Extending Planning Graphs to an ADL Subset

Jana Koehler, Bernhard Nebel, Jörg Hoffmann, and Yannis Dimopoulos

Institute for Computer Science
Albert Ludwigs University
Am Flughafen 17
79110 Freiburg, Germany
<last-name>@informatik.uni-freiburg.de

Abstract. We describe an extension of GRAPHPLAN to a subset of ADL that allows conditional and universally quantified effects in operators in such a way that almost all interesting properties of the original GRAPHPLAN algorithm are preserved.

1 Introduction

Planning with planning graphs [1] has received considerable attention recently. The impressive performance and in particular the theoretical properties such as soundness, completeness, generation of shortest plans, and termination on unsolvable problems motivated us to use the approach as the kernel algorithm for our own planner IP^2. But GRAPHPLAN also has its limitations. First, its performance can decrease dramatically if too much irrelevant information is contained in the specification of a planning task [7]. Second, its simple representation language is restricted to pure STRIPS operators – no conditional or universally quantified effects are allowed and it was unclear whether the underlying planning algorithm could be extended to more expressive formalisms [1, 3, 6].

name:	**move-briefcase**
par:	l_1:location, l_2:location
pre:	at-b(l_1)
eff:	ADD at-b(l_2), DEL at-b(l_1)
	$\forall x$:object [in(x) $\Rightarrow$ ADD at(x,l_2), DEL at(x,l_1)].

Fig. 1. Operator with conditional and universally quantified effects

In principle, sets of STRIPS operators can be used to encode conditional effects. For example, the *move* operator from the well-known Briefcase domain that specifies that all objects which are inside a briefcase move whenever the briefcase moves (Fig. 1) can be equivalently translated into a set of operators – one operator for each possible subset of objects, i.e., moving the empty briefcase, moving the briefcase with one object inside, with two etc. But such an encoding leads to exponentially more operators which can make even small planning problems practically intractable, see [5]. These observations motivated us to directly embed operators with conditional and universally quantified effects into planning graphs, while other features of ADL [8] that are e.g., available in UCPOP [9] can be reasonably handled by preprocessing [3].

2 A Semantics for Parallel ADL Plans

One of the distinguished features of GRAPHPLAN is its ability to produce *shortest* plans in the sense that it exploits maximal parallelism of actions in the plan – a property one would like to carry over to more expressive operators.

Definition 1. An *operator* is a 4-tuple consisting of

1. a *name*, which is a string,
2. a *parameter list* of typed variables,
3. the *precondition* φ_0, which is a conjunction of atoms, and
4. the *effect* as a conjunction of possibly universally quantified formulas that are of the form $\varphi_i \Rightarrow \alpha_i, \delta_i$ where φ_i is the so-called effect condition (limited to a set of atoms) and α_i, δ_i are the actual effects (also limited to sets of atoms with α_i being the Add effects and δ_i being the Del effects).

Note that as in GRAPHPLAN no explicit atomic negation is available in our language. Instead we model atomic negation by introducing an additional predicate *not-p(x)* if $\neg p(x)$ is needed. The universal quantifier may be absent and in the case of an unconditional effect $\varphi_i = \emptyset$ holds. An example of a valid operator is shown in Fig. 1. An example of a non-valid effect representation is $\varphi_i \Rightarrow \forall x : \text{ADD } p(x)$, because the effect condition lies outside the scope of the universal quantifier.

Definition 2. The set of all ground atoms is denoted with P. As usual, a *state* $S \subseteq P$ is a set of ground atoms.

Definition 3. An *action* o is a ground instance of an operator and has the form:[1]

$$\begin{aligned} o : \; & \varphi_0 \\ & \alpha_0, \delta_0; \\ & \varphi_1 \Rightarrow \alpha_1, \delta_1; \\ & \vdots \\ & \varphi_n \Rightarrow \alpha_n, \delta_n. \end{aligned}$$

[1] The ground instance of a universally quantified effect $\forall x\ [\varphi_i(x) \Rightarrow \alpha_i(x), \delta_i(x)]$ is the conjunction of ground instances $\bigwedge_{k=0}^{n}[\varphi_i([x/a_k]) \Rightarrow \alpha_i([x/a_k]), \delta_i([x/a_k])]$ if the domain $\text{DOM}(x)$ of the typed quantified variable x is $\{a_0, a_1, \ldots, a_n\}$.

