Validation and Verification Issues in a Timeline-based Planning System

A. Cesta[†] and **A. Finzi**[‡] and **S. Fratini**[†] and **A. Orlandini**[*] and **E. Tronci**[§]
[†] ISTC-CNR, Via S.Martino della Battaglia 44, I-00185 Rome, Italy
[‡] DSF "Federico II" University, Via Cinthia, I-80126 Naples, Italy
[*] DIA "Roma TRE" University, Via della Vasca Navale 79, I-00146 Rome, Italy
[§] DI "La Sapienza" University, Via Salaria 198, I-00198 Rome, Italy

Abstract

One of the key points to take into account to foster effective introduction of AI planning and scheduling systems in real world is to develop end user trust in the related technologies. Automated planning and scheduling systems often brings solutions to the users which are neither "obvious" nor immediately acceptable for them. This is due to the ability of these tools to take into account quite an amount of temporal and causal constraints and to employ resolution processes often designed to optimize the solution with respect to non trivial evaluation functions.

To increase technology trust, the study of tools for verifying and validating plans and schedules produced by AI systems might be instrumental. In general, validation and verification techniques represent a needed complementary technology in developing domain independent architectures for automated problem solving. This paper presents a preliminary report of the issues concerned with the use of two software tools for formal verification of finite state systems to the validation of the solutions produced by MrSPOCK, a recent effort for building a timeline based planning tool in an ESA project.

Introduction

Designing Artificial Intelligence (AI) planning and scheduling systems suitable for supporting human mission planners in their daily work is a challenging research stream at ESA (the European Space Agency) and in other space agencies. The APSI (Advanced Planning and Scheduling) initiative at ESA-ESOC is an example of research activity aimed at demonstrating the effectiveness of AI in support of internal long term programs. As already known, space applications introduce very challenging problems for Planning and Scheduling (P&S) technologies that compete with a number of reasoning and automated control tools in the aim to challenge their own autonomous features to face such a problems. Frequently, models and solutions proposed are very complex and even engineers, designers and scientists can have difficulties in validating and verifying them by simple inspection. Then, automated Validation and Verification (V&V) techniques are an important contribution, adding value in such kind of applications. In fact, a failure from an automated decision may have a dramatic impact in terms of loss of either science activities or money or even human life. It is quite commonly acknowledged that integration of V&V capabilities in general purpose P&S architectures may have a significant impact on the common use of our reference technology.

Validation of planning models has been studied in several works[1]. For instance, in (Pecheur and Simmons 2001; Khatib, Muscettola, and Havelund 2001) Livingstone and HSTS domain models are validated exploiting model checking techniques. In (Smith et al. 2005), formal verification is used in order to check the existence of undesirable plans with respect to the domain model. While, VAL (Howey and Long 2003) is a plan validation tool for PDDL that was successfully used during the International Planning Competition since 2002.

Current AI planning literature shows how timeline-based planning can be an effective competitor for classical planning to tackle complex domains which require the use of both temporal reasoning and scheduling features (see (Muscettola 1994; Jonsson et al. 2000; Frank and Jonsson 2003; Smith, Frank, and Jonsson 2000)). The work described here is connected to timeline planning because of a general effort to build a reusable software framework for modeling space missions problems using timelines (see (Cesta, Fratini, and Pecora 2007)). The timeline-based approach models the P&S problem by identifying a set of relevant *features* of the planning domain which need to be controlled to obtain a desired temporal behavior. Timelines model entities whose properties may vary in time and which represent one or more physical (or logical) subsystems which are relevant to a given planning context. The planner/scheduler plays the role of the controller for these entities, and reasons in terms of *constraints* that bound their internal evolutions and the desired properties of the generated behaviors (goals).

In our current work we plan to explore different perspectives in the integration of V&V with timeline based planning and scheduling techniques. The long term goal is to obtain a software environment in which both technologies are integrated and the application developers may take advantage of the co-existence of the two tools while knowledge engineering new application. Indeed, we are currently in an

[1]See also a specific workshop at ICAPS-05.

initial stage in our path and are reporting here such current work. In the attempt of developing a common terminology, we have started considering (a) a particular planning system, the application called MrSPOCK, for "Mars Express Science Plan Opportunities Coordination Kit", a recent effort for building a timeline based planning system for the ESA; and (b) two state-of-the-art software tools in formal verification, NuSMV and UPPAAL, and (c) focused on the problem of the validation of the solutions produced by MrSPOCK. Notwithstanding the initial extent of the current effort we have already received some interesting hints on the whole activity.

