AFRL-IF-RS-TR-2004-272
Final Technical Report
October 2004

STRUCTURED MODELING LANGUAGE FOR REPRESENTING ACTIVE TEMPLATE LIBRARIES (CAUSAL MODELING)

Rockwell Scientific

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J736

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

STINFO FINAL REPORT

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-2004-272 has been reviewed and is approved for publication

APPROVED: /s/

JOHN F. LEMMER
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	OCTOBER 2004	Final May 00 – Jun 04

4. TITLE AND SUBTITLE
STRUCTURED MODELING LANGUAGE FOR REPRESENTING ACTIVE TEMPLATE LIBRARIES (CAUSAL MODELING)

5. FUNDING NUMBERS
C - F30602-00-C-0038
PE - 63760E
PR - ATEM
TA - P0
WU - 09

6. AUTHOR(S)
Yousri M. El Fattah

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rockwell Scientific Company, LLC
1049 Camino dos Rios
Thousand Oaks California 91360

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

AFRL/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-272

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John F. Lemmer/IFSA/(315) 330-3657/ John.Lemmer@rl.af.mil

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

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

In this report we give a high-level description of the computational approach for the Causal Modeler (CModeler) tool. The tool provides a capability for capturing the cause/effect constraints in a Special Operations plan and for reasoning tasks in support of plan execution. The input to the tool is a plan created by a human planner in a mixed initiative environment using custom graphical interface (a program called SOFTools TPE) and the output is a minimal directed acyclic graph (DAG) representing a parsimonious potential causality graph. The nodes of the DAG are the actions and the directed arcs represent potential causal links. The term "potential" emphasizes the uncertainty in the abduced links since no requirement is placed on the availability of domain theory. Our approach relies on the structural information only; namely the temporal ordering of the actions and the task hierarchy of the plan.

We describe one main application of the tool for the Special Operations domain to support the task of run-time replanning. The replanning task takes the unexpected events in the execution of the plan (e.g., late or aborted actions) and uses the causal model to compute the impact on future actions and reconfigure the plan. We summarize at the end of the report our views of the lessons learned and give concluding remarks about future directions for developing this technology.

14. SUBJECT TERMS
Causal Modeling, Mixed Initiative Planning, Dynamic Execution, Temporal Constraints

15. NUMBER OF PAGES
35

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. Executive Summary

The objective of the DARPA Active Templates program is to develop lightweight planning tools in support of special operations. Rockwell Scientifics' effort culminates in two software systems: CModeler, a causal modeling Tool and SOFPlans, a graphical planning tool.

This report is focused only on the causal modeling tool which provides a capability for capturing the cause/effect constraints in a special operations plan and for reasoning tasks in support of plan execution. The CModeler tool captures automatically a default causal model from the plan authoring input entered by users using domain specific interfaces, e.g. the SOFTools Temporal Plan Editor (TPE) in the special operations domain.

We developed a computational approach based on the notion of potential causality to enable building a causal model elicitation environment that automates the process of building models as much as possible and updates those models incrementally with the plan authoring input. Instead of relying on knowledge bases and prior domain theories as in traditional knowledge acquisition, our approach uses only the temporal ordering of the actions and the task structure of the plan. The term "potential" is used to emphasize the uncertainty in the abduced causal ordering since no requirement is made on the existence of a complete domain theory as in standard partial order planning.

We give in this report a formalization and algorithm for extracting a parsimonious description of a potential causality relation, which is presented to the modeler as a representation of the candidate space of sets of causal links consistent with the authored plan. We describe one main application of the tool for the special operations planning domain to support the task of run-time replanning. The replanning task takes the unexpected events in the execution of the plan (e.g., late or aborted actions) and uses the causal model to compute the impact on future actions in the remainder of the plan.

We give in this report a general formulation of the task of abducing potential causality from plan authoring input in a mixed initiative framework and provide a high-level description of the reasoning algorithms. We give an example of a plan authored in SOFTools and describe the default model and the process of editing the causal graph. We use the same example to show how the causal model is subsequently used to support run-time replanning. The last sections in the report give our assessment of the lessons learned and our general conclusions.

2. Introduction

In standard automated planning [Weld 1999] the goal is to automatically synthesize a plan from a representation of goals, states and operators. A plan operator is typically represented by a list of preconditions and post-conditions expressed as sentences in some formal logic. The preconditions state which conditions must hold before an operator can be executed and the post-conditions explicitly state the results of executing the operator. Generally speaking, automated planning amounts to a search for an ordering of the plan operators so that starting from the initial state, the operators preconditions are satisfied at every step in the ordering and the end state entails the goal state.

In mixed initiative planning (MIP) plan synthesis uses a collaborative planning paradigm in which humans and machines collaborate to build effective plans more quickly and with greater reliability [Burstein & McDermott 1996]. The MIP approach alleviates the requirement for complete domain models as in automated generative planners and makes the approach viable for real-life large-scale planning domains such as in military and space. In MIP Human modelers and planning algorithms need to interact more effectively in order to develop safe efficient plans, and tools are required that ease the task for the modeler.

Our causal modeling approach aims at supporting ease of plan entry during MIP in which a human planner authors a plan by formulating the planning tasks and selecting and ordering the actions. The approach takes the plan authoring input and generates a concise description of a set of causal links for the plan. This causally linked plan can then be used for execution monitoring and plan revision management.

The DARPA Active Templates is an innovative approach to MIP in which human planners build the plans by instantiating and linking templates, which are kind of frames that consist of attribute-value pairs representing stereotyped tasks or situations in the domain. An example of a template is a travel planning template [Frank et al. 2001] where the attributes may include origin, destination, weather, transportation, etc. In general, the templates maintain built-in constraints and the attributes can be automatically defaulted using specialized functions and user interfaces. The user can also assign values to attributes and can override default values.

The implementation of our approach in the CModeler software tool [El Fattah 2003], is centered on the special operation planning domain, the focus for the Active Templates program. The ontology for that domain is based on the notion of movement of assets from one place to another. Places represent fixed geographic locations over spans of time and the assets represent resources such as aircrafts, ships, helicopters, vehicles etc. The CModeler program has a number of interesting capabilities:

- Automatic elicitation of causal models by interpreting temporal plan plots, input from the user's adopted plan authoring interfaces (SOFTools TPE)

- The program leverages multiple sources: the predefined ontology of the domain, the plan XML schema, the precedence ordering of the plan actions, the locality relation derived from the plan task hierarchy.
- The program enables the user to view a hierarchical layout of the causal graph computed automatically and allows the user to amend the graph by adding or deleting causal links.
- The program enables abstraction operations in the form of graph folding and unfolding to allow users to view the causal model at the desired representation levels; i.e., by shrinking unnecessary details and expanding interesting portions of the plan.
- The program enables constraint specification on the timing of the actions and embeds a versatile constraint solver (based on declarative constraint logic programming CLP formulation) to generate the plan schedule that satisfy all constraints.

3. SOF Planning

The ontology of the Special Operation Forces (SOF) planning domain is based on the notion of movement of assets from one place to another. Places represent fixed geographic locations over spans of time and the assets represent resources such as aircrafts, ships, helicopters, vehicles etc. Figure 1 gives a partial specification of SOF plan objects in the Unified Modeling Language, UML. The Figure says that a plan is composed of any number of events, movements, and places. A movement has a start and an end which are special events, and is associated with two places which are the origin and destination for the movement. A place or a movement can be associated with any number of events which are local to the place or the movement.

An event has type which can be plain or decisionpoint, a time to specify when the event should occur and a calltime to specify when the event actually occurs during execution. A place has a time interval specified by starttime and endtime, and a name intended to generically identify its physical location (or area of operation, in the case of a "mobile place" like a ship) and its role, e.g., FSB (forward staging base). A movement has assets which are the resources assigned to effect the move, e.g., two C-5 aircrafts.

The objects in Figure 1 also define the plan templates. Each template is a list of variable value pairs; the variables being the attributes of the objects and the values are restricted by domain constraints and the specified relations between the templates. For example, the duration of a movement is a function of the locations of the origin and destination of the movement and the assets used by the movement. An example of relations between the templates is that the location of the arrival event of a movement is the destination place for the movement.

