CommonP:
A Common Plan Representation for Air Campaign Plans

Yolanda Gil
gil@isi.edu
INSPECT
USC/ Information Sciences Institute

Robert M. MacGregor
macgregor@isi.edu
JFACC Ontologies
USC/ Information Sciences Institute

Karen L. Myers
myers@ai.sri.com
Continuous Planning and Execution
SRI International

Stephen F. Smith
sfs@cs.cmu.edu
Mixed-Initiative Scheduling
Carnegie Mellon University

William R. Swartout
swartout@isi.edu
INSPECT
USC/Information Sciences Institute

**** DRAFT ****

**** WORK IN PROGRESS ****

December 20, 1998

NOTE: This draft contains the desired representation capabilities of the language and some illustrative examples of the language as we envision it. The specification of the language will be included in future versions of this document.

1. Introduction

The goal of this work is to develop a language, called CommonP, to represent Air Campaign Plans that will support the needs of DARPA's JFACC Program [cite]. This work represents the needs and recommendations of the technology-base projects in the JFACC program. It draws on our previous experience on AI planning and problem solving, knowledge representation and knowledge modeling, and knowledge sharing. Within the JFACC program, the technologies that we are drawing from are:

- The ACT formalism for representing plan templates [Wilkins and Myers], which is the basis of SRI's project on Continuous Planning and Execution.
- EXPECT's language for representing goals and tasks [cite], which is a component of the EXPECT framework used in the INSPECT plan evaluation project from USC/ISI and was the basis for the ACP objective representations adopted by the JFACC program [cite]
- LOOM's knowledge representation and reasoning capabilities [cite], which is a knowledge representation system at the core of the JFACC ontologies project from USC/ISI
- The OPIS modeling language for resources and temporal constraints [cite], which is used in CMU's mixed-initiative scheduling project

An important additional source of information for this work is the representations used in ACPT family of tools for editing Air Campaign Plans according to the strategy-to-tasks approach [cite] and the representations of objectives for air campaign plans developed by ISI [cite]. Appendix A presents some background information about these systems and approaches.

Our approach to developing CommonP is to put together an initial version of the language that combines useful ideas from the above sources as we find them appropriate to support the JFACC program goals. We will test it by representing actual air campaign plans used in JFACC demonstration scenarios and by having our systems reason with those representations. Based on the lessons learned from this test, we will extend the representation and iterate this process.

Our work is also related to efforts in schema representations for the JFACC program, but is more focused on the representations needs that we anticipate from our research thrust within the program. Our work should complement and influence the plan representation components of the JFACC schemaworking group. CommonP can be viewed as a vehicle for input from the JFACC Tech Base R&D to the schema development work. Notice that CommonP is a representation that can be mapped into schemas to support implemented systems.

This effort is related to other efforts in designing common plan representations (including SPAR [cite], KRSL [cite], CPR [cite], etc) but is more focused and specific to the needs of the JFACC program. While CommonP will be closely aligned with these efforts, our

work is a smaller scale and more specific effort that will not need to address some of the hard planning issues that are not specific to this domain and that will be considered by these other groups. Our effort is empirical and practical in nature. CommonP will be extended and updated as we understannd through experience its adequacy to meet the needs of the JFACC program. We will also be responsive to progress in common plan representations from DARPA and from the planning community at large, extending CommonP accordingly as long as it fills a need within the scope of the JFACC program goals. As an empirical piece of work, it should serve as a useful point of reference for these other efforts regarding the representationsl needs of our specific domain.

This document represents work in progress. Comments and requests for updates should be sent to the authors.

We begin describing the assumptions that we make about planning air campaigns as a way of background on what the representation needs to support, then we present and discuss the CommonP language and plan representations, and finalize with a summary of issues that our work does not currently address and that we anticipate will drive future extensions to the language.

