Generating Space Telescope Observation Schedules

Nicola Muscettola, Stephen F. Smith,
Gilad Amiri, and Dhiraj Pathak

CMU-RI-TR-89-28

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

November 1989

This work was sponsored in part by the National Aeronautics and Space Administration under contract # NCC 2-531 and the Robotics Institute

Table of Contents

List of Figures

Abstract

In this paper, we describe HSTS, a system that constructs executable observation schedules for the Hubble Space Telescope (HST). HST observation scheduling is a complex task, requiring attendance to a myriad of constraints relating to orbit characteristics, power and thermal balance requirements, instrument capabilities, viewing conditions, guidance requirements, overall allocation objectives, and astonomer specific restrictions and preferences. HSTS provides a general framework for representing and solving such complex scheduling problems. Generally speaking, scheduling in HSTS is viewed as the process of constructing a prediction of the behavior of a physical system (e.g. the HST operating environment) that reflects specified goals and constraints. The HSTS architecture provides a *domain description language* for specifying the structure and dynamics of the physical system, a *temporal data base* for modeling possible system behaviors over time, and an opportunistic, constraint-directed *scheduling methodology* for constructing a system behavior (or set of behaviors) consistent with stated scheduling goals and constraints.

1 Introduction

The Hubble Space Telescope (HST) is a sophisticated $1.4 billion observatory that is currently scheduled to be placed in low earth orbit in early 1990; its expected operational lifetime will be approximately 15 years. Once deployed, HST will present the astronomical community with unprecedented opportunities for exploration of the universe. With its complement of six viewing instruments, HST will allow astronomers to observe and analyze celestial objects at a distance 7 to 10 times further than is currently possible from existing ground-based observatories. Contention for viewing time among prospective users is understandably expected to be high, and efficient management of telescope operations so as to maximize scientific usage is thus a critical operational concern.

The development of observation schedules for HST is a large and complex task. From a pool of observation programs accepted by the HST allocation committee in a given calendar year, several thousand observations must be selected and placed on a time line, and decisions are subject to a myriad of constraints relating to orbit characteristics, power and thermal balance requirements, instrument capabilities, viewing conditions, guidance requirements, overall allocation objectives, and proposer specific restrictions and preferences. Many of these constraints relate to the actual dynamics of carrying out telescope observations, and one source of complexity lies in generating feasible sequences of spacecraft activities (e.g., warming up and cooling down of instruments, pointing maneuvers, communication of data, etc.) that realize desired telescope behavior. Complexity also derives from the fact that it is generally not possible to satisfy all problem constraints. In situations of conflict, requirements must be selectively relaxed (e.g., by dropping specific observations or observation programs) and appropriate compromises among various objectives and preferences must be determined.

Current support for HST observation planning and scheduling is provided by the Science Operations Ground System (SOGS), a $70 million FORTRAN-based software system developed by TRW for use by astronomers at the Space Telescope Science Institute (STScI). SOGS has had a somewhat checkered past [17], due in part to the complexity of the scheduling problem and the constraints that must be taken into account, and in part to the difficulty of developing a solution via traditional specification-based software engineering practices and conventional programming languages. Evolving specifications of various telescope capabilities and operational constraints, for example, have necessitated rewrites to major portions of the system, and the effects of these changes on other aspects of the system are still being discovered and debugged. Under development since 1981, the software has only recently reached the point where it appears capable of providing an acceptable level of support. At the same time, the system implements a fairly rigid approach to solving the problem. The treatment of different constraints is inextricably bound to the underlying scheduling algorithm. Unanticipated constraints (e.g., the fact that an observation might be causally related to more than one other observation) cannot be easily accommodated, and many sources of scheduling flexibility admitted by the current HST proposal specifications [15] simply cannot be exploited. This, in turn, places limits the system's ability to effectively address the observation scheduling problem.

The Hubble Space Telescope Scheduling (HSTS) project was initiated at Carnegie Mellon by NASA with the goal of developing a more effective approach to scheduling problems, like that of managing HST operations, that require attendance to a large and complex set of constraints and objectives. It was motivated specifically by a desire to explore the broader applicability of techniques for *opportunistic, constraint-directed* scheduling previously developed and validated in complex factory scheduling domains [7, 14, 11]. Constraint-directed scheduling is an incremental methodology wherein heuristic knowledge relating to characteristics of current solution constraints is exploited as a means for dynamically structuring the search for a solution. Given its complexity and the perceived limitations of current solutions, the HST scheduling problem was selected as an appropriate domain for this work. Consideration of this problem has led to considerable generalization of previously developed techniques. Principal, in this regard, is the development of a general framework for representing and reasoning about constraints that reflect the dynamics of the physical system to be controlled, and a scheduling methodology that integrates such *state-dependent* problem solving with constraint-directed problem structuring. This extends the applicability of our previous work in factory scheduling to domains that require attendance to a wide variety of state-dependent conditions in addition to resource availability. In the context of HST, we believe this approach overcomes the inherent inflexibility of the existing SOGS scheduling software and provides a basis for constructing observation schedules that more accurately reflect the full range of HST scheduling constraints and objectives. More generally, we believe the approach is relevant to a much wider range of scheduling applications.

In this paper, we describe HSTS, a prototype system for short-term (one week to one month) telescope observation scheduling that implements this generalized approach. Generally speaking, scheduling in HSTS is viewed as the process of constructing a prediction of the behavior of a physical system that reflects specified goals and constraints. The HSTS architecture provides a *domain description language* for specifying the structure and dynamics of the physical system, a *temporal data base* for modeling possible system behaviors over time, and a *scheduling methodology* for constructing a system behavior (or set of behaviors) consistent with stated scheduling goals and constraints. To provide a context for describing this scheduling architecture, we first examine in more detail the nature of the HST scheduling problem and its constraints, and the concept of opportunistic, constraint-directed scheduling that motivates our approach.

2 The HST Observation Scheduling Problem

As indicated, the HST observation scheduling problem is one of detailing a timetable of telescope activities that implements, to the extent possible, a set of observation programs that have been previously selected for execution by the HST allocation committee from a larger set of submitted proposals. According to current telescope management policies, proposals will be reviewed on an annual basis, at which time selections of programs for the coming year will be made. The number of programs accepted in a given year is expected to exceed the actual viewing time available. Accordingly, some number of accepted programs will be designated as supplemental, implying that their inclusion in the telescope's schedule is not guaranteed. Generally speaking, the HST scheduling problem is one of maximizing the amount of science viewing time while attending as much as possible to the diverse preferences of specific observation programs and insuring feasibility with respect to the complex set of constraints surrounding operation of the telescope and execution of observations.

One broad class of scheduling constraints are those imposed by users to achieve desired scientific objectives. The HST observing proposal specification language [15] provides the astronomer considerable flexibility in defining observation programs. A variety of relationships among constituent observations may be specified, including partial orderings over sets of observations, temporal separation constraints between ordered observations, temporal grouping constraints over unordered observations, coordinated parallel observations with different viewing instruments, same telescope orientation constraints for multiple observations, and conditional execution of dependent observations. The observations in a given program may be individually prioritized, and it is also possible to specify preferences with respect to observation completion levels (e.g., 25% completion is minimally acceptable, 50% completion would be desirable, 75% completion is the maximum required). Uncoordinated parallel observation programs can also be defined, in which case constituent observations are intended to be executed in parallel with those of other programs if their constraints can be mutually satisfied.

The execution of a specific observation requires the satisfaction of many additional constraints, relating primarily to the dynamics of telescope operations. Consider the following hypothetical observation:

- OBSERVATION 1: Take a picture of Globular Cluster NGC 288 sometime between October 1st, 1990 and October 31st 1990. The picture has to be taken with the Wide Field camera (WF) in normal configuration. The exposure must have a duration of 1 minute.

