In L. Castillo, D. Borrajo, M. A. Salido & A. Oddi (Eds), "Planning, Scheduling, and Constraint Satisfaction: From Theory to Practice", IOS Press, 2005

Using a scheduling domain ontology to compute *user-oriented* explanations

Stephen F. Smith [1], Gabriella Cortellessa [2], David W. Hildum [1], Christian M. Ohler [1]

[1] The Robotics Institute Carnegie Mellon University
5000 Forbes Avenue - Pittsburgh PA 15213
{sfs,hildum,ohler}@cs.cmu.edu

[2] ISTC-CNR National Research Council of Italy
Viale Marx 15, I-00137 Rome, Italy
corte@istc.cnr.it

Abstract. One broad source of difficulty in transitioning automated planning and scheduling theories and algorithms into practical application systems is the need to integrate with user decision-making processes. Both user acceptance of system decisions in successful problem solving episodes and effective user involvement in circumstances where the system reaches a problem solving impasse require that the system be comprehensible, and this requirement, in turn, implies that the system be capable of explaining its decisions in user-understandable terms. Unfortunately, most current work in explanation instead forces users to understand the system's underlying search models. In this paper, we consider this problem of bridging the gap between user and system models in the context of a constraint-based scheduling system. Previous work has proposed the use of a scheduling domain ontology as a basis for translating user problem specifications into internal system models. Here we propose the complementary use of a domain ontology as a means of computing user-oriented explanations of system decisions. We focus specifically on the problem of explaining temporal constraint conflicts that may arise in the course of either solving a given scheduling problem or integrating new state information with a previously computed schedule. The central idea is to use domain ontology knowledge (1) to isolate those constraints that are meaningful to and can be manipulated by the user, and (2) to identify various constraint relaxation options that the user might consider to resolve the conflict at hand. Using the COMIREM planner/scheduler [10] as a reference model, we show examples of how this approach can generate effective user-level explanations of constraint conflicts from low-level descriptions of detected cycles in a temporal constraint graph. Along the way, we discuss the broader implications from the standpoint of mixed-initiative scheduling system design.

1 Introduction

One major obstacle to the practical application of new planning and scheduling theories and algorithms is the gap between these automated models and human planning and scheduling processes. Although there are certainly some exceptions, total automation of decision-making is not an appropriate goal in most practical domains. More typically, it is the case

that experienced users and automated planning/scheduling technologies bring complementary problem-solving strengths to the table, and the goal is to synergistically blend these combined strengths. Often the scale, complexity or general ill-structuredness of practical domains overwhelms the solving capabilities of automated planning and scheduling technologies, and some sort of problem decomposition and reduction is required to achieve solver tractability. Likewise, human planners often have deep knowledge about a given domain which can provide useful strategic guidance, but they are hampered by the complexity of grinding out detailed plans/schedules. In such cases, successful technology application requires effective integration of user and system decision-making. However, this is complicated by the fact that users do not reason about plans and schedules at the level of search spaces and temporal constraint graphs; the system must somehow bridge the gap between user and system models and representations.

Research into the design of mixed-initiative planning and scheduling systems is concerned fundamentally with solving this general problem of interfacing a user with the automated system and placing a user into the problem-solving loop. One basic issue concerns mechanisms for specifying planning and scheduling problems to the automated system, i.e., for translating user specifications of domain constraints and objectives into internal system models. However, to effectively close the loop and actually involve the user in the planning/scheduling process, a second, somewhat inverse issue must be addressed: that of explaining system results in user-comprehensible terms. Human planners tend to be skeptical of automated systems in general, and an ability on the system's part to provide user-level rationalization of generated plans and schedules can promote user acceptance. Perhaps even more important to effective mixed-initiative planning and scheduling behavior is an ability to provide explanatory support in the event of system failure. As suggested above, the experienced human user often possesses deeper knowledge of strategic decision options that is outside of system models. She could manage unforeseen situations and retract specific constraints to recover from system failure, if properly informed of the specifics of the impasse that the automated system has encountered. Most current systems provide no guidance in such cases, and force the user to diagnose the problem at the level of the system's internal model.