The preliminary analysis in this paper aims at presenting the approach we have followed as well as discussing the problems encountered and the lessons learned in trying the automatic validation and verification of MrSPOCK's solutions against the temporal and causal constraints in the symbolic model that drives the tool in building the plans.

Target Scenario

As said in the introduction we have focused our current activity in developing a validation tool for the MrSPOCK planner recently developed within the APSI project for MARS EXPRESS long, medium and short term planning. The open problem was to support the collaborative problem solving process between the science team and the operation team of the space mission. These two groups of human planners iteratively refine a plan containing all activities for the mission. The process starts at the long term plan (LTP) level – three months of planning horizon – and is gradually refined to obtain fully instantiated activities at short term plan (STP) level - one week of planning horizon. This process continuously leads to weekly STPs, which are then further refined every two days to produce final executable plans. Goal of MrSPOCK has been to develop a pre-planning optimization tool for spacecraft operations planning. Specifically, we have focused on the generation of a *pre-optimized skeleton LTP* which will then be subject to cooperative science team and operation team refinement (see (Cesta et al. 2008b) for a detailed description of the addressed problem).

We observe that, at the first step of the negotiation process, one source of approximation comes from the fact that the operation team has only partial information about the requested science operations for MARS EXPRESS. Payload Operation Requests (PORs), that is the requested science operations. Reference to such requests is given only when the Pointing Timeline Request is issued by the science team on the basis of the input skeleton plan generated by operation team. Indeed, these science requests often require the satellite to point to the planet, reducing its ability to obtain energy from the Sun and to send data back to Earth. Also, science operations consume power and exclude the possibility of performing maintenance operations. On the other hand, the operation team requires the spacecraft to perform maintenance and other service maneuvers in order to maintain the spacecraft operational. Overall, the two groups of managers have to cooperatively converge on a plan which resolves the complex interplay between science, pointing direction, power, data transmission and maintenance operations.

In this context, the challenge of MrSPOCK is to provide an automated procedure for producing a *good* skeleton plan, i.e., a LTP that takes into account the needs of both parties, thus reducing the effort in reaching an agreement on a medium-term plan – one month planning horizon. Overall, the generated LTP should be such that: (a) the number of (expensive) iterations between science and operation team is reduced; (b) a set of objective functions – the total volume of data for down-link operations; the number of pericentres for science operations; the number and the uniform distribution of uplink windows – are optimized.

Problem Required Constraints

For each orbit followed by the spacecraft, the baseline operations are split into three phases: (1) around the *pericentre* (the orbital closest to the target planet); (2) around the *apocentre* (the orbital more far away from the planet); (3) *between* the pericentre and apocentre passages. Around pericentre, the spacecraft is generally pointing to the centre of the planet thus allowing observations of the planet surface – generically referred to as *Science operation*. Between pericentre and apocentre passages, the spacecraft is generally Earth pointing for transmitting data to Earth. *Communication* with Earth should occur within a *ground station availability window*. Ground station visibility can either partially overlap or fully contain a pericentre passage. Additionally, *Maintenance* operations should occur around the apocentre passages.

At present, given these requirements, an initial skeleton plan for MARS EXPRESS is generated by the operation team by allocating over the planning horizon (which generally covers hundreds of orbits) three different types of decisions:

– selection of the *Maintenance* windows (generally centered around the apocentre events and used primarily for *momentum wheel-offloading*);

– selection of the *Communication* windows among the set of available ground stations (the so-called ground stations de-overlapping);

– selection of the windows for *Science* operations around pericentre events.

Additionally there are quite an amount of *hard* and *soft* constraints to be satisfied. Constraints on uplink windows frequency and duration require four hours uplink time for each 24 hours (soft constraint). Moreover, there should be given the possibility to split a four-hour uplink window in two two-hour uplink windows. Apocentre slots for spacecraft maintenance windows must be allocated between 2 and 5 orbits apart, and the maintenance duration is of 90 minutes to be centered around the apocentre interval.

Communication activities are source of several temporal constraints to be considered as hard. For example: (1) the minimum/maximal durations for the X-band transmitter in the *on* state, (2) the minimum duration for the X-band transmitter in the state *off*; (3) the periods in which the X-band transmitter has to be *off* (e.g., eclipses, occultations, slewing manoeuvres and non-Earth pointing status).