This specification accurately describes what the user wants to obtain from the telescope but leaves unspecified the operational constraints associated with actually executing the exposure. In fact, these constraints relate more to the physics of picture-taking with the telescope, and are usually independent of the particular observations to be executed.

Two of the conditions that have to be satisfied in order to perform any observation with the WF are:

1. The telescope must be pointed at the target while the picture is being taken;

2. The WF must be operational (i.e., exposing) for the specified duration.

Each of these conditions, in turn, places additional constraints on the required state of the telescope and/or of the surrounding environment with which the telescope interacts, which must be similarly established for the exposure to take place. The concatenation of constraints that is required is analogous to the expansion of planning goals in a planner [5, 18]. However the nature of the constraints to be represented is such that they are not easily expressible in current planning representation frameworks.

Returning to our example, if Condition 1 is not already satisfied, it can be achieved by rotating (or slewing) the telescope from the orientation required by the previous target to that of NGC 288. No slewing would be required if the observation scheduled to occur immediately before OBSERVATION 1 was also of NGC 288. More generally, the duration of a slewing operation depends on the position of the previous target on the celestial sphere, being a function of the sweeping angle between the two orientations and the slewing angular velocity.[1] Such a duration can therefore be calculated only when an observation sequence has been determined.

Pointing at a target, however, requires more than just slewing the telescope. For the entire duration of the observation, the telescope must be "locked onto" the target, so that the target will not move within the WF's field of view. This is accomplished through a closed loop position control system that locks two of the telescope's three Fine Guidance Sensors on a pair of known "guide stars" in the neighborhood of the target. Guide star acquisition requires time, and locking can only occur when the guide stars are visible to the telescope. As a first approximation, we can consider the visibility the guide stars to coincide with the visibility of the viewing target.

Since the telescope is orbiting at low altitude (355 to 380 miles), almost every target will be periodically occulted by the earth. Similarly, pointing restrictions relative to the sun and moon will restrict viewing opportunities. However, as indicated above, the visibility windows for any given target that are implied by these constraints will be deterministically known within the short-term scheduling horizon.

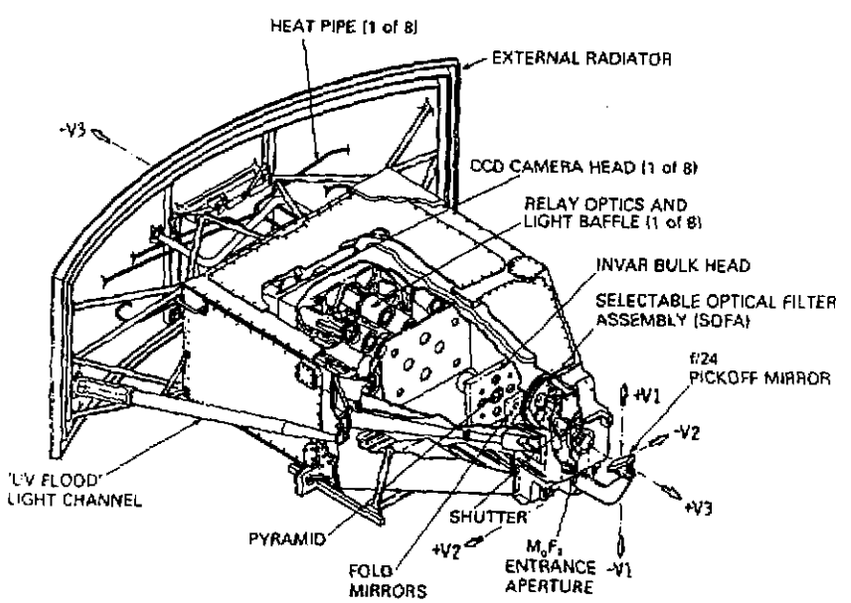

Figure 1: The Wide Field, Planetary Camera (WF/PC) (taken from [15])

The achievement of Condition 2 (WF must be operational for specified duration), typically requires the execution of complex sequences of setup operations. Relative to our example, the WF is in fact only part of a more complex instrument called WF/PC (Figure 1); it consists of two separate CCD sensors with different fields of view, the WF and the Planetary Camera (PC). Both detectors share a service base, which we will refer to as WFPC. The base instrument and both detectors have an associated operating status, which can undergo the warming up and cooling down transition sequences described in Figures 2 and 3 respectively.

These transition graphs, however, are not independent. A first constraint requires that PC remain in the off state

[1] Actually, pointing restrictions relative to the sun make the calculation a bit more complex.

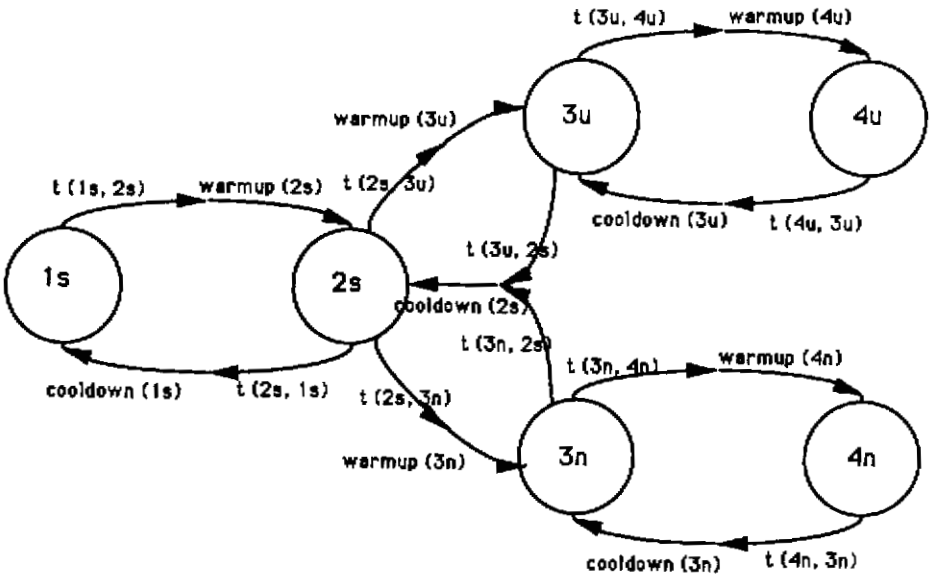

Figure 2: The WFPC warm-up/cool-down transition network

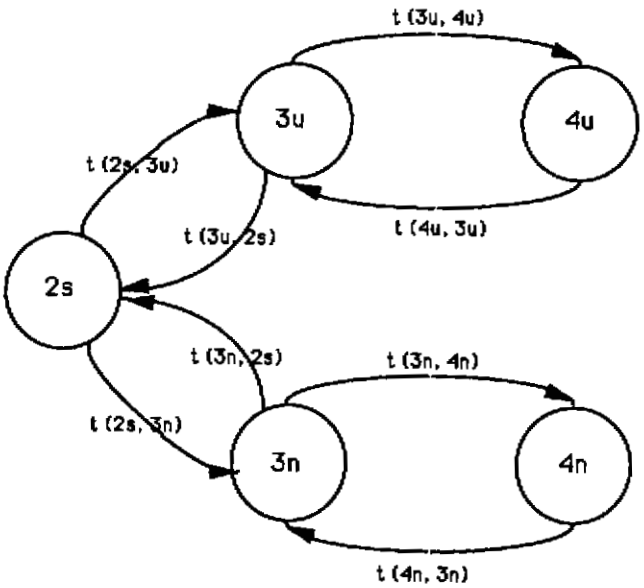

Figure 3: The warm-up/cool-down transition network for both the WF and PC detectors

(state 2*s* in Figure 3) during any time that WF is not in the 2*s* state, and vice versa. The WF (or PC) can also only be in one of the two possible operational states (4*u* and 4*n* in Figure 3) if the WFPC is in the corresponding operational state. Other more complex conditions require synchronization with an entire transition process. While the WF (or the PC) is in 3*n* (3*u*) state, for example, the WFPC can only be in any of the states pertaining to warm-up or cool-down sequences of transitions between the corresponding 3*n* (3*u*) and 4*n* (4*u*) states depicted in Figure 2.