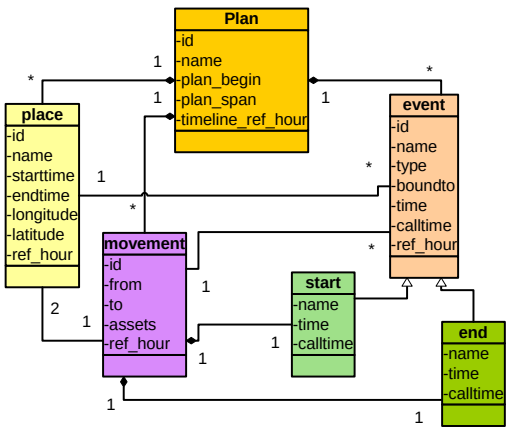

Figure 1 SOF plan objects (templates).

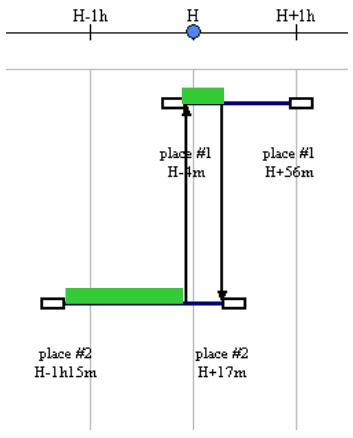

Figure 2 Timing constraints

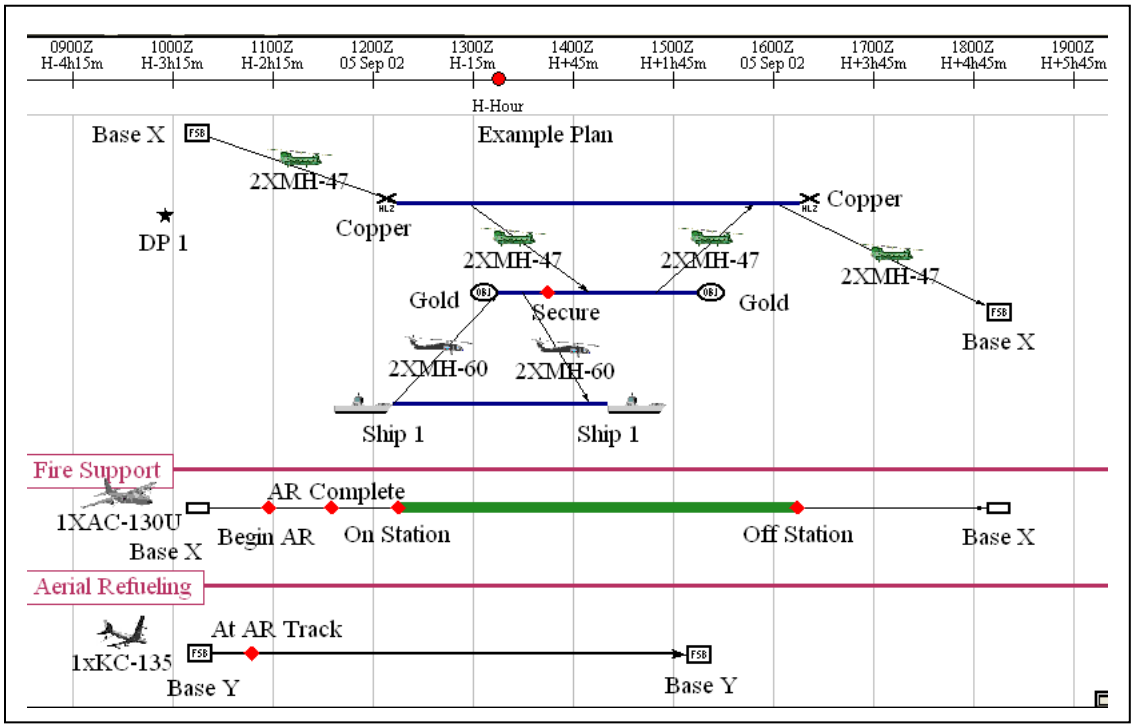

Figure 3 Example plan in SOFTools TPE

In addition there also are integrity constraints enforced on the timing of the plan events and tasks. For example, the arrival event must occur later than the departure event of the same movement; the arrival event be earlier than the endtime for the destination of the movement; and so on. See Figure 2.

Using a graphical editor, called SOFTools TPE, a planner can author a plan by a drag and drop of graphical icons representing the templates. Events and decision points are depicted as time points (diamond shaped). Places are diagrammed by horizontal lines specifying the start and end time. Movements are represented by arrows directed from an origin place to a destination.

An example of a plan diagram entered in SOFTools that we will use throughout this report is shown in Figure 3. The plan can be summarized as two black-hawk (MH60s) flying to objective from ship to perform a "secure operation" before the arrival of a Chinook (MH-47). The Chinook flies to objective from a forward staging base (FSB) via helicopter landing zone (HLZ) then returns to FSB. A gunship (AC-130U) is to perform the fire support to protect the Chinook flight from FSB and the black-hawks arrival at objective. The fire support gunship requires aerial refueling (AR) by a KC-135. Notice that some of the knowledge in our narration is no where represented in the plan diagram. Specifically the intention of using the fire support for the purpose of protecting the blackhwaks arrival at objective is not explicitly represented. Generally speaking the plan authoring input is only a partial specification of a plan focused only on the extensional representation only. The intentional representation of the plan is not explicitly specified. By intentional representation we mean a formal specification of cause/effect relations between the actions in the plan that enable answering queries about plan change. For example what would be the effect of an event such as the start of the fire support being delayed? In the absence of knowledge about causal links it is not possible to determine what effect such event will have on other events in the plan since we do not the independencies or the dependencies between the plan actions.

The main motivation for this work has been to provide an automated tool to abduce a default causal model from the extensional temporal .plan representation. We focus on the structure of the cause/ effect relations in the plan and represent the structure in the form of a directed acyclic graph whose nodes are the plan events/ actions and the links represent "causal links" and "threat resolution links" as commonly defined in partial-order planning [Weld 1999]. Informally, a causal link is a link between an action pair such that the postcondition for one action achieves a precondition necessary for the other. While a threat resolution link is a temporal ordering that must be enforced to protect preconditions one action be clobbered by the postconditions of the other.

4. Potential Causality

In this section we formalize our causal modeling approach in the context of automated planning. We give an abstract formulation of the plan authoring input and a general statement of the task of reasoning about potential causality.

Plan authoring in our framework is a triple (T, P, ρ) . T is a rooted directed tree whose nodes *V(T)* are the plan tasks. The root of the tree is the top-level task and the parent-children relation represents task decomposition. P is a totally ordered (t.o.) plan $P = \langle a_1, \ldots, a_n \rangle$, represented by the pair $\langle A, \prec_P \rangle$ where A is the set of actions and $\prec_P$ is a total order defined by the t.o. plan: for $1 \le i, j \le n$, $a_i \prec_P a_j$ iff $i < j$. $\rho \subset A \times V(T)$ is a binary relation that associates the actions with the tasks.

Given a plan authoring input (T, P, ρ) the problem we pose is to infer a partial-order plan (p.o. plan) given by a tuple $\langle A, \prec \rangle$ where $\prec$ is a partial order on A[1]. The partial order represents a parsimonious description of a potential causality relation on the actions consistent with the t.o. plan P. We represent the partial order $\prec$ by a directed acyclic graph (DAG), G=(A,E), called causal graph, such that there is a directed path from a_i to a_j in G iff $a_i \prec a_j$. The directed edges $(a_i, a_j) \in E$ represent both "causal links" and "threat resolution links" as commonly defined in partial-order planning [Weld 1999]. A causal link from a_i to a_j means a_i achieves some precondition for a_j while a threat resolution link is a temporal ordering that must be enforced to protect conditions achieved by some causal link. The causal graph G represents the p.o. plan which is a compact representation for a set of t.o. plans that can be obtained as the topological sort of G. A topological sort of G is a sequence obtained by ordering the nodes A such that if $(a_i, a_j) \in E$ then a_i is before a_j in the sequence.

