From Abstract Crisis to Concrete Relief: A Preliminary Report on Combining State Abstraction and HTN Planning

Susanne Biundo and Bernd Schattenberg
University of Ulm
Institute of Artificial Intelligence
D-89069 Ulm, Germany
firstname.lastname@uni-ulm.de

Abstract
Flexible support for crisis management can definitely be improved by making use of advanced planning capabilities. However, the complexity of the underlying domain often causes intractable efforts in modeling the domain as well as a huge search space to be explored by the system. A way to overcome these problems is to impose a structure not only according to tasks but also according to relationships between and properties of the objects involved, thereby using so-called decomposition axioms. We outline the prototype of a system that is capable of tackling planning for complex application domains. It is based on a well-founded combination of action and state abstractions. The paper presents the basic techniques and provides a formal semantic foundation of the approach. It introduces the planning system and illustrates its underlying principles by examples taken from the crisis management domain used in our ongoing project.

1 Introduction
When trying to exploit planning technology for realistic applications like system support for crisis management, one of the main problems to be tackled is the complexity of the underlying domain. Not only does it cause intractable modeling efforts, a huge search space has to be explored by the system as well. Furthermore, such a system has to be flexible in the sense that mixed initiative planning has to be supported and incoming information as well as most recently arising tasks should be considered and integrated during runtime. In order to meet multiple requirements like in this case, hybrid planning approaches have to be developed to provide enough flexibility and lucidity as has repeatedly been argued by other authors likewise (cf. (Estlin, Chien, and Wang 1997) and (Kambhampati, Mali, and Srivastava 1998)).

We introduce a planning approach for system support in the realistic and complex application domain of crisis management. It integrates hierarchical action- and state-based techniques in a consequent way by imposing hierarchical structures on both operators and states. The hierarchical concept is partly adopted from traditional hierarchical task network (HTN) planning (cf. (Erol, Hendler, and Nau 1994)). Therefore, basic notions like primitive and abstract tasks as well as methods for decomposing the latter stepwise into primitive ones are among the core concepts. However, tasks do show pre- and postconditions –like operators do in classical state-based planning– on every level of abstraction. This provides the flexibility to make use of state-based planning techniques by introducing additional tasks when trying to establish missing preconditions, and enables the system to integrate incoming tasks on any level of abstraction at any time. This is done by allowing for so-called decomposition axioms, which are defined as part of the domain model. The planning approach is based on a formal semantics, which relies on previous work on logic-based planning (Stephan and Biundo 1996). A first prototype implements our integrative approach.

The paper is organized as follows. In Section 2 we introduce the formal semantics. The application domain –a mission of the German Federal Agency for Technical Relief at a flood– is briefly introduced in Section 3. Section 4 describes our planning method and illustrates the techniques by means of examples taken from this crisis management domain. In Section 5, we shortly report on the implementation and the lessons learned from this experiment. Section 6 is devoted to related work and finally we conclude with some remarks in Section 7.

2 Formal Framework
The Logical Language: The semantics of our planning approach is based on a many-sorted first-order logic. The logical language $L = (Z, R, R, C, V)$ consists of a finite set $Z$ of sort symbols, $Z$-indexed families of finite disjoint sets of rigid and flexible relation symbols ($R$ and $R$, resp.), a $Z$-indexed family $C$ of disjoint sets of constants, and a $Z$-indexed family of disjoint sets of variables. Formulae over $L$ are built as usual.

The formal planning language $P$ is obtained by extending $L$ by $O, T,$ and $E$. $O$ and $T$ are $Z$-indexed families of finite disjoint sets of operator and task symbols, respectively. For all $\in Z$, the sets $R_{x}, R_{y}, O_{x}$, and $T_{x}$ are supposed to be mutually disjoint. $E$ denotes a $Z$-indexed family of so-called elementary operation symbols. It provides for each flexible relation symbol $R$ a so-called add-operation $\text{+}R$ and a delete-operation $\text{–}R$. As for the semantics, we adopt some essential features of the planning formalisms introduced in (Stephan and Biundo 1993) and (Stephan and Biundo 1996),

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
which are based on programming and temporal logics. Following a state-based planning approach, we use operators and tasks to take us from one state to another. The flexible symbols provided by our planning language are used to express the changes caused by these state transitions. Consequently, we introduce states as interpretations of the flexible symbols.