The specification of an exposure also implies specific communication requirements, for transfer of results to the ground, and, in some cases, for real-time analysis of the data. Communication constraints are a function of the specified instrument and viewing mode (the WFPC has only a single "imaging" mode), as well as various user specified special requirements (e.g., immediate down-linking of data). The execution of supporting communication activities is subject to constraints that typically differ from those required by the actual exposing activity. For example, transmission of data requires both visibility of a communication satellite and the availability of an appropriate communication link; storage of results for later transmission requires sufficient on-board tape recorder capacity. Observations may also be designated as "interruptible" in the event of earth occultations, in which case additional activities are required to reestablish the target lock after each occultation period. Periodic passage of the telescope through the South Atlantic Anomaly (SAA), a portion of the Van Pelt radiation belt, also restricts viewing opportunities.

3 Opportunistic, Constraint-Directed Scheduling

The magnitude of the overall observation scheduling task (i.e., producing a year long observation schedule) suggests a hierarchical decomposition of the problem over different time scales and a successive refinement approach to the development of a detailed solution. This view is further supported by the fact that there are various degrees of uncertainty associated with different constraints over different time horizons. For example, constraints relating to the telescope's position in its orbit (e.g., when the telescope will be in the earth's shadow, when a communication satellite will be visible) can only be known with certainty up to a month or so in advance. At the same time, these constraints can often be treated in a statistical sense over longer planning horizons (e.g., percentage of dark time expected in a given time period). Such constraint models, when considered in combination with other constraints whose implications can be seen at coarser temporal granularities (e.g., constraints restricting how close the telescope may point toward the sun, which may preclude viewing a target for a two month period during the coming year), can provide a basis for aggregate allocation decisions. These ideas have been effectively exploited in [9] to schedule observations into successively finer grained time segments (e.g., monthly/weekly time buckets).

We presume a similar approach to managing the complexity of the overall problem, and our work focuses specifically on the short-term (one week to one month) observation scheduling problem. In this context, all orbital constraints are known with certainty, and the objective is a directly executable observation schedule. The character of this problem differs in one important respect from that of the coarser granularity planning described above. At aggregate levels, it is not possible to ensure feasibility of the schedules produced (i.e., that there is sufficient viewing time in a given time segment to execute all scheduled observations), and it makes little sense to treat "viewing capacity" constraints in a rigid absolute manner. In fact, a certain amount of oversubscription is desirable to provide short term scheduling flexibility, and the aggregate scheduling concern is more one of appropriately distributing capacity requirements over the long term scheduling horizon. On the other hand, construction of a detailed short term schedule does require a guarantee of executability. This implies absolute consistency with all constraints relating to viewing capacity (i.e., the actual "physics" of carrying out telescope observations), and definite decisions as to what observations can and can't be performed in the current short-term horizon.

One approach to detailed scheduling that has demonstrated an ability to effectively cope with a large and conflicting set of constraints is *constraint-directed scheduling* [7, 14, 11]. This approach forms the basis of the OPIS scheduling system [14], and its effectiveness has been experimentally validated in the context of complex factory scheduling problems [11]. In brief, constraint-directed scheduling is an incremental problem solving methodology based on repeated global analysis of the characteristics of constraints implied by the current partial solution (e.g., intervals of likely resource contention, relative flexibility of different activity time constraints, conflicts in the current partial schedule) as a means for structuring and exploring the underlying search space. The approach presumes an ability on the part of the problem solver to decompose the overall problem in different ways.

Heuristic knowledge relating the implications of analysis results (i.e., perceived needs and opportunities for optimizing along different dimensions) to the relative strengths and weaknesses of alternative problem decomposition strategies provides a basis for focusing attention at each step (i.e., which decisions to consider next and which scheduling criteria to emphasize). Commitments are thus made in an *opportunistic* manner (i.e., there is no a priori constraint on the order in which decisions are made). As specific commitments are made, current solution constraints are updated to reflect their consequences. This approach represents a radical departure from traditional dispatch-based scheduling methodologies [12], wherein commitments are generated in a strict forward time order. It enables the scheduler to focus immediately on those decisions most critical to overall optimization of scheduling objectives as opposed to encountering them only after other restricting commitments have necessarily been made.

In considering the applicability of this previous work in constraint-directed factory scheduling to the HST short-term scheduling problem, one distinction of consequence between domains is the relative complexity of state-dependent constraints that must be attended to. In the factory scheduling domains previously considered, necessary state-dependent activities (in this case concerning the "setup" of resources for specific manufacturing operations) were simplistic and themselves relatively unconstrained. Given these constraint characteristics, it was possible to operate with modeling assumptions only slightly more sophisticated than those of classical scheduling [2]. In particular, OPIS adopts a simplified view of the state of resources over time, explicitly modeling only their available capacity, and assuming all other aspects of their state to be a function of the last activity performed. Resource setups are implicitly modeled as adjustments to the durations of activities that require them.

In the HST domain, in contrast, it is simply not possible to operate under such modeling assumptions. The complexity and interacting nature of the state-dependent constraints on telescope observation prevent the definition of accurate implicit models (e.g., observation "setup durations"), instead requiring an explicit model of the actual world state and the on-line expansion of sequences of activities to satisfy observation setup constraints. At the same time, given the large-scale nature of the overall problem (e.g., the scheduling of several hundred exposures over a 1 week horizon), such detailed reasoning can only be feasibly approached once some commitment has been made relative to where on the time line specific observations are to be placed. Thus, it is evident that analysis and opportunistic commitment with regard to specific observations must take place relative to approximate models of current solution constraints, and as such, these commitments can, at best, provide constraints on the actual decisions that must ultimately be taken. Such commitments must be subsequently refined so as to both insure their feasibility (i.e., that requisite activities can be accomplished in a manner consistent with the decision and the current partial schedule) and attend, as much as is possible, to their optimality (i.e., that the final placement of all constituent activities on the time line reflects relevant scheduling objectives).

This approach to scheduling forms the basis of the HSTS scheduler. In the following sections, we describe the system components that enable such decision-making, considering in turn the domain description language, the temporal data base, and the problem solving methodology.

4 Modeling the Dynamics of Telescope Operations

As indicated at the outset of this paper, scheduling in HSTS is viewed as the process of constructing a behavior of a system that satisfies given constraints. Assuming this view, the first problem to address is that of describing the structure and dynamics of the system to be managed. Our approach reflects the following broad modeling requirements:

- **Representational adequacy:**

  In-depth analysis of the HST scheduling problem has led to the identification of several representational requirements:

  1. the ability to model actions and states that have definite, and often context-dependent, durations (e.g., slewing time).
  2. the ability to deal with actions and events that depend on the occurrence of particular combinations of states as opposed to the execution of explicit actions (e.g., a lock on a target is lost if the visibility window closes)

3. the ability to model not only sequence constraints among actions and states but also constraints on their parallel occurrence (e.g., constraints on WFPC reconfiguration).

The HSTS modeling framework addresses all of these issues.

- **Independence from the particular application:** It is evident that the representational issues identified above are not unique to the HST domain, but are instead common to a wide range of scheduling problems. For example, a representation of context-dependent durations is fundamental to management of automated factories where mobile robots are used to move parts around, as optimization of paths and travel times would necessarily play an important role.

  The HSTS modeling framework provides a general approach to the representation of physical systems, which we have used to construct a specific model of the HST operating environment.