[1] A partial order is a binary relation that is irreflexive, asymmetric and transitive. A binary relation on a set X is a subset $R \subset X \times X$. The relationship is irreflexive if for all (x, x) in $X \times X$, then $(x, x) \notin R$. The relationship is asymmetric if for all $(x, y) \in R$, then $(y, x) \notin R$. The relationship is transitive if for all x, y, z, if $(x, y) \in R$ and $(y, z) \in R$ then . $(a_i, a_j) \in E$ $(x, z) \in R$

Definition 1 Given a plan authoring input $\Pi = (T, P, \rho)$; $P = (A, \prec_P)$ we say a partial order $\prec$ on A *is a* Π-compatible ***potential causality*** order if it satisfies the following postulates: (i) the total order $\prec_P$ obeys the partial order $\prec$,i.e., $\prec \subseteq \prec_P$ (ii) if two actions a_i, a_j belong to same task $t \in V(T)$, i.e., $(a_i, t) \in \rho$ and $(a_j, t) \in \rho$ then the actions must be ordered following the t.o. plan, i.e., $a_i \prec a_j$ if $i < j$ and $a_j \prec a_i$ if $j > i$ (iii) if two actions a_i, a_j belong to tasks $t_i, t_j \in V(T)$,i.e., $(a_i, t_i) \in \rho$ and $(a_j, t_j) \in \rho$ such that either t_i or t_j is an ancestor of the other in T, then the actions must be ordered following the t.o. plan, i.e., $a_i \prec a_j$ if $i < j$ else $a_j \prec a_i$.

The intuition behind the potential causality postulates is as follows. The temporal ordering of the actions for each task guarantees that each action has effects that are required by the preconditions of the later actions in the same task and therefore provide sufficient condition on the causal ordering. The tasks whose actions do not overlap (do not share actions) are likely to be independent and could be executed in any order. Tasks that are ancestor (or descendant) of one another in the tree are potentially inter-dependent and therefore their temporal ordering should be preserved by the causal ordering. Potential causality provides a space of partial orders from the most to least constrained. The most constrained order is the input plan ordering $\prec_P$ which is a complete ordering. The least constrained causal order can be defined using optimization criteria such as set minimality as defined next.

Definition 2 Given two Π-compatible potential causality orders $P = (A, \mathrm{p})$ and $P' = (A, \mathrm{p}')$ we say that P' is less constrained than ***P*** iff $\mathrm{p}' \supset \mathrm{p}$. A Π-compatible potential causality order $P = (A, \mathrm{p})$ is minimal iff there is no other Π-compatible potential causality order $P' = (A, \mathrm{p}')$ such that P' is less constrained than *P*.

Example 1. Consider the planning problem of putting shoes and socks. A plan authoring input will have: (i) a tree T with a root node and two leaves which are the primitive tasks right foot and left foot; (ii) a t.o. plan e.g., ‹start, left sock, left shoe ,right sock, right shoe, finish›; (iii) mapping of actions to primitive tasks: {(start ,plan),(finish, plan),(left sock, left foot) ,(left shoe, left foot), (right sock, right foot), (right shoe, right foot)}. Here start and finish are “dummy” actions to encode the initial state and the goal state for the plan task which is the root node of the tree. It is easy to see that a minimal potential causality order is represented by the causal graph having the directed edges {start→right sock, start→left sock, right sock →right shoe, left sock →left shoe, left shoe →finish, right shoe →finish}. The graph is a compact representation of 6 possible t.o. plans.

We now describe a polynomial time algorithm that computes a minimal potential causality graph (PCG) from a plan authoring input. The algorithm has two main steps. In the first step it computes a minimal potential causality order $\prec$ starting from the complete input plan order $\prec_P$, and in the second returns a directed acyclic graph (DAG) from the transitive reduction of the partial order. The transitive reduction can be computed in $O(|A| \cdot |\prec|)$.

Algorithm PCG
Input: authored plan Π=(T,P,ρ); P=(A, $\prec_P$)
Output: potential causality graph, G
begin
$\prec \leftarrow \prec_P$ // initialize partial order
// compute minimal potential causality order
for each $(a,a') \in \prec$
$S = \{t \in V(T) \mid (a,t) \in \rho\}$ // tasks for a
$S' = \{t \in V(T) \mid (a',t') \in \rho\}$ // tasks for a'
violation $\leftarrow$ false
repeat while $\neg$ violation
for each $t \in S$ and $t' \in S'$
if $S \cap S' \neq \varnothing$ then violation $\leftarrow$ true
else if $t' \notin$ desc$(t) \wedge t \notin$ desc(t') then violation $\leftarrow$ true
loop
if $\neg$ violation then $\prec \leftarrow \prec -\{(a,a')\}$
//causal graph as transitive reduction
return minimal DAG, $G = (A,E)$, such that there is a
directed path from a to a' iff $a \prec a'$
End

5. Causal Modeler

We built a tool that implements our approach for inferring potential causal links in the SOF planning domain. The tool, called Causal Modeler (or CModeler) [El Fattah 2003] captures cause/effect models from SOFTool plans based on potential causality (e.g. aircraft arrival depends on aircraft departure) CModeler can generate plans using the causal information and can automatically revise plans/ schedules when events occur late or are aborted and automatically update SOF execution checklist. The computing architecture is shown in Figure 4.

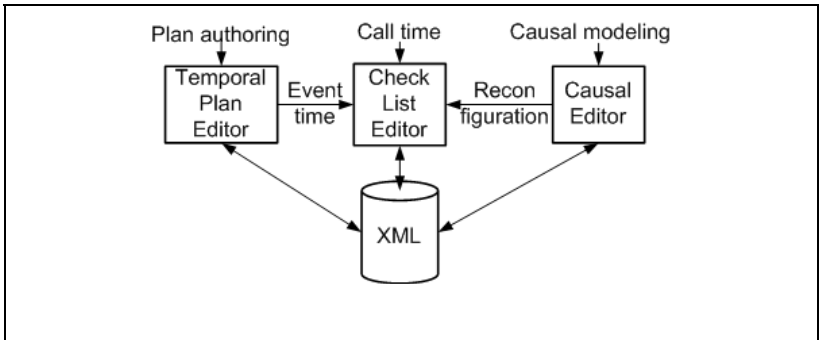

Figure 4 Causal Modeler architecture

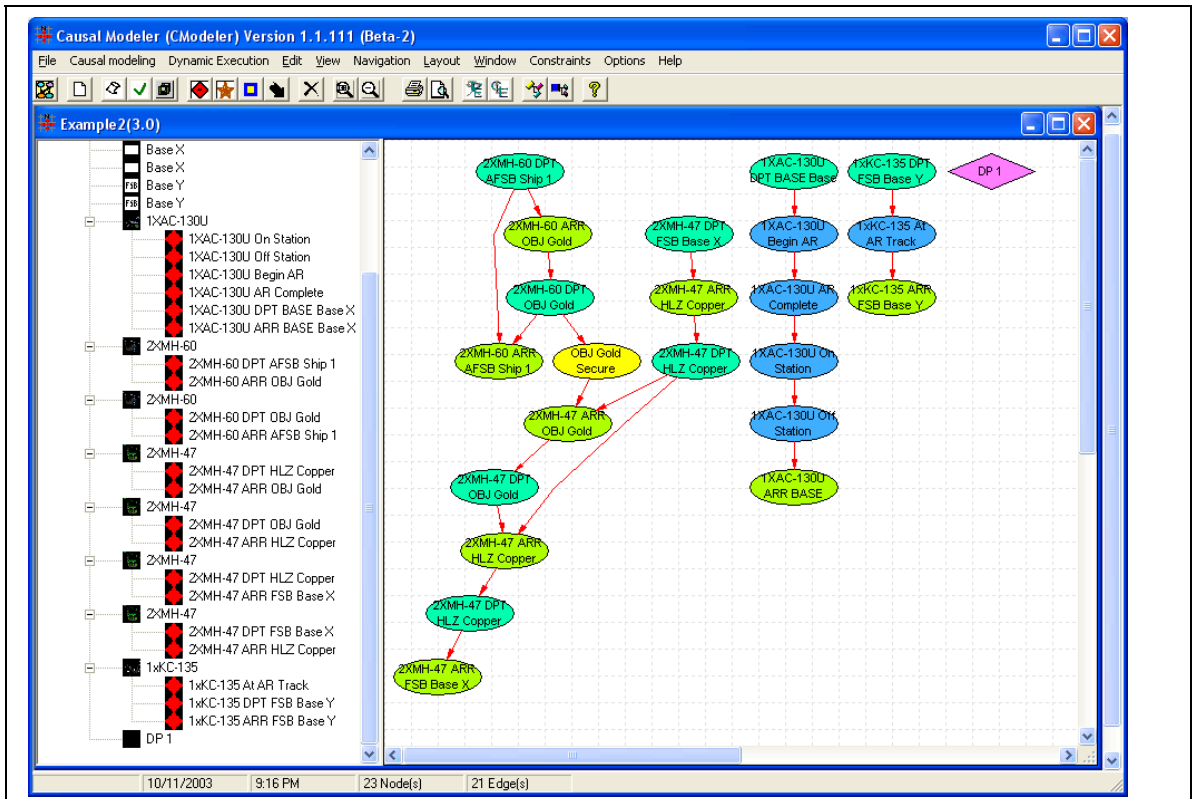

Figure 5 Causal modeler showing default graph.

