The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

On the Relationship Between State-Dependent Action Costs and Conditional Effects in Planning

Robert Mattmüller, Florian Geißer
University of Freiburg, Germany
{mattmuel, geisserf}@informatik.uni-freiburg.de

Benedict Wright, Bernhard Nebel
BrainLinks-BrainTools, University of Freiburg, Germany
{bwright, nebel}@informatik.uni-freiburg.de

Abstract

When planning for tasks that feature both state-dependent action costs and conditional effects using relaxation heuristics, the following problem appears: handling costs and effects separately leads to worse-than-necessary heuristic values, since we may get the more useful effect at the lower cost by choosing different values of a relaxed variable when determining relaxed costs and relaxed active effects. In this paper, we show how this issue can be avoided by representing state-dependent costs and conditional effects uniformly, both as edge-valued multi-valued decision diagrams (EVMDDs) over different sets of edge values, and then working with their product diagram. We develop a theory of EVMDDs that is general enough to encompass state-dependent action costs, conditional effects, and even their combination. We define relaxed effect semantics in the presence of state-dependent action costs and conditional effects, and describe how this semantics can be efficiently computed using product EVMDDs. This will form the foundation for informative relaxation heuristics in the setting with state-dependent costs and conditional effects combined.

Introduction

Both from the modeling and from the computational perspective, it makes sense to allow planning tasks with state-dependent action costs, which can be more natural, elegant, compact, and structured than tasks with state-independent costs only. Recent work (Geißer, Keller, and Mattmüller 2015; 2016) has shown that state-dependent action costs (SDAC) can be handled efficiently by representing cost functions as edge-valued multi-valued decision diagrams (EVMDDs) (Ciardo and Siminiceanu 2002; Lai, Pedram, and Vrudhula 1996). Such decision diagrams exhibit additive structure in the cost functions. This structure can then be exploited in various ways, such as in compilations of SDAC to constant-cost tasks, or within the relaxed planning graph (RPG) when computing relaxation heuristics, or to efficiently obtain abstraction heuristics (Geißer, Keller, and Mattmüller 2015; 2016).

However, it turns out that one needs to be very careful when dealing with SDAC and conditional effects (CE) simultaneously, in particular in a delete-relaxed setting based on the accumulation semantics (Hoffmann 2005), and if there is an action whose cost and effect share dependencies on common variables. If this is the case, and if SDAC and CE are handled separately, one may obtain a useful but expensive effect at an unrealistically low cost by choosing different values of a relaxed variable when determining relaxed costs and relaxed active effects. This can lead to unnecessarily low and thus uninformative heuristic values, which hurts the search that uses this heuristic. Let us illustrate the problem with a concrete example (see Fig. 1). Assume that there is a corridor in which we can only move one cell to the left or to the right in each step. The position in the corridor is denoted by the state variable x with possible values $0, \ldots, 5$. Initially, $x = 0$, and in the goal, $x = 5$. The *move-right* action is always applicable, and it has the conditional effect $x' := x + 1$[1], which we read as an abbreviation for $(x = 0 \rhd x' := 1) \wedge \cdots \wedge (x = 4 \rhd x' := 5)$. Moreover, the further to the right one gets, the more costly the movements become, which is reflected by the cost function $\mathit{cost}(\textit{move-right}) = x + 1$. The *move-left* action works similarly, with the same cost function as *move-right*. An optimal unrelaxed plan is to move to the right five times in a row, at an overall cost of $1 + 2 + 3 + 4 + 5 = 15$.

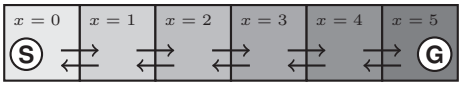


Figure 1: Corridor example. Initial position left, goal position right. Darker shades indicate higher costs to move.

Assume that we want to obtain a relaxation heuristic value for the initial state s_0, say $h^+(s_0)$, and assume that we ignore the interaction of SDAC and CE in the relaxation. This means that in a relaxed state s^+ with $s^+(x) \subseteq \{0, \ldots, 5\}$, where x takes several values simultaneously, the cost of *move-right* is the *minimal* cost the action has for any value of x in s^+, and that the *effect* is the *union* of the effects it has for any value of x in s^+. For example, for $s^+(x) = \{0, 1, 2\}$, we get $\mathit{cost}(\textit{move-right})(s^+) = 1$ from $0 \in s^+(x)$, never-

[1] Notice that we call the variable x after the update x'. For clarity, we will follow this pattern of using primed copies of variables to refer to their value after an update throughout the paper.

theless the next relaxed state will include the value 3, because $2 \in s^+(x)$, meaning that we moved one cell to the right at cost 1, although it should have cost us 3. This can lead to severe underestimations of the actual goal distances. E. g., we get $h^+(s_0) = 5$ instead of $h^*(s_0) = 15$. Even worse, instead of decreasing when moving closer to the goal, the heuristic values first increase. For instance, if s_1 is the state with $x = 1$, then $h^+(s_1) = 6 > 5 = h^+(s_0)$, although we are closer to the goal. The reason is that we first have to pay two units for moving to the left, just to get an excuse for assuming unit cost values of the subsequent four actions of moving to the right from the initial position $x = 1$. In general, the resulting heuristic values can become arbitrarily inaccurate.