- **Independence from the problem solving strategy:**

  Use of an opportunistic scheduling methodology implies the need to variably adopt different problem solving strategies during the construction of system behaviors (e.g. backward chaining, forward simulation). Thus, the system description must truly encompass all possible behaviors of the system, and be decoupled from any assumptions about the nature of problem solving strategy to be applied.

  The HSTS modeling framework achieves such independence by clearly separating the description of the structure and the dynamics of the system, which is of general use, from any heuristics and preferences that might be added to the model to specialize it with respect to a specific problem solving strategy.

The development of frameworks for the representation of system behaviors has always been a central component of AI research in planning. However, this work falls short when viewed in terms of the specific modeling requirements identified above. Traditional STRIPS-style representation formalisms [5, 18] are clearly inadequate, since they represent actions as instantaneous transitions among states and therefore assume an implicit model of time. More recent work has addressed many of the limitations of this approach. However, in some cases [1, 8, 16, 13] the specification of actions has retained a central role, making it difficult to treat parallelism constraints among different action/state sequences. In other cases [4, 6], the domain representation language is primarily justified in terms of the particular problem solving style that is applied to the system model.

In the following subsections, we describe the salient features of the HSTS modeling framework. We first consider the basic primitives for specifying system structure and dynamics, and then the extensions necessary to accommodate multiple levels of representation.

4.1 The HSTS Domain Description Language

Within the HSTS domain description language, a system is defined, at the basic structural level, as a collection of interacting parts or **individuals**. An individual is seen as an instance of a prototype **class**. Each class is characterized by a particular set of **properties**. For example, the optical system of HST is defined as an instance (the sole instance in this case) of the class of space telescope optical systems. For purposes of scheduling, the state of an optical system is fully specified once one knows what it is pointing at. Thus, the sole property of this class is POINTING STATUS. Another important class of individuals is that of **fixed targets** (stars, globular clusters, galaxies, etc.) (see Figure 4). One of their fundamental properties is their position on the the celestial sphere, identified by a <Right Ascension, Declination> coordinate pair; another is their visibility with respect to the space telescope.

{{**fixed-target**
LOCATION:
VISIBILITY: }}

Figure 4: The class of fixed targets

A fundamental assumption relative to the specification of properties is that it must be conceptually possible to

associate one and only one value to each property of each individual in the system model at any instant of time. In general, a value of a property of an individual is seen as a description of some relation existing among that individual and other individuals in the system. Some properties are **static**, i.e., their value does not change over time. Others are **dynamic**, i.e., their value might change over time. Referring again to the example in Figure 4, a fixed target's LOCATION is a static property while its VISIBILITY is dynamic (i.e. some times it will be visible and other times it will be occulted). Other examples of dynamic properties in the HST domain include the POINTING STATUS of the telescope's optical system and the OPERATING STATUS of an instrument.

Dynamic properties of individuals are referred to as **state variables**. A **behavior** of the system is an an evolution over time of the values of the state variables that is consistent with the system's specifications. Given the previously stated assumption regarding the association of values to properties, it follows that a behavior of the system is completely specified once a value has been associated with each state variable for each instant of time. Scheduling is concerned with the construction of such behaviors, and we will consider the representation of specific behaviors in a subsequent section. For now, we are concerned with specification of the *possible* behaviors that can be realized by the system.

To fully define a given state variable, it is necessary to specify the set of values that it can possibly assume. Since values represent relations among individuals, a set of values is a set of tuples belonging to one or more relations. Each tuple is represented as a predicate calculus assertions, with predicate names designating specific relations and arguments denoting variables or constants; by convention variable arguments are preceded by a question mark. Returning to our examples from the HST domain, the VISIBILITY state variable of a given fixed target *?T* is defined to take on one of the two possible values at any point in time: *VISIBLE* (*?T*) or *NOT–VISIBLE* (*?T*). The set of possible values for the POINTING STATUS of the optical system of the telescope is given in Figure 5, where the variables *?T*, *?T*1 and *?T*2 designate arbitrary targets.

LOCKED (*HST*, *?T*)
UNLOCKED (*HST*, *?T*)
LOCKING (*HST*, *?T*)
SLEWING (*HST*, *?T*1, *?T*2)
UNLOCKING(*HST*, *?T*)

Figure 5: Possible values of POINTING STATUS

Besides the "structure" of the system, the other fundamental aspect that a model has to cover is the specification of the "laws" that govern the possible behaviors of the system. In the HSTS domain description language, such laws are expressed as constraints on the possible values that the state variables can assume over time; in particular it is possible to specify constraints of simultaneity and sequentiality with respect to specific state values.

Such constraints are organized around each possible value of each state variable as **value descriptors**. A value descriptor can be seen as the specification of patterns of values relative to the variables of the system and extending over time; each value has a single value descriptor. In order for a value v to be present in a behavior B of the system, it must be possible to recognize in B one and only one of the patterns specified in the value descriptor of v.

In more detail, a value descriptor specifies two different types of information. The first is the **intrinsic duration** of the value. An intrinsic duration is a constraint on the amount of time that a value can appear continuously in a behavior of the system; it is represented as a pair of temporal distances $[d, D]$, $d \leq D$, where d and D are respectively the **lower bound** and the **upper bound** on the duration. For example, $[c, c]$, where c is a constant, can be associated to a value with a constant intrinsic duration. The couple $[0, +\infty]$ denotes an indefinite duration; in this case the duration of the value is totally determined by the occurrence of other values that constrain its start and end time.

In general, both d and D may be functions of the arguments of the predicate that represents the associated value. The intrinsic duration of *SLEWING* (*HST*, ?*T*1, ?*T*2), for example, is represented by the pair:

[*slewing–time* (*HST*, ?*T*1, ?*T*2), *slewing–time* (*HST*, ?*T*1, ?*T*2)]

This will return a single duration once the two targets ?*T*1 and ?*T*2 are known.

The second component of a value descriptor specifies how the occurrence of other state values in a behavior constrains the continuous occurrence of the corresponding value. These constraints, referred to as **compatibilities**, are expressed as temporal relations between the start and/or end times of the continuous occurrences of two values. For example, the constraint that a target ?*T* must be visible in order to take a picture of it with viewing instrument ?*I* in operational status ?*S*, is expressed as:

VISIBLE (?*T*) {*contains*, [0, +∞], [0, +∞]} *EXPOSE* (?*I*, ?*S*, ?*T*)

This indicates that, for any ?*I*, ?*S* and ?*T*, if *EXPOSE* (?*I*, ?*S*, ?*T*) appears in the behavior of the system, then the value *VISIBLE* (?*T*) has to appear continuously during an interval of time such that its start precedes the start of *EXPOSE* (?*I*, ?*S*, ?*T*) by an indefinite amount of time and its end follows the end of *EXPOSE* (?*I*, ?*S*, ?*T*) by an indefinite amount of time. Another temporal relation available in the domain description language is {*before*, [*d*, *D*]}, which specifies that the end of the constraining value must precede the start of the constrained value, and the time interval separating the two events is constrained by [*d*, *D*]; The relation {*before*, [0, 0]}, for example, requires the simultaneity of the two events.

The compatibilities defined by a value descriptor are partitioned into **preconditions**, which indicate the state values which may enable the occurrence of the described value, and **postconditions**, which indicate the state values that may be caused by the occurrence of the described value. Both preconditions and postconditions may consist of one or more sets of alternative compatibilities. These structures are represented in the domain description language as AND/XOR trees of compatibility sets. Any set of compatibilities that once satisfied, or true, would satisfy, or make true, the overall compatibility tree, is a sufficient condition for the corresponding value to arise. Figure 6 illustrates the precondition and postcondition compatibility structure of the value *EXPOSE* (*WF*, 4*n*, ?*T*), which corresponds to taking a picture of a given target with the WF in operational status 4*n*.