CModeler provides two dynamic execution capabilities for computing the effect of change during execution and for updating the plan execution checklist. The changes currently supported are aborted and delayed events. The causal models are represented as directed graphs whose nodes are plan elements (type movements; places; events and decision points) and the edges are directed from causes to effects. The structure of the causal model is represented as compound graph visualized in two views: **tree view** and **graph view**. The tree view shows the plan task hierarchy and the graph view shows the causal graph induced by contracting or expanding nodes in the tree. Figure 5 shows CModeler displaying the default causal graph for the plan example shown in Figure 3.

CModeler allows full graphical editing capabilities for:

a) adding/ removing causal links between events

b) adding/ removing plan elements

c) editing properties of plan elements as in SOFTools. CModeler applies causal heuristics to infer default causal model incrementally as plan elements are added. A user can enable/disable the default causal modeling by selecting or deselecting the use of heuristics from CModeler's menu. If the heuristics are deselected then no causal links are created automatically in the graph view and the user can graphically enter the causal dependencies.

CModeler provides two ways to build causal models (1) **interactive way** - by authoring a plan interactively in CModeler; (2) **non-interactive way**- by authoring a plan in the SOFTools TPE then loading the plan into CModeler

.

Having an automated tool for default causal modeling has filled a need in this SOF planning domain for eliciting and capturing causal knowledge during the plan authoring. The causal views computed by CModeler are linked to the temporal and spatial plan views so that changes in one view can propagate to the other views. This is done by representing and capturing the temporal plan constraints and the causal dependencies between the actions in the plan. By viewing the default causal graph a user is able to determine if causal links are missing or existing ones are spurious and can amend the default model by appropriately adding or deleting causal links.

Incremental Causal Construction

CModeler applies the heuristics as the user is plotting the temporal plan diagram. The heuristics are local to action subsets, enabling incremental construction of causal models. The basic idea of the incremental construction is a splicing operation; at each step replacing arcs in existing causal model by chains containing new nodes. Figure 6 shows a splicing operation after adding a plan-event type action to the plan. The Figure shows the colored node Act_i being added to the task tree and spliced in the current causal model by removing an edge between two nodes (the predecessor and successor in the temporal sequence) and adding a chain that links the new node to its neighbors. CModeler maintains two data structures for plan authoring the task tree and the causal graph. At each step in plan authoring a user can add to the plan a new object: a place task; a movement task; or an action of type plan-event, place-event, movement-event, or decision-point. CModeler interprets the added object and will perform any of these splicing operations:

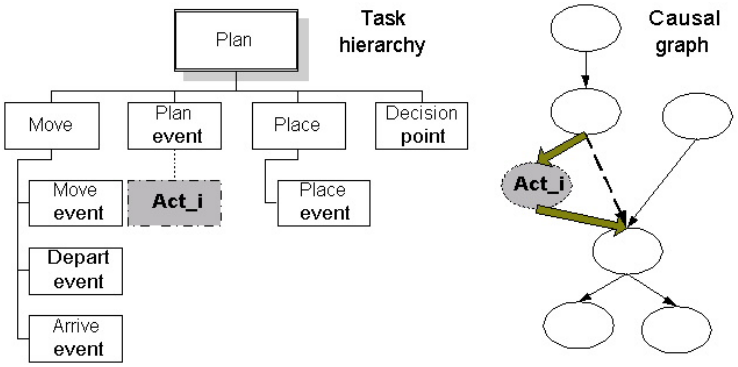

Figure 6 Causal modeler showing the default causal graph.

- If plan element is a place then add a place node to the tree.
- If plan element is a movement then add a movement node to the tree and create two graph nodes for depart and arrive events and a causal link between them
- If the plan element is an event bound to a place (movement) then add a child to the respective place (movement) node in the tree and splice the event node in the place (movement) subgraph
- If the plan element is a plan event (decision point) then add a node to the tree and splice the event (decision) in the plan event (decision) subgraph.

Abstraction

For a large plan containing a large number of actions the causal graph can become too large to visualize and details become obscured. CModeler supports two abstraction strategies: (a) folding: which conceals "unwanted" details of a graph allowing a user to temporarily hide certain nodes and edges and; (b) unfolding: which reintroduces the hidden details of folded nodes. At any time during a sequence of folding or unfolding operations, the causal graph has a valid topology allowing a user to perform layout. Thus, folding and unfolding can be used to coarsen and refine causal graph abstractions and to create new summary views, called causal views. We use the plan task hierarchy as a basis for computing abstraction views of the causal graph. The abstraction views are nested graphs [Sugiyama and Misue, 1991] (sometimes called a hierarchical or compound graph) allowing nodes to contain graphs. Each view is depicted as a directed graph whose nodes contain blobs denoting sets of nodes in the causal graph. Every set of interest is represented by a unique blob and is labeled by an associated task in the plan. The causal views have directed edges induced from the causal graph as shown in Figure 7. The Figure shows that an edge from node b to node c in the causal graph induces an edge from the blob T2 to the blob T1. At the most abstract level the causal view will have just one blob containing the entire graph.

CModeler shows causal views as a pair: a tree and a directed graph as shown in Figure 5. The views are coupled so that the leaves in the tree view always match up with the nodes in the graph view. Initially the tree in the tree view is the task hierarchy and the graph in the graph view is the "flat" causal graph. To fold an internal node in the tree view, a user clicks the minus (-) sign next to the node. This contracts the node in the tree view and coarsens the graph view by hiding the subgraph for the descendants of the contracted node inside a blob matching the contracted node in the tree view. Blobs are shown as rectangles and plan actions or events are shown as ellipses.

To unfold a node in the tree view, a user clicks the plus (+) sign next to the node in the tree view. Unfolding will expand the tree one level below the unfolded node showing the children of that node in the tree view and the subgraph for the children in the graph view. Induced edges are created dynamically to and from the blobs in the graph views to maintain consistency with the "flat" graph. An induced edge is created from (to) a blob if there exists an edge in the graph view from (to) a descendant of the blob. Note that unlike the causal graph which is acyclic, the abstracted causal views may contain directed cycles. For example, if in Figure 7 there is an additional arc from node a to node d then there will be a directed cycle between the blobs T1 and T2.The folding and unfolding explorations to coarsen and re-fine the causal views interactively can be implemented efficiently in time linear in the number of changes induced by any exploration operation [Buchsbaum and Westbrook, 2000]

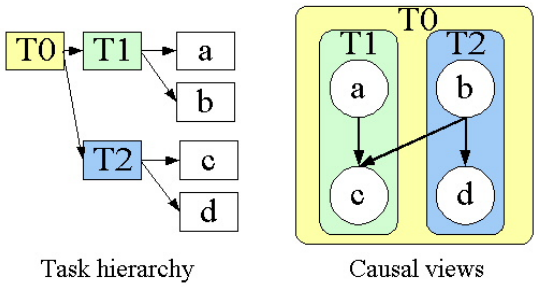

Figure 7 Causal views

6. Constraint Processing

To enable specification of temporal constraints on the plan tasks, we use the causal graph to define a temporal constraint network. Temporal constraints are assigned to the nodes and edges of the graph and the solution to the constraint network determines the start and finish dates for all the plan tasks.

A solution of a constraint network is an assignment of values to the variables that satisfies all the constraints. A constraint network is consistent if there exists a solution. A constraint network is minimal if each primitive constraint is satisfied in a solution of the network; i.e. there are no primitive constraints that do not participate in at least one solution. Interesting problems (typically NP-hard) are to:

(a) determine consistency

(b) compute the minimal network

(c) compute one or all solution(s)

(d) check consistency of an assignment

We restrict ourselves to binary constraint networks. The value domain is the real domain for all the time variables. The constraint specification for the plan events depend on whether the event is independent, i.e., having no parents in the causal graph. If the event has parents then the constraints are specified as the minimum lag time from each parent that the event depends upon. If the event is independent then the constraint is specified as either before, on, or after the H-hour. Minimum lead time is specified for the before constraint and minimum lag time for the after constraint which both translate to inequality constraints. For the on constraint a lead or lag time can be specified which translates to an equality constraint.

