Splitting Multiple Situations in Conditional Planning

Alfredo Milani

Dipartimento di Matematica, Università di Perugia
Via Vanvitelli 1
Perugia, Italy 06100
milani@gauss.dipmat.unipg.it

Abstract

Most current conditional planners are based on explicit representation of alternative outcomes of conditional actions. The explicit representation of possible outcomes induces a split in the search space which rapidly exponentially grows.

A general strategy of *minimal splitting of multiple situationl,* for conditional planners, is introduced in this paper, in order to reduce the amount of computational resources required to manage conditional branches.

The strategy is based on the definition of conditional formulas, they allow to represent set of situations avoiding explicit combinatorial representation of them.

Conditional formulas are used in order to keep the search space and plan representation as small as possible while splitting on conditional branches is avoided if not pertinent to the problem at hand.

The proposed strategy also minimises the use of observing and sensing operations during plan execution monitoring.

Introduction

The conditional planning approach modifies the classical planning scheme in order to take into account of interaction with the environment and execution monitoring phase.

Conditional planners do not avoid replanning at all, but uncertainty is taken into account inside the plan itself and different outcomes of actions can due to a different development of the same plan.

In some sense conditional planners are able to generate *robust plans* , between the extreme cases of *universal planning* (Schoppers 1987) and *reactive planning* (Kaelbling1987), in the first case all possible contingencies are conditionally considered, in the second one the plan is not proper conditional but each low level action is conditionally executed by monitoring conditions which trigger a rective behaviour.

This research has been partially supported by Special Project "Pianificazione Automatica" of C.N.R., Italian National Research Council.

Conditional planners can solve problems in environments where the sources of uncertainty can be predicted (for example: "Look for Mary, I know that she is in this building, but I don't know in which room"). They can plan in advance for possible future outcomes of actions which can due to invalid executions (for example: "I'm glad that I remembered to bring an extra key" (Peot&Smith 1992).

Generating plans with complex control structures, as iteration loops and recursions, requires to face the topic of conditional actions (*Recursive planning* introduced in (Sani & Steel 1991)).

Conditional plans can drive monitoring systems in focusing on those aspects which really condition plan execution (for example: "Watch the road while driving!"); finally there are problems for which only conditional solutions exist, that is not all the developments of the plan due to a solution (for example: "Toss a coin in order to obtain tail", "Find a job").

Conditional Planning and Explicit Representation

It is worth to underline some representational aspects common to the former conditional planners and planning models. From the point of view of action representation those planners do not say much new with respect to*Warplan-C* , (Warren 1976)(Warren 1990) one of the first conditional planners: representation of conditional actions in all those planners require to *explicitely represent the set of their different outcomes.*

(Brewka & Hertzberg 1990) proposed a planning scheme where a conditional action is represented by a list of mutually disjoint preconditions, each precondition has a list of mutually disjoint postconditions (each pre- or post-condition being a conjunctive formula). In (Sani & Steel 1991) preconditions of case nodes (which represent conditionals) are allowed to be disjunctive formulas, each element of disjunction representing a precondition for a branch, while effects are unique for each branch. When a case node is executed the branch which have verified preconditions is chosen. In CNLP (Peot & Smith 1992) an extension of Strips (Fikes&Nillson 1971) operator is introduced to take into account of conditional outcomes

with mutually disjoint set of postconditions; dependency and conditioning links are use to represent the conditional structure of plan .

In her thesis (Antognoni 1992) proposes a conditional nonlinear planner, where outcomes of conditional actions are explicitly represented. When a conditional action is included in a plan, then a set of actions, called *action reductions*, are added to the plan, one action reduction is added for each different outcome of the original conditional action. A plan is then a *nonlinear set of action reductions*.

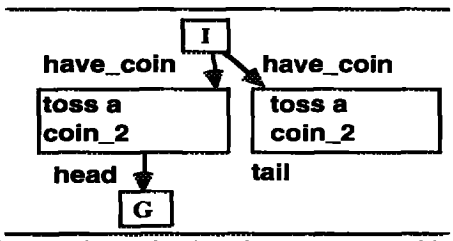

Fig.1 Action reductions in (Antognoni 1992).

Each action in the plan is labelled with the set of action reductions on which its executions *depends*, this represent conditional branches in a way somewhat equivalent to (Peot & Smith 1992).

Unfortunately when it is required a conditional action to represent all its mutually disjoint outcomes, the length of action representation rapidly exponentially grows, and similarly it grows up the number of contexts the conditional planner have to analyse.