In this paper, we consider this latter issue of explaining situations of solution failure in user-oriented terms. We focus specifically on the problem of explaining temporal constraint conflicts encountered by a constraint-based scheduling system. In contrast to previous work on explanation in Constraint Satisfaction Problem Solving (CSP) domains which operates at the level of individual decision variables and constraints, we propose use of a scheduling domain ontology to compute higher-level explanations. The reference system for our work is COMIREM, a web-based mixed-initiative problem solver [10]. We demonstrate how COMIREM's ontology can be used to isolate user-relevant constraints in a given conflict situation and provide guidance in exploring resolution options. Before describing our approach, we first review relevant aspects of the COMIREM ontology and problem solver and then briefly summarize prior work in explanation.

2 COMIREM: A Mixed-Initiative Planner/Scheduler

COMIREM is a web-based system devoted to the problem of interactive and dynamic allocation of resources to activities over specific time intervals. The system is based on a CSP paradigm and promotes a problem solving process that combines the actions of the auto-

mated solver and the human planner. COMIREM is composed of two main modules, named respectively *Automated Solver* and *Interaction Module*. The first is devoted to modeling domain entities through a CSP representation and provides the algorithm to solve the problem. It models the domain through the OZONE *scheduling domain ontology* which allows a user-interpretable description of an application domain to be mapped to application system functionalities [9]. Domain objects and features are represented in terms of entities very close to the human representation level of abstraction, and can be easily presented to the user through the module devoted to the user-system interaction. The Interaction Module directly interacts with the user, and allows her to take part in the process of finding a solution via advanced interactive facilities. It represents the communication channel between the user and the automated solver and a means to exploit various features of the automated system. Major aspects of the COMIREM architecture are summarized in the subsections below.

2.1 The Underlying OZONE *Scheduling Ontology*

The main primitives for constructing domain models in COMIREM derive from the OZONE Ontology [9]. Considered originally as a vehicle for high-level specification of scheduling problems, entities in the ontology can be broadly subdivided into *Activities, Resources and Constraints*:

- Activities. An activity represents a process that can be executed over a certain time interval. Execution of an activity requires resources. In COMIREM activities can be organized hierarchically into multi-level activity networks.
- Resources. A resource is an entity that supports or enables the execution of activities. Resources are generally in finite supply and their availability constrains *when* and *how* activities execute. An important objective of scheduling is to make efficient use of resources that support multiple competing activities. Two types of resources are modeled in COMIREM: single and multi-capacity resources.
- Constraints. Generally speaking, a constraint restricts the set of values that can be assigned to a variable. COMIREM provides the means to model three types of constraints: (a) *temporal constraints*, which constrain the start times and/or end times of one or more activities; (b) *resource constraints*: which require sequentialization of activities competing for the same resources; (c) *causal constraints*: which define what conditions must be met before an activity can be executed.

2.2 The Automated Solver

COMIREM utilizes an opportunistic constraint-posting scheduling procedure to allocate resources to activities over time, relying on a planning sub-procedure as necessary to determine appropriate resource reconfiguration actions. COMIREM takes as input an initial *plan sketch* that specifies, at some level of abstraction, the actions necessary to accomplish certain end goals for a given scenario. For example, to rescue people from an embassy in a foreign capital, an initial plan describing necessary actions such as securing the local airport, transporting rescuers to the airport, etc. is provided by the human planner, together with any associated causal dependences and temporal constraints. Starting from this initial plan, the scheduling

procedure tries to feasibly allocate resources to input activities. In some cases, feasible assignment entails the generation of resource support plans (e.g., for "positioning" an aircraft to the location where it is needed), such planning subproblems are solved dynamically as specific resource assignments are considered. If successful, the procedure returns a detailed plan, where each activity is assigned the resources it requires and is designated to execute in a specified finite time interval. Due to its interactive nature, the system can exploit human-planner knowledge and decision making, and in fact promotes a mixed-initiative problem solving process. Through the Interaction Module it is possible to either generate a solution automatically or iteratively build a solution through a *step by step* mixed-initiative procedure that interleaves human choices with system calculation of consequences.