The $\varphi_i, \alpha_i, \delta_i$ are sets of ground atoms with φ_0 denoting the preconditions of o, α_0 being the *Add* list and δ_0 being the so-called *Delete* list of o. A conditional effect contains the effect condition φ_i, Add list α_i and Delete list δ_i.

Figure 2 shows a ground instance of the *move* operator for DOM(*object*)= {*letter, toy*} and DOM(*location*) = {*office, home*}.

name:	**move-briefcase**	
par:	office, home: location	
pre:	at-b(office)	
eff:	$\emptyset$	$\Rightarrow$ ADD at-b(home), DEL at-b(office);
	in(letter)	$\Rightarrow$ ADD at(letter,home), DEL at(letter,office);
	in(toy)	$\Rightarrow$ ADD at(toy,home), DEL at(toy,office).

Fig. 2. A possible ground instance of the *move* operator

In the following, we define the result of applying a single action to a given state.

Definition 4. Let *Res* be a function from states and sequences of actions to states. The result of applying a single-action sequence $\langle o \rangle$ to a state S is defined as

$$Res(S, \langle o \rangle) = \begin{cases} \big(S \cup A(S,o)\big) \setminus D(S,o) & \text{if } \varphi_0 \subseteq S \\ \text{undefined} & \text{otherwise} \end{cases}$$

with

$$A(S,o) = \bigcup_{\varphi_i \subseteq S, i \geq 0} \alpha_i \qquad \text{and} \qquad D(S,o) = \bigcup_{\varphi_i \subseteq S, i \geq 0} \delta_i \ .$$

The result of applying a sequence of actions to a state is recursively defined as usual.

For a planning language of simple STRIPS actions, the definition of *Res* can be extended to a set of parallel actions in a straightforward way [1] such that the resulting state is uniquely defined. However, when conditional effects are allowed, it is very difficult and even too restrictive to guarantee the uniqueness property of *Res* [5]. Therefore we define a function R that yields a set of resulting states.

Definition 5. Let $\mathcal{S}$ be a set of states and R be a function from sets of states and sequences of sets of actions to sets of states. The result of applying a one-set sequence of parallel actions $\langle Q \rangle = \langle \{o_1, \ldots, o_n\} \rangle$ is defined as:

$$R(\mathcal{S}, \langle Q \rangle) = \begin{cases} \{T \subseteq P \mid S \in \mathcal{S}, q \in Seq(Q), T = Res(S,q)\} & \text{if } Res(S,q) \text{ defined} \\ & \forall S \in \mathcal{S}, q \in Seq(Q) \\ \text{undefined} & \text{otherwise} \end{cases}$$

with $Seq(Q)$ denoting the set of all linearizations of the action set $Q = \{o_1, \ldots, o_n\}$. In the special case of the empty sequence we obtain $R(\mathcal{S}, \langle\rangle) = \mathcal{S}$. The result of applying a sequence of sets of parallel actions is defined as

$$R(\mathcal{S}, \langle Q_1, \ldots, Q_n\rangle) = \begin{cases} R(R(\mathcal{S}, \langle Q_1, \ldots, Q_{n-1}\rangle), \langle Q_n\rangle) & \text{if } R(\mathcal{S}, \langle Q_1, \ldots, Q_{n-1}\rangle) \\ & \text{is defined} \\ \text{undefined} & \text{otherwise} \ . \end{cases}$$

3 Planning Graphs for an ADL Subset

The algorithm to construct planning graphs for a given planning problem in the ADL subset of operators differs not very much from the original algorithm described in [1].

Definition 6. A *planning problem* $\mathcal{P}(\mathcal{O}, D, I, G)$ is a 4-tuple where $\mathcal{O}$ is the set of operators, D is the domain of discourse (a finite set of typed objects), and I (the initial state) and G (the goal state) are sets of ground atoms.

Given $\mathcal{O}$ and D, the set of all actions O is the set of possible ground instances of all operators in $\mathcal{O}$.

Definition 7. As in GRAPHPLAN, we define a *planning graph* $\Pi(N, E)$ as a directed leveled graph with the set of nodes $N = N_O \cup N_F$ where N_O and N_F contain the sets of action nodes and ground atom (also called fact) nodes with $N_O \cap N_F = \emptyset$, respectively. The set of edges $E = E_P \cup E_A \cup E_D$ is split into the three disjoint sets E_P, the precondition edges $(N_F \times N_O)$ and E_A, E_D which contain the Add effect edges and Del effect edges $(N_O \times N_F \times 2^{N_F})$.[2]

In a leveled graph, action and fact nodes are arranged in alternating levels and each level is associated with a time step (a natural number). The smallest level $N_F(0)$ comprises fact nodes (one per atom from the initial state I) and is followed by a level of action nodes $N_O(0)$, followed by $N_F(1), N_O(1), N_F(2), \ldots, N_F(max)$, i.e., the graph starts and ends with a fact level. Edges are restricted to link only nodes from two adjacent levels. All this is identical to GRAPHPLAN.