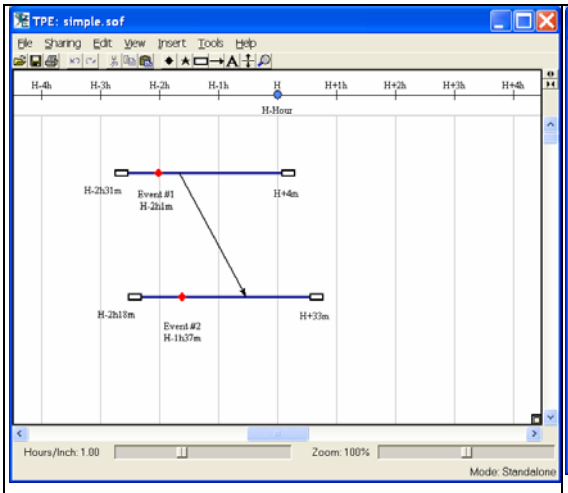

Figure 8 Simple temporal plan

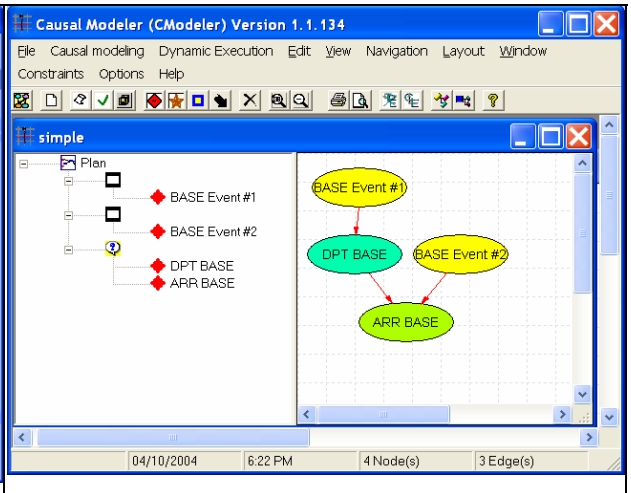

Figure 9 Causal graph for simple plan

The default constraints abduced from the temporal plan are specified as follows:

- For each independent event (root node in the causal graph) the constraint is before or after constraint with a minimum lead or lag time specified by the difference between the event time and the H-hour.
- For each dependent event (having parents in the causal graph) create constraint for each directed edge and specify a minimum lag time computed as the difference between the event time and the time for its parent. The constraint for the dependent event is a set of inequalities; each specifying the time for the event minus the time for the parent be greater or equal the minimum lag time specified for the respective edge in the causal graph.

As an example consider the temporal plan shown in Figure 8. The plan consists of two bases, a movement from one base to the other and a place event at each base. The default causal graph is shown in Figure 9.

The default constraints abduced for the model is shown in Figure 10. There are two independent events in the graph which are the base event #1 and base event #2. The Figure shows the constraint for base event #1 being before H-hour with a lead time computed from the temporal plan. An example of dependent event is "Dpt base" which has one parent "Base event #1" and the constraint is a lag time constraint relative to the parent with the lag computed from the difference between the event time of "Dpt base" and that of the parent. All the default constraints are given by inequalities with the value domain being the real values. The variables are the event times for the events in the plan plus the time of the plan H-hour. For the example, there are 4 variables and there are 5 constraints: 2 for the 2 root nodes plus a constraint for each directed edge in the causal graph.

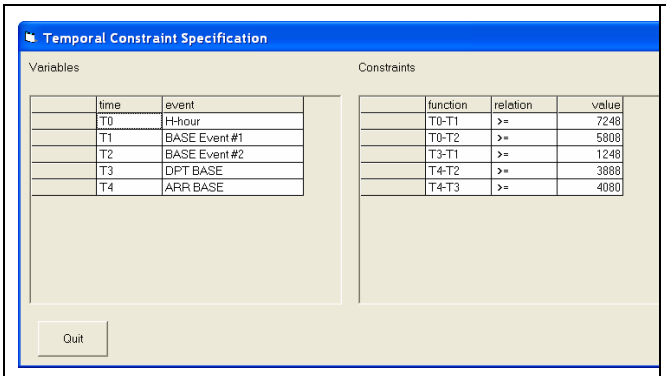

Figure 10 Default constraints for the simple plan

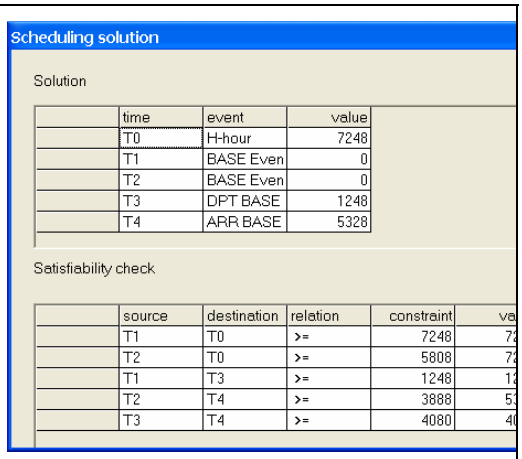

Figure 11 Solution of the default constraints

The solution for the constraint problem is computed by a constraint solver using the constraint logic programming language over real domain CLP(R). A query is created by assembling all the temporal constraints and assigning a temporal variable for each event. There can be many solutions that satisfy the constraints. To restrict the set of solutions we formulate the problem as an optimization problem by minimizing the sum of all the event times. For our simple plan, the constraint query is stated as follows:

```
schedule([T0, T1, T2, T3, T4]):-
      {T0 >= 0, T1>= 0, T2>= 0, T3>= 0, T4>= 0, T0-T1 >= 7248,
      T0-T2 >= 5808, T3-T1>= 1248, T4-T2>= 3888, T4-T3>= 4080,
      Z=T0+ T1+ T2+ T3+ T4},
      minimize(Z).
```

The solution returned is as shown in Figure 11. Notice that the solution generally differs from the initial values assigned in the original temporal plan. For example, both the independent events are assigned the same time while in the initial plan they are not.

7. Knowledge Capture

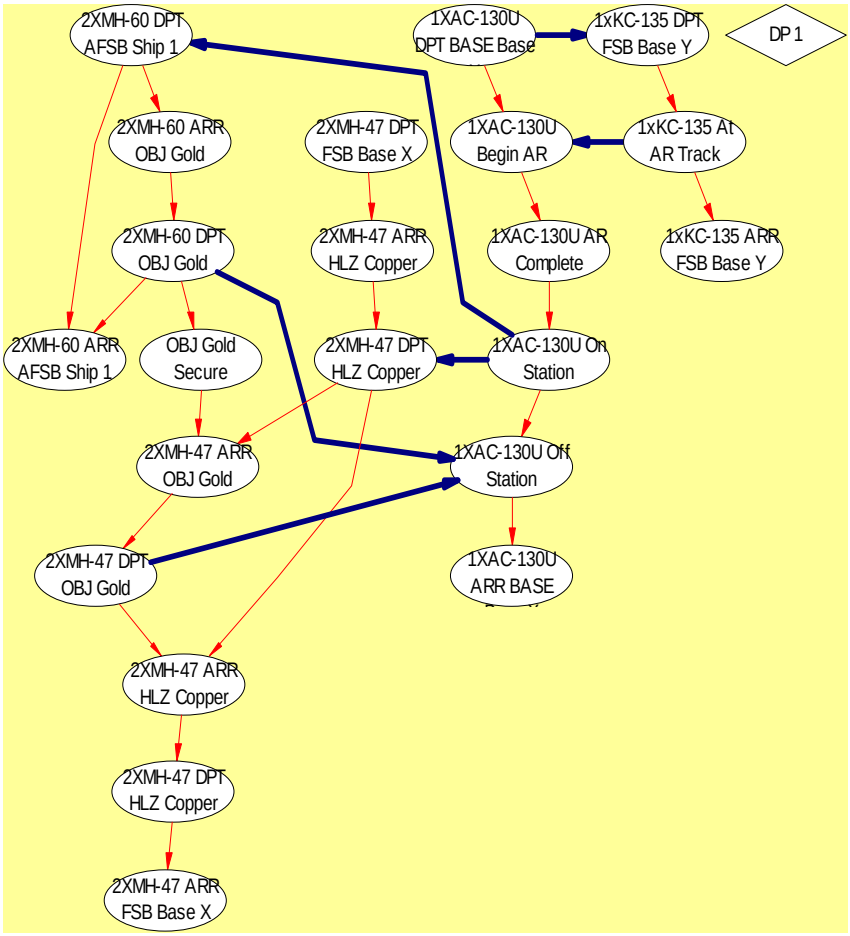

Figure 12 Potential causality graph with thick arcs added by domain expert

An important result of our work is that it enables round-trip engineering between the causal and temporal representations of a plan. Planner in our SOF domain are accustomed to authoring plans based on the temporal/spatial view. Having an automated tool for default causal modeling has filled a need for eliciting and capturing causal knowledge during the plan authoring. This means capturing and explicitly representing the plan constraints and the causal dependencies between the events in the plan.