States and State Transitions: For a logical language L a model denotes a structure \( M = (D, S, I) \), where \( D \) is a Z-indexed family of carrier sets, \( S \) is a set of states, and \( I \) is a (state-independent) interpretation that assigns elements of the respective carrier sets to constants and a function resp. relation of appropriate type to each rigid symbol. As usual, sort preserving valuations \( \beta : V \to D \) are used for variables. Given a model \( M = (D, S, I) \), an atomic formula \( R(\tau_1, \ldots, \tau_n) \) is valid in a state \( s \in S \) under a valuation \( \beta \) denoted by \( s \models_{M, \beta} R(\tau_1, \ldots, \tau_n) \) according to the following definition.

For \( R \in R_v : s \models_{M, \beta} R(\tau_1, \ldots, \tau_n) \)
iff \( (I_1(\tau_1), \ldots, I_n(\tau_n)) \in I(R) \)

For \( R \in R_f : s \models_{M, \beta} R(\tau_1, \ldots, \tau_n) \)
iff \( (I_1(\tau_1), \ldots, I_n(\tau_n)) \in s(R) \)

Based on these definitions, validity of complex formulae is defined as usual. Now we are ready to turn from states to state transitions. To this end, we first assume that our models are natural ones (Stephan and Biundo 1993). This means, the carrier sets are supposed to be finite and we restrict the set of states to those, which assign finite relations to the symbols in \( R_f \). Furthermore, for each flexible symbol \( R \in R_f \), \( \exists \) states \( z \) such that \( \beta(z) \) is valid in \( s \) for \( R \). A function \( d : D_z \times \ldots \times D_z \to S \times S \) and \( a : D_z \times \ldots \times D_z \to S \times S \) are defined as follows. For a given model \( M = (D, S, I) \), a valuation \( \beta \), and a valuation \( \beta' \), a formula \( \varphi \) is invariant against an operator \((O(\bar{x}), prec, e))\) iff for all states \( s, s' \) with \((s, s') \models_{M, \beta} (O(\bar{x}), prec, e) \) \iff \( s \models_{M, \beta} prec \) \and \( s \models_{M, \beta} \varphi \) \and \( s' \models_{M, \beta} \varphi \). A formula post is generated by a sequence \( O_1 \ldots O_n \) of operators if it is generated by some \( O_i \) \( (1 \leq i \leq n) \) and is invariant against each \( O_j \) \( (i < j \leq n) \).

Tasks and Methods: Given a planning language \( P \), a task is a triple \((T(x), prec, post))\), where \( T \) is a task symbol, \( x = x_1 \ldots x_n \) is a list of variables, and \( prec \) and \( post \) are sub-formulae over \( L \). For a given model \( M = (D, S, I) \) and a valuation \( \beta \), the task transforms a state \( s \) into a state \( s' \), denoted by \((s, s') \models_{M, \beta} (T(x), prec, post))\) \iff \( s \models_{M, \beta} prec \) \and \( s' \models_{M, \beta} post \) \and \exists \text{ a finite sequence } s_1 \ldots s_n \text{ of states} \and \exists \text{ a finite sequence } O_1 \ldots O_{n-1} \text{ of operators}, \text{such that } s = s_1, s' = s_n, (s_i, s_{i+1}) \models_{M, \beta} O_i \text{ for all } 1 \leq i < n, \text{ and } O_{n-1} \models_{M, \beta} \text{ post} \rightarrow \text{ post}. \text{ The task generates a formula post'} \text{ if in addition } s_n \models_{M, \beta} \text{ post} \rightarrow \text{ post'}. \text{ The hierarchical structure of planning domains is reflected in two ways. First of all so-called methods are used to specify how an abstract task can be subdivided into a set of (primitive) sub-tasks, like it is usually done in HTN planning. Secondly, a hierarchy is imposed on the formulae that are used to express the pre- and postconditions of tasks. To this end, user-defined decomposition axioms of the form } \varphi \leftrightarrow (\exists \psi_1 \lor \ldots \lor \psi_m) \text{ specify how an abstract condition } \varphi \text{ can be refined into a more concrete one, each } \psi_i \text{ being a possibility to do so. A method } \{(T(x), prec, post), T\} \text{ is given by a task and a set} \text{ (primitive) sub-tasks, like it is usually done in HTN planning. Secondly, a hierarchy is imposed on the formulae that are used to express the pre- and postconditions of tasks. To this end, user-defined decomposition axioms of the form } \varphi \leftrightarrow (\exists \psi_1 \lor \ldots \lor \psi_m) \text{ specify how an abstract condition } \varphi \text{ can be refined into a more concrete one, each } \psi_i \text{ being a possibility to do so. A method } \{(T(x), prec, post), T\} \text{ is given by a task and a set}
\( \mathcal{T} \) of task sequences. For each such sequence \( t_1 \ldots t_n \) the \( t_i \) may be primitive or non-primitive. A method is called legal iff each task sequence \( t_1 \ldots t_n \in \mathcal{T} \) is a legal decomposition of the task.