4.2 Levels of Representation

The domain description language just outlined provides a basis for specifying explicit models of system behavior. With respect to the HST domain, this is fundamental to the generation of executable observing schedules. However, as indicated earlier, the complexity of large scale scheduling problems also requires an ability to reason with simpler approximate models of system dynamics, as global problem analysis and focus of attention cannot be feasibly accomplished relative to the full complexity of the detailed model. To accommodate such decision-making, the HSTS modeling framework provides primitives for augmenting the specified detailed model with an abstract model. Values in the abstract model provide high level descriptions of system activities; in particular, representing the specific scheduling goals that constitute the current problem. In the HST domain, abstract values refer to specific "observations" (e.g., *OBSERVE*(?*P*, ?*I*, ?*S*, ?*T*, ...), where ?*P* designates an observation program, ?*I* designates a viewing instrument, ?*S* designates the required operating state of ?*I*, and ?*T* designates a target), which decompose into appropriate value configurations at the detailed level (e.g., an *EXPOSE*(?*I*, ?*S*, ?*T*) *READOUT*(?*C*, ?*I*) sequence, with ?*C* designating a communication device).

The primary distinction with respect to the representation of abstract values is the implicit treatment of state-dependent "setup" constraints. These constraints correspond precisely to the compatibilities associated with those system state variables over which the scheduler has control (e.g., the OPERATING STATUS of specific instruments, the POINTING STATUS of the telescope's optical system, etc.). At the abstract level, such constraints are instead modeled implicitly as adjustments to the intrinsic durations of values (i.e. as was done in the OPIS scheduler), and are represented as functions in the abstract value's descriptor. For example, the complex of detailed activities necessary to position HST for viewing can be modeled as a context-dependent duration, in this case a function of the proximity of the previous and current targets, and the time required for guide star acquisition. Similarly, instrument reconfiguration time can be estimated via a configuration distance matrix. Such models obviously ignore much of

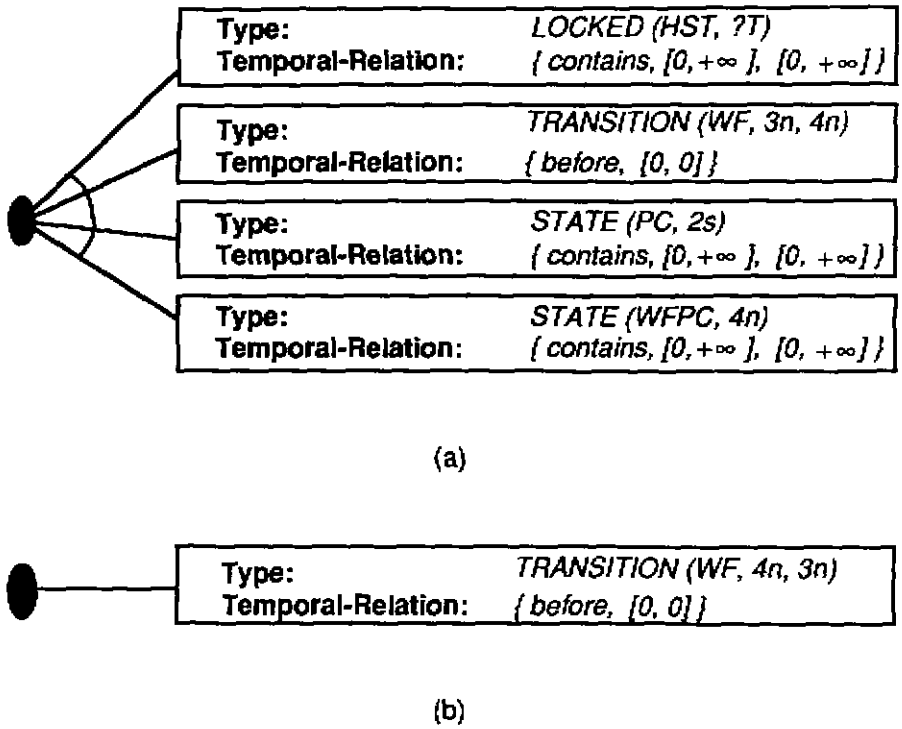

Figure 6: Compatibilities for *EXPOSE* (*WF*, 4*n*, ?*T*): (a) Precondition tree, (b) Postcondition tree

the details of the actual system dynamics (e.g., distinct setup activities are assumed not to interact). At the same time, they enable a level of representation of the scheduling problem wherein all activities to be placed on the time line can be assumed to be known in advance, providing a framework for global analysis of current solution constraints and opportunistic subproblem formulation.

Specification of the mapping between abstract and detailed values in the model is accomplished through the association of **refinement descriptors** with abstract values. The refinement descriptor of a given abstract value specifies a corresponding set of detailed state values, and a set of temporal relations constraining their occurrence over time. For example, the refinement descriptor for *OBSERVE*(?*P*, ?*I*, ?*S*, ?*T*, ...) listed in Figure 7 specifies the occurrence of two values, *EXPOSE*(?*I*, ?*S*, ?*T*) and *READOUT*(1*Mbyte-link*, ?*I*) on the OPERATING STATUS state variables of the instrument and communication device respectively, such that the end time of the *EXPOSE* and the start time of the *READOUT* are within the duration of a single orbit. Furthermore, the start and end time of the *EXPOSE* coincide respectively with the start and end time of the abstract *OBSERVE*. These latter constraints provide a basis for downward imposition of time constraints, as well as upward propagation of detailed scheduling decisions.

5 Representing System Behaviors

Given a description of the system to be managed, a second broad architectural issue concerns the manner in which specific system behaviors (i.e. schedules) constructed by the scheduler are represented. Within HSTS, this is accomplished through the use of an underlying temporal data base. The HSTS temporal data base has the following general characteristics:

1. **It stores behaviors of a system**: In fact, the data base satisfies a stronger requirement, since its *only*

```
((refine-desc-1
    VALUE:
        OBSERVE(?P, ?I, ?S, ?T, ...)
    SUBVALUES:
        V1 (OPERATING-STATUS EXPOSE(?I, ?S, ?T) )
        V2 (OPERATING-STATUS READOUT(?I,1Mbyte-link) )
    ORDERING-CONSTRAINTS:
        V1   [ before, [0, *orbit-duration* ] ]   V2
    SAME-START: V1
    SAME-END: V1   ))
```

Figure 7: Refinement descriptor for *OBSERVE(?P, ?I, ?S, ?T, ...)*

legal states are those that satisfy the constraints on the dynamics of a pre-specified system model;

2. **It is a constraint network:** Behaviors are represented implicitly by a series of constraints that have been either externally imposed (e.g. by requirements of the problem) or directly extracted from the system model. This provides a representation of a partial schedule as a state of the database where several aspects of the system behavior currently under construction are left underspecified.
3. **It supports opportunistic scheduling:** At any point during scheduling, several parts of the data base might require refinement (through additional constraint posting) to produce a complete specification of the final schedule. The database leaves complete freedom as to the order in which these refinements are made.

The HSTS temporal data base extends in several ways the philosophy of the time map formalism developed in [3]. Perhaps the most fundamental departure in our approach is the tight connection that is established between the state of the data base and the model of a system This association provides a strong basis for enforcing database consistency.

In this section, we first introduce the main representational primitives of the HSTS temporal data base, and we then discuss the temporal data base visualization facilities provided by the HSTS scheduler.

5.1 The HSTS Temporal Data Base

As in [3], the HSTS temporal data base explicitly represents the start and the end of the occurrence of a particular state variable value as nodes, or **time points**, in a directed graph. Each edge of the graph is a temporal distance constraint between the two connected points; each edge has an associated pair $[d, D]$, with $D \geq d \geq 0$, designating current lower and upper bounds on the distance.

The occurrence of a value is represented as a triple $<st, et, type>$, called **token**, where *st* and *et*, respectively the token's start and end, are two time points and *type* is a set of values.