2. Designing the Representation

Because ACPT is based on domain expert's view of the planning process, it captures a number of interesting features that we will retain:

- hierarchical decomposition of the plan in terms of strategy-to-tasks planning
- a verb-based representation of objectives, which reflects how users talk about plans
- priorities of different objectives
- success criteria in terms of measures of merit

From our perspective, additional requirements in terms of what the language needs to capture include:

- distinction between ***objectives*** and ***actions*** (*what to achieve* versus *how it can be done*). ACPT only captures the former. This distinction is widely used in AI planning systems
- plan design: chosen plan versus possible alternatives versus unfeasible alternatives. Current representations only capture chosen options
- typical decompositions into subobjectives as plan fragments, which is necessary in order to re-use past experience
- information about the execution environment: execution status of actions, monitoring of events, external events including enemy actions and reactions, etc.
- capture resource allocation to objectives
- uncertainty

- strategy behind the plan
- user advice and guidance
- context for objectives: relating force application objectives to force support and force enhancement objectives
- support for explanation and paraphrasing: for mixed-initiative to work, people must understand plans

3. Planning Air Campaigns

The JFACC program follows the strategy-to-tasks approach to air campaign planning. This approach views planning as an iterative, hierarchical process, where higher level strategies and objectives are decomposed into lower level tasks down to activities. This is well aligned with an AI approach to planning that is known as Hierarchical Task Network (HTN) planning [Erol *et al*, which is the approach taken by the AI planning technology base projects represented in the JFACC program.

We assume the following steps in generating air campaign plans, whether automatically generated by planning software or manually generated and captured by a plan editor. At each iteration, there are two layers of decisions:

- *what is to be achieved*. Depending on the context this is often referred to as a goal, a task, an objective, an activity, etc. Because each one of these terms has many meanings in different contexts, we will denote them as **PG**s.
- *what can be done to achieve something*. This is often referred to as an action, an operator, etc. We will refer to them as **PA**s.

The distinction between PAs and PGs is present in state-of-the-art AI planners but is notably absent in ACPT-style tools. We consider this distinction important for several reasons. First, because it captures the decision making process of generating alternatives and options during planning, and is better suited to representing the notion of *shadow plans*, i.e., plans that are currently not pursued but are valid alternatives that may be used in response to a turn of events. It also enables the representation at each step of the plan of the effects and enabling conditions associated with the actions taken in the world, which makes it easier to represent causal information in the plan. Finally, it makes a finer distinction of what are parent and children of an air campaign objective as they relate to the decision making process during planning.

Air Campaign plans are formed by instantiations of PGs and PAs. We will also represent *PG templates* and *PA templates*, which are non-instantiated descriptions of PAs and PGs. Templates capture generalized knowledge about air campaign planning, for example how

to achieve air superiority over a region. Instantiations of templates result in the plan, for example achieve air superiority over SW Irak is an instantiation of the previous PG template. PG templates represent knowledge about what kinds of things are achieved during air campaign planning, while PA templates represent knowledge about what kinds of things can be done to achieve them.

Air campaign planning is a complex search process through the space of possible alternative plans. Alternatives are generated as follows. For a given PG there can be several PAs that can achieve it. These cause several options in the plan, one of which will be selected as the preferred option, others may be rejected after some consideration, and others may be kept as feasible alternatives to the currently selected option. Some alternatives may be left unexplored. A PA in turn can be decomposed into smaller-grained PGs, that will be achieved by PAs.

Air Campaign planning occurs in the context of a specific military operation, which determines the environment in which the plan will be used. The information about this environment that is used for planning will be referred to as the *world model.* A key part of planning is determining what actions to take within a given situation in order to reach a situation with designated characteristics. We refer to individual situations as a*world state* or simply as *state*. A state may be completely or incompletely specified, with the latter designating a set of states that satisfy the partial specification

Air campaign planning is immersed in a universe of continuous change caused by the execution of actions in the plan, by external events, or by other agents in the environment. We will use *monitors* to represent useful information about the plan's execution environment. Monitors specify how the planner should respond to the occurrence of certain events during execution.

Measures of Merit or Measures of Effect are used by air campaign planners to assess progress towards accomplishing the objectives. To express this information, we will use two different fields:

- intended effects: aspects of the state that we desire to achieve.