For a given model \( M = (D, S, I) \) and a valuation \( \beta \) a task sequence \( t_1 \ldots t_n \) is a legal decomposition of a task \((T(x), \text{prec}, \text{post})\) iff the task sequence transforms a state \( s \) into \( s' \) such that for the precondition \( \text{prec}' \) of task \( t_1 \):

\[
\forall \mathcal{I} \in \mathcal{M}, \beta \quad s' \equiv M, \beta \quad \text{prec}' \rightarrow \text{prec}
\]

and the sequence \( t_1 \ldots t_n \) generates a formula \( \text{post}' \) such that \( s' \equiv M, \beta \quad \text{post}' \rightarrow \text{post} \).

The above mentioned axioms are thereby used to justify legality of decompositions when specifying methods. Illegal decompositions can be detected during construction of the domain.

### 3 The Application

Our planning domain is crisis management as being provided by organizations like THW. The German *Technisches Hilfswerk* is a governmental disaster relief organization that provides technical assistance at home as well as humanitarian aid abroad. Their mission within the flood disaster at the river “Oder” in July 1997 is used in our ongoing project to build a first realistic domain model for the planner. In the following, examples from this domain will be used to demonstrate our approach. The tasks of the THW are rich and widespread, they cover all aspects of crisis management, ranging from first measures after a hazardous event to long term supplies after clearing some disaster area. Therefore, Figure 1 only shows a relative small part of the complex task hierarchy, and most task networks are depicted as single, more “self-explanatory” actions.

The most abstract task is named *flood-disaster*. It comprises a management and communication task to determine which areas are endangered to what level. Furthermore, the logistics and supplies have to be installed, e.g. quarters for the relievers to be set up. The evacuation and the securing of the embankment are the most crucial sub-tasks in reality, and during all the activities, the relievers have to clear the area continuously, i.e. to check damaged buildings, to remove perished animals, etc.

In our examples we will focus on the evacuation task, which consists of two sub-tasks: one informs the population about the relief measures, the second brings people to safe areas. The methods for the latter define two expansions, depending on the initial situation. The relievers can help the people to leave the endangered area by themselves, or the circumstances might require the population to be moved by the THW.

#### 4 Combining Hierarchical Action- and State-based Planning

Our planning approach flexibly combines classical HTN and classical state-based partial order causal-link (POCL) planning relying on the formal framework introduced in Section 2. The prototype implements a simple top-level algorithm comparable to (Russell and Norvig 1995, p. 374), which is basically a classical nonlinear planning algorithm with decomposition of abstract tasks as an additional plan modification step. Although it looks very similar to those used by existing hybrid planning systems (see Section 6), we will use it to outline the underlying principles in the subroutines. First of all, we will focus on the closing of open preconditions.

In order to enable the planner to reason about the plans’ causal structures and dependencies at all levels of abstraction, complex tasks do carry preconditions and effects like the primitive operators do. For the time being they are assumed to be conjunctions of positive and negative literals.

While the relation between abstract and primitive tasks is given by a number of methods as in classical HTN planning, in our approach relations between the respective preconditions and effects are specified by the decomposition axioms. The example in Figure 2 shows two methods for the expansion of the abstract transport task in the evacuation context. The ordering constraints represent all possible sequences of sub-tasks, sort information for the variables is assumed to be given in the task definitions.