Depending on the cardinality of the type, a token can have two different interpretations:

- **value token**, if the type is a single value. A value token indicates the actual occurrence of that value in the set of behaviors currently represented by the temporal data base.
- **constraint token**, if the type is a set of values of cardinality greater than one. This kind of token denotes a sequence of values of indefinite length (possibly empty). Each value of the sequence is constrained to belong to the *type* of the token, while *st* and *et* represent respectively the start of the first value in the sequence and the end of the last token in the sequence.

Tokens are organized into sequences that reflect the values assumed by each system state variable over time in the behavior currently represented in the data base. Note that start and end points of value occurrences (tokens) in the

data base are not necessarily uniquely determined; instead, time bounds on their occurrence can be obtained by propagating the distance constraints of each edge across the entire time points graph.

A tight connection to the system model is maintained by allowing a token to appear in the time map only if the occurrence of its associated value is consistent with the system's dynamics and any other constraints imposed by the scheduling problem (e.g., a constraint that one activity must precede another). Consistency with the system's dynamics entails the following two structural properties:

1. The distance between the start and the end point of each value token must be consistent with the intrinsic duration of the value; this implies the existence of a distance constraint between the token's start and end points which reflects the lower and upper bounds on duration defined in the corresponding value descriptor.

2. For each value token, there must be a set of satisfied compatibilities among the alternatives defined in the value descriptor. This requires the existence of a constraining token for each compatibility of the set, whose value matches the value pattern in the compatibility and whose start and end points can be connected to the extremes of the constrained token according to the temporal relation in the compatibility. The incrementality of the HSTS temporal data base lies in the fact that compatibilities can be achieved incrementally. More precisely, each token maintains an instantiation of the precondition and postcondition trees specified in the value descriptor of its type. When a precondition (postcondition) is satisfied, this fact is propagated through the precondition (postcondition) tree of the appropriate token. If further preconditions (postconditions) remain to be achieved to justify the occurrence of a given token, the corresponding precondition (postcondition) tree is said to be still *open*.

5.2 Visualization of the Temporal Data Base

The HSTS scheduler provides a powerful graphical interface to access the information stored in the temporal data base.

A sequence of tokens associated to a state variable can be visualized in two different ways.

For some state variables, the characteristics of the dynamics of the system or of the scheduling problem at hand uniquely determine the times of occurrence of the transitions among values; these state variables are therefore outside of the control of the scheduling system. In the HST domain, for example, this is the case with the visibility state variables of viewing targets. Within the interface, the sequences of values on all state variables of this kind are visualized within a two window pane (Figure 8). The left window contains the names of the state variables, which are mouse sensitive in order to provide additional information on their attributes. The succession of their values is represented as a sequence of rectangles on an horizontal time line. Different shadings are used to represent different values; for example, in Figure 8 each box corresponds to a state of visibility of the target, with black designating periods of occultation and white designating periods of visibility. Each box is mouse sensitive to provide textual information on their value and start and end times.

For the state variables that can be manipulated by the scheduler, it is necessary to visualize the time bounds on the start and end times of each token. This is obtained by associating a three window pane to each state variable; Figure 9, for example, displays four of the state variables in the HST model, three representing the OPERATING STATUS of the WFPC and its two detectors, and the other representing the telescope optical system's POINTING STATUS. The sequence of tokens assumed by the state variable is stacked vertically, with the earliest tokens in the sequence at the top. The left of each pane contains the names of the tokens; clicking on each of them will provide information regarding the status of the precondition and postcondition trees. The middle window displays the type of each token; each of them can be enlarged by selecting it with the mouse. The right window contains a graphical representation for the time bounds of each token, i.e., its earliest start time, earliest end time, latest start time and latest end time. Each point is represented by an appropriately shaded box. Clicking on the displayed time bound will visualize the numeric times associated with each extreme point.

A variety of graphical tools are available to query additional information. Facilities for focusing the visualization on different parts of the time map are also available; they include 2D panning and zooming, with the possibility of