- evaluation function: an expression that combines various observable properties of the state to produce an indication of how the intended effects are being achieved.

4. CommonP Representations

The steps that we have followed to develop this representation are:

1. First, we took the Act formalism to provide the structure for plan templates as *Acts*. This will be the basic structure for plans.
2. Then, we extended it with the EXPECT goal/capability description language to describe Act capabilities and goals.

3. We then turned to OPIS scheduling technology to specify resources and temporal constraints.
4. The final step is the mapping of ISI ACP objectives representations to the overall framework.

We begin by describing the core structures in CommonP: PGs, PAs, and monitors. We then go into detail about important substructures of these: capabilities, decompositions, temporal restrictions, resources, states, and measures of merit.

4.1 Core Structures

4.1.1 Representing PGs

PG templates are represented with the following information:

- statement: a description of what needs to be accomplished
- intended effects: desired state characteristics
- evaluation function: function to measure progress regarding intended effects
- temporal restrictions: temporal constraints including those in relation to other PGs
- priority: an indication of the importance or urgency of accomplishing the PG

PGs are represented with the following information:

- statement: a description of what needs to be accomplished
- template: the PG template that the PG is an instance of
- bindings: a collection of constraints that specifies the correspondance between this PG and its PG template
- alternatives: a set of possible approaches for achieving the PG, defined in terms of Pas. They are annotated according to their status. For example, one of them may be selected as currently preferred, others may be feasible but not currently chose, others may be unfeasible or undesirable. This set need not be exhaustive
- intended effects: desired state characteristics
- evaluation function value: value of the evaluation function as plan is executed
- temporal restrictions: temporal constraints including those in relation to other PGs
- parents: a set of PAs each of whose decompositions generated this particular PG. A top-level PG that is given as input to the planer will not have parents.
- priority: an indication of the importance or urgency for accomplishing this PG
- execution status: indicates how the execution of this PG is proceeding (e.g., inactive, in progress, completed, etc)
- MOM status: indicates the current value of the MOM for this PG during the execution of the plan

4.1.2 Representing PAs

A PA template is represented with the following information:

- capability statement: a description of what the PA can accomplish
- preconditions: a set of constraints that describe the class of states where this PA can be applied
- setting: constraints that determine specific instances for this PA
- effects: expected state of the world after PA is applied
- plot: a specification of how the PA can be decomposed into PGs
- primitive: indicates whether the PA is directly executable in the world. If a PA is primitive, no plot needs to be specified.
- resource requirements: a specification of the resources needed to perform the action

A PA is represented with the following information:

- template: the PA template that the PA is an instance of
- bindings: a collection of constraints that specifies the correspondence between this PA and its PA template.
- PG: a PG that the PA is accomplishing
- intended effects: effects of the PA that were desired when the PA was chosen to accomplish the PG.
- instantiated plot: an instantiation of the plot of the PA template followed by this PA
- start
- end
- duration
- children: a set of PGs that are generated from the plot

4.1.3 Representing Monitors

Monitor templates are defined by the following information:

- monitored conditions: an event or collection of events in the world that trigger invocation of the monitor
- preconditions: a set of constraints that describe the class of states where the monitor is relevant
- plot: the tasks to be performed when the monitored conditions are detected (e.g., invocation of PAs, posting of PGs, etc.)

4.2 Substructures

4.2.1 PG Statements and PA Capabilities

As planning proceeds, the capability of a PA that is used to accomplish a PG must be the same as the PG statement. As a result, there is a close correspondence between PG

statements and PA capabilities, and as a result they are expressed using the same language. We use a verb-clause representation composed of:

- a main verb
- a list of arguments, each with a name and a parameter

We will adopt the EXPECT language for stating them, as described in the Appendix.