Fortunately, there is a way out of this problem. We must not handle SDAC and CE separately by minimizing over the costs and taking unions of effects separately, but rather take the interaction between them into account. In the example, this means that we still have to take the union over all possible effects in s^+, but that we have to assign different costs to different effects. Then, in state s^+ from above, we still get the effects $x' := 1$, $x' := 2$, and $x' := 3$, but at separate costs of 1, 2, and 3, respectively, which leads to the perfect heuristic value $h^+(s_0) = 15$. The question is how to connect SDAC and CE in the right way. Following a naïve approach, we might simply work with an exponentially large tabular representation mapping partial states over the relevant state variables to cost-effect pairs. This would essentially trade memory for heuristic accuracy. However, we can do better: The key observation behind our proposed solution is that both conditional costs and effects can be thought of as functions from states to elements of certain monoids: to cost values from $\mathcal{N} = (\mathbb{N}, +, 0)$ for SDAC[2], and to sets of active effects from $\mathcal{F} = (2^F, \cup, \emptyset)$ for CE, where F is the set of facts of the planning task. Having monoid structures with addition and union, respectively, allows us to assign partial costs that are already unavoidable and partial effects that are already guaranteed to happen to partial variable assignments, and to incrementally derive total costs (via addition) and total effects (via set union) by systematically evaluating the current state fact by fact. This observation, together with the observation that EVMDDs already proved useful for state-dependent costs, suggests representing conditional effects as EVMDDs over $\mathcal{F}$, just as state-dependent costs can be represented as EVMDDs over $\mathcal{N}$, and then combining these representations, provided that both use the same variable ordering. The reader who is curious about what those diagrams look like for the motivating example may already have a quick glance at Figs. 2, 3, and 4, which we will discuss in more detail below. The product diagram in Fig. 4 solves our problem with the running example. Recall the relaxed state s^+ with $s^+(x) = \{0, 1, 2\}$. Before, we had *cost*(*move-right*)$(s^+) = 1$ for all effects that *move-right* produced in s^+, i. e., for $x' := 1$, $x' := 2$, and for $x' := 3$ alike. Now, *cost*(*move-right*)(s^+) is no longer a single value, but rather it associates different costs to different effects, specifically cost i to effect $x' := i$ for $i = 1, 2, 3$, i. e., cost 3 to $x' := 3$. The combined decision diagrams for SDAC and CE can then be used similarly as EVMDDs for SDAC alone (Geißer, Keller, and Mattmüller 2015; 2016). The rest of the paper is concerned with formalizing this idea and generalizing it to arbitrary SDAC and CE.

[2]We use $\mathbb{N}$ instead of $\mathbb{Z}$ or $\mathbb{Q}$, because having a well-founded set with a minimal element makes some discussions a bit easier.

Preliminaries

Planning with State-Dependent Action Costs and Conditional Effects

We consider planning tasks with SDAC and CE, and base our work on the formalism of Geißer et al. (2015).

A *planning task with SDAC and CE* is a tuple $\Pi = (\mathcal{V}, A, s_0, s_\star, (c_a)_{a \in A})$ consisting of the following components: $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of *state variables*, each with an associated finite domain $\mathcal{D}_v = \{0, \ldots, |\mathcal{D}_v| - 1\}$. A *fact* f is a pair (v, d), where $v \in \mathcal{V}$ and $d \in \mathcal{D}_v$. We refer to the set of all facts as F. A *partial variable assignment* s over $\mathcal{V}$ is a consistent set of facts. If s assigns a value to each $v \in \mathcal{V}$, s is called a *state*. Let S denote the set of states of Π. A is a set of *actions*. An action is a pair $a = \langle p, e \rangle$, where p is a partial variable assignment called the *precondition*, and e is a *conditional effect*. We assume, without loss of generality, that conditional effects are given in effect normal form (ENF), which is a special case of Rintanen's unary conditionality (UC) normal form (Rintanen 2003). An effect in ENF is a conjunction $e = \bigwedge_{i=1,\ldots,k} e_i$ of sub-effects e_i of the form $\varphi_i \rhd (w' := d')$, where φ_i is a propositional formula over F, and where $w' := d'$ is an atomic effect (a primed fact) with a variable $w \in \mathcal{V}$ and value $d' \in \mathcal{D}_w$. In ENF, every atomic effect may occur at most once in e. We furthermore assume that there is no state s in which two contradicting atomic effects are enabled, i. e., whenever e includes two conjuncts $\varphi_i \rhd (w' := d')$ and $\varphi_j \rhd (w' := d'')$ for $d' \neq d''$, then $\varphi_i \wedge \varphi_j$ is unsatisfiable. If some $\varphi_i = \top$, then the corresponding sub-effect is unconditional. The state $s_0 \in S$ is called the *initial state*, and the partial state $s_\star$ specifies the *goal* condition. Each action $a \in A$ has an associated *cost function* $c_a : S \to \mathbb{N}$ that assigns the application cost of a to all states where a is applicable.