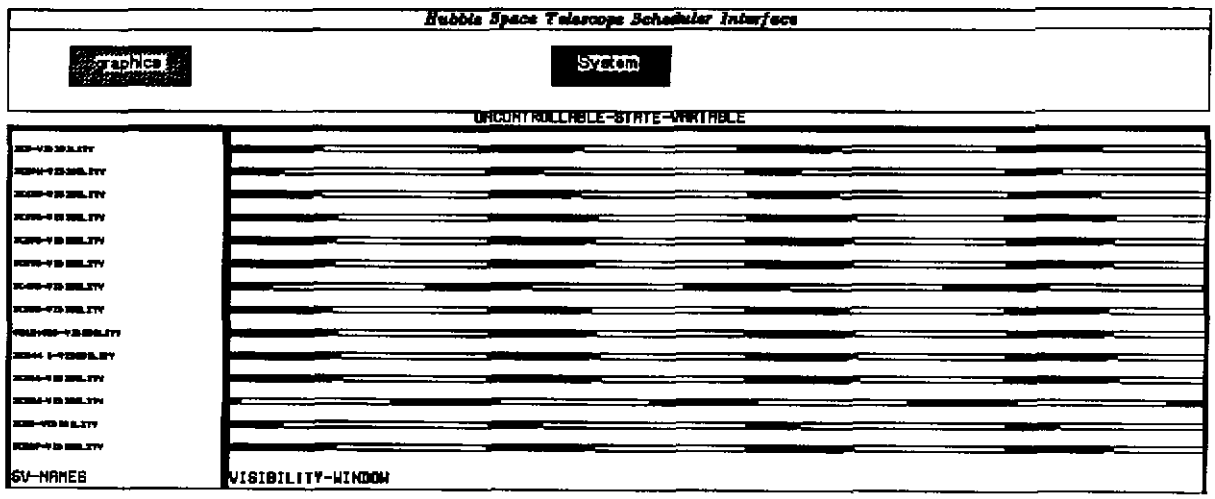

Figure 8: Visibility of viewing targets

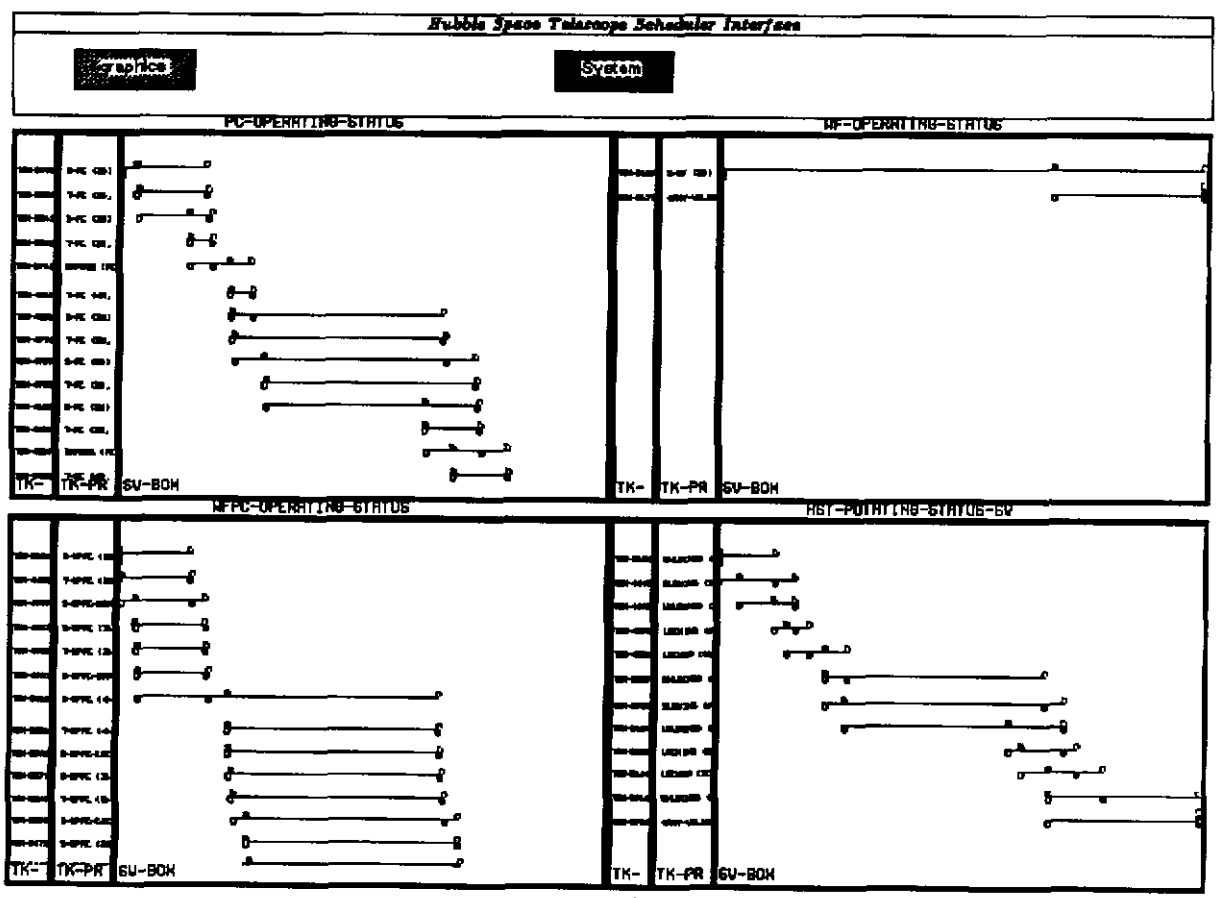

Figure 9: Controllable state variables

specifying the state variables to be displayed and the associate token subsequences. It is also possible to select among layouts of windows of different sizes.

6 Integrating opportunistic scheduling and planning

The third component of the HSTS scheduling architecture is the problem solving methodology. This methodology has the following distinguishing characteristics:

1. **It advocates opportunistic problem decomposition and solution development**: Schedules are constructed incrementally through repeated formulation and solution of specific scheduling subproblems. The framework provides complete flexibility to assemble subproblems according to characteristics of the current partial schedule.

2. **It utilizes multiple levels of representation**: An abstract model of overall scheduling problem, suitable for global analysis of current solution constraints, provides a basis for problem decomposition and subproblem structuring. The actual schedule is constructed according to the detailed model of the system.

3. **It integrates planning and scheduling**: the construction of the schedule at the detailed level is carried out by a temporal planner, which enables expansion of requisite setup operations at problem solving time.

The HSTS scheduling methodology, therefore, implements opportunistic, constraint-directed scheduling [14] with respect to a much more generally applicable set of modeling assumptions.

During the schedule generation process, the current solution state is reflected by the contents of two distinct temporal data bases:

- **Scheduling Problem Time Map (SPTM)**: This is a representation of the scheduling goals and temporal scheduling constraints that constitute the current scheduling problem.

- **Current Solution Time Map (CSTM)**: This represents the set of behaviors of the system that are consistent with the scheduling goals selected so far.

In the context of the HST scheduling problem, the SPTM initially contains the pool of observations to be scheduled together with the temporal constraints implied by proposer specified requirements (e.g., their relationships to other observations in the same observing program). The CSTM initially reflects external events that are deterministically known (e.g., periods of target visibility) and assumptions regarding the initial state of other system state variables (e.g., the initial state of each of the instruments, the initial pointing status, etc.)

Generation of a schedule proceeds incrementally, by repeatedly selecting and constraining one or more as yet unachieved scheduling goals from the SPTM (or alternatively selecting one or more previously achieved goals to be retracted), and constructing a system behavior that extends the CSTM to include achievement of the newly posted goals (or desired retraction of previously achieved goals). This cycle is repeated until either all of the scheduling goals have been achieved or it has been determined that it is not possible to achieve those that remain. In the following subsections, we describe the HSTS scheduling cycle in more detail.

6.1 Sub-Problem Formulation

The first step of the iterative scheduling process is **Sub-Problem Formulation**, which is concerned with specifying how to further extend (or revise) the detailed partial solution developed thus far in the CSTM. Subproblem formulation operates with respect to the high level representation of the overall problem provided by the SPTM. At this level of representation, as indicated earlier, state-dependent constraints on the achievement of scheduling goals are implicitly modeled as additional temporal constraints. Thus, from the standpoint of subproblem formulation, the problem is strictly one of optimizing the allocation of resource capacity to a known set of goal activities subject to current scheduling constraints.

Sub-problem formulation is organized as a two step process of problem analysis and heuristic commitment. During problem analysis, the constraints associated with currently unachieved scheduling goals are analyzed to produce a coarse characterization of the structure of the current solution space. The objective here is to estimate the level of resource contention that exists over different portions of the scheduling horizon, given the diversity and current temporal flexibility of outstanding resource requests. This characterization is used to differentiate among different problem decomposition and subproblem structuring alternatives, and consistently focus the scheduler on the most appropriate subproblem.

Commitment to a specific subproblem is accomplished via the application of problem structuring heuristics. These heuristics encode knowledge which relates aspects of the current solution state to needs, opportunities and strategies for optimization relative to various scheduling objectives and preferences. Minimally, the result of heuristic commitment is the selection of an existing sub-map in the current SPTM; for example the selection of a particular *OBSERVE* to achieve together with the time bounds currently implied by its relationships with other observations. More generally, however, heuristic commitment entails the imposition of additional constraints. Problem structuring heuristics may be based on problem specific optimization objectives (e.g., establishing sequencing constraints between selected observations so as to minimize slewing and/or instrument reconfiguration time), or reflect more general resource allocation strategies (e.g., restricting a selected observation's time window to an expected period of low resource contention). Heuristic commitment may also involve specification of preferences over the scheduling alternatives that exist within formulated subproblem constraints (e.g., "schedule the observation as early as possible within the imposed time constraints").

Given the abstract model upon which subproblem formulation is based, it may in fact prove appropriate to revise aspects of the current solution as scheduling proceeds and solution constraints become clearer. In such cases, heuristic commitment actually reflects decommitment, and the formulated subproblem specifies one or more previously achieved goals to be retracted. Note, however, that such decisions must be heuristically motivated [14], as it will generally not be possible to satisfy all scheduling goals and exhaustive consideration of the overall solution space is not a feasible worst case scenario.

6.2 Sub-Problem Integration

The **Sub-Problem Integration** step provides the interface between sub-problem formulation and detailed planning. There are two issues here.

The first issue concerns translation of the subproblem, formulated in terms of the more abstract model, into a representation of goals at the detailed level. In the case of new goals to be achieved, this is accomplished by interpreting the refinement descriptor associated with each top level scheduling goal designated in the formulated subproblem. In the case of previously achieved goals to be retracted, the connection has already been established.

The second issue concerns the integration of the resulting set of new planning goals into the current CSTM (or the removal of any previously achieved goals selected for retraction). Generally speaking, this task consists of specifying the portion of the current CSTM over which detailed planning will be permitted to range, and requires consideration of the problem constraints currently reflected by the CSTM in relation to those additionally imposed by the current sub-problem. Considering an example from the HST domain, suppose that the current scheduling subproblem consists of taking an exposure of target $T3$ between two previously scheduled observations, the first on target $T1$ and the second on target $T2$. Prior to integration of the new goal, the sequence of values on the telescope's pointing status in the CSTM will contain the subsequence required to transition from being locked on target $T1$ to being locked on target $T2$. To accommodate the new observation, it is necessary to remove this subsequence and substitute it with constraint tokens that can be refined to obtain the two subsequences necessary to move first from target $T1$ to target $T3$ and then from target $T3$ to target $T2$. In general, the integration of a sub-problem into the CSTM requires determination of the tokens and/or causal connections in the CSTM that can be retained and those must be retracted or underconstrained to provide an appropriate degree of flexibility for detailed planning.