4.2.2 Alternatives

The alternatives for achieving a PG are a set of partially instantiated Pas that are relevant to that PG. Important types of alternatives are:

- chosen action: an action that is currently preferred among all the possible PAs that can achieve this PG
- feasible alternatives: a set of actions that have been considered and determined to be feasible alternatives to the chosen action to achieve this PG
- failed alternatives: a set of actions that were explored as possible alternatives to achieve the PG and were abandoned for being undesirable or unfeasible options

Each alternative is annotated with information about its status. The status indicates either computational status (e.g., FULLY-EXPLORED), and/or a status indicated by the user (e.g., UNDESIRABLE).

4.2.3 Plots

Plots specify collections of PGs that can serve as the response for a monitor or the decomposition of a PA. The plot can be represented as a directed graph, where nodes are PG statements and links represent sequencing constraints among the nodes. Temporal constraints among nodes can be used to provide additional ordering information such as bounds on start or end times. The decomposition may include conditional branches, parallel nodes, and loops.

We will adopt the Act language to specify plots, as described in the Appendix.

4.2.4 Temporal Restrictions

Temporal restrictions are specified using Allen's temporal logic relations [cite] that includes the following:

- X before Y
- X equal Y
- X meets Y
- X overlaps Y
- X during Y
- X starts Y
- X finishes Y

It is possible to use an extended set, as in:

- X before Y [lb ub]
- X finishes Y [lb ub]
- X same-end-window Y [ub]

where lb and ub indicate lower and upper bound respectively.

4.2.5 Resources

The resource requirements for a PA are specified as:

- resource type
- amount of resource required
- interval of resource requirement

The resources themselves are part of the state description, and include a specification of

- resource type
- resource availability over time
- location

This state information about resources is updated as allocation decisions are made.

4.2.6 World Model and States

Planning requires information about the world in which plans are to be executed. Some of this information is background information that defines and characterizes key objects in the domain (e.g., force structures, munitions, airframes) or the basic physics of the world (e.g., geographic information, models of time). Such background information is static and need be defined only once in initializing the knowledge bases for a system. Additionally, dynamic properties of and the weather may change frequently.

CommonP adopts the following representational structures for world model and state information:

- Class hierarchy for representing domain objects
- Typed predicate representation for representing basic physical constraints of the world and dynamic world constraints

4.2.7 World Model Declarations

Certain conventions can be adopted to simplify the process of representing and reasoning with world state information.

- Static/Dynamic
- Open/Closed
- Functionality with respect to an argument subset

4.2.8 MOMs

MOMs are very important in air campaign planning for several reasons. First, in specifying MOMs for an objective users can ensure that it is a reasonable objective in terms of it being achievable and measurable. Good MOMs are an indication of good plan design. Second, specifying a MOM for an objective supports Campaign Assessment in that it is possible to measure progress towards objectives as the plan is executed. Third, MOMs are useful in a continuous planning and execution environment in that they are an indication of when an objective is becoming unachievable and should be deactivated to pursue other alternative options.

Because MOMs are so central to many aspects of Air Campaign Planning that have not yet been investigated in depth, we expect that their specification willb ecome more ellaborate and precise as the JFACC program progresses. Initially we will follow [Valente 1996], where MOMs are represented with a description of a desired state, one or more qualifiers, and one or more references to other states that the measure is relative to.

5. The CommonP Language for Air Campaign Plans

The development of the CommonP language also follows the steps outlined in Section 4:

1. First, we took the Act formalism to provide the structure for plan templates as *Acts*. This will be the basic structure for plans.
2. Then, we extended it with the EXPECT goal/capability description language to describe Act capabilities and goals.
3. We then turned to OPIS scheduling technology to specify resources and temporal constraints.
4. The final step is the mapping of ISI ACP objectives representations to the overall framework.

Thus, this section is still in a draft form, as we evolve the language from working on examples taken from actual air campaign plans and from the knowledge bases that we have developed. Work to date has concentrated on steps 1 and 2, and started on step 3.