One of the decomposition axioms that will be applied in the respective expansion steps, will e.g. look like this (assuming the intuitive sub-sort relationships):

\[
\text{At(\text{Unit } u, \text{Location } l)} \leftrightarrow \\
\quad \left[ \text{Standing-at(\text{Vehicle } u, \text{Location } l, \text{Road } r)} \lor \ldots \lor \right. \\
\quad \text{Aircraft-at(\text{Aircraft } u, \text{Location } l, \text{Height } h)} \lor \right. \\
\quad \left. \text{At(\text{Container } c, \text{Location } l)} \land \text{In(\text{Container } c, \text{Unit } u)} \right]
\]

The specified decomposition axioms together with the sort and sub-sort definitions, represent a hierarchy on the relations and objects in the domain. We can make use of this...
knowledge when closing open preconditions with tasks on different levels of abstraction. When some (possibly abstract) effect of a task is needed to establish the precondition of another, the planner can provide this by choosing some -- according to the decomposition axioms-- suitable tasks in the partial plan to close the open condition. We then add causal links like in classical POC1 planning to represent causality. If no establisher can be identified, the planner introduces a suitable establisher for the open condition from the domain description. Figure 3 shows the planning process in such a situation.

Figure 3: Closing an open precondition along the condition hierarchy.

Figure 2: Example for a method definition
equivalent conditions. Our solution lies in the decomposition axioms, according to which we distribute the abstract effects and conditions among the tasks of the expanded network. This means, the decomposition axioms are used to inherit causal links from an abstract level to a more concrete one. Figure 4 shows the result: the passengers are boarded on some vehicle, driven to the camp and then unboarded again. The more abstract link carried the \( \text{At} \) relation between a vehicle \( ?u \) and the location \( \text{area4} \). The decomposition axioms justify a specialization of this link into \( \text{Standing-at} \) for vehicles. Please note, that the vehicle boarding task may carry the same precondition, which leads to a second inherited causal link derived from the abstract causal relation. Furthermore, when the abstract \( \text{move} \) task is specialized, it has to be checked against the decomposition axioms, whether the newly introduced causal links are inheritable or not. If not, the plan has to be considered inconsistent.

Now we will discuss threat handling, where the system can make use of the decomposition axioms like it did for condition establishment. Conflicts can be detected and resolved between arbitrary expansion levels, as the negation of an abstract condition implies the negation of every concrete one specified in the decomposition axioms, and a negation of one of the concrete conditions threatens the abstract one. This mechanism guarantees correct solutions when using the standard POCCL conflict resolution strategies at any time the algorithm chooses to check for threats. In addition, besides orderings and non-codesignation, expansion becomes a reasonable threat resolution mechanism. Due to a finer granularity, the conflicting effect might turn out to be harmless at the primitive operator level where the conflicting tasks may overlap.

The example in Figure 4 shows the expansion of a support procedure with a second movement task, and how it interferes with an established condition. The light line indicates the conflict at an abstract level, the dark one does so for the more concrete link. The negated effect can be specialized in a way that makes the plan inconsistent. A simple non-codesignation or some ordering constraint can solve this situation, for a conflict involving several layers of abstraction.

5 Implementation

We implemented a first prototype of the planning system in Java. It integrates task decomposition with state-based planning techniques for conflict handling and closing of preconditions. The main algorithm performs non-deterministic steps in a least commitment fashion according to a very simple strategy: It first tries to resolve threats in the current plan, then to close open preconditions and to bind variables, and with least priority expands complex tasks into task networks. This procedure is able to perform a systematic planning strategy. As an additional choice point in the POCCL steps, we added the expansion of the respective task: the planner can close open preconditions by establishing an appropriate causal link, by inserting a suitable task, and by expanding some of the abstract tasks (which may introduce suitable sub-tasks), etc. While doing so, the system prefers to establish causal links at the current level of abstraction, to bind variables (if necessary) as abstract as possible, and to resolve threats with the classical procedures. With that strategy, a plan is mainly developed at the most abstract level.

The forthcoming version of the system will use a more flexible top level routine which determines the appropriate sequence of plan manipulating steps by analyzing the visited and expected plan space, thereby projecting causal interactions. As already argued above, it seems very useful to concentrate expansion within the conflicting tasks in order to rule out inconsistent solutions at a level as early as possible.

To increase the system’s performance, we use a conservative algorithm for manipulating a global plan structure representing the expanded networks. By doing so, we have the additional advantage of automatically bookkeeping the performed expansion steps as well as all other choices made.
by the algorithm (cf. decomposition links in (Young, Pol-
lack, and Moore 1994)). When looking for an appropriate
effect to close an abstract precondition, the system can eas-
ily inspect already expanded abstract tasks and follow their
decomposition to less abstract levels.