By viewing the default causal graph a user is able to determine if some causal links are missing and can amend the default model by adding or deleting links Our heuristic approach provides "useful" results with incomplete information and the results improve as additional knowledge sources become available, e.g. additional heuristics and more complete specification of plan constraints.

For example, the default causal model shown in Figure 12 has some of the causal relationships unspecified. Those are the relationships between the fire support and their objectives and between the aerial refueling and the fire support. They are missing in the default model because no transaction constraints were specified.

To amend the default causal graph the user will add those causal links as shown in Figure 3 in thick lines. The Figure shows 6 causal links added to the default model. Link 1 says the start of the fire support necessitates the start of aerial refueling movement. Link 2 says that being at aerial-refueling track enables the refueling action. Links 3, 4 specify the causal condition for the supported missions to begin. The causal links say the fire support "on-station" event enables the two Blackhawk to depart Ship and the Chinooks to depart drop zone Copper. Links 5, 6 specify the causal condition when the fire support is to end. The links say that the departure of both the Chinooks and the Blackhawk from objective Gold enables fire support "off-station" event.

8. Dynamic Execution

One of the main functions enabled by our CModeler program is the support for dynamic plan execution. Most plans do not execute according to schedule and when this occurs there is limited time to respond. When changes occur during execution, the task of plan reconfiguration consists in determining all actions impacted by the change and recomputing the plan schedule to meet known constraints on the affected actions. The affected actions can be determined by performing a reachability analysis on the causal graph. For example, if an action such as the departure of an aircraft is delayed then we must delay all its causally-dependent actions (its descendants in the causal graph), e.g., the arrival of the aircraft at destination and the arrival-dependent chain of events. Note that those descendants are not necessarily contiguous in the plan execution checklist making the determination of causal impact non-trivial. Also the temporal relationships inferred from the plan temporal diagram are not adequate to determine the required reconfiguration. For large plans the reconfiguration task can be difficult and time-consuming to perform without automation. We have implemented the following algorithm for plan reconfiguration (PR) in CModeler.

```
Algorithm PR
Input: checklist L; causal graph G=(A,E).
Output: reconfigured checklist L'.
  begin
  S←Ø  //execution state
  ⟨a_1,…,a_n⟩←topological sort of G
  for i=1 to n do
   if (calltime(a_i,L)=?) //not yet executed
   then if //has some parent in abort state
        (x in pa(a_i,G): state(x,abort) in S)
     then //set action state to abort
        S←S∪{state(a_i,abort)}
     else // max delay of the parents
        Δ=max{ Δ_x | x∈pa(a_i,G),state(x,delay(Δ_x)∈S}
        S←S∪{state(a_i, Δ)}
    else if //action has been aborted
           (calltime(a_i,L)=abort)
         then //set action state to abort
            S←S∪{state(a_i,abort)}
         else //compute delay
            Δ=⌈ calltime(a_i,L)- eventtime(a_i,L)⌉
  for i=1 to n do
   if (state(a_i,abort)∈ S)
    then eventtime(a_i,L)←abort
   if (state(a_i,delay(Δ))∈ S)
    then eventtime(a_i,L)← eventtime(a_i,L)+Δ
  return L'←
  end.
```

The plan reconfiguration algorithm PR takes the plan causal graph and the execution state and outputs an updated checklist of the plan. A plan execution state consists of a determination in real-time for each event, relative to its call time and scheduled time, whether: the event is late; has not occurred; been aborted; called in early; called in on time; called in late. Our reconfiguration algorithm first orders the events along a causal order which is just a topological sort of the causal graph (which is acyclic). The algorithm propagates the execution state by aborting an event if any of its parents is aborted; otherwise delaying the event by the maximum delay over its parents. The algorithm shifts the scheduled time for delayed events and outputs a new execution list ordering the events by increasing event time.

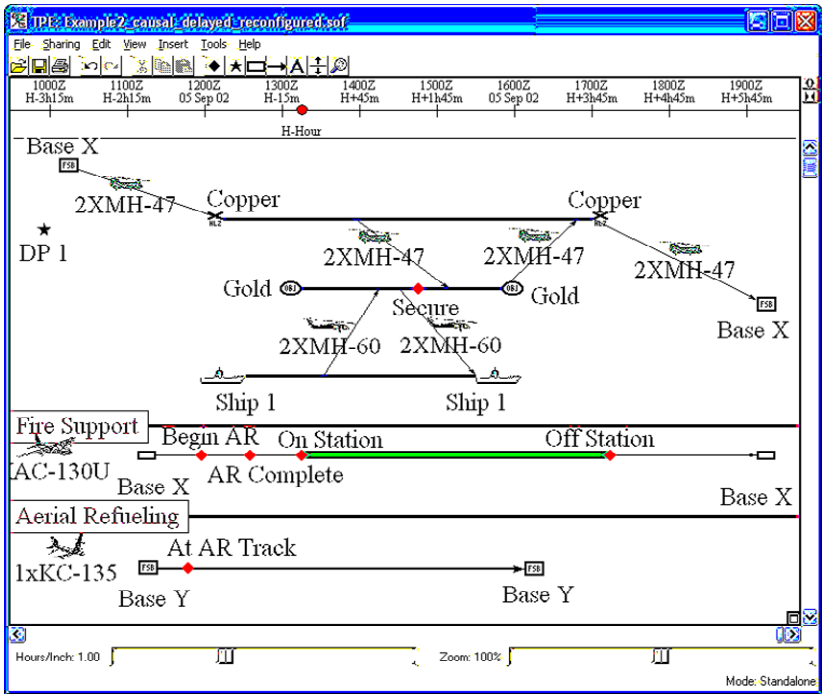

Figure 13 Reconfigured plan after fire support delayed 1 hour

Figure 13 shows a reconfiguration example for our SOF plan of Figure 3, where the depart event of the fire support gunship (AC-130U) is delayed 1 hour. The reconfiguration is computed based on the amended causal graph shown in Figure 3. Compared to the original plan in Figure 2, we see that (a) the event of KC-135 at air-refueling track is shifted back 1 hour; (b) the departure of the chinook from Copper and the Blackhawk from Ship1 are all shifted by same amount to meet the fire-support coverage constraint; (c) all causally dependent events are properly shifted; (d) all independent events remain unchanged, e.g. the decision point the departure of the Chinook from baseX and arrival to Copper. Note that the end times for places are shifted by proper amount to satisfy the condition that reconfigured place events be within the start and end time for the place.

9. Plan Reuse

Our causal modeling approach can speed-up plan authoring by reusing previous cases from an experience library. The plan reuse consists of two parts: (1) retrieval of existing plans from the experience library that resemble current partial plan; (2) adaptation of retrieved plan to extend and complete current partial plan.

An important task is to determine similarity between existing plan and current partial plan. We assume that previous plans are cached as cases in an experience library and each case consist of a plan annotated with its causal model.

Causal graphs provide a structured framework for computing similarity between plans. During plan authoring we can start with a partial plan and query a historical database for best matching completed plans. Once we have a matching plan and a mapping between similar parts of the plan we can then adapt the matching plan to construct a suitable current plan. The matching can be performed at different levels of abstraction.