Furthermore, there are preferences to follow for ground station selection (called *de-overlapping* in mission terminology). Ground stations have different features like different dish diameters (there are 70 meters dish antennas, 35 meters and 34 meters). Usually, they allow both uplink and downlink communications, but there are cases where it is only possible to downlink. Additionally, there are ground stations owned by different agencies and they should be used according to some policy restriction.

A hybrid solver

In building MrSPOCK, we have followed a hybrid approach. The timeline representation and management features of a general purpose planning and scheduling system, called Open Multi-Component Planner and Scheduler (OMPS (Fratini, Pecora, and Cesta 2008)), are exploited for modeling some features of the problem and for computing and maintaining timelines. In addition, a domain dependent specific planner exploits the domain independent underlying planning system by guaranteeing the satisfaction of the problem's constraints not modeled in the domain description and performing plans optimization exploiting a genetic approach.

OMPS is designed as a layered architecture: there is an underlying temporal database (that provides primitives to represent and manage time points and temporal constraints), a timeline management and representation layer above the temporal database (that provides primitives to represent temporal flexible plans as timelines) and a portfolio of domain independent planning and scheduling procedures on top. MrSPOCK uses the OMPS Timeline Representation Framework (TRF) and proposes a domain dependent search procedure on top of the TRF.

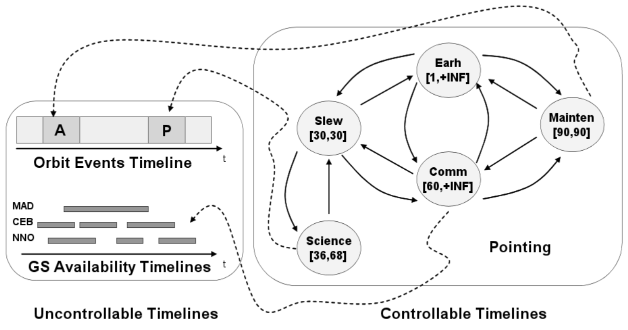

Figure 1: The domain model

In MrSPOCK, we use the TRF features by representing the domain with two different types of timelines (see Figure 1): (1) *Controllable State Variables*, which define the search space of the problem, and whose timelines ultimately represent the solution to the problem; (2) *Uncontrollable State Variables*, representing values imposed over time which can be only observed. Here, we use a single controllable state variable to model the spacecraft's pointing mode, which specifies the temporal occurrence of science and maintenance operations as well as the spacecraft's ability to communicate. The values that can be taken by this state variable, their durations (represented as a pair $[min, max]$) and the allowed transitions among them are synthesized by the automaton in the right in Figure 1. This is an effective way to capture part of the set-up necessary constraints in the solution.

In addition, we instantiate two uncontrollable state variable to represent contingent events such as orbit events and communication opportunity windows. One state variable type component maintains the temporal occurrences of pericentres and apocentres ("P" and "A" values on the timeline in Figure 1, left, top) of the spacecraft's orbit (they are fixed in time according to the information found in an orbit events file), while the other state variables maintains the visibility of three ground stations ("MAD","CEB" and "NNO" timelines in Figure 1, left, bottom). These state variables have as allowed values $\{$Available(?rate,?ul_dl,?antennas), Unavailable()$\}$, where the ?rate parameter indicates the bitrate at which communication can occur, ?ul_dl indicates whether the station is available for upload, download or both, and the ?antennas parameter indicates which dish is available for transmission. Any valid plan needs temporal synchronizations among the pointing timeline and the uncontrollable timelines (represented as dotted arrows in Figure 1): science operations must occur during Pericentres, maintenence operations must occur during Apocentres and communications must occur during ground station visibility windows. In addition to those synchronization constraints, the pointing mode timeline must respect the transition among values specified by the automaton and the minimal and maximal duration specified for each value (in the automaton as well).

On top of this model, MrSPOCK's solver allocates science, maintenance and communication activities and exploits the TRF features for synchronizing them with orbit events and ground station visibilities. By querying the TRF that maintain the actual temporal occurrence of previously allocated activities, orbit events and available visibility windows, the domain dependent planner allocates new activities enforcing constraints on uplink windows and maintenance frequencies. The choice between science, maintenance and communication activities (when available stations allow a communication activity) is driven by a chromosome. Once the complete temporal plan has been instantiated, it is measured and used in a genetic optimization loop aiming at maximizing science allocation (see (Cesta et al. 2008a) for a MrSPOCK's more detailed description).