Furthermore, we keep the original notion that two actions *interfere* if one unconditionally deletes a precondition or an unconditional ADD effect of the other, i.e., they cannot simultaneously be executed in *any* world state considering only their unconditional effects. This implies that conditional effects are ignored when defining *interference* between actions, because whether two actions

[2] We write effect edges as triples $N_O \times N_F \times 2^{N_F}$ where an edge is drawn between an action node $o \in N_O$ and its effect (a fact node) $f \in N_F$ that is labeled with a set of fact nodes, which represents the (conjunctive) effect condition under which o achieves f.

interfere based on their conditional effects cannot be decided in advance, but depends on the specific state in which the actions are to be executed. We also adopt GRAPHPLAN's original notion that if two actions are mutually exclusive of each other no possible state of the world exists where both could be executed.

Definition 8. Two actions are *mutually exclusive* of each other

- at time step 0: iff they interfere;
- at time step $n \geq 1$: iff they interfere or the actions have competing needs.

Definition 9. Two actions o_1 and o_2 at an action level $N_O(n \geq 1)$ have *competing needs* iff there exist facts $f_1 \in \varphi_0(o_1)$ and $f_2 \in \varphi_0(o_2)$ that are mutually exclusive.

Definition 10. Two facts $f_1, f_2 \in N_F(n \geq 1)$ are *mutually exclusive* iff there is no non-exclusive pair of actions $o_1, o_2 \in N_O(n-1)$ or no single action $o \in N_O(n-1)$ that conditionally or unconditionally adds f_1 and f_2.

Intuitively, facts are exclusive if no possible state of the world can make both true – considering only their unconditional effects.

Definition 11. An action o is *applicable in a fact level* $N_F(n)$ iff $\varphi_0(o) \subseteq N_F(n)$ and all $f \in \varphi_0(o)$ are non-exclusive of each other.

Based on the previous definitions, IP^2 is *optimistic* when building planning graphs, i.e., analysis and resolution of conflicts between actions caused by conditional effects are deferred to the planning phase. Although actions are non-exclusive if potential conflicts between their conditional effects occur, one can sometimes propagate exclusivity information from preconditions and effect conditions to conditional effects [5].

Technically, planning graphs are constructed in the following way: Given a planning problem, the set of actions is determined as all possible ground instances of all operators. The atoms from the initial state form the first fact level $N_F(0)$. Each action level $N_O(n)$ contains two kinds of action nodes: so-called no-ops (one per atom in $N_F(n)$) and "ordinary" action nodes (one per action that is applicable in $N_F(n)$). No-ops are GRAPHPLAN's solution to the frame problem. For each fact $f \in N_F(n)$, a no-op nop_f is added to $N_O(n)$ with precondition $\varphi_0(nop_f) = \{f\}$ and the only unconditional effect $\alpha_0(nop_f) = \{f\}$. For each "ordinary" action node o, precondition edges between each fact in $N_F(n)$ that is a precondition of o and the action node are established. IP^2 planning graphs differ from GRAPHPLAN only wrt. the effect edges and how the next fact level $N_F(n+1)$ is built. Given a conditional effect $\varphi_i(o) \Rightarrow \alpha_i(o), \delta_i(o)$, IP^2 first proceeds over the individual Add effects $f \in \alpha_i(o)$. They are added to $N_F(n+1)$ iff the following conditions are satisfied:

1. $\varphi_i(o) \subseteq N_F(n)$
2. all facts in $\varphi_i(o)$ are non-exclusive of each other in $N_F(n)$
3. all facts in $\varphi_i(o)$ are non-exclusive of the facts in $\varphi_0(o)$ in $N_F(n)$

The first two conditions imply that the effect condition is applicable, similar to Definition 11. Condition 3 tests if precondition and effect condition can ever hold in the same state – although one can assume that any "reasonable" operator definition will satisfy this condition.