6.3 Planning

It is now necessary to establish that the underspecified CSTM into which the new subproblem has been integrated actually contains some consistent behavior of the system, i.e., a partial schedule. The **Planning** phase of the cycle carries out the search for such behaviors.

The planner refines the underconstrained CSTM through an iterative process. At any iteration, there may be several open compatibilities (i.e., preconditions and/or postconditions) in the CSTM. More precisely, a compatibility is open if it belongs to an open precondition (postcondition) tree and does not have a corresponding implementation in the CSTM.

The planning cycle proceeds in two steps. The first step is to select a specific open compatibility to achieve, along with the token this compatibility constrains (referred to as the constrained token). This is equivalent to choosing a constraint on the behavior of the system that will eventually constitute the schedule. The next step is to determine if the current CSTM is consistent with that constraint. This is accomplished by posting the constraint in the CSTM, an operation consisting of:

1. selecting, from the time map, a constraining token that matches the value required by the compatibility, i.e., a token whose type has a non empty intersection with that required by the compatibility, and
2. connecting the constraining token to the constrained token with the temporal relation indicated in the compatibility.

When a constraint is posted, a temporal constraint propagation process spreads the consequences to the rest of the CSTM.

During execution of this planning cycle, choices may arise at two points: when selecting the compatibility and when selecting the constraining token. These choice points originate a search process. The application of appropriate heuristics permits to select the most promising alternative; the architecture allows the introduction of arbitrary general purpose or domain specific heuristics.

If the introduction of a compatibility constraint in the current CSTM leads to detection of an inconsistency during constraint propagation, this implies that the set of behaviors implicitly represented by the CSTM is empty. An inconsistency requires the planner to backtrack in the search process, or to fail if no alternative paths are still open in the search tree. The planning cycle is repeated until either the CSTM contains no remaining open compatibilities, or all possibilities have been exhausted.

7 Current Directions

An initial version of the above scheduling architecture has been implemented and tested with respect to a simplified but representative model of the HST operating environment. The current HSTS prototype is capable of flexibly and efficiently constructing system behaviors (schedules) consistent with formulated subproblems. It has already demonstrated an ability to handle most types of constraints considered by the current SOGS scheduling system, and, in some cases, provides functionality that is beyond SOGS's current capabilities. The scheduling cycle is fully automated, although the system currently operates with a fairly unsophisticated subproblem formulation component (the development of a more sophisticated subproblem formulator is currently underway - see below). The system may also be operated interactively, in which case the user plays the role of the subproblem formulator and focuses the temporal planner.

Our research to date has focused on specification of the principal architectural components of the scheduler. Particular emphasis has been placed on definition of a domain description language suitable for representing the complex constraints that govern space telescope operations, and development of a temporal planner that effectively exploits this representation to generate executable observation schedules. Current research is aimed at the refinement of current system components, and the development and incorporation of more powerful scheduling heuristics.

One area of work concerns the development of more sophisticated constraint-directed strategies for sub-problem formulation. The current sub-problem formulation component operates with respect to a fairly simple "setup minimization" heuristic which assumes only limited knowledge of the structure of solution space and accounts for only a fraction of HST scheduling objectives and preferences. We are investigating the use of previously developed preference representations [7, 10] as a basis for more general characterizations of current solution constraints, and the development of problem decomposition heuristics that operate with respect to these representations. On a related note, we currently have little insight into performance tradeoffs concerning the relative amount of problem solving effort expended at the subproblem formulation and detailed planning levels, and the relative amount of problem solving responsibility that should be ascribed to each component. Experimental analysis of this tradeoff is required.

An important issue relating to sub-problem integration that requires further investigation concerns determination of the portion of the CSTM that detailed planning should range over in solving the current subproblem (as this has a direct bearing on the amount of search required). The current implementation exploits previously achieved "goal" tokens in the CSTM as a means for establishing the planner' initial state. We are currently investigating more context-sensitive approaches that take better advantage of aspects of the CSTM relevant to the solution of the current subproblem.

With respect to detailed planning, we have found the use of heuristics to limit the number of alternatives expanded and minimize backtracking to be extremely important, and we are currently pursuing the further development of such heuristics. Our work so far seems to suggest that their nature depends in part on the general structure of the domain (as described in the system dynamics specification) in which case they should be transferable to other scheduling domains, and in part on the specifics of the particular domain. Furthermore, while the planner presently builds a plan by posting one compatibility at a time, we believe much power can be gained by storing sub-plans (i.e., appropriate networks of tokens), and assembling them in order to build the new CSTM. This is also an area of current investigation.

We believe we have made considerable progress toward the development of a better solution to the HST observation scheduling problem. In conjunction with the work described above, our current plans for the coming year include a scaling up of the current model of the HST domain for purposes of conducting a comparative performance analysis with the existing SOGS scheduling system relative to actual sets of observation proposals. This head to head competition will provide a realistic assessment of the power of our approach.

References

[1] Allen, J. and Koomen, J.A.
Planning Using a Temporal World Model.
In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741-747. 1983.

[2] Baker, K.R.
Introduction to Sequencing and Scheduling.
John Wiley and Sons, New York, 1974.

[3] Dean, T.L. and McDermott, D.V.
Temporal Data Base Management.
Artificial Intelligence 32:1-55, 1987.

[4] Dean, T. and Firby, R.J. and Miller, D.
Hierarchical Planning Involving Deadlines, Travel Time, and Resources.
Computational Intelligence 4:381-398, 1988.

[5] Fikes, R.E., Hart, P.E. and Nilsson, N.J.
Learning and Executing Generalized Robot Plans.
Artificial Intelligence 3:251-288, 1972.

[6] Forbus, K.D.
Introducing Actions into Qualitative Simulation.
In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1273-1278.
Morgan Kaufmann, 1989.

[7] Fox, M.S. and Smith, S.F.
ISIS: A Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, 1984.

[8] Hogge, J.C.
The Compilation of Planning Operators from Qualitative Process Theory Models.
Technical Report UIUCDCS-R-87-1368, University of Illinois at Urbana-Champaign Department of Computer Science, 1987.

[9] Johnston, M.D.
Knowledge-based Telescope Scheduling.
Lecture Notes in Physics 329.
Springer-Verlag, 1989.

[10] Muscettola, N. and S.F. Smith.
A Probabilistic Framework for Resource-Constrained Multi-Agent Planning.
In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1063-1066.
Morgan Kaufmann, 1987.

[11] Ow, P.S. and Smith, S.F.
Viewing Scheduling as an Opportunistic Problem Solving Process.
In R.G. Jeroslow (editor), *Annals of Operations Research 12*. Baltzer Scientific Publishing Co., 1988.

[12] Panwalker S.S. and Iskander W.
A Survey of Scheduling Rules.
Operations Research 25:45-61, 1977.

[13] Sathi, A. and Fox, M.S. and Greenberg, M.
Representation of Activity Knowledge for Project Management.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7:531-552, 1985.

[14] Smith, S.F.
A Constraint-Based Framework for Reactive Management of Factory Schedules.
In M.D. Oliff (editor), *Intelligent Manufacturing*. Benjamin Cummings Publishers, 1987.

[15] STScI.
Proposal Instructions for the Hubble Space Telescope.
Technical Report, Space Telescope Science Institute, 1986.

[16] Vere, S.
Planning in Time: Windows and Durations for Activities and Goals.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5, 1983.

[17] Waldrop, M.
Will the Hubble Space Telescope Compute ?
Science 243:1437-1439, March, 1989.

[18] Wilkins, D.E.
Practical Planning.
Morgan Kaufmann, 1988.