Model Checking

As previously discussed, MrSPOCK proposes solutions taking into account quite an amount of temporal and causal constraints defined by ESA science and operation team members. Thus, they have to accept and use such a solutions as basis for further refinement until complete STPs are defined. An effective introduction of AI planning and scheduling has to deal with users' trust in related technologies. In fact, automated P&S systems often brings solutions to users which are neither "obvious" nor immediately acceptable for them.

Furthermore when a hybrid approach that mixes different solving procedures is followed, as we have done in MrSPOCK, the need of an indipendent solution verification process increases. In fact, since not all the problem constraints are taken into account in the domain description of the general planning system, the proof of correctness of the domain independent planning system can not ensure the correctness of the produced solutions. Moreover a genetic optimization process is performed within MrSPOCK and the chromosome management performed by the genetic algorithm might invalidate the solution as well. Hence the need of verifying the solutions of such a kind of hybrid solver.

Aiming at fostering users to act more confidently with MrSPOCK and increasing the effectiveness of the supporting tool, we perform an additional effort to verify solutions validity with a different technology. In fact, it is worth noting that performing an independent model verification can provide support also during design phase, helping in validating the domain model.

Thus increasing the overall framework robustness.

In this sense, V&V techniques represent a needed complementary technology in developing domain independent architectures for automated problem solving. In particular, model checking is a well known verification technology used to verify requirements and design for a variety of real-time embedded and safety-critical systems. Then, two prominent software tools in model checking, NuSMV and UPPAAL, are considered, focusing on the problem of the validating solutions produced by MrSPOCK.

NuSMV (Cimatti et al. 2002) is a model checker for temporal logics. It has a dedicated modelling language, which permits the definition of concurrent finite state systems in an expressive, compact, and modular way. The SMV specification uses variables with finite types, grouped into a hierarchy of module declarations. Each module states its local variables, their initial value and how they change from one state to the next. The properties are expressed in Computation Tree Logic (CTL). CTL is a branching-time temporal logic, which means that it supports reasoning over both the breadth and the depth of the tree of possible executions.

UPPAAL (Larsen, Pettersson, and Yi 1997), whose acronym comes from joining the names of UPPsala and AALborg universities that built it, is a tool box for modeling, simulation, and verification of real-time systems. The verifier covers the exhaustive dynamic behavior of the system for proving safety and bounded liveness properties. A UPPAAL model consists of a set of timed automata, a set of clocks, global variables and synchronizing channels. A node in an automaton may be associated with an invariant, for enforcing transitions out of the node. An arc may be associated with guards, for controlling when this transition can be taken. On any transition, local clocks may get reset and global variables may get re-assigned. Channels are used in order to synchronize transitions on different automata. Analogously to NuSMV, verified properties are stated in CTL.

The rest of the paper reports on our current achievements using these two tools for validating MrSPOCK output.

Plan Validation within MrSPOCK

In this section a possible mapping between plan validation and model checking tasks is presented and discussed. In particular, in Figure 2, an integrated plan validation architecture is presented.

MrSPOCK domain and plan are encoded in a new model, while a property assuring plan validity is defined. Then, both the model and property are provided as inputs to model checking tool. In this sense, an initial goal is to build the new model, such that it can be encoded both in NuSMV and UPPAAL input languages, and state a suitable property for plan validity check.

Starting to address such an issue, target application domain description and a completely specified temporal solution plan, generated by MrSPOCK, are to be encoded in the new model. Such a model is to be translated in the model checkers input languages. Thus, plan validity property is defined according to such a model.

In order to simplify the presentation, let us consider in the following no parameters handling. Although such a feature is managed in our modelling, it is not quite relevant with respect to the current discussion.

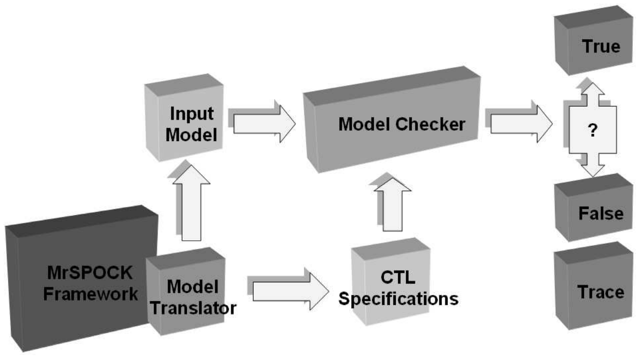

Figure 2: Plan validation architecture exploiting model checking.