The algorithm allows recursive task expansion schemata
to model loops. If for every recursive task a terminating
method is provided and if an appropriate search algorithm
is used (here e.g. iterated deepening) then the recursion is
harmless with respect to program termination and “increas-
ing the incompleteness” of the planning process. These
loops are very useful in the presented examples, e.g. evacu-
ation has to be performed “until all persons are safe”.

6 Discussion and Related Work
Hierarchical task network (HTN) planning as described and
analyzed in (Erol 1995) is the basis for systems like O-Plan,
UMCP and Shop. In contrast to our approach, which makes
use of state abstractions in condition achievement, abstract
tasks in O-Plan (Currie and Tate 1991) do not carry pre-
conditions and effects. Instead, the system relates condi-
tions of primitive operators over different levels in the plan
generation process by introducing condition types in the ab-
stract expansion schemes (Tate, Drabble, and Dalton 1994).
These types specify how conditions of the tasks in the ex-
pansion can be achieved: by the effect of a task that is (a)
inside or outside the current expansion and (b) introduced
at the current plan generation level, above, or below. This
technique requires the domain encoder to structure the task
hierarchy very carefully as its pruning affects the system’s
search space structure than the structure of the plan space.
Compared to O-Plan, UMCP (Erol, Hendler, and Nau 1994)
is a much more puristic implementation. The search space is
constraint pruned, down to the most concrete operator level,
where the typical conditions are introduced. Both systems
merely rely on action abstraction.

A new direction in the HTN paradigm is given by the
Shop system (Nau et al. 1999), that proposes ordered task
decomposition, using if-then-else cascades in method selec-
tion. The main idea is to plan all tasks in the order they
are later executed. This enables the system to deduce com-
plete state descriptions, beginning with the initial state. The
developers met the criticism on their linearity assumption
with a modified system, called M-Shop (Nau et al. 2000)
which can handle planning problems with parallel goals in
the initial task. Many realistic domains may meet this partial
linearity property, our crisis management domain however,
does not, because task execution itself is highly distributed
and the execution order for most tasks is not known in ad-

ance.

Planning using state abstraction was the earliest form of
hierarchical planning in linear planning systems. Nonethe-
less, the Abstrips system (Sacerdoti 1974) is still discussed
(Giunchiglia 1997), and has influenced many modern plan-
ners, because this kind of abstraction does not work at the
control level and can therefore be easily combined with
other search techniques and heuristics. Classical state ab-
straction works by deleting certain sets of preconditions,
thereby defining criticality levels for each of which the sys-
tem plans in a classical manner. Alpine (Knoblock 1994) au-
tomatically generates these levels, building abstraction hier-
archies with ordered monotonicity, i.e. detailed action levels
do not interfere with more abstract established conditions.
Similar work in the context of nonlinear planning has been
done by Yang in the Abtweak planner (Yang, Tenenberg, and

Exploiting object hierarchies for state abstraction is a
comparatively new technique in planning, an example be-
ing object centered planning (McCluskey, Kitchin, and Por-
teous 1996). In this paradigm objects are organized in static
and dynamic sorts, and each instance of a dynamic sort has
its own local state which is defined by a set of predicates.
Consequently, predicates are owned by exactly one sort, the
key attribute of the predicate, thereby becoming static or dy-
namic themselves. For all sorts legal local states are speci-

ymmetric, and transitions over these legal states are the basis for
operator definitions. OCLh (McCluskey 2000) extends this
formalism to action abstraction by introducing a sort hierar-
chy, in which dynamic predicates are inherited from super-
sorts. So-called guards play the role of pre and postcondi-
tions of objects transition sequences that build the semantics
for abstract tasks. The planning algorithm in this framework
repeats an expand then make-sound cycle: after expanding
one level of task networks, the system is checking for incon-
sistencies and repairing them. Although our state abstraction
is similar, we can handle the refinement of objects and pred-
icates, likewise, and are not restricted to a fixed planning
strategy.