Each cost function c_a depends on a certain subset of the state variables. Throughout the paper, we assume without loss of generality that for all variables v that are mentioned in the precondition p of an action a, neither c_a nor any effect condition φ_i of its effect depends on v. Otherwise, one could substitute the precondition value of v in the cost function or the effect condition, respectively, and simplify. The semantics of planning tasks are as usual: an action a is applicable in state s iff $p \subseteq s$. To define the result of an action application, we need the change set of e in s (Rintanen 2003).

Definition 1. *Let $s \in S$ be a state and e an effect in ENF over the state variables of s. Then the* change set *of e in s, symbolically $[e]_s$, is defined as follows:*

(1) $[e_1 \wedge \cdots \wedge e_n]_s = [e_1]_s \cup \cdots \cup [e_n]_s$ *, and*
(2) $[\varphi \rhd \mathsf{f}]_s = \{\mathsf{f}\}$ *if $s \models \varphi$, and* $[\varphi \rhd \mathsf{f}]_s = \emptyset$*, otherwise.*

A change set will never contain two contradicting effects $w' := d'$ and $w' := d''$ for $d' \neq d''$ because we as-

sume that contradicting effects have inconsistent conditions. Therefore, removing primes from primed variables, we can view change sets as partial variable assignments. Then, applying an applicable action a to s yields the state s' with $s'(v) = [e]_s(v)$ where $[e]_s(v)$ is defined, and $s'(v) = s(v)$ otherwise. We write $s[a]$ for s'.

A state s is a goal state iff $s_\star \subseteq s$. Let $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ be a sequence of actions from A. We call π *applicable* in s_0 if there exist states $s_1, \ldots, s_n$ such that a_i is applicable in s_i and $s_{i+1} = s_i[a_i]$ for all $i = 0, \ldots, n-1$. We call π a *plan* for Π if it is applicable in s_0 and if s_n is a goal state. The *cost* of plan π is the sum of action costs along the induced state sequence, i.e., $cost(\pi) = \sum_{i=0}^{n-1} c_{a_i}(s_i)$.

Edge-Valued Decision Diagrams

Both action cost functions and conditional effects can be represented as EVMDDs, though over different monoids. Recall that a monoid is a structure $\mathcal{M} = (M, +, 0)$ consisting of a carrier set M, a binary operation $+$ on M, and an element $0 \in M$ such that $+$ is associative, and that 0 is the neutral element. In our scenario, $+$ will be used to aggregate partial costs or effects, and the neutral element will be the cost value zero or the empty set of effects, respectively. We only consider *commutative* monoids to avoid issues with different variable orderings.

To define when a (reduced ordered) EVMDD is canonical, we need to ensure that edge labels are not arbitrarily shifted up and down along the edges. This is achieved by requiring that there is nothing that sibling edge labels originating in the same parent node still have in common that could not be taken care of earlier in the decision diagram. For $\mathcal{N}$, this means that the minimum edge label of any edge leaving the same parent node is zero (and hence, any excess cost has been pulled upward into the incoming edge label). Similarly, for $\mathcal{F}$, it means that the intersection of the labels of the edges leaving the same parent node is empty (and hence, all partial effects that happen for all possible values of the current decision variable are pulled upward into the incoming edge label).

To accomplish this in general, we require that $\mathcal{M}$ has a *monus* operator $\dot{-}$ (Amer 1984). For that, we need the relation $\leq$ on M defined as $a \leq b$ iff there is a $c \in M$ with $a + c = b$, for $a, b \in M$. This relation is reflexive and transitive by definition, and we additionally assume it to be antisymmetric, giving us a partial order. Next, we assume that for all $a, b \in M$, there is a unique smallest element $c \in M$ such that $a \leq b + c$. Then $\mathcal{M}$ is called a *commutative monoid with monus (CMM)* (Amer 1984) and the monus $a \dot{-} b$ of a and b is this unique c. Finally, we require that the partial order $\leq$ is a *meet-semilattice* order, i. e., that $(M, \leq)$ has a *greatest lower bound* for any nonempty finite subset $M' \subseteq M$, denoted by $\bigwedge M'$, as well as that the operation $+$ on M can be distributed over the greatest lower bound operator $\bigwedge$, and that $\bigwedge M = 0$. If all these requirements are met, we call $\mathcal{M}$ a *meet-semilattice ordered CMM*. We will often assume a meet-semilattice ordering implicitly without always mentioning it.