Since one action can achieve the same atomic effect f under different effect conditions, more than one edge can exist between an action node and a fact node in the next level. For each atomic Del effect $f \in \delta_i(o)$, the same tests are performed on the effect condition $\varphi_i(o)$ and a Del edge is established if $f \in N_F(n+1)$. After each level is completed, the mutual exclusive pairs of facts in $N_F(n+1)$ and actions in $N_O(n)$ are determined. The planning graph construction terminates when all goal atoms occur in $N_F(max)$ or if the goal state is unreachable, see Theorem 13.

Given the following sets of actions $\{Op1, Op2, Op3\}$ together with the initial state $I = \{d1, d2, d3, x, y, z\}$ and goals $G = \{a, b, c\}$, Fig. 3 shows the generated planning graph (without no-ops).

Op1	Op2	Op3
pre: d1	pre: d2	pre: d3
eff: ADD a, DEL d1.	eff: ADD b, DEL d2;	eff: ADD c;
	y $\Rightarrow$ ADD x;	y $\Rightarrow$ ADD x;
	x $\Rightarrow$ DEL a.	z $\Rightarrow$ ADD y.

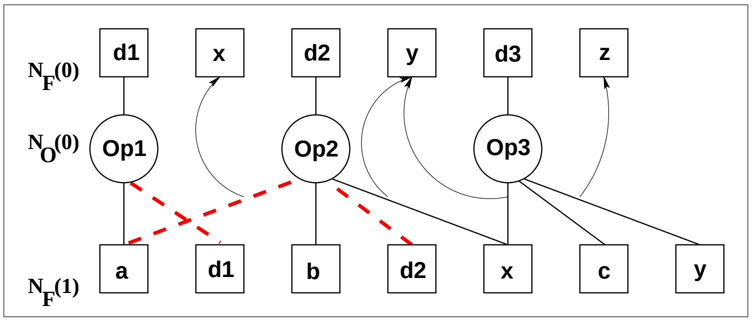

Fig. 3. An example planning graph. The delete edges are drawn with dashed lines and each conditional effect edge has a pointer to its effect condition. The three actions are non-exclusive because they interfere only over their conditional effects.

Planning graphs for the ADL subset inherit all properties of the original planning graphs as described in [1]: First, their size is polynomially restricted in their depth and in the number of actions and facts in the initial state. Second, fact levels grow monotonically, which allows us to incorporate the original "level-off" test into IP^2 that defines a simple and sufficient, but not necessary criterion that a planning problem has no solution.

Definition 12. Let $\mathsf{Mutex}(n)$ be the set of all exclusive pairs at fact level n. We say that the planning graph has *leveled off* at level n iff $|N_F(n)| = |N_F(n+1)|$ and $|\mathsf{Mutex}(n)| = |\mathsf{Mutex}(n+1)|$.

Theorem 13. *[Blum & Furst 95] A planning problem $\mathcal{P}(\mathcal{O}, D, I, G)$ has no solution if its planning graph has leveled off at time step n and either*

1. *one atomic goal is not contained in $N_F(n)$ or*
2. *at least two goal atoms are marked as mutually exclusive.*

4 Finding a Valid Plan in Planning Graphs

The first part of the planning algorithm is identical to GRAPHPLAN: The planning graph is built until the goals are reached for the first time or the graph has leveled off and it turns out that the goal state is not reachable. Note that the planning graph contains only the initial fact level $N_F(0)$ if $G \subseteq I$, i.e., if the goal state holds already in the initial state. If the goals are unreachable, $\perp$ ("no solution exists") is returned. If they are already satisfied in the initial state, $\langle\rangle$ ("empty plan") is returned.

A. initial creation of planning graph
call the planning graph generation algorithm
returns $\Pi(N, E)$ or $\perp$ `/* shortest graph containing goals */`
max `/* the number of proposition levels generated */`
$\mathsf{Mutex}(n)$`/* all exclusive pairs at level` n `*/`
if $max = 0$ then return $\langle\rangle$ and stop planning.
if $\Pi(N, E) = \perp$ then return $\perp$ and stop planning.

If none of the special cases applies, a recursive search algorithm $\mathsf{search}(n)$ over the planning graph is initialized that starts at the last fact level ($n = max$) and terminates when actions from level $N_O(n-1)$ have been successfully selected to achieve the goals at fact level $N_F(n)$ for all levels $1 \leq n \leq max$. The search algorithm differs in three points from its GRAPHPLAN predecessor:

1. the input, which is a pair of sets,
2. the selection procedure for actions at each level, which takes into consideration that an action can possibly achieve the same goal atom under different effect conditions, and
3. the resolution of conflicts caused by conditional effects.