Integrating state-based nonlinear planning capabilities in
the fashion of e.g. UCPop (McAllester and Rosenblitt 1991),
into an action abstracting system promises many advantages.
As pointed out in (Estlin, Chien, and Wang 1997) it adds
the strengths of both, at the same time softening their weak
points. This is reflected in the modeling process: task net-
works more naturally represent hierarchy and modularity
and enable the user to represent domains in an object ori-
ented form which is easier to write and reason about. De-
composition rules can refer to either low- or high-level forms
of a particular object or goal, as the information pertaining
to specific entities is contained in smaller, more specialized
rules. The drawback of this technique is that inter-modular
constraints (Estlin, Chien, and Wang 1997), i.e. exceptions
or special cases in action execution, cannot be represented
adequately, which often leads to overly-specified reduction
rules. This can be seen in the example in Figure 3, where
classical hierarchical planners would introduce expansion
schemata for every kind of support task to be ordered be-
fore the evacuation. Operator-based techniques on the other
hand help encoding implicit constraints, as their kind of plan
refinement is more general and provides more compact rep-
resentations. In addition, it provides an early detection of
inconsistencies at an abstract level, together with means of
resolving the conflicts. But using solely operators, certain
aspects are difficult to represent (for a discussion about the
expressive power of HTN planning, see (Erol 1995)). The
advantages of a natural mixed domain knowledge represent-
a obvious, although difficult to evaluate quantita-
Such hybrid systems had been watched suspiciously a long time, because the planning paradigms were considered to be conflictive. New AI textbooks present this approach in the style of state abstraction planning in (Yang, Tenen-berg, and Woods 1996), i.e. the abstract tasks carry preconditions and effects from a subset of the less abstract tasks. Yang suggests in (Yang 1997) to keep hierarchical models restricted in such a way, that in every reduction schema there is one task carrying the main effects of the network and hence those of the associated abstract task. In such domains the downward solution property (all consistent abstract solutions can be refined into consistent primitive solutions) holds as a basis for effective search space reduction. A similar approach is presented by Russell and Norvig (Russell and Norvig 1995), who allow distribution of conjuncts of conditions among the sub-tasks of the network. One of the very few existing systems is DPOCL (Young, Pollack, and Moore 1994). It decomposes abstract tasks into networks with additional initial and final steps which carry the conditions of the abstract tasks. Some of the techniques used there raise the crucial question of user intent. The system prunes unused steps and takes condition establishers from every level of abstraction, even from sub-tasks of potential establishers. The problem of when to insert new tasks, and where to use decomposition rules only, is very hard to solve, as it depends in part on the modeller’s intention. So far, we have provided our system with a switch for explicitly not inserting new tasks in precondition achievement, as well as an output, indicating the inserted tasks. Moreover, premature insertion of new tasks may lead to non-optimal short plans, but we postpone this problem for this time as a matter of “good” search strategies, like it is solved for classical state-based nonlinear planners –but of course it will be tackled in the future.

Closely related to our approach is the work of Kambhampati (Kambhampati, Mali, and Srivastava 1998). It integrates HTN planning in a general framework for refinement planning, thereby making use of operator-based techniques. In this view, the algorithm uses reduction schemes where available, and primitive actions otherwise. Causal interaction is analyzed also at the abstract level, and refined by mapping conditions and effects of abstract tasks on conditions and effects in their sub-tasks. Abstract conditions are closed by phantom establishers that are identified at a later stage, while our algorithm just postpones such steps if no suitable task is less abstract enough. Conflict detection and resolution can only be done at the primitive level, as in contrast to our methodology, there is no “vertical” link between causalities in the different levels of abstraction. Kambhampati addresses user intent by defining a subset of abstract effects explicitly for condition establishment, and by explicitly representing the incompleteness of scheme definitions. For the latter, a specific predicate prevents insertion of new steps.

7 Conclusions and Future Work

We have introduced a planning approach that integrates hierarchical task network and state-based POCL planning techniques by imposing hierarchical structures on both tasks and state descriptions. Tasks on all abstraction levels are extended by pre- and postconditions, which enable the flexible integration of hierarchical decomposition and nonlinear planning. Decomposition axioms allow for a consistent interleaving of the different types of planning steps in this context. A planning system has been presented, which implements this integrative planning approach. It will be used to flexibly generate mission plans for environmental disaster situations.

Future work will start with the implementation of a more flexible search strategy. Furthermore, the example domain strongly demands resource reasoning, especially time, and a specialized loop mechanism.

References