Modelling MrSPOCK Domain and Plan

First, domain and plan are to be encoded in the new model. Within the framework, a domain model is represented as a set of components (i.e. timelines). For each of them, a set of *consistency features* are defined in order to state the nominal component's behaviors. Consistency features represent transition and temporal constraints related to one singular timeline. Looking at Figure 1, consistency features related to the spacecraft's pointing mode are, for instance, the duration constraint on Science activity (within $\{36, 68\}$ interval) or the transitions from *Maintenance* to *Earth* or *Comm* states.

In addition, some inter-components relations are specified in order to state constraints between different timelines. Such a relations compose an overall *domain theory*. In such a way, a component's timeline can be related with other

```
// General variables definition
Define a clock for each component;

Define a counter for each component;

// Components description
For each component,
    Generate a corresponding automaton;
    For each related consistency features
      Generate into the automaton description an appropriate transition;

    Clock counts how much time a certain value holds.
    When a transition occurs, clock resets to zero.

// Plan description
For each behavior in the plan
    Generate a corresponding automaton;
    For each value change in the plan
      Generate into the automaton description an appropriate transition;

    Counter counts how much times a value change occurs in the behavior.

// Monitor description
Generate a monitor automaton with two different states {OK, ERR};
  For each value and time interval in the plan
    Generate a transition from OK to ERR
         if component automaton presents a different value;
  For each domain theory relation
    Generate a transition from OK to ERR if relation is violated;

// Files Generation
NuSMV and UPPAAL input language files are generated.
```

Figure 3: An algorithm for mapping MrSPOCK domain and plan models into reference model for model checkers.

different components' behavior. Synchronizations between pointing's and uncontrollable timelines constitute the set of domain theory constraints. Science operations occurrences during pericentre orbits or ground station availability needed during communications are examples of such a constraints.

A solution plan is represented as a set of decisions on the components' behaviors. Focusing on a singular component, the solution plan is a sequence of allowed values that the component has to assume in a given time frame. Also the time points, at which such a changes occur, are fixed and provided within the plan.

An algorithm for mapping MrSPOCK domain and plan into the new model is presented in figure 3. First, general variables are defined. For each component, a clock and a counter are generated. The clock counts how many times a value holds in a component, while the counter enumerates the plan steps within the behavior.

For each component, a corresponding automaton is generated stating its consistent behaviors. In fact, for each corresponding consistency features an appropriate transition is defined. Thus, temporal constraints can be stated as temporal guards on transitions or as state invariants.

Then, for each timeline, a corresponding automaton is generated and appropriate transitions are introduced to define values changes.

In order to validate plan behaviors, components and planned values are to be related within the time frame considered. That is, components and planned timelines have to follow the same evolution. In this way, it is guaranteed that the solution plan is consistent w.r.t. the domain model. Such a relation, can be defined by synchronizing plan behaviors and components changes. Thus, when a plan behavior requires a certain value for a component, the related component automata has to assume such a value.

Once such a relation is realized, a monitor automaton is generated to check whether plan and domain are consistent or not. If an inconsistency occurs, an error in the plan exists. In such a case the monitor changes its status from *normal* to *error*. Of course, the monitor checks also domain theory violations.

Since our goal is not only to validate solution plans, but also to support users during problems diagnosis, we define the monitor automaton with multiple error states. In this way, users can have additional information about which are the failing components or which domain theory relations are violated. Such a feature can be easily introduced in the validation architecture, introducing suitable transitions from normal to failing states.

Referring to MrSPOCK 's scenario, the monitor automata checks if the planned pointing's timeline is consistent with its allowed temporal evolutions. For instance, *Earth, Slew, Science, Slew, ...* (with appropriate durations) is a consistent behavior, while *Earth, Slew, Science, Earth, ...* is not. Moreover, the monitor verifies that all the timelines are consistent w.r.t. the domain theory constraints: for instance, whenever the *Science* status is present on pointing timeline, the orbit events' timeline has to present *Pericentre* status.

Validation Properties

Once the model previously defined is provided as input to model checkers, a validity plan property has to be defined. So, model checkers can perform verification.

In order to state such a property, we refer to monitor automaton. With respect to the model presented above, the property to be checked is the following: *for each timeline, the last plan step can be reached and the monitor remains normal*. Since, model checkers expect a CTL formula, we use the following:

```
AG (last plan steps can be reached) and
        (Monitor is normal).
```

Where **AG** means that for **A**ll the possible system evolutions is **G**lobally true that the property is satisfied. Whenever the above formula does not hold, model checkers produce an execution trace of the system witnessing that the monitor automaton reach an *error* state. That is, such a trace represents a system execution showing how the timelines are inconsistent with respect to the components' behaviors. Thus, the reported trace can be used to identify the plan error, or the domain inconsistency, and diagnose from which conditions it origins. In this way, completely specified temporal plans can be validated while very useful information can be provided to users whenever an error occurs.