5.1 Examples

Below are samples of a few PAs represented in CommonP (as we explain below, they are based on some SIPE operators for air campaign planning):

```
(defPA
 :IDENTIFIER AOASPD
 :COMMENTS "Achieve offensive air superiority, reducing
      threats in ingress-sector to LOW and others to MEDIUM levels"

 :CAPABILITY (achieve (obj (?oas is (subtype-of offensive-air-
superiority)))
                      (during (?p is (phase))))
 :QUALIFIERS (TEMPORAL (on (?t is (day))))
 :APPROACH ()
 :RESOURCES ()

 :parameters ()
 :constraints ()
 :annotations ()

 :preconditions ()
 :plot
    BEGIN-SEQUENCE
    :Task T1 (breach (obj (subtype-of iads)) (on ?t) (with LOW-RATING))
    :Task T2 (extend (obj (subtype-of air-superiority))
                     (from ?t)
                     (through ?p)
                         (with MEDIUM-RATING))
    END-SEQUENCE
 :resource-requirements ()
 :effects ()
 )


=========================================================

;; Here are three operators that achieve the same goal (breach IADS)
;; but in different ways:

(defPA
 :IDENTIFIER BI1S
 :COMMENTS "Breach IADS in one single sector"

 :CAPABILITY (breach (obj (subtype-of iads))
                          (with (?r is (rating))))
 :QUALIFIERS (TEMPORAL (on (?t is (day))))
 :approach ((in (?s is (subtype-of single-sector))))
 :RESOURCES ()

 :parameters  ((?region is (region))
                 (?ingress-sector is (sector)))
 :constraints ((blue ?region)
```

```
               (red ?ingress-sector) ; original SIPE operator does not
have this
               (borders ?ingress-sector ?region))
 :annotations (INSTANTIATE (?region ?ingress-sector))

 :preconditions ()
 :plot
    :Task T1 (breach (obj ?ingress-sector) (with ?r))
 :resource-requirements ()
 :effects () ;; shouldn't this say "IADS inactive in sector"?
 )

(defPA
 :IDENTIFIER BI2S
 :COMMENTS: "Breach IADS at two sectors"

 :CAPABILITY (breach (obj (subtype-of iads))
                (with (?r is (rating))))
 :QUALIFIERS (TEMPORAL (on (?t is (day))))
 :approach ((in (?s1 is (subtype-of sector)))
             (and-in (?s2 is (subtype-of sector)))
            )
 :RESOURCES ()

 :parameters  ((?region is (region))
               (?s1 is (sector))
               (?s2 is (sector)))
 :constraints ((blue ?region)
               (red ?s1)
                   (red ?s2)
                   (NOT (EQUAL ?s1 ?s2)))
 :annotations (INSTANTIATE (?region ?s1 ?s2))

 :preconditions ()
 :plot
  BEGIN-PARALLEL:
    :Task (breach (obj ?s1) (on ?t) (with ?r))
    :Task (breach (obj ?s2) (on ?t) (with ?r))
  END-PARALLEL
 :resource-requirements ()
 :effects ()
 )

(defPA
 :IDENTIFIER BIHB
 :COMMENTS: "Breach IADS in all hostile border sectors"

 :CAPABILITY (breach (obj (subtype-of iads))
                (with (?r is (rating))))
 :QUALIFIERS (TEMPORAL (on (?t is (day))))
 :approach ((along (?h is (subtype-of hostile-border))))
 :RESOURCES ()

 :parameters  ((?region is (region)))
```

```
 :constraints ((blue ?region))
 :annotations (INSTANTIATE (?region))

 :preconditions ()
 :plot

    ; how to specify forall vars: their types & then constraints
    FOREACH (PARAMETERS ((?sector is sector))
             CONSTRAINTS ((red ?sector)
                          (borders ?sector ?region)))
     :Task (breach (obj ?sector) (with ?r))
 :resource-requirements ()
 :effects ()
 )




==========================================================

(defPA
 :IDENTIFIER ID39

 :CAPABILITY (destroy (obj (subtype-of enemy-SSM-launch-sites))
                          (on (?a is (NW-area))))
 :additional-qualifiers ((with (?w is (subtype-of precision-weapons))))
 :approach ((using (?a is (subtype-of stealth-aircraft))))
 :RESOURCES ()

 :parameters
 :constraints
 :annotations (instantiate ())

 :preconditions (
             )
 :plot
 :resource-requirements ()
 :effects ()
 )


; need to specify that SEAD aircraft is just an escort to
;  the real aircraft that fly the mission.
(defPA
 :IDENTIFIER ?

 :CAPABILITY (destroy (obj (subtype-of enemy-SSM-launch-sites))
                          (on (?s is (area))))
 :additional-qualifiers ()
 :approach ((using (?a is (subtype-of SEAD-aircraft))))
 :RESOURCES ()

 :parameters
```

```
 :constraints
 :annotations (instantiate ())

 :preconditions (
             )
 :plot
 :resource-requirements ()
 :effects ()
 )



 :temporal-constraints ((END (ON-OR-BEFORE D+5)))
```

