01/01" and "1998 01/02", "Machine #1" DOES NOT CEASE TO BE utilized for "Product #1".
2. Between "1998 01/01" and "1998 01/02", "Machine #2" DOES NOT CEASE TO BE utilized for "Product #2".
3. Between "1998 01/01" and "1998 01/02", "Machine #3" DOES NOT CEASE TO BE utilized for "Product #1".
4. Between "1998 01/01" and "1998 01/02", "Machine #3" DOES NOT CEASE TO BE utilized for "Product #2".
5. Between "1998 01/01" and "1998 01/02", the production time for "Product #1" DOES NOT CHANGE.
6. Between "1998 01/01" and "1998 01/02", the production time for "Product #2" DOES NOT CHANGE.
7. Between "1998 01/02" and "1998 01/03", "Machine #1" CEASES TO BE utilized for "Product #1".
8. Between "1998 01/02" and "1998 01/03", "Machine #3" CEASES TO BE utilized for "Product #2".
9. Between "1998 01/03" and "1998 01/04", the production time for "Product #1" INCREASES.

These background assertions provide a partial account of changes and non-changes occurring over related intervals of time, with the first six background assertions describing changes occurring prior to the period of time covered by the event pattern, the next two background assertions describing changes occurring during the first time interval of the event pattern, and the last assertion describing a change occurring during the second time interval of the event pattern.

As a first case, suppose the human participant has classified the event pattern as a "consequent" event—that is, the human participant is looking to see if the causal progression specified by the event pattern might indeed be expected to occur within the time period from "1998 01/02" to "1998 01/04". The first interval of a consequent event must be confirmed by the computer, and thus the computer

begins the instantiation of the event pattern (Figure 5) by flagging event assertions 1 and 2, and leaving event assertions 3 and 4 unflagged.

Next, the computer enters the repeat loop of Figure 5. The first step of this loop involves a computer determination of candidate values for variables in the event pattern. The algorithm for accomplishing this is provided in Figure 6; however, other approaches such as one employing a generate-and-test procedure could equivalently be utilized.

In the algorithm of Figure 6, the computer first determines likely circumstances during the period of occurrence for the event pattern. The algorithm for accomplishing this is provided in Figure 7; however, other approaches such those involving the application of heuristics, analogy-based reasoning, or forward-chaining rules could equivalently be utilized.

The algorithm of Figure 7 iterates over the event assertions in the event pattern. The discussion here focuses on event assertion 1, with the remaining event assertions processed in a similar manner. The computer first collects all background assertions that match or conflict with this assertion. In this case, the following assertions are collected: background assertions 7 and 8, because they describe machines becoming no longer utilized for particular products, and background assertions 1 through 4, because they similarly describe machines utilized for particular products, and their ending time point, "1998 01/02", overlaps with the beginning time point of event assertion 1.

For the second step of the iterative loop in Figure 7, the computer determines that the flag for event assertion 1 is indeed set, and thus it applies a persistence assumption and a closed-world assumption to the background assertions as a means of extending its estimation of prevailing circumstances during the time period of the event pattern. Ensuing circumstances for background assertions 1 and 4 are already specified in the form of background assertions 7 and 8; however, background assertions 2 and 3 have no corresponding specifications of ensuing circumstances. In this case, the application of the persistence and closed-world assumptions to background assertions 2 and 3 produces two new assertions specifying that Machines 2 and 3, respectively, remain utilized for Products 2 and 1, respectively, over the interval from "1998 01/02" to "1998 01/03".

For the final step of the iterative loop in Figure 7, the computer determines that the event pattern does indeed have variables, and thus the appropriate action is to continue with processing of the next event assertion. After processing the remaining three event assertions, the computer returns the collected assertions to the calling procedure. Continuing with the second step in Figure 6, the computer must complete the determination of candidate values for variables. The algorithm for accomplishing this is provided in Figure 8. This algorithm is a simplified algorithm for constraint propagation; however, other approaches such those involving the application of heuristics, analogy-based reasoning, forward-chaining rules, or

use of another algorithm for constraint propagation (e.g., Waltz's algorithm) could equivalently be utilized.

The algorithm of Figure 8 consists of an iterative loop that begins by examining each event assertion and listing, as appropriate, either a set of "good" values (all others being bad) or a set of "bad" values (all others being good) for each variable contained within that assertion. For the first event assertion, the presence of background assertions 7 and 8 among the collected assertions for this event assertion generate direct matches that generate the "good" values "Machine #1" and "Machine #3" for the variable "a machine" and "Product #1" and "Product #2" for "a product". The closed-world and persistence assumptions effectively specify that no other machines cease to be utilized for products over the concerned interval, and thus all other possible values for these variables would be considered "bad."