The input consists of two sets of facts instead of only one: (1) the *goals* $\mathcal{G}_n$ that have to be achieved at level n by selecting actions at $N_O(n-1)$ and (2) so-called *negative goals* $\mathcal{C}_n$ that the selected actions have to avoid, i.e., no state in which $\mathcal{G}_n$ is established must entail $\mathcal{C}_n$.[3] If one of the negative goal atoms were established by the selected actions, effect conditions of undesired conditional effects would be made true leading to harmful interaction among actions that were selected to achieve the goals at level $n + 1$.

[3] Note that the input is $\mathcal{G}_{max} = G$ and $\mathcal{C}_{max} = \emptyset$ for the initial call $\mathsf{search}(max)$.

B. level-guided expansion and search

loop
/* initialization of search parameters /*
n := max /* current level */
$\mathcal{G}_{max}$:= G /* fixed goal set at max level */
$\forall\, \mathcal{G}_{1\leq n\leq max-1}$:= $\emptyset$ /* variable goal set at other levels */
$\forall\, \mathcal{C}_{1\leq n\leq max}$:= $\emptyset$ /* negative goals at level n */
$\forall\, \Delta_{0\leq n\leq max-1}$:= $\emptyset$ /* set of selected edges at each level */
call search(max) /* find plan in given planning graph */
if *FAILURE*
  then if $\mathcal{G}_{max}$ is unsolvable (Theorem 14)
    then return $\perp$ and stop planning
    else call expand($max+1$) /* expand graph by one level */
  endif
  else plan found: return set of used actions at each level
endif
endloop

Given a set of goal atoms $\mathcal{G}_n \subseteq N_F(n)$ and a set of negative goals $\mathcal{C}_n \subseteq N_F(n)$, action selection proceeds as follows: First, a set of Add edges $\Delta_{n-1} \subseteq E_A(n-1)$ is selected at action level $N_O(n-1)$ that enable the atoms in the goal set $\mathcal{G}_n$. In contrast to GRAPHPLAN that selects an action, IP^2 selects a particular Add edge that is linked to the goal, because the same action can achieve a goal atom under different types of effects (conditional or not) or under different effect conditions. By selecting an edge, an action is chosen indirectly (marked as "used").

1. choice point: select set of Add edges Δ_{n-1}
for each goal atom $g \in \mathcal{G}_n$
  if a "used" action $o \in N_O(n-1)$ has an unconditional Add edge to g
    then $\Delta_{n-1} := \Delta_{n-1} \cup \{(o, g, \emptyset)\}$ and $\mathcal{G}_n := \mathcal{G}_n \setminus g$
    /* skip goals that are already unconditionally achieved */
    else select edge $(o, g, \varphi_i(o))$ of an action o such that
    a) already "used" actions at level $n-1$ and o are non-exclusive
    b) o does not unconditionally
      - delete a goal $g \in \mathcal{G}_n$ or effect condition of another selected edge
      - add a negative goal $c \in \mathcal{C}_n$
    c) no already "used" action unconditionally deletes an atom in $\varphi_i(o)$
  endif
endfor

The conditions a) to c) guarantee that the "used" actions are independent of each other wrt. their preconditions, unconditional effects, effect conditions of used conditional effects, and that no negative goal is unconditionally added.

Given the example in Fig. 3 and the goals $\mathcal{G}_1 = \{a, b, c\}$ and $\mathcal{C}_1 = \emptyset$, the planner has as its only choice the edges $(Op1, a, \emptyset)$, $(Op2, b, \emptyset)$, and $(Op3, c, \emptyset)$. All conditions are satisfied, i.e., the actions are non-exclusive and no unconditional Del effects occur.

As a next step, IP^2 computes the goals at level $n-1$ based on the preconditions and effect conditions of the selected Add edges. If the resulting set $\mathcal{G}_{n-1}$ is mutually exclusive, the planner backtracks to a new choice of edges, because mutually exclusive goals can never be achieved in the same state.

2. compute goal set $\mathcal{G}_{n-1}$:
$\mathcal{G}_{n-1} := \bigcup_k \varphi_0(o_k) \cup \bigcup_k \varphi_i(o_k)$ with $(o_k, g, \varphi_i(o_k)) \in \Delta_{n-1}$
`/* note that` $\varphi_i(o_k) = \emptyset$ `for an unconditional effect */`

In the small example from Fig. 3, this step is trivial, because all selected edges are unconditional and the new goals $\mathcal{G}_0$ are obtained as the preconditions $\{d1, d2, d3\}$ of the actions.