An example of an objective:

```
PO:  (breach (obj (subtype-of IADS)) (on D+5) (with LOW-RATING))
```

5.2 Discussion

We developed the examples above by taking some SIPE operators and rewritting them into something that could be a first cut at CommonP. The two basic things we did were:

- take out ACT-specific information
- express their purpose, goals, and subtasks in EXPECT's language for goals and capabilities

For example, we took the SIPE operator:

```
OPERATOR: achieve-offensive-air-superiority
;;Reduce threats in ingress-sector to LOW, others to MEDIUM level
ARGUMENTS: phase1, when1;
PURPOSE: (achieve-oas phase1 when1)
PLOT:
   TASK: (breach-iads when1 LOW)
   TASK: (extend-air-superiority phase1 when1 MEDIUM)
```

and expressed it as:

```
(defPA
 :IDENTIFIER AOASPD
```

```
 :CAPABILITY (achieve (obj (?oas is (subtype-of offensive-air-
superiority)))
                      (during (?p is (phase)))
                      (on (?t is (day))))
 :approach
 :preconditions ()
 :effects ()
 :plot
    BEGIN-SEQUENCE
    :Task T1 (breach (obj (subtype-of iads)) (during ?t) (with LOW-
RATING))
    :Task T2 (extend (obj (subtype-of air-superiority))
                         (during ?p) (on ?t) (with MEDIUM-RATING))
    END-SEQUENCE
 :primitive NOT
 :resource-requirements ()
 )
```

We run into several problems. One was that it became hard to express the goals and capabilities, specifically the "approach". Another was that the slot "instantiate" was needed by SIPE but did not look like it would be of interest to a human planner. Finally, we were not sure whether the constraints on the parameters should be separate or part of the preconditions (ex: that the region has to be blue). The rest of this section summarizes some of our thoughts and proposals to address these problems.

One thing that did not work very well is to try to do a straightforward mapping of the SIPE operator's purpose into EXPECT's language for capabilities. Instead, we had to go back and think about what the arguments meant, which ones were likely to appear in an operator and which ones would appear in an objective, and separate them. This really amounts to going back to the stage of modelling the knowledge that the operator is supposed to capture.

We structured the goal and operator capability descriptions in several subfields, each subfield has parameters of different nature:

- capability: operator capabilities should have the parameters that are intrinsically part of the capability and have to be present in a goal in order to match with the operator. ex: breach iads.

- context: additional parameters that restrict the operator capability further . This includes, for example, variables that essentially pass along book-keeping information that doesn't directly impact the details of the PA. For example, (on ?day). It's common to have to pass along a lot of these kinds of keeping variables within planning systems, which can make the purpose description for operators rather involved and confusing.

- approach: specifies/summarizes the rationale behind the decomposition proposed by the operator. For example, a goal to breach iads can appear in two operators, one takes the approach of breaching in one sector and another breaching in two sectors. The parameters in the approach may be posted in the goal, if so they really represent guidance on how to achieve the goal.

- resources: specifies a resource constraint or assignment for achieving the goal. For example, using stealth aircraft.

The approach may be represented more appropriately following the style of the strategic properties in SRI's Advisable Planner.