In all, the four event assertions yield the following lists of "good" and "bad" variable values ("bad" values are listed for event assertions 3 and 4, because, absent the constraint provided by the application of a persistence and closed-world assumption, all values are "good" unless explicitly appearing in background assertions that conflict with these event assertions):

1. (from event assertion 1)
    - "a machine" has "good" values "Machine #1" and "Machine #3"
    - "a product" has "good" values "Product #1" and "Product #2"
2. (from event assertion 2)
    - "a product" has "good" values "Product #1" and "Product #2"
3. (from event assertion 3)
    - "a machine" has no "bad" values (all other values are "good")
    - "a product" has no "bad" values (all other values are "good")
4. (from event assertion 4)
    - "a product" has no "bad" values (all other values are "good")

Next, the algorithm of Figure 8 combines the "good" and "bad" lists for each variable. Two "good" lists combine by set intersection to produce a new "good" list. Two "bad" lists combine by set union to produce a new "bad" list. When a "good" list and a "bad" list are combined, the contents of the "bad" list are excluded from the "good" list by set difference, and the result is labeled a "good" list.

Considering the variable "a machine", the combination process produces a "good" list containing the values "Machine #1" and "Machine #3". For the variable "a product", the this process produces a "good" list containing the values "Product #1" and "Product #2".

In the third step of the iterative loop in Figure 8, the computer examines the "good" and "bad" lists for the variables to determine if any variable has only a single "good" value listed. If so, the computer substitutes the value for the variable and jumps to the top of the loop. In this case, neither variable has a single "good" value, so the computer returns the value lists to the calling procedure, the algorithm in Figure 6. This procedure also completes, returning the value lists to the algorithm of Figure 5.

Continuing with the algorithm in Figure 5, the computer tests the event pattern to determine if it has no remaining variables. In this case, the event pattern does have variables, so the computer proceeds to check whether the human participant wishes to replace a variable with a specific value. At this point, suppose the human participant instructs the computer to replace "a machine" with "Machine #3". The computer does this and returns to the top of the loop in Figure 5, thus proceeding with a new call to the subprocedure in Figure 6, which begins with a call to the subprocedure in Figure 7.

This time, the gathering operation for the new event assertion 1 (with "Machine #3" substituted for "a machine") produces only three background assertions—3, 4 and 8—because only these specify utilization status for "Machine #3" either during or overlapping the time interval of the event assertion. Applying the persistence and closed-world assumptions to the repository, a further assertion is generated, specifying that "Machine #3" continues to be utilized for "Product #1" over the interval from "1998 01/02" to "1998 01/03". Completing its processing of the new first event assertion in the procedure in Figure 7, the computer determines that the event assertion has remaining variables, thus the appropriate action is to continue with processing of the new second event assertion. After processing the remaining three new event assertions, the computer returns the generated, related assertions to the procedure in Figure 6.

The second step of the algorithm in Figure 6 again calls the procedure in Figure 8, which this time results in the following value lists for the single remaining variable "a product":

1. (from the new event assertion 1)

    - "a product" has a single "good" value "Product #2"

2. (from the new event assertion 2)

    - "a product" has "good" values "Product #1" and "Product #2"

3. (from the new event assertion 3)

    - "a product" has no "bad" values (all other values are "good")

4. (from the new event assertion 4)

    - "a product" has no "bad" values (all other values are "good")

Next, the algorithm of Figure 8 combines the "good" and "bad" lists for the variable "a product", resulting in a list of "good" values containing a single value "Product #2". This time, in the third step of the iterative loop, the computer replaces "a product" with the value "Product #2" and continues with an additional iteration of the loop. Repeating the first step of the loop, it considers each new event assertion in turn (also pruning the associated lists of related assertions to consider only those assertions that relate to the now further-instantiated event assertions), determining that each new event assertion contains no remaining variables and thus resulting in a return from the procedure in Figure 8 to the calling procedure in Figure 6. This procedure then returns to the procedure in Figure 5. Finally, the second step of the iterative loop in the algorithm in Figure 5 determines that there are no remaining variables and exits, returning the mapped version of the original event pattern with "a machine" replaced by "Machine #3" and "a product" replaced by "Product #2". In effect, through a collaboration of computer and human, the event instantiation algorithm has generated a prediction that the removal of "Machine #3" from utilization for "Product #2" may result in increased production time for "Product #2".