Another consideration can be done for uncertain environments: not all the uncertainty contained in an action execution could be pertinent to the problem at hand (consider for example: "I'm not sure to win this tennis match, but I just need to do some sport, then my goal is satisfied and I don't care about my uncertainty").

In this work we argue that action representation should be as compact as possible, in order to keep plan smaller and manageable. But it is also required to have a flexible representation which possibly allows to obtain the detailed explicit representation. The need of an explicit representation can arise dinamically by the current problem under development (consider for example: "I'm not sure to win this tennis match, but my goal is to win the one million dollar prize in order to pay my hotel room, so it is better that I explicitely consider the possiblity of not winning").

In the following we define a compact and manageable representation for conditional action and then a flexible strategy for detailing (i.e.*splitting*)contexts when needed by the problem under development.

Conditional Formulas and Multiple Situations

Our planning model is based on the concept of *conditional formula* and *alt* operator (Milani 1994). Let's introduce some definitions in order to give the semantic of *alt* operator, the definitions are based on assignments.

Definition: Assignment. An *assignment*, for a formula is a set of literals (affirmed or negated ground predicate).
If a literal p (p can appear or not in the formula) is not in the assignment we say that p is *unknown* in that assignment.
Definition: Consistent assignment. A *consistent assignment* is an assignment in which the same literals does not appear negated and affirmed at the same time.

Definition: Atomic formula. An *atomic formula* is a literal, (an affirmed or negated ground predicate), it denotes a set of assignments with the literal assignment as only element.
Example:p , atomic predicate, denotes the single assignment *{{p}}*;
not p where p is an atomic predicate, denotes the single assignment *{{not p}}*.
Note that negation can only be applied to atomic formula, this is not a real limitation for most planning domains.

Definition : "alt " operator. The *alt* operator (*alt* stands for *alt*ernate) applied to two formulas p and q denotes, by definition, the minimal consistent sets of truth assignments which separately verifies p and q . If p and q are formulas which respectively denote the set of assignments P and Q then the set of assignments denoted by p ***alt*** q is $P \cup Q$ .We equivalently say, by definition, that p *alt* q is *verified by* each assignments in $P \cup Q$
Example:p ***alt*** q denotes the assignments *{{p}, {q}}*
p ***alt not*** p denotes the assignments *{{p}, {not p}}*

Definition: "and " operator. Let $P=\{P_1,...,P_m\}$ the assignments which verify p, and let $Q=\{Q_1,...,Q_n\}$ the assignments which verify q then
p *and* q is verified by assignments $R=\{R_1,...,R_{mxn}\}$ each R_k such that $R_k = P_i \cup Q_j$ for each $[P_i,Q_j]$ in P x Q, if each R_k is consistent, the whole formula is consistent, otherwise it is inconsistent.
Example: if p and q are literals then
p ***and*** q denotes the set {p,q}
p ***and not*** p denotes the inconsistent assignment

Definition: Conditional formula. A *conditional formula* is an atomic formula, or it is built by compounding atomic formulas or conditional formula by *alt* and *and* connectives. A conditional formula denotes a set of assignments.

PO. Properties of "and"
And operator has the usual properties of logical *and*:
p **and** p = p;
p **and** q = q **and** p;
(p **and** q) **and** r = p **and** (q **and** r);
p **and not** is p false, i.e. denote an inconsistent assignment
The proofs of the properties, not given here, are based on showing that equivalent formulas denotes the same sets of assignments.

P1. Alt Idempotent: p **alt** p = p
(the situations where *p* is alternative to *p* reduce to only one in which *p* is verified).
P2. Associativity: (p **alt** q) **alt** s = p **alt** (q **alt** s)
P3.Commutativity: p **alt** q = q **alt** p
P4.And/alt distribution
(p **alt** q) **and** s=(p **and** s) **alt** (q **and** s)

P5. Normal Conditional Form
Every conditional formula can be reduced to a *Normal Conditional Form (NCF)*:
A_1 **alt** A_2 **alt** ... **alt** A_n
where A_i are conjunctive formulas without ***alt*** operators.

The NCF, explicitly represents all the situations described by a formula, the NCF of a given formula can be computed by iterating application of property P0-P4. Each A_i term represents a situation.