The Parameters slot is used to specify taxonomic-type constraints on variables. The Constraints slot is used to represent other constraints on the parameters, including constraints involving multiple variables. Preconditions represent conditions that can be subgoaled on, i.e., that can be achieved by other steps added to the plan.

Preconditions and effects can be specified either as part of the operator as a whole, or they can be associated to specific substeps within the plot.

An annotation slot would include information that users may not necessarily want to see. Some of this information may be system-specific. For example, this would hold a parse tree for the capability, which is used to pass information from the MASTERMIND objectives editor to the INSPECT air campaign plan critiquer. In previous work (MAPViS/TIE 97-1) we ended up putting this information as a comment in Acts.

For now, the annotations slot also holds the "instantiate" information found in the Acts. This tells SIPE that it should select a value for the variable when applying the operator, rather than continuing with a ``least commitment'' approach of simply accumulating constraints. Eventually, this information would need to go into a slot that specifies this and other kinds of search control knowledge (such as ordering information among constraints/preconditions to indicate preferred search paths), but we have not done this work yet. This search control knowledge indicates a critical `decision point', and it is of relevance to human planners. For example, within the PA with ID BI1S, the Instantiate could be interpreted as a `hook' to requesting information from the user about which sector should be used as the ingress point.

6. Discussion

CommonP is being designed to support the representation of a wide range of information about an air campaign plan, but does not require that all the information is provided at every step. This is because plans will often be represented incompletely in practice. During manual plan creation, a user may not specify some of the items because they may not be available or because of the time required to specify them. When parts of a plan are

generated by planning software, some items may not be specified because they are unknown to the system.

7. Conclusion and Future Directions

Below is a collection of issues that we anticipate will be needs within the JFACC program that will need to be addressed in future extensions of the representation:

- specifying states
- measures of effect, end states
- centers of gravity and their relation to the air campaign plan
- uncertainty
- user advice and guidance for the planning task
- control knowledge to guide and influence choices
- external events
- enemy's actions and reaction

Even though our representations are specific to represent air campaign plans they may need to be further refined to accommodate important distinctions in this domain. The JFACC program includes in its scope planning for force application, force support, and for force enhancement. Our representations will accommodate any necessary distinctions as more details about the requirements and functionality of these planning subdomains are understood as the program progresses.

Appendix A: Background

A.1 The Act Formalism

An Act describes a set of activities that can be undertaken to fulfill some specified purpose in designated circumstances. The purpose could be either to satisfy a goal or to respond to some event. As such, an Act can represent, among other things, an executable procedure, a planning template, or a plan at a given level of refinement.

Each Act is comprised of the following components.

- **Name** unique identifier for the act
- **Cue** the purpose of the Act (i.e., the reason for invoking it)
- **Preconditions** gating world-state conditions on the applicability of the Act
- **Setting** conditions for binding local variables
- **Resources** resources that are held or consumed for the scope of the Act
- **Properties** user-definable attributes, temporal constraints
- **Plot** a partially-ordered set of activities that will lead to the accomplishment of the purpose of the Act

The first six of these components describe the conditions under which one might apply the Act. In contrast, the plot describes a set of activities that can be executed to accomplish the purpose of the Act.

An Act is constructed from a set of *goal expressions*, each of which consists of one of a predefined set of *goal modalities* applied to a logical formula. The goal modalities designate different types of desired activity, including goals of achievement, maintenance, and testing (among others). The logical formula specifies either an action, or desired world-state characteristics. While general formulas are supported, most domains focus on simple formulas consisting of relations or negated relations. Below are several example goal expressions within Act:

```
(TEST (Threat Kennedy Schanjok-Sector))
(ACHIEVE (Breach-IADS Sentani-Sector LOW)
(WAIT (BDA-Report Mission-5 Rockatoon-Sector))
```

The Cue of an Act can be constructed from a range of different goal modalities. Most often, Acts are written to *test* conditions, to *achieve* tasks, or to *respond* to new events or information. Preconditions, Setting, and Resources are formulated primarily in terms of `TEST` goals (although additional goals are useful in certain situations).