2.3 Mixed-Initiative Problem Solving in COMIREM

As just mentioned, COMIREM provides a user with two options: (a) automatically generate a solution to the problem; (b) iteratively build a solution. In the first case a user decides to completely entrust the system with the task of finding a solution. In the second case, the system provides constraint checking and option generation support for user decisions. When an initial plan is loaded, COMIREM performs a temporal feasibility check, and creates new activities as necessary to carry out entailed supporting actions. A visual representation of the problem and its main features is provided to the user through a graphical spreadsheet-style model. For each unassigned activity in the plan, COMIREM maintains the current set of feasible allocation options and presents them to the user through the Interaction Module. At any time and in any order, the user can manually specify resource assignments for particular activities. Whenever a user allocates a resource to a given activity, the impact of the user's choice is reflected in the plan and the system updates the set of possible options available for other pending decisions. At any point in the process, the automatic algorithm can be invoked to make all remaining assignments and produce a complete solution. Both activity and resource attributes can also be edited to change the constraints and requirements of the problem. The system provides a general ability to *undo* any user action (or sequence of actions) previously taken, providing a flexible framework for what-if analysis.

The interaction module in COMIREM can be seen as an intelligent blackboard that allows a user to reason incrementally on a solution, providing both (1) visualization functionalities to inspect problem and domain features and (2) interactive services to involve a user in the problem solving. The ambitious idea behind COMIREM is to capture different skills that a user and an automated system can apply to the resolution process. Typically an automated algorithm is better suited to conducting repetitive search steps that are not possible for a human user, while a user typically has more specific knowledge about the target domain that is difficult to formalize in general terms to be used by an algorithm. The development of principles for mixed-initiative interaction [2, 3] represents a key to the development of more powerful problem solving environments.

2.4 Explanation as a means to foster mixed-initiative problem solving

Among the numerous aspects involved in the development of mixed-initiative systems, one important requirement is the need to maintain continuous communication between the user and the automated problem solver. Current interactive systems are usually lacking with respect to providing such a continuity. System failures that may be encountered in finding a

solution typify this sort of deficiency. Typically, when a planning/scheduling system fails during the problem solving, or when the solution is found to be inconsistent due to introduction of new world state information, the user is not properly supported and left alone to determine the reasons for the break (e.g., no solution exists, the particular algorithm did not find a solution, there was a bug in the solver etc.).

This leads to an interruption in the problem solving process that could be otherwise used as an event to determine a shift in the *initiative*. A conflicted situation might, for example, be resolved by a user who has a deeper knowledge of the domain and/or agrees to slightly change the problem in order to obtain a solution. Obviously in transferring the initiative to the user, a system should inform her about the reasons of the failure or problem encountered to ease and promote her participation. On this subject the concept of *explanation* is becoming of increasing interest in many different research communities. Our interest in this paper focuses on the use of explanation within COMIREM as a means to explain and inform a user about the reasons underlying system search failures and solution inconsistencies.