For example the following formula:

*(open(door) **alt not** open(door)) **and** (empty(room) **alt not** empty(room))*

can be reduced to the following NCF which represents (more explicitly) the same set of situations:

*((open(door) **and** empty(room)) **alt***
*((open(door)**and not** empty(room)) **alt***
*((**not** open(door) **and** empty(room)) **alt***
*((**not** open(door) **and not** empty(room))*

It is easy to see that the number of terms in a NCF it is worst-case exponential with the number of ***alt*** operators which appear in the original formula.

Representing Conditional Actions and Multiple Situations

The alternate operator is a powerful expressive tool for *representing sets of alternative situations.* It avoids most of problems which arise with misuse of or-disjunction in planning model. A set of planning situations can be easily represented by a conditional formula. On the same basis conditional actions can be defined.
Details about the complete planning model can be found in (Antognoni, Marcugini & Milani 1993), while in (Milani 1994) a semantic of action execution is given.

Definition: State
A *state* (or a *multiple situation*) in conditional planning is a set of situations. A state can then be represented by a conditional formula.

Definition: Conditional action
A *conditional action* is represented by two conditional formulas which describe *preconditions* and *effects* (the set of situations which allow action execution and the set of situations determined by action execution), and a *context* for execution, which describes conditions over other actions in order to execute the current one.

Conditional actions have multiple alternative situations as preconditions, see (Sani & Steel 1991) and (Brewka& Hertzberg 1990) and have multiple situations as effects.

Splitting Multiple Situations

Representing a conditional plan by mean of conditional formulas and conditional actions adds great flexibility to a planning system. Each conditional action has not a unique representation but it can be represented by a wide number of equivalent set of action reductions.

Let suppose to have the following dummy conditional action *A* with null preconditions and effects represented by c. formula *F= (p **alt** q **alt** s **alt** t)* then action A can be represented for example in the different form as shown in figure 2.

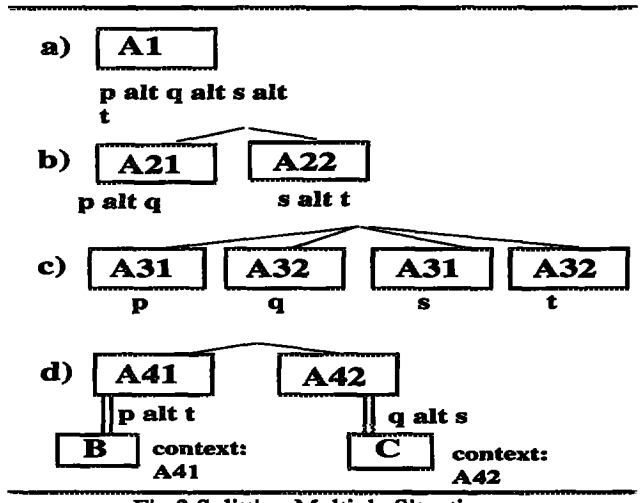

Fig.2 Splitting Multiple Situations.

The first form *a)* is the most compact form, the full-detailed representation is shown in *c)* , a possible intermediate form is *b)* . The boxed letters represents *actions reductions,* that is different contexts after action A execution (in our model set of action reductions derived from the same original action are managed as a whole with respect to precedence and dependency constraints).

It is easy to see that the set {A1} {A21,A22} and {A31,A32,A33,A34} denotes the same set of situations, since each reduction embed, in the conditional formula representing its effects, a subset of situations of the compact multiple situation represented by F.

The three representations are not equivalent from the computational point of view, because in the full detailed representation *c)* , the planner has to represent and to analyse four different contexts.

On the other hand no representation is better than another, since, for example, the most compact form *a)* does not suffice to represent a *conditional branch* (or equivalently a *conditional dependency*) as in fig.2*d)* , the figure represents that action B is executed only in case A41, and action C is executed only in context of effects A42.

Then it is necessary to determine *when* detailing (or *splitting situations*) is needed and *how* to split, since there is not unique way to split a multiple situation.

Most conditional planners and conditional planning models (Sani& Steel 1991) (Brewka & Hertzberg 1991) (Peot & Smith 1992) (Antognoni 1992) require all the resulting states of an action to be explicitly represented.
In those models, the search space is splitted in a number of alternative spaces *since representation phase*, then the full-detailed form is necessarily adopted.

The basic point of the minimal splitting technique, which we propose, is flexibility during plan development. The technique is based on representing sets of situations in the *most compact* form by conditional formulas, and on generating the split of the representation only if needed.