The plot of an Act consists of a directed graph, whose nodes represent actions and whose arcs impose a partial order for execution. Associated with each plot node is a list of goal expressions, which characterize the activities that must be accomplished to fulfill the

purpose of the Act. Additionally, plot nodes can contain a variety of attributes, including temporal constraints based on the Allen relations [Allen] (thus enabling general temporal relationships among nodes). The Act plot further embeds control information that enables conditional branching, iteration, and parallelism.

Links between Acts and between nodes in different Acts can be specified to indicate a variety of inter-Act relationships. In particular, decomposition links can be specified that show how a high-level objective has been refined to a set of increasingly more detailed Acts.

A.2 EXPECT's Language for Goals and Tasks

Goals are represented as verb clauses with typed arguments. Each argument has a name and a parameter. The parameters may be of the following types:

- a specific instance, ex: the USS Coronado, represented as `(the-instance USS-Coronado)`
- an abstract concept , ex: air superiority, represented as `(the-concept air-superiority)`
- an instance type, ex: battleship, represented as `(battleship)`
- a concept type, which includes all its subtypes, ex: C2 structure, represented as `(subtype-of C2-structure)`
- a set of instances, intensionally or extensionally specified , ex: Mexico and Canada, represented as `(the-set Mexico Canada)`
- a set of concepts, intensionally or extensionally specified , ex: MRCs and OOTW, represented as `(the-set MRC OOTW)`

An instance of an objective would be:

```
(disrupt (OBJ (the-set EWR IADS)) (IN (SW-Irak)))
```

The first argument is the direct object of the verb, marked with the notation `OBJ`.

A PA template that could be applied to achieve this objective would be:

```
(disrupt (OBJ (set-of (subtype-of C2-structure))) (IN
(region)))
```

A.3 The ACPT Family of Editors for Air Campaign Plans

Editors for air campaign objectives capture the following information about air campaign objectives:

- objective title
- action (subset of the title)
- effect (subset of the MOM)
- MOM
- COGs
- location
- % completed
- review status
- parents
- temporal constraints
- priority

A.4 ISI's Structured Representations of Air Campaign Objectives

ISI's EXPECT group developed the INSPECT air campaign plan critiquer as part of their participation in the Fourth Integrated Feasibility Demonstration of the DARPA/Rome Laboratory Planning Initiative (ARPI's IFD-4). For IFD-4, ACPT was extended by integrating several components, including SRI's SIPE and ISI's INSPECT. We found early on that in ACPT the objectives were described as strings, which lack the structure that an automated tool (such as INSPECT) needs in order to reason about them. We embarked in an effort to provide a formalization of air campaign objectives, which turned out to improve the editor and to be useful to other air campaign planning tools as well. Operational users found our structured representations very useful, because an editor would enable them to be more precise in representing objectives and because they resulted in standard statements of objectives that could be shared and understood by everyone. For example, ACPT allowed them to state an objective as "conduct operations", which is too vague, or "gain air superiority", which is imprecise because it does not specify the geographical area within the theater that is intended.

Working with Air Force experts from the "Checkmate cell" at the Pentagon, we developed
a structured representation of air campaign objectives [cite] that is based on the EXPECT representation for goals (see Appendix A.2). In subsequent work, we developed grammars for other kinds of objectives, including force support and defensive air operations. In our initial analysis, which was concerned with offensive air operations, less than 30 different verbs were used. Each verb had a precise meaning that the experts would agree upon after some discussion. For example, the experts would point out that an objective to destroy radars and early warning systems was inadequate, and that it was preferable to state the truth intent, for example to disrupt the facilities whether they were

destroyed or not. Once the main verbs were agreed to, we discovered that the verb itself put considerable constraints on the roles that were applicable, making a case grammar an appropriate way to structure the objectives. Several editors that make use of these grammars have been built, the most widely used one is the Mastermind objectives editor built with adaptive forms [cite].