We performed some preliminary tests in order to verify plan validation architecture performances. We run tests on a linux workstation endowed with a 64-bit AMD Athlon CPU (3.5GHz) and 2GB RAM. We validate plans generated by MrSPOCK ranging within 1 to 10 days of activity, handling from 45 to 335 tasks on all the timelines. The results, depicted in Figure 4, show that UPPAAL performs better than NuSMV[2].

[2]NuSMV with BMC instead of BDD model checking.

UPPAAL works on-the-fly, which means that it does not preconstruct a global state graph, or Kripke structure, as a prerequisite for the verification of system properties. Since NuSMV performs such a kind of preconstruction, the experimental results collected do not surprise us.

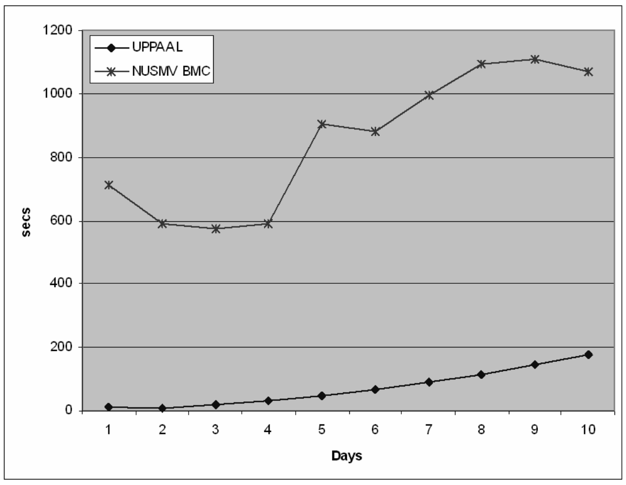

Figure 4: Experimental Results for UPPAAL and NuSMV.

Although the validation technique presented here is preliminary, we have already obtained some interesting results. For example, our verification tool allowed us to detect and solve an inconsistency in MrSPOCK domain. Namely, we found that MrSpock could generate solutions not consistent with the apocentre-maintenance occurences constraint, which is an implicit requirement (i.e. not represented in the temporal model) for the hybid solver. Using the proposed validation architecture, it was possible to detect the inconsistency and, exploiting the additional information provided by the reported trace, diagnose and fix the error.

Flexible Temporal Plan Validation

In order to address the general plan validation problem, it is necessary to handle flexible temporal plans. Then, an extension of the model presented above is required. Namely, time points are to be represented as time intervals rather than as fixed time instants. Such a representation can be easily introduced in the model by simply considering temporal variables over a certain interval of values. For instance, if a flexible time point can assume values in a given temporal interval $[min, max]$, then a variable is to be considered in the model with value in $\{min...max\}$. In this way, model checkers can explore and validate all the possible temporal evolutions of the plan.

By suitably modelling synchronization between such a variables, domain model components and plan behaviors, we can use the proposed architecture also for flexible plan validation.

Nevertheless, some problems araise in handling such a flexibility. First, during property verification, model checkers consider all the possible temporal combinations over variables, even those that have not to be taken into account because of temporal inconsistency with respect to overall constraints.

To better figure out such a problem, consider the following simple example. A plan can contain a task with start and end times defined by two different flexible temporal points t_s and t_e. The activity starts within the interval $[5, 10]$, while it ends within the interval $[15, 20]$. In addition, a duration constraint is defined as follows: $t_e - t_s \leq 10$.

Following our approach, two variables are defined within the flexible model, v_s and v_e on $\{5...10\}$ and $\{15...20\}$, respectively. In this case, not all the possible combinations are to be checked. In fact, if $v_s = 5$ then v_e can assume only 15 as admitted value, for all the other values v_e can assume are inconsistent and can be avoided during validation. Analogous reasoning can be repeated for each assignment on v_s.

In this sense, we are far away to present significant experimental results for flexible plan validation, but different considerations can be done in order to analyse such a problem. On the one hand, considering inconsistent combinations does not influence the verification process, but increases the time needed for the computation. In fact, a larger number of evolutions are to be considered and verified, even though already known to be not valid, as previously shown in the example. On the other hand, we can not still use witness trace in order to diagnose the problem. In fact, they can return as failing trace exactly one of the inconsistent behaviors, adding no useful information. This is due to our model, which does not permit to perform constraints propagation over flexible temporal intervals.