Notice that $\mathcal{N}$ with the natural order $\leq$ as partial order, truncated subtraction as monus, and minimum operator $\min$ as greatest lower bound is a meet-semilattice ordered CMM, and that the same holds for $\mathcal{F}$ with subset relationship $\subseteq$ as partial order, set difference as monus, and the intersection operation $\cap$ as greatest lower bound. Since we will consider product EVMDDs that combine costs and effects, we also require that Cartesian products of meet-semilattice ordered CMMs are again meet-semilattice ordered CMMs. With component-wise partial orders, monuses, and greatest lower bounds, this is indeed the case. During EVMDD construction, the greatest lower bound operator $\bigwedge$ will be used to determine what the common part of sibling edge labels is, whereas the monus operator $\dot{-}$ will be used to determine which partial cost or effect is left at each EVMDD edge after the common part of sibling edge labels has been "subtracted" and pulled upward into the incoming edge label.

Definition 2. *Let $\mathcal{M} = (M, +, 0)$ be a meet-semilattice ordered CMM. An* EVMDD *over $\mathcal{M}$ and over $\mathcal{V}$ is a tuple $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$, where $\kappa \in M$ and $\mathbf{f}$ is a directed acyclic graph consisting of two types of nodes: (i) there is a single* terminal node *denoted by $\mathbf{0}$. (ii) A* nonterminal node *$\mathbf{v}$ is a tuple $(v, \chi_0, \ldots, \chi_k, w_0, \ldots, w_k)$ where $v \in \mathcal{V}$ is a variable, $k = |\mathcal{D}_v| - 1$, children $\chi_0, \ldots, \chi_k$ are terminal or nonterminal nodes of $\mathcal{E}$, labels $w_0, \ldots, w_k \in M$, and $\bigwedge_{i=0,\ldots,k} w_i = 0$.*

By $\mathbf{f}$ we also refer to the root node of $\mathcal{E}$. Edges of $\mathcal{E}$ between parent and child nodes are implicit in the definition of the nonterminal nodes of $\mathcal{E}$. The *label* of an edge from $\mathbf{v}$ to child χ_i is w_i. An EVMDD over a commutative monoid $\mathcal{M}$ with carrier M and variables $\mathcal{V}$ denotes a function from the set of states S over $\mathcal{V}$ to M. Intuitively, to determine the function value for a given state $s \in S$, one has to follow the unique path in the EVMDD determined by s by always following the unique edges consistent with s, collect the edge labels along the way, and combine them with $+$. E. g., if the edge labels are numbers and $+$ is addition, then one has to add up all the encountered edge labels.

Definition 3. *An EVMDD $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$ over meet-semilattice ordered CMM $\mathcal{M} = (M, +, 0)$ and variables $\mathcal{V}$ denotes the function $\kappa + f$ from the states over $\mathcal{V}$ to M, where f is the function denoted by $\mathbf{f}$. The terminal node $\mathbf{0}$ denotes the constant function 0, and a node $(v, \chi_0, \ldots, \chi_k, w_0, \ldots, w_k)$ denotes the function given by $f(s) = w_{s(v)} + f_{s(v)}(s)$, where $f_{s(v)}$ is the function denoted by child $\chi_{s(v)}$. We write $\mathcal{E}(s)$ for $\kappa + f(s)$.*

In the graphical representation of an EVMDD, $\mathbf{f}$ is represented by a rooted DAG and κ by a dangling incoming edge to the root node of $\mathbf{f}$. The terminal node is depicted by a solid black circle. Edge constraints d, corresponding to the valuation of the decision node, are written next to the edges, edge labels w_d in boxes on the edges.

Let us return to our example. The action cost function *cost*(*move-right*) $= x+1$ can be represented by the EVMDD over $\mathcal{N}$ depicted in Fig. 2. Similarly, the conditional effect $x' := x+1$ of *move-right* can be represented by the EVMDD over $\mathcal{F}$ depicted in Fig. 3. Notice that in the latter, the edge labels are generally *sets* of effects that fire, not just single effects. They only happen to be singleton sets in this example for $x = 0, \ldots, 4$. For $x = 5$, when the right end of the

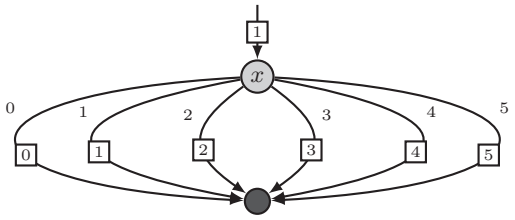


Figure 2: EVMDD over $\mathcal{N}$ for cost function $x + 1$.