Now, the action set can be tested for minimality. For each used action, all unconditional Add edges and all conditional Add edges are collected, whose effect conditions are completely contained in the new goals $\mathcal{G}_{n-1}$. The action set is minimal if each action (which can also be a no-op) achieves at least one goal fact that is not achieved by any other action considering the collected edges. If the action set is non-minimal, the planner backtracks and the next choice of edges and actions is computed.

Now, the planner can determine the new negative goals $\mathcal{C}_{n-1}$. This process is quite complex and comprises two main tasks:

1. The planner has to decide if negative goals from the set $\mathcal{C}_n$ are explicitly destroyed by actions at level $n-1$ or progressed through the set $\mathcal{C}_{n-1}$.
2. The selected actions at level $n-1$ are tested for interference caused by conditional effects. To prevent harmful conditional effects, atoms from the corresponding effect conditions are added to the new negative goals $\mathcal{C}_{n-1}$.

For the first task, the planner checks if there is a negative goal c in the set $\mathcal{C}_n$ that is non-exclusive of the new goals $\mathcal{G}_{n-1}$ and that is added by a no-op. This means that c holds in the state where $\mathcal{G}_{n-1}$ is achieved and that it could "survive" the execution of the selected actions at time step $n-1$ if the no-op is non-exclusive of the selected actions, i.e., there is no action in the set Δ_{n-1} that deletes c. The planner can deal with this conflict in two ways: either by adding c to the set of new negative goals $\mathcal{C}_{n-1}$ or by selecting a Del edge from a new or already used action to destroy c. In the first case, we obtain the first entry for the new negative goals. In the second case, the new goals have to be augmented by the preconditions (and possibly effect conditions) of the selected action. Note that this action needs to be a valid choice wrt. interference with already used actions as tested in Part 1 of the search algorithm. To avoid the generation of non-optimal plans, progression is tried first and in the second case, Del edges of used actions are preferred.

3. deal with negative goals $\mathcal{C}_n$:
determine the set $\mathcal{C}_n^-$ of negative goals in C_n that are non-exclusive of $\mathcal{G}_{n-1}$
$\mathbb{C} = \emptyset$`/* initialization */`
for all $c \in \mathcal{C}_n^-$

if c is Add effect of a no-op
then $\mathbb{C} := \mathbb{C} \cup \{c\}$
else nothing
endif
endfor
for all $c \in \mathbb{C}$ **choice point**:
progress c as a new negative goal: $\mathcal{C}_{n-1} := \mathcal{C}_{n-1} \cup \{c\}$
or
select a Del edge from $E_D(n-1)$
$\Delta_{n-1} := \Delta_{n-1} \cup \{(o, c, \varphi_i(o))\}$
$\mathcal{G}_{n-1} := \mathcal{G}_{n-1} \cup \varphi_0(o) \cup \varphi_i(o)$
endfor

For the second task, the planner has to deal with conditional interference among actions that was totally ignored until now. The difference between the two types of effects is that unconditional effects always occur when an action is selected, while conditional effects additionally require the effect condition to hold. This gives the planner the possibility to prevent an undesired conditional effect by making sure that its effect conditions do not hold in the state in which the action is executed. But IP^2 does not know which specific linearization will be chosen for execution because it selects a set of "parallel" actions. Therefore, it has to address two tasks:

1. By forming a set of negative goals at level $n-1$ it guarantees that the undesired effect condition does not hold in the state in which the "parallel" action set is executed.
2. By testing that no action in the parallel set adds the undesired effect condition it guarantees that any linearization leads to a valid execution sequence.

Four different possibilities for conditional interference have to be tested: A selected action is not allowed to conditionally delete a goal (an atom from the set $\mathcal{G}_n$), a precondition of another selected action (an atom from the set $\mathcal{G}_{n-1}$), an effect condition of a desired conditional effect (also an atom from $\mathcal{G}_{n-1}$), or to conditionally add a negative goal from the set $\mathcal{C}_n$.