In this sense, addressing such a problem is the next challenging issue. Our feelings are that this should be possible since all constraints are part of the problem description model but should be handle conveniently.

Related Works

In this section, some related works concerning plan verification and validation systems based on temporal models are briefly discussed.

Closely related to our work is the framework proposed in (Abdedaim et al. 2007). Here, the authors investigate and compare Constraint Based Temporal Planning techniques and Timed Game Automata methods for representing and solving realistic temporal planning problems. In this direction, they propose a mapping from IxTeT planning problems to UPPAAL-TIGA game-reachability problems and present a comparison of the two planning approaches. This paper is mainly focused on robust plan synthesis while we are interested in plan validation and verification. Indeed, our aim is to address validation and verification issues arising when planning in complex domains and depolying hybrid solving algorithms; these problems are not considered in (Abdedaim et al. 2007).

Our overall approach is more similar to the one proposed in (Smith et al. 2005; Havelund et al. 2008), where model checking (in SPIN) is deployed to guarantee that plans produced by generic automated planning systems meet certain desirable properties. However, differently from our case, the focus is on model validation: planning models are tested and refined to prevent the generation of undesirable plans. The modelling framework and the properties considered in this paper are quite different from the one we are interested in. Indeed, real-time temporal properties and flexible temporally plan verification and validation are not addressed.

A more expressive temporal model is considered in (Khatib, Muscettola, and Havelund 2001) where the authors propose a mapping from interval-based temporal relations models (i.e. DDL models for HSTS) to timed automata models (UPPAAL). This mapping was introduced as a preliminary step towards the application of verification and validation techniques in timeline-based temporal planning, however this direction is not fully developed. In particular, the authors do not propose methods for plan validation.

In (Vidal 2000), a mapping from Contingent TCN into TGA models is proposed. However, also in this case, methods for temporal plan verification are not presented.

In the framework of PDDL, we can find the VAL plan validation tool (Howey and Long 2003) that was extended to permit the validation of plans with durative actions. Although VAL was successfully used during International Planning Competitions since 2002, several open modelling issues are still to be addressed (Fox et al. 2006).

Formal verification for timeline-based temporal planning is considered also in the ANML framework, a timeline-based specification framework under development at NASA Ames. The work in (Siminiceanu, Butler, and Munoz 2008) presents a translator from ANMLite (abstact version of ANML) to the SAL model checker. Given this mapping, the authors present preliminary results to assess the efficency of model checking in plan synthesis. Plan validation and verification issues are not discussed.

Less closely related, we can find other frameworks where model checking and temporal plan verification are deployed to support robust plan synthesis. For instance, in the CIRCA framework (Goldman et al. 2002), a Controller Synthesis Module (CSM) automatically synthesizes hard real-time reactive plans; the CSM is modeled through a timed automaton model and a model-checking plan verifier is used as a support for robust reactive planning. Here, the main concern is on-the-fly synthesis of control sequences, hence issues and methods (e.g. reactive plan generation and verification) are different from the ones discussed in our paper.

Conclusion

In this work, we have presented a preliminary approach to plan validation and verification in a timeline-based planning system. In particular, we have considered V&V issues arising in the MrSPOCK framework, a recent effort for building a timeline-based planning system for the European Space Agency. In this context, plan validation and verification tools are particularly relevant. Indeed, the P&S system of MrSPOCK is based on a hybrid approach where not all the domain constraints can be explicitly represented in the plan domain, therefore the soundness of the plan with respect to the model does not necessarily ensure the soundeness of the produced solution. In this case, an independent solution verifier is needed to test plan consistency with respect to implicit requirements. Furthermore, from the end user perspective, validation and verification tools offer an independent testing environment which can enhance end user trust on the complex and (sometimes) counterintuitive solutions generated by MrSPOCK.

The aim of this paper was to present the approach followed, discussing the problems encountered and the lessons learned in the attempt of deploying V&V techniques and modelling tools in MrSPOCK. In this context, we have presented our integrated plan validation system where plan validation is based on model checking. More specifically, we have defined a translation from our domain models to specifications in UPPAAL and NuSMV in so enabling formal verification of domain and plan properties. In the MrSPOCK domain, we have discussed relevant issues reporting on some preliminary results in temporal plan verification. We have also tackled the problem of flexible temporal plans validation, however, further investigations are necessary to obtain an efficient and effective verification process in this setting.