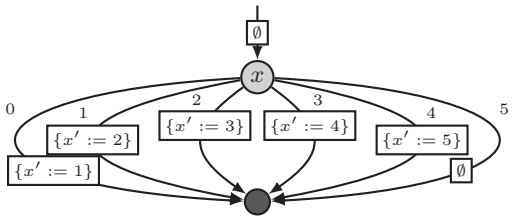


Figure 3: EVMDD over $\mathcal{F}$ for conditional effect $x' := x+1$.

corridor has been reached, the conditional effect is empty, as witnessed by the corresponding edge label $\emptyset$. Similarly, the empty set at the dangling incoming edge represents the fact that there are no *unconditional effects* in this example. Otherwise, they would be found there. The product of those two EVMDDs (Fig. 4) is obtained by combining decision nodes of (the quasi-reduced form of) one diagram with nodes of (the quasi-reduced form of) the other diagram on the same level, i. e., with the same associated decision variable, with corresponding paths leading there, and combining edges and edge labels accordingly. It is, by construction, an EVMDD over the Cartesian product $\mathcal{N} \times \mathcal{F}$. In this example, there is only one decision node with associated decision variable x in both diagrams, and therefore just one product node.

Notice that for general meet-semilattice ordered CMMs, being reduced and ordered, and satisfying $\bigwedge_{i=0,\ldots,k} w_i = 0$ at each decision node is *not* sufficient for uniqueness, since

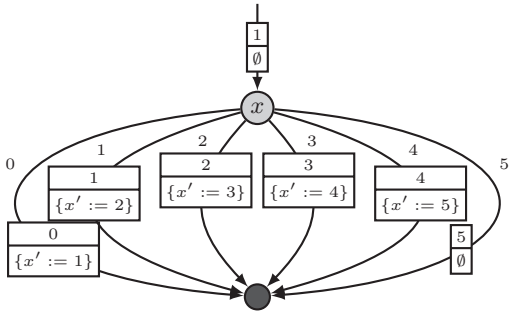


Figure 4: EVMDD over $\mathcal{N} \times \mathcal{F}$ for cost function $x + 1$ and conditional effect $x' := x + 1$ combined. Edge labels have their $\mathcal{N}$-part on top, and their $\mathcal{F}$-part at the bottom.

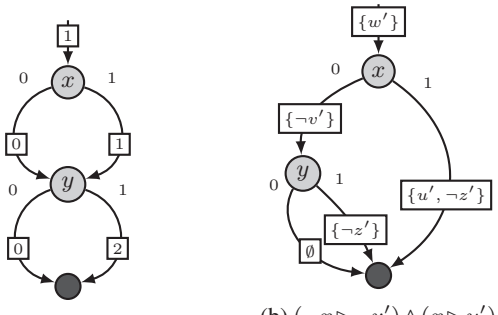
there can be redundant edge labels along a path that are not ruled by any of these conditions. We conjecture that an additional *redundancy freedom* condition of the form $(a + b) \dot{-} b = a$ for labels a and b on the same EVMDD path may restore uniqueness. For $\mathcal{N}$ and $\mathcal{F}$, the *apply* procedure described below, which we use to construct EVMDDs, guarantees this condition, which is sufficient for the purpose of this paper. We conjecture that the same holds for other meet-semilattice ordered CMMs.

EVMDD Construction

In this section, we will discuss how EVMDDs over $\mathcal{N} \times \mathcal{F}$ can be constructed that encode SDAC and CE for unrelaxed states in one diagram. In the subsequent section, we will show how the same diagrams can also be used to determine SDAC and CE for *relaxed* states.

Before we discuss the individual constructions for state-dependent action costs, conditional effects and the combination of both, we want to revisit the *apply* procedure (Lai, Pedram, and Vrudhula 1996). The original algorithm only considered EVMDDs over the same monoid; we consider a more general definition which allows us to combine EVMDDs over different monoids. For that we assume that a state space S and a variable ordering are fixed. Let $\mathcal{L} = (L, +_L, 0_L)$, $\mathcal{R} = (R, +_R, 0_R)$ and $\mathcal{T} = (T, +_T, 0_T)$ be three meet-semilattice ordered CMMs and let $\circ : L \times R \to T$ be an operator. Assume further that $f : S \to L$ and $g : S \to R$ are two functions and that, by slight abuse of notation, we view $\circ$ also as an operator on such functions in the obvious way via $(f \circ g)(s) = f(s) \circ g(s)$. Furthermore, let $\mathcal{E}_{(\cdot)}$ be the construction that turns a function f into the reduced ordered EVMDD $\mathcal{E}_f$ representing it. We would like to have an operator $\circ_\mathcal{E}$ on EVMDDs that mimics the behavior of $\circ$ on EVMDDs, i.e., such that $\mathcal{E}_{(f \circ g)} = \mathcal{E}_f \circ_\mathcal{E} \mathcal{E}_g$. The *apply* procedure does exactly that. In the literature, the application of $\circ$ on the EVMDD level, $\mathcal{E}_f \circ_\mathcal{E} \mathcal{E}_g$, is usually written as $apply(\circ, \mathcal{E}_f, \mathcal{E}_g)$. Algorithmically, the *apply* procedure traverses both input EVMDDs $\mathcal{E}_f$ and $\mathcal{E}_g$ from top to bottom in a synchronized manner, propagating edge labels downward, recursively applying $\circ$ to pairs of corresponding subgraphs with the same edge constraint, and pulling up excess edge weights (labels) again when the recursive computation has terminated. In the base case, when both EVMDDs represent constant functions encoded in their bottom-most edge labels w_L (in $\mathcal{E}_f$) and w_R (in $\mathcal{E}_g$), those get combined into the new edge label $w_L \circ w_R$. If, due to one of the EVMDDs being Shannon-reduced at some point where the other is not, the decision variables on both sides do not match, then the Shannon reduction on one side has to be conceptually undone by virtually introducing a new decision node with all outgoing edges carrying the "empty" label 0_L (0_R) before proceeding.