In contrast, it is also useful to consider how the event instantiation algorithm will operate given alternate event classifications supplied by the human participant. If the human participant classifies the event pattern as "reported," then the persistence and closed-world assumptions are not applied during the gathering step for any of the event assertions. This places less constraint on the range of acceptable values for variables appearing within these assertions. In particular, the computer will determine that "Machine #2" is also an acceptable value for "a machine", because it has been given no authority to assume that "Machine #2" continues to be utilized for "Product #2" over the first time interval covered by the event pattern, "1998 01/02" to "1998 01/03". Thus, the computer will present to the human participant an expanded list of candidate variable values—Machines 1, 2 and 3 for "a machine" and Products 1 and 2 for "a product"—effectively deferring to the human participant's stronger knowledge in the context of this reported event.

On the other hand, if the human participant classifies the event pattern as "inferred," then the persistence and closed-world assumptions are applied not only when processing event assertions 1 and 2 (covering the interval from "1998 01/02"

to "1998 01/03"), but also when processing event assertions 3 and 4 (covering the interval from "1998 01/03" to "1998 01/04"). In this case, during the processing of generating related assertions for event assertion 4, an additional assertion is generated specifying that the production time for "Product #2" does not change between "1998 01/03" and "1998 01/04". This assertion in turn prevents "Product #2" from being accepted as a "good" value for "a product" in event assertion 4, thus resulting in complete instantiation of the event pattern by the computer as it first replaces "a product" with "Product #1", then replaces "a machine" with "Machine #1". In this case, the human has deferred to the computer's stronger knowledge in inferring whether or not an instance of the event pattern can be substantiated as having occurred solely on the basis of what is specified in the repository.

6 Discussion

The split confirmation technique goes a long way toward supporting interactive, shared-initiative planning and monitoring, and in itself, is reasonably straightforward to implement. However, as with many research efforts, the ensuing development and deployment stages of the effort can nevertheless hold relatively significant difficulties. For IMPACT, one such difficulty lies in translating numerous currently-used formats for specifying timestamped information into a common format—here, stylized English—for entry into the system. Other hurdles are the design and construction of powerful, easy-to-use environments for human-driven negotiation of terminology for describing events, as well as specification of event definitions and causal rules. Finally, careful consideration must be given to determining the proper role for computer inference in the system, such that the computer is usefully employed where applicable, yet doesn't perform too much behind the human user's back or when incapable of reaching dependable conclusions. Future plans call for increasing the interconnectivity of the IMPACT system to accommodate new data formats, new types of network interaction, and richer GUI-based interaction.

Eventually, IMPACT or its underlying split-confirmation functionality could be repackaged as a general-purpose software application akin to current spreadsheet applications but dealing with circumstances and events rather than with numbers. Such an application might be targeted to contexts where continual change necessitates the interleaving of plan construction, modification and tracking activities. Some examples are: resource allocation planning, organizational restructurings, multiparty engineering or software design, logistics/transportation planning and military operation planning. In addition, there are myriad situation-specific project planning efforts within corporations and other organizations which could be aided by the use of such a system.

References

[1] Borchardt, G. C., "Understanding Causal Descriptions of Physical Systems," *Proc. AAAI Tenth National Conference on Artificial Intelligence,* 1992, 2–8.

[2] Borchardt, G. C., *Thinking between the Lines: Computers and the Comprehension of Causal Descriptions,* MIT Press, 1994.

[3] Cohen, P., "Plan Steering and Mixed-Initiative Planning," in Tate, A. (ed.), *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative,* AAAI Press, 1996.

[4] Cook, T. M. and Russell, R. A., *Introduction to Management Science,* Prentice Hall, 1993.

[5] Date, C. J., *Database Systems,* Sixth Edition, Addison-Wesley, 1995.

[6] Ferguson, G., Allen, J. and Miller, B., "TRAINS-95: Towards a Mixed-Initiative Planning Assistant," in Drabble, B. (ed.), *Proc. Third International Conference on Artificial Intelligence Planning Systems,* AAAI Press, 1996.

[7] Georgeff, M. P. "Reasoning About Plans and Actions," in Shrobe, H. E. (ed.), *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence,* Morgan Kaufmann, 1988, 173–196.

[8] Lowery, G., *Managing Projects with Microsoft Project 4.0 for Windows and Macintosh,* Van Nostrand Reinhold, 1994.

[9] Microsoft Corporation, *User's Guide: Microsoft Excel, Version 5.0,* 1994.

[10] Reiter, R., "Nonmonotonic Reasoning," in Shrobe, H. E. (ed.), *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence,* Morgan Kaufmann, 1988, 439–482.

[11] Shrobe, H. E. (ed.), *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence,* Morgan Kaufmann, 1988.

[12] Tate, A., Drabble, B. and Dalton, J., "O-Plan: A Knowledge-Based Planner and Its Application to Logistics," in Tate, A. (ed.), *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative,* AAAI Press, 1996.

[13] Winston, P. H., *Artificial Intelligence,* Third Edition, Addison-Wesley, 1992.

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.