The following is an algorithm for computing the similarity between pairs of plans. Similarity is evaluated using an index between 0 and 1, with 0 indicating no match 1 perfect match and the higher the index the better the match. The algorithm compares pairs of plans based on their causal models.

```
Algorithm P-Sim
Input: Two causal graphs G1, G2 of two distinct plans.
Output: Similarity measure between the two plans
Begin
    Compute  causal orders d1=(u1,…,uk1}), d2=(v1,…,vk2) for G1 and G2;
      k1≤k2.
    index←1; #matches←0
    for i=1 to k1
       match←false
       j=index
       while (not match∧j<k2)
          if (type(ui)==type(vj)) ∧(|pa(ui)|==pa(vj)|)
          then match=true;
          j=j+1
       if (match)
       then #matches←#matches+1; index←j
    return Similarity_index = #matches/k1;
End.
```

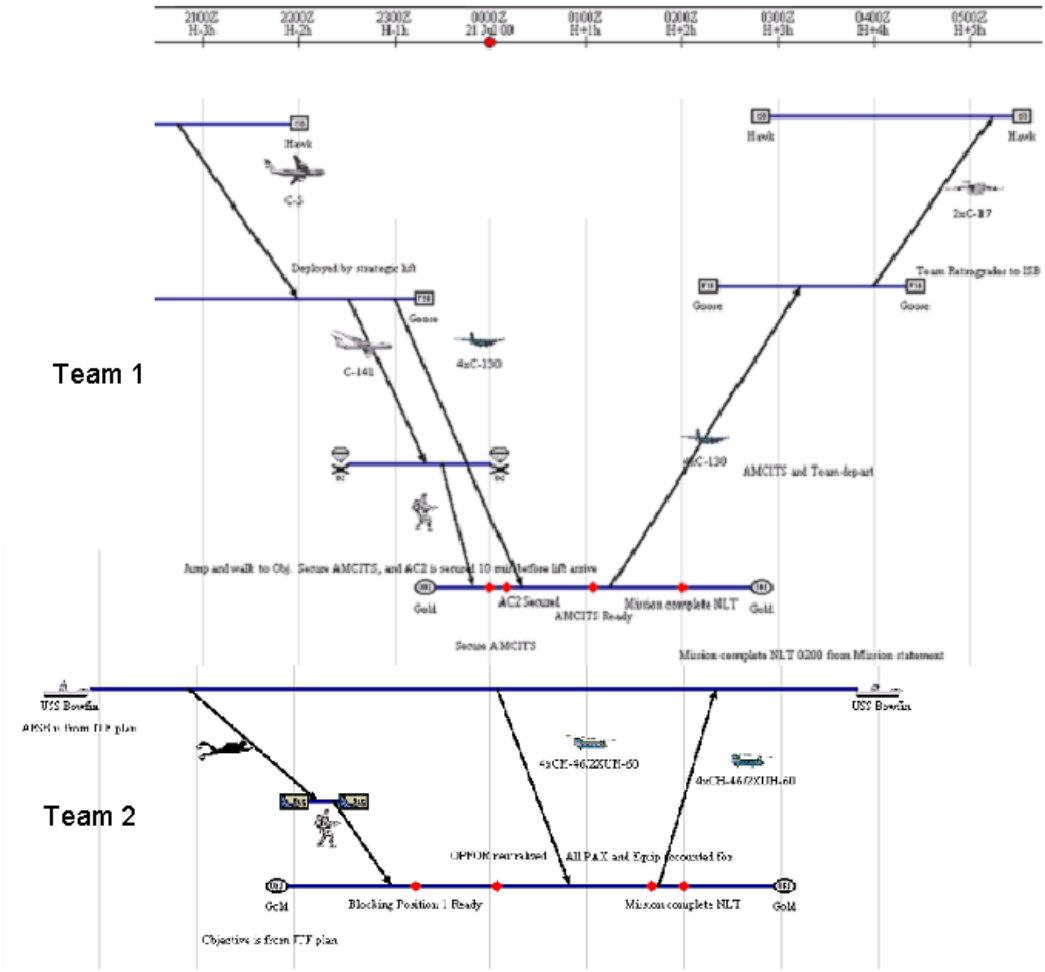

Figure 14 Two plans: Team1 is air-based and Team2 is sea-based.

The idea of the algorithm is to order the nodes in a causal ordering and determine for each pair of nodes whether they match based on their type and the number of parents (i.e., the size of the causal families). The algorithm can be applied to compare plans based on task views (see Section 5.2) by first folding the causal graphs and then applying the algorithm to the folded causal graphs where nodes are blobs representing plan tasks, each containing subsets of events. If the folded graphs have cycles then it can be reduced to an acyclic graph by lumping the strongly connected components in super blobs and then comparing the reduced graphs.

Figure 14 shows two SOF plans one air-based called Team1 and the other sea-based called Team2. Figure 15 shows the causal graphs for both plans. The Figure also shows a causal ordering of the nodes in the graphs. Clearly, there is a very close match between the two plans: 7 out of 8 nodes in the graph of plan-1 match respective nodes in the graph of plan-2. The similarity mapping between plan-1 and plan-2 is: 1→1, 2→2, 3→4, 4→5, 5→6, 6→7, 8→10.

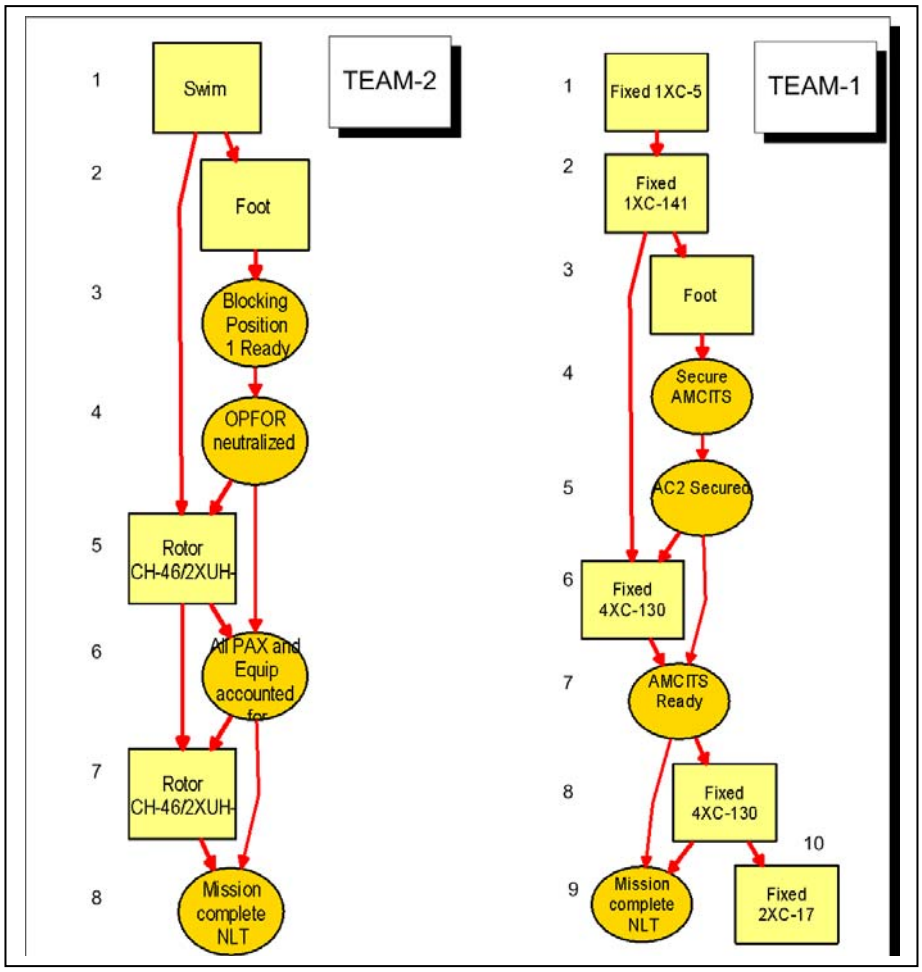

Figure 15 Comparing two causal graphs

10. Related Work

There has been work on capturing plan rationale and causal structure recovery of plans in the context of partial order planning [Kambhampati & Kedar 1994], case-based planning [Veloso & Carbonell 1993], and MIP [Veloso 1996]. A plan rationale aims to explicate why a plan is the way it is or the reason as to why the plan decisions are taken. Unlike previous work which requires complete domain theory of the planning domain, our approach can generate potential causal links in the plan by analyzing the temporal ordering and the task structure of the plan. Requiring complete domain models or expecting the human planner to specify every condition required by the plan operators is not realistic in large real-life planning applications such as in military and space. Our approach complements another direction of research in AI planning focused on qualitative reasoning about plans [Myers 2001]. In that research the focus is on qualitative cause-effect relations and on calculus for reasoning about change. Our work is complementary in the sense that it offers an approach for initial causal models that can be supplemented with qualitative knowledge to address the problem of lack of complete domain models.