Minimal Splitting

Splitting the search space and observing the outcome of an action is necessary if that outcome gives a (positive or negative) contribution to the plan. On the other hand it is not necessary to observe an alternative outcome of a conditional action, if that outcome does not give any contribution to the plan.

On the basis of this idea we develop a planning strategy for our conditional planner, that is called *minimal splitting*. *Action splitting* is the operation which substitutes an action in the plan with an equivalent set of *action reductions*.

A *minimal splitting* substitutes intermediate action reductions which can be further splitted to the lowest level. The splitting of a conditional action in its possible outcomes (or *action reductions*) will be:

- *dynamical*, i.e. it take place during the planning phase;
- *delayed* until needed, i.e. the splitting is made *if needed* to solve the problem;
- *partial*, the minimal required detail level will be generated.

Definition: Splitting Condition
An action A is *candidate to splitting* if a predicate p in its effects meets the following conditions:
i) p is a *clobberer*[1] of an other action B, or
p is an *estabilisher* of an other action B; and
ii) p appear under the scope of an *alt* operator (it appear somewhere inside either the operands of *alt*)
In other words p is required to have a positive or negative *role* in the plan and to be *uncertain* .

Definition: Minimal Action Splitting
If an action A is a candidate to splitting by verified splitting conditions on predicate p and
i) $Effects(A) = x(M \ alt \ N)y$ where x and y are pre/postfix of the conditional formula which represent the effects, and
ii) $(M \ alt \ N)$ it is the outmost *alt* operator under which scope p is;
then action A *splits* into two (or more) alternative actions Ai, recursively obtained as following:

a) create actions A_1, A_2 with
$Preconditions(A1)=Preconditions(A)$
$Preconditions(A2)=Preconditions(A)$
$Effects(A1)=x(M)y$ $Effects(A2)=x(N)y$
b) let actions $A1$ and $A2$ substitute action A in the plan, make they belong to the same class of reductions, see (Antognoni 1992) and make they inherited all precedences and dependencies of A.

If conditions for splitting are still verified for some A_i , iterate this same minimal splitting procedure i)-ii) until p is no more under the scope of some *alt* operator.
It can be proved the following property for the Minimal Action Splitting.

P6. Equivalence of Splitting.
The set of action reductions obtained by iteration of Minimal Action Splitting from a given action reduction is *equivalent* to the initial one, that is they denote the same multiple situation denoted by the given action reduction.

Similarly a plan P' obtained from plan P by iterating the Minimal Action Splitting is *equivalent* to P.

Planning Strategy

The plan generation steps proceed with the usual phases of *estabilishing goals* and *conflicts removal* ,(Chapman 1987), extended with the use of contexts (Peoth& Smith 1992) (Antognoni 1992).

Initially action instances are added to the plan by the planner with their complete effects descriptions (i.e. compact conditional formula). Actions representations are splitted only when *splitting conditions* are met.

When an action is splitted it is substituted in the plan by an equivalent set of action reductions which inherited all action attributes. In this case new alternative contexts are generated, the contexts corresponding to the different outcomes. The generated contexts have to be solved and completed with the usual nonlinear conditional planning strategy.

The resulting final plan will show an intermediate form between full splitted and compact one.

Splitting and Observing Actions

Any conditional planner assumes an underlying *plan monitoring system* capable of *distinguishing contexts*, in order to decide which action has to be executed next or which *conditional branch* is under execution. The observing task can involve positioning of sensors and complex inferences from sensors data; avoiding useless observations can therefore result in relevant resources saving.

The underlying plan monitoring task is reflected by the structure of splitting and contexts in the final plan. Each splitting requires an, explicit or implicit, *observe action* in order to establish which possible outcome has taken place. Observe actions (Peoth & Smith 1992) are actions which

allow the executor to know the outcome of an action and to decide which is the current context, so that it is able to determine feasibility of next action execution according to the conditional structure of the plan.

It must be noted that the needs of observe actions is not inherently required by a conditional action but it is certainly inherent to a splitting as it has been defined. The uncertainty contained in an action description can be inessential for solving a certain problem (in that case no observations are needed) while it can be relevant for a different problem (in this latter case observations and contexts dependencies are needed).

Although there exists an equivalence between a conditional action and the set of splitted actions it generates (in fact a *class of reductions* (Antognoni 1992) globally describes exactly the same preconditions and effects situations of the unsplitted conditional action) there is an important difference in the way they reflect the underlying plan monitoring task, and they lead to plans which require different computational resources in order to be developed and monitored. The strategy of mininal splitting of multiple situations minimises the number of observations needed to execute a plan since it avoids splitting if not needed.