In the following, we will assume that cost functions and conditional effects are given as syntactic terms (e. g., costs as multivariate polynomials), and can thus be represented by abstract syntax trees (ASTs). To construct corresponding EVMDDs, we will then traverse the ASTs bottom-up, and perform the operations represented by inner nodes of the AST on the EVMDDs already constructed for the subtrees of the AST, using the *apply* procedure. For an alter-



(a) $x + 2y + 1$.

(b) $(\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge w'$.

Figure 5: EVMDDs for costs and conditional effects.

native construction using repeated Shannon expansions into cofactors we refer to a previous workshop version of this paper (Mattmüller et al. 2017).

Construction for State-Dependent Action Costs

Let $c : S \to \mathbb{N}$ be the function we want to represent, given as a multivariate polynomial. We first generate the abstract syntax tree of c where leaf nodes represent variables and constants, and inner nodes represent arithmetic operators. For each leaf node, we create an EVMDD $\mathcal{E}$; if the node represents a constant n, then we set $\kappa = n$ and get $\mathcal{E} = \langle n, \mathbf{0} \rangle$, i. e., $\mathcal{E}$ represents the constant function n. If the leaf node represents a variable v, then we create the EVMDD $\mathcal{E} = \langle 0, \mathbf{f} \rangle$ representing the function v, where $\mathbf{f} = (v, \mathbf{0}, \ldots, \mathbf{0}, 0, 1, \ldots, k)$ with $k = |\mathcal{D}_v| - 1$. Intuitively, $\mathcal{E}$ consists of a single non-terminal node representing variable v, with an edge of cost d to the terminal node for each domain value d of v. For inner nodes representing operators $\circ$, we recursively construct the EVMDDs $\mathcal{E}_L$ and $\mathcal{E}_R$ for the left and right child and call the *apply* procedure to generate $\mathcal{E}_L \circ_{\mathcal{E}} \mathcal{E}_R$.

Proposition 1. *Let $c : S \to \mathbb{N}$ be an arithmetic function and let $\mathcal{E}_c$ be the reduced ordered EVMDD for c constructed as described above, for an arbitrary variable ordering. Let $s \in S$ be a state. Then $c(s) = \mathcal{E}_c(s)$.* □

Example 1. *Fig. 5a depicts an EVMDD over $\mathcal{N}$ with variable ordering x, y representing the function $c(x, y) = x + 2y + 1$, where the domains of all variables are binary. The left hand side of Fig. 6 illustrates the construction of this EVMDD, showing the required* apply *calls in the inner nodes, and the EVMDDs corresponding to the base cases in the leaf nodes. Following the unique path through the EVMDD corresponding to a given state s, say with $s(x) = s(y) = 1$, summing up the edge labels along the way, results in the correct function value $c(x, y) = 4$.*