3 Explanation and CSPs

As already mentioned current work on explanation in CSP forces users to understand the system's underlying search model and reasoning in terms of variables and constraints that are usually not comprehensible for the final users. The CSP paradigm is a powerful means for representing and solving problems but it is far from users' models. In order to provide useful information a translation from the low level technicality to the high level human model is necessary. In classical constraint programming, an explanation is a set of constraints justifying propagation events generated by the solver (e.g., value removal from the domain of a certain variable, bound update, contradiction). In [6, 5] two kinds of explanation are introduced: (a) *contradiction explanation* and (b) *eliminating explanation*. The former is a subset of the current problem constraints, which if left alone, leads to a contradiction. It is composed of two parts, a subset of the original constraints, and a subset of the decision constraints introduced so far in the search. The latter, is an implication justifying the removal of a value from the domain of a variable. A very similar approach to this problem is the notion of *justification* introduced by Bessier [1]. A justification is an additional piece of information that is stored each time a value is deleted from a variable's domain. One general approach to computing explanations, then, is to make explicit the knowledge that the solver has while, for example, removing a value from a domain or dealing with an inconsistency. In this way, each time an event is generated (e.g., a value removal), the corresponding explanation is computed within the propagation code of the constraints and the *trace* of the solver reasoning is used as an explanation. Computation of explanations in this manner assumes that the basic CSP model is understandable, and neglects the issue of finding effective ways to present them to a naive user. In [11] the explanation problem is investigated and the need of designing effective ways to organize and present it to the user is highlighted. A crucial aspect in the presentation problem is one of finding effective ways to structure and present the information in order to enhance a user's ability to understand and possibly solve an occurred problem. In [7] a set of tools for providing user-friendly explanations in an explanation based constraint programming system is introduced. The basic hypothesis this work relies on is that all aspects of a constraint-based application can be represented in a hierarchical way. The implicit hierarchy that appears when encoding the problem is used to group constraints into "user-friendly boxes" which are used to add structured information while posting the constraints. In partic-

ular a textual representation of the set of constraints is introduced in the system which is used as a user-comprehensible explanation when a conflict arises.

4 Using the OZONE ontology to compute *user-oriented* explanations

In this section, we outline an approach to translating system-level explanations into a more user-comprehensible and user-actionable form via the use of an underlying domain ontology. Our previous work has argued the use of a scheduling domain ontology to facilitate user construction of an executable system model in a given application domain [9]. Here we consider its complementary use in driving the generation of user-oriented explanations. We focus specifically on the problem of explaining temporal constraint conflicts within COMIREM. Our idea is to use the COMIREM domain ontology knowledge (1) to compute the set of constraints that form the explanation, (2) to identify various constraint relaxation options that the user might consider to resolve the conflict, and (3) to provide content for generating user-understandable explanations of conflicts and possible resolving actions. Figure 1 illustrates the layers of a domain model in COMIREM. The first layer in the picture models the

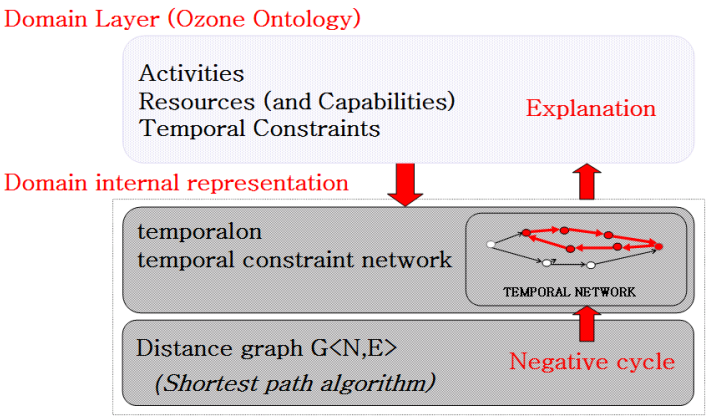


Figure 1: Domain Modeling in COMIREM

constraints of the target domain in terms of *activities*, *resources* and *constraints* using the scheduling ontology described in Section 2.1. This level of description provides an abstraction of the low level CSP representation used internally, that can be reasonably considered to be closer to the human model of the domain.

The second layer shows how this translation is implemented within COMIREM. The problem is represented internally as a Simple Temporal Problem (STP) constraint network [4] and the core scheduling procedure relies on an incremental STP constraint network solver. Planning/scheduling decisions generally correspond to the introduction of new constraints into the network (e.g., a sequencing constraint between two activities that require the same resource) or the adjustment of one or more existing constraints (e.g., refinement of an activity's duration, modification of an anchor). In either case, constraint propagation updates the bounds of affected nodes and checks for negative cycles in the resulting network. The lack of any such cycle ensures continued temporal feasibility of the plan. Otherwise a *conflict* has been detected, and either backtracking or some amount of constraint relaxation is necessary.