Example.

Let suppose to have the following action descriptions:

Action: throw_a_dice
Preconditions: have_dice, hand_free
Effects: (face(6) **alt** face(5) **alt** face(4) **alt** face(3) **alt** face(2) **alt** face(1)) **and** (noise **alt not** noise)

Action: clap_hands
Preconditions: hand_free
Effects: noise

The first given description captures the facts that *throwing a dice* will cause one value on his top face and it can (or cannot) cause some noise (i.e. because it hit a hard surface).

Let us suppose we have the goal that someone pays attention to us, and we know that making some *noise* is a way to reach the goal. Here we make no hypotheses about planner goal selection or action selection strategy.

Let then suppose the planner builds an initial plan generating an instance of the action *throw_a_dice.*

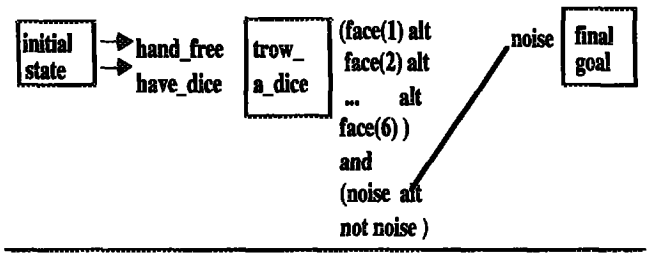

Fig.3 Initial partial plan

In this problem we need to separately analyse the case we succeeded in making the *noise* by *throwing a dice* and the case we observe we do not succeed, in this latter one we need to plan an alternative.

As shown in figure 3, action *throw_a_dice* meets conditions for splitting and the two action reduction **throw_a_dice1** and **throw_a_dice2** are generated by Minimal Splitting strategy.

The planning task can now proceeds on the two *alternative contexts*, giving a solution as shown in figure 4.

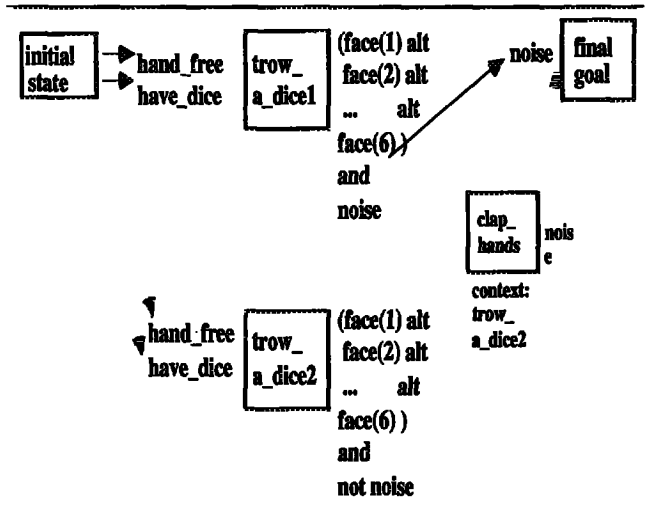

Fig.4 Minimal Splitting Posting by *noise* predicate.

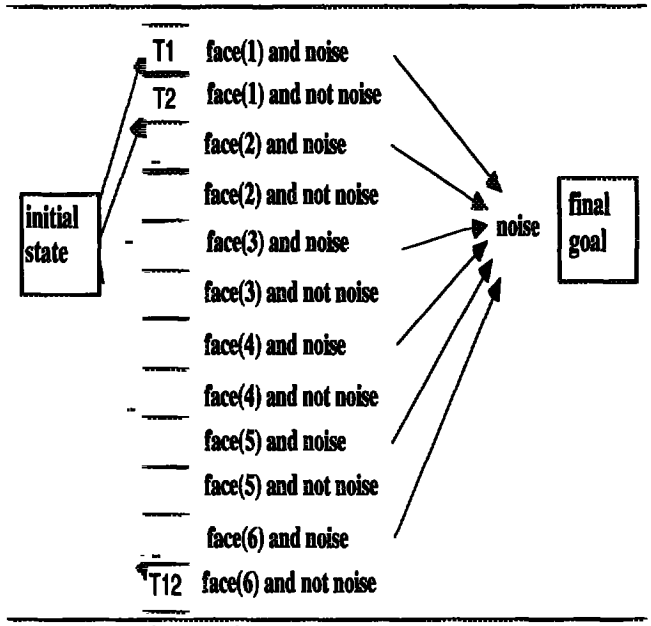

Fig.5 Splitted action representation.