Construction for Conditional Effects

For CE, using the monoid $\mathcal{F} = (2^F, \cup, \emptyset)$, the construction works in a two-fold manner. Let $e = (\varphi_1 \rhd \mathsf{f}_1) \wedge \cdots \wedge (\varphi_n \rhd \mathsf{f}_n)$ be an effect in ENF. For each φ_i, we generate the expression tree where leaf nodes represent either constants $\top$ or $\bot$, or facts $v = d$. Similar to above, we generate EVMDDs, now over the monoid $\mathcal{B} = (\{\top, \bot\}, \vee, \bot)$: for constants, the corresponding EVMDD is either $\mathcal{E} = \langle \top, \mathbf{0} \rangle$ or $\mathcal{E} = \langle \bot, \mathbf{0} \rangle$. For a fact $v = d$, the resulting EVMDD is $\mathcal{E} = \langle \bot, \mathbf{f} \rangle$ where $\mathbf{f} = (v, \mathbf{0}, \ldots, \mathbf{0}, w_0, \ldots, w_k)$ with $k = |\mathcal{D}_v| - 1$, $w_d = \top$, and $w_i = \bot$ for $i \neq d$, i.e. for $v = d$ the EVMDD evaluates to true, and otherwise to false. Inner nodes, representing Boolean connectives, are, once again, processed via the *apply* procedure. We call the resulting EVMDD $\mathcal{E}_{\varphi_i}$. For each fact f_i, we generate the corresponding EVMDD (over $\mathcal{F}$) representing that effect, $\mathcal{E}_{\mathsf{f}_i} = \langle \{\mathsf{f}_i'\}, \mathbf{0} \rangle$. Finally, we generate the EVMDD representing conditional effect $\varphi_i \rhd \mathsf{f}_i$ by applying the operator $\trianglerighteq : \{\top, \bot\} \times 2^F \to 2^F$ to $\mathcal{E}_{\varphi_i}$ and $\mathcal{E}_{\mathsf{f}_i}$, where $\top \trianglerighteq F' = F'$ and $\bot \trianglerighteq F' = \emptyset$ for $F' \subseteq F$. Intuitively, this EVMDD evaluates a state s to $[\varphi_i \rhd \mathsf{f}_i]_s$, i. e., to f_i, if $s \models \varphi_i$, and to $\emptyset$, otherwise. Finally, we represent e as a single EVMDD $\mathcal{E}_e$, by applying the union operator $\cup$ to the sub-EVMDDs.

Example 2. *Consider the conditional effect in ENF $e = (\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge w'$. We have binary domains for all variables and consequently use the abbreviations $\neg v$ and v for $v = 0$ and $v = 1$ (and $\neg v'$ and v' for $v' := 0$ and $v' := 1$ in effects). The right hand side of Fig. 6 illustrates the construction of this EVMDD, showing the required* apply *calls in the inner nodes and the EVMDDs corresponding to the base cases in the leaf nodes. Fig. 5b depicts an EVMDD over $\mathcal{F}$ with variable ordering x, y, z, u, v, w representing the effect e. Following the unique path through the EVMDD corresponding to a given state s, say with $s(x) = s(y) = 1$, and taking the union of the edge labels along the way, results in the effect $\{w', u', \neg z'\}$.*

For CE, the semantics of an effect ℓ applied to a state s is its change set $[e]_s$. Therefore, the analogue to Prop. 1 for CE, which follows inductively from the correctness of the *apply* procedure, reads as follows.

Proposition 2. *Let e be a conditional effect in ENF, and let $\mathcal{E}_e$ be the reduced ordered EVMDD for e constructed as described above, for an arbitrary variable ordering. Let $s \in S$ be a state. Then $[e]_s = \mathcal{E}_e(s)$.* □

Product EVMDDs

The more general definition of the *apply* procedure allows for a compact and succinct definition of product EVMDDs. Given two meet-semilattice ordered CMMs $\mathcal{L} = (L, +_L, 0_L)$ and $\mathcal{R} = (R, +_R, 0_R)$, and two EVMDDs $\mathcal{E}_f$, $\mathcal{E}_g$ representing functions $f : S \to L$, $g : S \to R$, the *product* of $\mathcal{E}_f$ and $\mathcal{E}_g$ is $\mathcal{E}_{f,g} = \mathcal{E}_f \otimes_{\mathcal{E}} \mathcal{E}_g$ with $\otimes : L \times R \to L \times R$ being the identity function $\otimes(l, r) = (l, r)$ for $(l, r) \in L \times R$. A schematic view of this construction is depicted in Figure 6. The left half of the tree describes the cost function, and the right half describes the conditional effects. The leaf nodes are constants, variables, or facts, and can directly be represented as EVMDDs (some examples for such EVMDDs are shown in red). Inner nodes correspond to operations on their underlying monoids, annotated by the corresponding *apply* call.

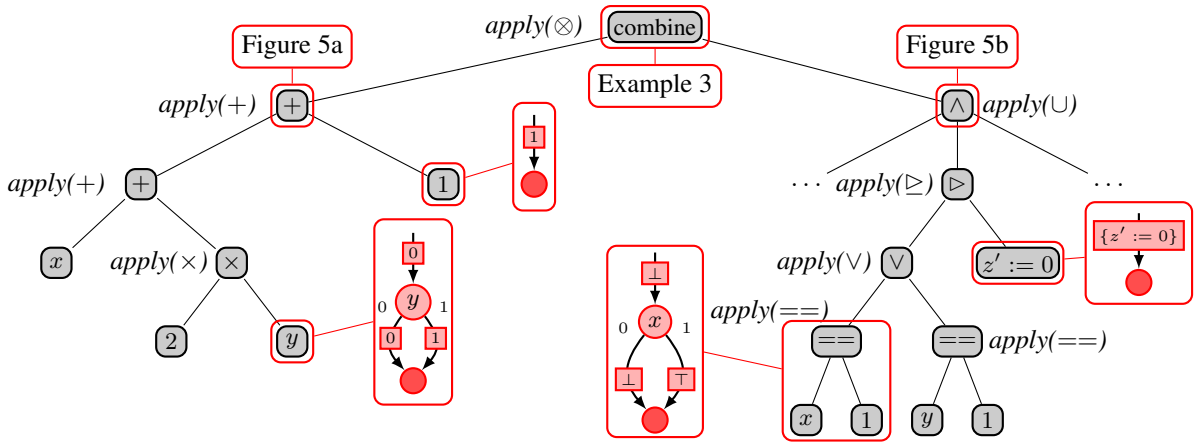
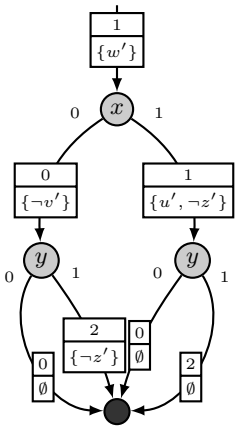