The intermediate level in Figure 1 represents a level of description in an OZONE domain model that captures various temporal constraints using a construct in the ontology called a

temporalon. A temporalon [8] designates a temporal constraint between two time points. Two basic types of temporalons are definable. The first type is *temporalon-activity*, in which case the temporalon represents the temporal characteristics of an activity. Its two time points designate the activity's start and end, and the distance constraint (i.e., lower-bound, upper-bound pair) defines the activity's duration. The second basic type is *temporalon-link*, used to specify temporal relations between activities (e.g., Activity A must precede Activity B), or to designate absolute temporal restrictions (e.g., Activity A cannot start earlier than t1). Temporalons of this form either link the start/end points of distinct activities or link the start/end point of a single activity to an absolute time origin. Temporalons are specialized into a hierarchy of different types, each of which models and captures a particular aspect of the scheduling domain (e.g., *due-date* temporalon, *release-date* temporalon, *move-activity* temporalon, etc.). There is a one-to-one correspondence between the set of temporalons defined in the domain model and the temporal constraints contained in the underlying STP constraint network.

It is possible to notice some similarity between the user-friendly boxes introduced in [7] and temporalons. In [7], an assumption is made according to which each problem can be represented as a hierarchy of constraints. When this hierarchy of constraints is posted, some amount of information (user-friendly boxes) is coded into the system which can be reused to present explanation to the user. Within COMIREM the extension of the OZONE ontology based on temporalons, provides an analogous categorization and gives information at a higher level of abstraction about the constraints involved in a conflict. The temporalon layer is indeed domain oriented and its embedded knowledge can be integrated, interpreted and translated to support human intelligible explanations. The ontology used in COMIREM represents an attempt to capture different aspects of the domain and represent them through the use of domain entities close to a human model of the world. This effort previously made to model scheduling domains and problems (*user* $\rightarrow$ *system* translation) grants us useful structures that can be reused when the contrary translation is needed (*user* $\leftarrow$ *system* translation).

4.1 A filtering classification to compute the explanation set

Within a COMIREM domain model, we can distinguish several categories of temporalons. Some temporalons represent input constraints that have been imposed by the user (e.g., Activity A must end by t2). Others represent inherent properties of the domain theory defined by the scheduling domain ontology (e.g., the fact that a "move activity" decomposes into a "load", "travel", "unload" sequence of finer level activities). Still others designate decisions that the automated solver has taken (e.g., use resource R first for activity A and then for activity B). Although all temporal constraints look identical at the level of the STP constraint network, the sets of constraints falling into each of the above categories have different implications from the standpoint of conflict explanation and resolution. For example, constraints relating to the structural characteristics of a hierarchical domain model, though likely to be identified as contributing constraints in a detected conflict, are not really relevant to understanding the conflict at the user-level. Alternatively, constraints directly attributable to user decisions are clearly relevant and may be retractable if they are found to be problematic. In general, one obvious pre-requisite to generating meaningful, actionable explanations is to first isolate the subset of those constraints returned by the STP cycle detector that are relevant to user decision-making. We call this subset the *explanation set*.

To make this notion more precise, we classify COMIREM temporalon subtypes into three main categories:

- *Problem constraints.* They derive directly from the user's specification of the problem and represent user requirements. Since these constraints originated from the user, they represent one set of constraints that she may be willing to compromise when faced with an *over-constrained* problem. Temporalons falling into this category include various user-specifiable activities (*move* and *paired-event*[1] temporalons), timing restrictions (*release-date*, *due-date*, *est* and *lft* temporalons), and various activity to activity synchronization constraints (*before, same-start* and *same-finish* temporalons).

- *Structural constraints.* These constraints either model physically motivated causal dependencies in the ontology's domain theory that cannot be relaxed, or express structural temporal relationships between activities residing at different levels in hierarchical activity networks (*activity-subactivity*, *demand-activity* temporalons, etc.).