If the planner discovers conditional interference, it checks if the effect conditions $\varphi_i(o)$ of a harmful conditional effect are contained in the set $\mathcal{G}_{n-1}$. In this case, backtracking to choose a new Δ_{n-1} is necessary, because $\varphi_i(o)$ is causally linked to the new goals $\mathcal{G}_{n-1}$, i.e., whenever they are achieved, the context for the harmful side effect is established as well. If actions at level $N_O(0)$ have harmful conditional effects, backtracking is also necessary because the effect conditions hold in the initial state, which cannot be altered by IP^2. On all levels $n \geq 1$, the set $\varphi_i(o) \setminus \mathcal{G}_{n-1}$ of effect condition atoms that are not in the new goal set is added to a set of sets $\mathbb{S}$ and a (not necessarily minimal) hitting set for $\mathbb{S}$ forms the set of new negative goals $\mathcal{C}_{n-1}$. This means, one element out of each set in $\mathbb{S}$ is selected to form $\mathcal{C}_{n-1}$, because a conditional effect is only achieved if the effect condition as a whole (set) is established, i.e., preventing one single atom per effect condition already avoids the undesired effect. Considering the set difference $\varphi_i(o) \setminus \mathcal{G}_{n-1}$ avoids that goals and negative goals overlap.

In the example from Fig. 3, the planner finds out that $Op2$ deletes the goal a under condition x, which becomes the first new negative goal, i.e., $\mathcal{C}_0 = \{x\}$. Now the planner has to resolve the "linearization problem" by making sure that no selected action adds any of the new negative goals in $\mathcal{C}_0$. The planner checks if any used actions other than $Op2$ add x, which is the case for $Op3$ that adds x under condition y. This means, if $Op3$ is executed before $Op2$ in a state in which y holds, the condition x becomes true and $Op2$ will delete the goal a. Therefore, y must be added as a new negative goal to make sure that this effect is prevented as well. If the effect conditions were a set of atoms instead of a singleton, one of the atoms would be selected as a new negative goal, i.e., the planner has a choice point. Since $\mathcal{C}_0$ has been extended, the whole test is repeated with $\mathcal{C}_0 = \{x, y\}$. Fortunately, only $Op3$ can add the effect condition y of its own harmful conditional effect and this does not matter. Thus, a fixed point is reached and all effect conditions of effects that potentially add new negative goals are already prevented by $\mathcal{C}_0$. But since $\mathcal{C}_0 \neq \emptyset$, no plan can be extracted.

4. compute negative goals $\mathcal{C}_{n-1}$
  for all actions o marked as "used" in $N_O(n-1)$
    if exists an edge $(o, g, \varphi_i(o)) \in E_D(n-1)$ with $g \in \mathcal{G}_n$ or $g \in \mathcal{G}_{n-1}$ or
      $(o, g, \varphi_i(o)) \in E_A(n-1)$ and $g \in \mathcal{C}_n$
      then add $\varphi_i(o) \setminus \mathcal{G}_{n-1}$ to the set of sets $\mathbb{S}$
      else nothing
    endif
  endfor
  if $\emptyset \in \mathbb{S}$ (exists $\varphi_i(o) \subseteq \mathcal{G}_{n-1}$) or $[n = 1$ and $\mathbb{S} \neq \{\emptyset\}]$ (actions at level $N_O(0)$)
    then backtrack
    else **<u>choice point</u>:** choose $\mathcal{C}_{n-1}$ as a hitting set for $\mathbb{S}$
  endif
  complete $\mathcal{C}_{n-1}$ by effect conditions of used actions that add atoms in $\mathcal{C}_{n-1}$
  if $\mathcal{G}_{n-1}$ and $\mathcal{C}_{n-1}$ are disjoint
    then invoke search$(n-1)$
    else backtrack
  endif

The completion process fails if an effect condition is completely contained in $\mathcal{G}_{n-1}$, because this would make $\mathcal{G}_{n-1}$ and $\mathcal{C}_{n-1}$ overlap (a relevant fact had to be made true and false at the same time), or if an action unconditionally adds a new negative goal. During backtracking, all choices of new negative goals (i.e., hitting sets) are tried before a different choice of Δ_{n-1} becomes necessary.

If all possible choices of actions to achieve the goals at level n have failed, the planner has to backtrack to action level n to change one of the sets $\mathcal{G}_n$, $\mathcal{C}_n$ because it has proven the pair to be unsolvable.

Backtracking at the maximum level of the graph is not possible and leads to a failure of search(max) on the planning graph, i.e., no valid plan could be extracted. The planning graph is extended by another action and fact level and the planner searches again on the extended graph. This process of interleaved graph expansion and search terminates if either a set Δ_n was successfully selected at each level of the graph – the plan is the set of all actions that are marked as "used" – or the problem turns out to be unsolvable, cf. Theorem 14.