It must be noted that the technique of *splittinng multiple situation* with representation based on conditional formulas keep actions representation small and it avoids:

-*useless splittings*, for example the "explicit combinatorial" representation of action *throw_a_dice* would require to represent twelve different situations;
-*useless observe operations*, the solution to our problem does not require to distinguish between the twelve possible situations after action *throw_a_dice* (see fig.5); it is necessary to distinguish only two outcomes (*noise alt not noise*).

In this case the plan execution monitoring system can focus on observing those outcomes and related low level actions (i.e. *sound sensing* in order to check *noise* predicate vs. *image sensing* in order to check *face()* predicate), in other words: we do not care about which value is on the *face* of the dice if we throw it in order to do some *noise*, then we better have to check if it really make *noise* when thrown.

Conclusions

Representing multiple situations with conditional formulas based on *alt* operator keeps action descriptions as small as possible, and it allows a policy of dynamical and minimal splitting of multiple situations.

Alternative outcomes which are not focused by the current problem can be kept apart, this avoids useless combinatorial explosion of the search space, those outcomes can be focused and can produce splitting in a different problem space with different problem goals.

Another advatege is that the minimal splitting of multiple situation avoids useless observe operations in order to discriminate contexts during execution phase.

Splitting the search space cannot be avoided at all and we expect this as natural in conditional planning. Since conditional actions bring with them a number of alternatives which cause a combinatorial explosion of the search space, basing on the idea that not all alternatives are of interest in all problem spaces we can reduce this explosion by:
-having a compact representation for multiple situations, i.e. *conditional formula*;
-starting splitting on multiple situations only when strictly necessary for the problem solution (clobbering or establishing), i.e. *minimal splitting*;
in this way we expect to manage the exact amount of combinatorial explosion inherent to the problem space.

References

Antognoni, G. 1992. Pianificazione di Azioni con Effetti Alternativi: un Modello di Rappresentazione", Tesi di Laurea, Dipartimento di Matematica, Università di Perugia, Perugia, Italy (1992)

Antognoni, G.; Milani, A.; Marcugini,S. 1993. Extending a Conditional Planning Model with Multiple Situations Management. Technical Report n.12 January 1993, Dipartimento di Matematica, Università di Perugia, Perugia, Italy

Brewka,G.; Hertzberg, J. 1990. How To Do Things with Worlds: On Formalizing Actions and Plans. Technical Report Tasso-report n.11, GMD Institute Bonn, Germany

Bylander, T. 1991. Complexity Results for Planning. In Proceedings of IJCAI-91, 274

Chapman, D. 1987. Planning for conjunctive goal, Artificial Intelligence n.32

Fikes,R.E.; Nillson,N.J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence n.2, 189

Ginsberg, M.L.; Smith,D.E. 1988. Reasoning About Action I: A Possible Worlds Approach Artificial Intelligence n. 35 , 165 (1988)

Hanks, S. 1990. Pratical Temporal Projection, In Proceedings of AAAI-90

Kaelbling, L.P. 1987. An Architecture for Intelligent Reactive Systems. In Proceedings of 1986 Workshop Reasoning About Actions and Plans, Timberline, OR, Morgan Kaufmann

Milani, A. 1994. A Representation for Multiple Situtations in Conditional Planning. In Current Trends in AI Planning. Backstrom C.; Sandewall, E. Eds. IOS Press, 226

Peot, M.A.; Smith, D.E. 1992. Conditional Nonlinear Planning. In Proceedings of the 1st Int.Conf.on A.I.Planning Systems, AIPS92, J.Hendler Ed., Morgan Kaufmann, 189

Sani, R.G.; Steel S. 1991. Recursive Plans, in Proceedings of the 1st European Workshop on Planning EWSP 1991, Sankt Augustin, Germany, Lecture Notes in Artificial Intelligence 522, Springer Verlag, 53

Schoppers, M.J. 1987. Universal Plans for Reactive Robots in Unpredictable Domains. In Proceedings of IJCAI-87 1039

Warren, D.H.D 1976. Generating Conditional Plans and Programs. In Proceedings of AISB-76 Summer Conference, Edinburgh, 277

Warren, D.H.D. 1990. Warplan: A System for Generating Plans. In Readings in Planning; J. Allen, J.Hendler, A. Tate ed., Morgan Kaufmann