- *Search constraints*. These are constraints introduced by the search algorithm in the course of solving the problem (*sequencing* temporalon), and of course can also be retracted in the event of a conflict.

Given this categorization, our approach to determining the explanation set for a given conflict is straightforward. Upon detection of a cycle, the STP network solver returns two pieces of information: the set of constraints (temporalons) involved in the cycle, and a magnitude m indicating the amount by which some temporal constraint (or combination of constraints) must be relaxed to resolve the inconsistency. The first filtering step simply removes all *structural temporalons* from the set of constraints returned by the STP cycle detector. The resulting subset is then further pruned by eliminating those temporalons that cannot be feasibly relaxed (e.g., would result in an activity with a negative duration). The remaining temporalons constitute the explanation set and are used to characterize the conflict to the user.

4.2 Conflict resolution options

A temporal constraint conflict detected in the temporal network can be resolved by modifying the bounds of one or more temporalons in the explanation set (using the magnitude m reported with the original conflict). Given the domain level typing of temporalons, the low level action of modifying temporalon bounds can be mapped directly to higher-level, user-oriented actions (i.e., *conflict resolution options*). For each temporalon contained in the explanation set, information captured in the domain ontology is used to compute a type-specific set of possible resolution options. The union of all computed options is then presented to the user.

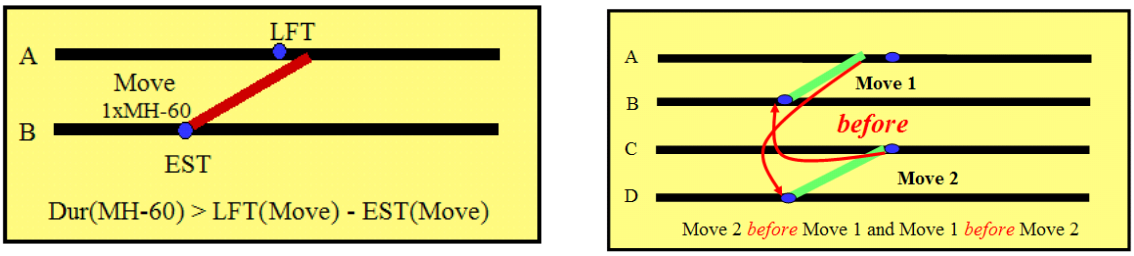
5 Examples of conflict explanation and resolution options

To illustrate the basic conflict explanation procedure just described consider the following two examples.

Example 1 Figure 2(a) shows an example of a temporal conflict discovered during feasibility checking of an input plan in COMIREM. The computed duration Dur(MH-60), of a *move* activity from location B to location A using resource (helicopter) MH-60 is greater than the difference between the latest finish time and the earliest start time constraints on the activity.

[1] a paired-event is an activity that takes place at one location.

The solver detects the inconsistency and returns the set of involved temporalons along with the conflict magnitude m.



(a) Example 1 (b) Example 2

Figure 2: Two examples of possible temporal constraints conflicts

In this example, there are three conflicting temporalons: a *move* temporalon, a *release-date* temporalon and a *due-date* temporalon. Since all belong to the *problem constraints* category (see Section 4.1) and each can be feasibly relaxed, no filtering is possible and this conflict set becomes the explanation set. Relevant resolutions are computed for each temporalon and the results are shown in Table 1.

explanation set	**explanation**	**resolution options**
	Temporal inconsistency is due to:	choose an option:
move	duration of activity move(A,B)	1 override computed duration
		2 use a faster resource
release-date	release-date constraint	3 deploy earlier
due-date	due-date constraint	4 delay engagement

Table 1: Explanation and resolution options for the conflict of Example 1

Example 2 Figure 2(b) shows a conflict due to an attempt to assert conflicting sequencing constraints between two move activities $move_1$, from location B to location A and $move_2$ from location D to location C (i.e., both $move_1$ before $move_2$ and $move_2$ before $move_1$). Given the hierarchical structure of move activities, the cycle detector actually returns a cycle with sixteen temporalons, in particular 8 structural temporalons, 2 move temporalons, 4 paired-event [2] and 2 *before* temporalons. The higher complexity of this example (16 temporalons against 3) highlights the general difficulty in providing a user with a comprehensible explanation. Using the classification, the eight structural temporalons are first eliminated from the explanation set. Subsequently, both the four paired-event and the two move temporalons are eliminated since none can be feasibly relaxed. Table 2 shows the final explanation set and the generated resolution options.