A lot of work remains to be done, our long term goal is to obtain a software environment in which V&V technologies are integrated in a complex P&S system and the application developers may take advantage of the co-existence of these tools while knowledge engineering new application.

Acknowledgments. Cesta, Fratini and Tronci are partially supported by the EU project ULISSE (Call "SPA.2007.2.1.01 Space Science"). Orlandini is partially supported by ISTC-CNR. Cesta and Fratini are also partially supported by European Space Agency (ESA) within the Advanced Planning and Scheduling Initiative (APSI).

References

Abdedaim, Y.; Asarin, E.; Gallien, M.; Ingrand, F.; Lesire, C.; and Sighireanu, M. 2007. Planning robust temporal plans: A comparison between cbtp and tga approaches. In *Proceedings of ICAPS-2007*, 2–10.

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2008a. Looking for MrSPOCK: Issues in Deploying a Space Application. In *ICAPS Workshop on Scheduling and Planning Applications, Sydney, Australia.*

Cesta, A.; Fratini, S.; Oddi, A.; and Pecora, F. 2008b. APSI Case#1: Pre-planning Science Operations in MARS EXPRESS. In *i-SAIRAS-08. Proceedings of the 9^{th} Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*. JPL, Pasadena, CA.

Cesta, A.; Fratini, S.; and Pecora, F. 2007. A Multi-Component Framework for Planning and Scheduling Integration. In *PlanSIG'07. Proceedings of the 26th Workshop of the UK Planning and Scheduling Special Interest Group, Prague, Czech Republic, December 17-18.*

Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. Nusmv 2: An opensource tool for symbolic model checking. In *14th International Conference on Computer-Aided Verification (CAV'2002).*

Fox, M.; Long, D.; Baldwin, L.; Wilson, G.; Woods, M.; Jameux, D.; and Aylett, R. 2006. On-board timeline validation and repair: A feasibility study. In Giuliano, M., ed., *Proceedings of 5th International Workshop on Planning and Scheduling for Space*.

Frank, J., and Jonsson, A. 2003. Constraint based attribute and interval planning. *Journal of Constraints* 8(4):339–364.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying planning and scheduling as timelines in a component-based perspective. *Archives of Control Sciences* 18(2):5–45.

Goldman, R. P.; Musliner, D. J.; ; and Pelican, M. J. 2002. Exploiting implicit representations in timed automaton verification for controller synthesis. In *Proceedings of the 2002 Hybrid Systems: Computation and Control Workshop*.

Havelund, K.; Groce, A.; Holzmann, G.; Joshi, R.; and Smith, M. 2008. Automated testing of planning models. In *Proceedings of the Fifth International Workshop on MODEL CHECKING and ARTIFICIAL INTELLIGENCE*, 5–17.

Howey, R., and Long, D. 2003. Val's progress: The automatic validation tool for pddl2.1 used in the international planning competition. In *Proceedings of the ICAPS 2003 workshop on The Competition: Impact, Organization, Evaluation, Benchmarks*, 28–37.

Jonsson, A.; Morris, P.; Muscettola, N.; Rajan, K.; and Smith, B. 2000. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)*.

Khatib, L.; Muscettola, N.; and Havelund, K. 2001. Mapping temporal planning constraints into timed automata. In *TIME-01. The Eigth International Symposium on Temporal Representation and Reasoning*, 21–27.

Larsen, K. G.; Pettersson, P.; and Yi, W. 1997. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer* 1(1-2):134–152.

Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., ed., *Intelligent Scheduling*. Morgan Kauffmann.

Pecheur, C., and Simmons, R. G. 2001. From livingstone to smv. In *FAABS '00: Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*, 103–113. London, UK: Springer-Verlag.

Siminiceanu, R. I.; Butler, R. W.; and Munoz, C. A. 2008. Experimental evaluation of a planning language suitable for formal verification. In *Proceedings of the Fifth International Workshop on MODEL CHECKING and ARTIFICIAL INTELLIGENCE*, 18–34.

Smith, M. H.; Holzmann, G. J.; Cucullu, G. C.; and Smith, B. D. 2005. Model checking autonomous planners: Even the best laid plans must be verified. In *Aerospace, 2005 IEEE Conference 5-12 March 2005*, 1 – 11. IEEE Computer Society.

Smith, D.; Frank, J.; and Jonsson, A. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15(1):47–83.

Vidal, T. 2000. A unified dynamic approach for dealing with temporal uncertainty and conditional planning. In *Proceedings of AIPS-2000*, 395–402.