Theorem 14. *Let the planning graph for a planning problem $\mathcal{P}(\mathcal{O}, D, I, G)$ be leveled off at some level n. Let $m \geq n$ be the current maximum fact level. The problem $\mathcal{P}(\mathcal{O}, D, I, G)$ is unsolvable if and only if extending the planning graph to level $m+1$ and searching the extended graph leads to the same number of goal/negative goal pairs at level n, i.e., $|\{\langle \mathcal{G}_n^m, \mathcal{C}_n^m \rangle\}| = |\{\langle \mathcal{G}_n^{m+1}, \mathcal{C}_n^{m+1} \rangle\}|$.*

See [5] for the proof where we also show that this unsolvability test remains valid under subset memoization.

Theorem 15. *Let $\mathcal{P}(\mathcal{O}, D, I, G)$ be a planning problem. If IP^2 returns a parallel plan $\mathcal{P} = \langle O_1, O_2, \ldots O_n \rangle$ then*

$$\bigcap R(I, \langle O_1, O_2, \ldots, O_n \rangle) \supseteq G$$

holds, i.e., $\mathcal{P}$ is a solution.

To prove soundness we rely on the semantics as developed in Definition 5 and proceed by induction over the length of the plan. Completeness follows from soundness and the fact that IP^2 terminates on unsolvable planning problems. The generation of shortest plans cannot always be guaranteed wrt. our liberal execution semantics. Certain special cases of conditional interference can result in an unnecessary separation of actions [5].

5 Selected Results from the Empirical Evaluation

In the empirical evaluation [5] we were mainly interested in the following questions:

1. Does the extension to a more expressive language lead to a computational overhead?
2. How does IP^2 compare to other planners supporting operators with conditional and universally quantified effects such as Prodigy [2] and UCPOP?

Due to space restrictions we can only sketch a few of the results here and have to refer the reader to [5]. To answer Question 1, we compared IP^2 to GRAPHPLAN on the original GRAPHPLAN test suite. On most of the examples, IP^2 can outperform GRAPHPLAN because we have not simply extended GRAPHPLAN's code, but made significant changes where we thought to have a more efficient solution. We also ran both systems on SATPLAN [4] examples to evaluate the influence of our improved algorithm for subset memoization yielding convincing results. To answer Question 2, we compared IP^2 to UCPOP and Prodigy in the Briefcase domain and a variant of the Scheduling domain. In both domains, IP^2 can usually outperform the other systems. We show the results for the Scheduling domain in Table 1.

Table 1. IP^2, Prodigy, and UCPOP in the scheduling domain

Problem	time steps	actions	UCPOP	Prodigy	IP^2
sched1	3	6	0.94	2.07	0.95
sched2	5	8	4.20	10.27	1.29
sched3	6	9	4.80	56.71	1.57
sched4	7	11	103.00	2.42	1.63
sched5	9	16	-	78.28	5.01
sched6	9	17	-	37.8	6.19

6 Conclusion

We have presented an extension of planning graphs to a subset of ADL that preserves allmost all interesting theoretical properties of the original GRAPHPLAN algorithm. A detailed empirical evaluation showed that this extension of the original GRAPHPLAN system comes with no computational overhead if carefully implemented and that it competes very well with other planners that support ADL subsets. The resulting system is the kernel algorithm of our own interference progression planner IP^2 that we intend to use as the planning algorithm for a mobile robot platform.

References

1. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2):279–298, 1997.
2. E. Fink and M. Veloso. Prodigy planning algorithm. Technical Report CMU-94-123, Carnegie Mellon University, 1994.
3. B. Gazen and C. Knoblock. Combining the expressivity of UCPOP with the efficiency of Graphplan. In Steel [10].
4. H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *AAAI-96*, pages 1194–1201.
5. J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. Technical report, 1997. http://www.informatik.uni-freiburg.de/˜ koehler/ipp.html.
6. D. McDermott. A heuristic estimator for means-ends analysis in planning. In *AIPS-96*, pages 142–149.
7. B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In Steel [10].
8. E. Pednault. ADL: Exploring the middle ground between STRIPS and the Situation Calculus. In *KR-89, Morgan Kaufman*, pages 324–332.
9. J. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *KR-92, Morgan Kaufman*, pages 103–113.
10. S. Steel, editor. *Proceedings of the 4th European Conference on Planning*, LNAI. Springer, 1997.