6 Conclusions

One potential source of difficulty in transitioning automated planning and scheduling theories and tools into practice is the gap between the user and system's models. Bridging this gap re-

[2] paired-event temporalons derive from the decomposition of a move activity into a "load", "travel" and "unload"sequence of activities.

explanation set	explanation	resolution options
	Temporal inconsistency is due to:	choose an option:
before	move1(A,B) before move2(C,D)	1 relax *before* constraint
before	move2(C,D) before move1(A,B)	2 relax *before* constraint

Table 2: Explanation and resolution options for the conflict of Example 2

quires a system to translate information from internal algorithmic representations into higher level descriptions that are more comprehensible to the user. In this paper we considered the problem of closing the loop with the user in situations where the automated solver reaches an impasse, so that initiative can be redirected back to the user and cooperative problem solving can continue. We focused specifically on the problem of explaining temporal constraint conflicts, and using the COMIREM planner/scheduler as a reference model, proposed the use of a scheduling domain ontology as a means of generating user-oriented explanations. We showed a couple of explanation generation examples, which give preliminary evidence of the potential of our approach.

Acknowledgments

Stephen F. Smith, David Hildum and Christian Ohler were funded in part by the Department of Defense Advanced Research Projects Agency (DARPA) under contract F30602-00-2-0503 and the CMU Robotics Institute. Gabriella Cortellessa's work is partially supported by ASI (Italian Space Agency) under projects ARISCOM and SACSO. This work has been developed during her visit at the CMU Robotics Institute as a visiting student scholar.

References

[1] C. Bessiere. Arc consistency in dynamic constraint satisfaction problems. In *Proceedings AAAI'91*, 1991.

[2] Mark Burstein and Drew McDermott. Issues in the development of human-computer mixed-initiative planning. In B. Gorayska and J.L. Mey, editors, *Cognitive Technology*, pages 285–303. Elsevier, 1996.

[3] R. Cohen, C. Allaby, C. Cumbaa, M. Fitzgerald, K. Ho, B. Hui, C. Latulipe, F. Lu, N. Moussa, D. Pooley, A. Qian, and S. Siddiqi. What is initiative? In S. Haller, S. McRoy, and A. Kobsa, editors, *Computational Models of Mixed-Initiative Interaction*, pages 171–212. Kluwer Academic Publishers, 1999.

[4] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[5] N. Jussien and V. Barichard. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, pages 118–133, Singapore, September 2000.

[6] Narendra Jussien. e-constraints: explanation-based constraint programming. In *CP01 Workshop on User-Interaction in Constraint Satisfaction*, Paphos, Cyprus, 1 December 2001.

[7] Narendra Jussien and Samir Ouis. User-friendly explanations for constraint programming. In *ICLP'01 11th Workshop on Logic Programming Environments*, Paphos, Cyprus, 1 December 2001.

[8] S. Smith and H. Hildum. Interacting with freon: A quick overview. Technical report, Carnegie Mellon University.

[9] S. F. Smith and M. A. Becker. An ontology for constructing scheduling systems. In *Working Notes of 1997 AAAI Symposium on Ontological Engineering*, Palo Alto, CA, March 1997. AAAI Press.

[10] S.F. Smith, D.W. Hildum, and D.A. Crimm. Interactive Resource Management in the Comirem Planner. In *IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems*, Acapulco Mexico, August 2003.

[11] Richard Wallace and Eugene Freuder. Explanation for Whom? In *CP01 Workshop on User-Interaction in Constraint Satisfaction*, Paphos, Cyprus, 1 December 2001.