In automated planning there has been work on the problem of removing unnecessary orderings in a total order plan (linear plan) in order to produce a "least constrained" [Veloso et al. 1990] or "shortest parallel execution" [Regnier & Fade 1991] partial-order plan. It has been shown that the problem is generally NP-hard [Bäckström 1993]. Our work differs in that our partial order is representing potential causal links without requiring a domain theory and an automated plan validity test as in [Bäckström 1993].

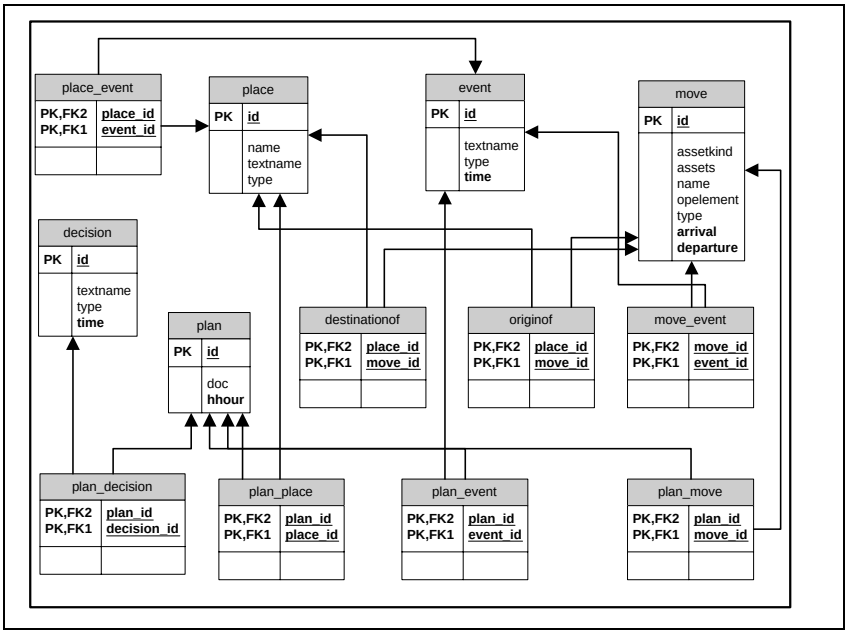

Figure 16 Temporal plan schema for SOFTools.

11. Lessons Learned

The focus of our effort in the initial phase of the program had been to develop a relational database schema for active templates and a set of tools for analyzing the cause-effect relationships between assets, actions, and objectives in a plan. Specifically we built a relational schema and a program that maps the temporal plan XML file to the database. We advocated that bringing planning into the realm of relational databases has several advantages. Databases provide robust mechanisms (commit/rollback) for distributing applications. By developing a transactional model of plan construction, we can allow multiple users (for example, the ground, maritime and air components) to access and edit the same plan. Our initial relational database approach has subsequently been subsumed by an extensible relational model called the Structured Data Model (SDM). The goal of the SDM has been to provide a uniform, easy to use, and complete representation of planning data so that tools developed by various AcT researchers can more easily interoperate.

The most important challenge has been to abduce the cause-effect structure of a special operation plan from the temporal representation entered in SOFTools Temporal Plan Editor. The value of the causal graph was recognized early on as a representation that captures the dependencies in the plan that were not otherwise explicitly represented in the temporal plan representation. As we further developed our causal modeling approach the concept of the relational representation became secondary. The primary focus emerged as the automated building of default causal models from the plan authoring input directly using the XML representation. The causal dependency depicted in the graph is only potential causality expressing consistent "happened before" relation. The heuristics exploit the locality inherent in the temporal editor where plans are composed of subplans of movements and places. The potential causality information can then be edited by the user by adding or removing edges to reflect causal semantics not captured in the temporal plan representation.

Causal views can provide significant enhancement to the conventional approach of monitoring plan execution using checklists. Execution checklists can become very large, and events that indicate related activities at a high-level view of the plan may appear to be widely separated by other events and by time. And there may be a lot of irrelevant events mixed in with ones of interest. This means that when we view a large execution checklist, and an event that indicates a failed or delayed execution turns up, we cannot immediately focus on the other events in the checklist that led up to the failure, nor can we quickly determine impact on future events. Instead, we have to use a lot of knowledge about the mission and the executing plan to try to figure it out. The causal views can be used to monitor event logs at any abstraction level during plan execution, to detect inconsistencies with prior events, to determine violations of critical requirements and impact on future operations.

Causality can also enhance plan authoring in a different way. We can use causality to retrieve similar plans from historical databases. We developed an algorithm for computing complete or partial matching between plans based on their causal models. The basic idea of the algorithm is that two plans are similar if their causal structures overlap. That is, if they have consistent causal ordering and the nodes in the ordered causal families have the same type and same number of parents. We did some preliminary work in that direction in collaboration with other team members in the context of case based reasoning and hierarchical task networks.

12. Conclusions

We developed in this project a computational approach and an automated tool for abducing parsimonious representation of causal dependencies in a plan authored by humans in mixed initiative environment. The tool is designed for the SOF planning domain and takes as input a plan created by a human planner using custom graphical interface (a program called SOFTools TPE) and outputs a minimal DAG representing a parsimonious potential causality graph. The nodes of the DAG are the actions and the directed arcs represent potential causal links and/or threat resolution links as commonly defined in partial order planners. The term "potential" emphasizes the uncertainty in the abduced links since no requirement is placed on the availability of domain theory. Our approach relies on the structural information only; namely the temporal ordering of the actions and the task hierarchy of the plan. The DAG is presented graphically to the human modeler for validation and further refinement. The plan author can amend the potential causality structure by graphically adding or deleting directed arcs in the DAG. The validated causal graph captures the knowledge relevant to the causal dependencies in the plan and can then be used for reasoning tasks such as run-time replanning. There are a number of areas for future extensions of the technology. One area is to exploit the causal model to speed up the plan authoring by fetching similar plans from a case-based library of plans. We have developed an algorithm to compute plan similarity that can be embedded in a case-based reasoning system. Another area is to integrate with qualitative reasoning approaches in planning for reasoning with incomplete information. Requiring complete domain models or expecting the human planner to specify every condition required by the plan operators is not realistic in large military and space applications. Our potential causality approach can be supplemented with qualitative knowledge to address the problem of lack of complete domain models.

13. References

Weld, D. 1999: Recent advances in AI planning. AI Magazine 20, pp. 93–123.

Frank, M.; Muslea, M.; Oh, J.; Minton, S.; and Knoblock, C. 2001. An intelligent user interface for mixed-initiative multi-source travel planning. In Proceedings of the 6th international conference on Intelligent user interfaces, ACM Press pp. 85--86.

Burstein, M.; and McDermott, D. 1996. Issues in the development of human-computer mixed-initiative planning, In Cognitive Technology, B. Gorayska and J.L. Mey (eds.), Elsevier pp. 285-303.

Kambhampati, S., and Kedar S. 1994. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans, Artificial Intelligence, 67, pp. 29-70.

Veloso, M., and Carbonell J. 1993. Derivational analogy in prodigy: Automating case acquisition, storage and utilization. Machine Learning, pp. 249–278.

Veloso, M..1996. Towards mixed-initiative rationale-supported planning. In Austin Tate, editor, Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative,. AAAI Press pp. 277-282.

Myers, K.L. 2001. Toward a theory of qualitative reasoning about plans. Technical report, A.I. Center, SRI..

Veloso, M., Perez, and M., Carbonell, J. 1990. Nonlinear planning with parallel resource allocation. In Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control, pp. 207-212.

Regnier, P., and Fade, B. Complete determination of parallel actions and temporal optimization in linear plans of action. European Workshop on Planning, volume 522 of Lecture Notes in Artificial Intelligence, Springer, pp. 100-111.

Bäckström, C. 1993. Finding Least Constrained Plans and Optimal Parallel Executions is Harder than We Thought. Proceedings of the Second European Workshop on Planning.

El Fattah, Y. 2004. Potential Causality in Mixed Initiative Planning. Proceedings of the 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE)

El Fattah, Y. 2003. Causal Modeler ver. 1.1 Software User Manual.

Sugiyama, K. and K. Misue 1991. Visualizing structural information: Hierarchical drawing of a compound digraph. IEEE Transactions on Systems, Man and Cybernetics, 23(2):235–282.

Buchsbaum A. L. and J. R. Westbrook, 2000. Maintaining hierarchical graph views. In *Proceedings of ACM-SIAM symposium on Discrete algorithms*, pages 566 – 575, New York, NY, USA. ACM Press.