Figure 6: Combined AST for cost function $x + 2y + 1$ and conditional effect $(\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge w'$ is depicted in gray. Exemplary EVMDDs corresponding to the nodes of the AST are depicted in red.

Example 3. *Consider the two EVMDDs from Ex. 1 and 2. Their product is depicted to the right. The diagram contains a full binary tree over x and y, which means that there is no potential of exploiting shared structure due to the involved combination of costs and conditional effects. This is, however, specific to this example. In general, product diagrams can be much smaller.*



By evaluating the product EVMDD for a state s, we still get the correct cost values and change sets back via projection. This holds in general for EVMDDs over arbitrary meet-semilattice ordered CMMs.

Proposition 3. *Given f,g as above, assume a fixed variable ordering and let $s \in S$. Then $\mathcal{E}_{f,g}(s) = (f(s), g(s))$.*

Proof. Follows immediately from the fact that $apply(\circ, \mathcal{E}_f, \mathcal{E}_g) = \mathcal{E}_{f \circ g}$, instantiated with $\circ = \otimes$. More precisely, $\mathcal{E}_{f,g} = \mathcal{E}_f \otimes_{\mathcal{E}} \mathcal{E}_g = apply(\otimes, \mathcal{E}_f, \mathcal{E}_g) = \mathcal{E}_{f \otimes g}$, by definition of product EVMDDs, and definition and correctness of the *apply* procedure, respectively. Therefore, $\mathcal{E}_{f,g}(s) = \mathcal{E}_{f \otimes g}(s) = (f, g)(s) = (f(s), g(s))$. □

Corollary 1. *Assume a fixed variable ordering. Let $c : S \to \mathbb{N}$ be an arithmetic function and e be a conditional effect in ENF. Let $\mathcal{E}_c$ and $\mathcal{E}_e$ be the EVMDDs for c and e constructed as described. Let $\mathcal{E}_{c,e} = \mathcal{E}_c \otimes \mathcal{E}_e$, and let $s \in S$ be a state. Then $\mathcal{E}_{c,e}(s) = (\mathcal{E}_c(s), \mathcal{E}_e(s)) = (c(s), [e]_s)$.* □

The size of a product EVMDD $\mathcal{E}_f \otimes_{\mathcal{E}} \mathcal{E}_g$ is always bounded by the product of the sizes of the factors $\mathcal{E}_f$ and $\mathcal{E}_g$. Moreover, there are two special cases where the construction incurs no blowup whatsoever. First, if $\mathcal{E}_f$ and $\mathcal{E}_g$ share an identical graph topology and only differ in their edge labels (as in Figs. 2 and 3), i. e., their evaluation proceeds "in lockstep", then the product also shares the same topology. This happens whenever there is a one-to-one correspondence between sub-effects and partial costs associated with them. Second, if the set of variables $\mathcal{V}_f$ on which $\mathcal{E}_f$ depends is disjoint from the set of variables $\mathcal{V}_g$ on which $\mathcal{E}_g$ depends, and if $\mathcal{V}_f$ and $\mathcal{V}_g$ are *not* interleaved in the variable ordering, then $\mathcal{E}_f \otimes_{\mathcal{E}} \mathcal{E}_g$ will essentially be $\mathcal{E}_f$ and $\mathcal{E}_g$ sequentially "glued together", with the label of the disappearing second dangling incoming edge moved to the first dangling incoming edge instead. This is the case whenever there is no relation between costs and effects at all.

Relaxed Semantics for SDAC and CE

In this section, we first *declaratively* define a relaxed semantics in the presence of SDAC and CE, and then show how this semantics can be efficiently *computed* using the previously constructed product EVMDDs over $\mathcal{N} \times \mathcal{F}$. Whenever we mention relaxed states, the reader should keep in mind that the same discussion works for arbitrary Cartesian states (Ball, Podelski, and Rajamani 2001; Seipp and Helmert 2013), of which relaxed states are merely a special case, in particular also for states of a Cartesian abstraction.

Declarative Definition

A relaxed state s^+ assigns to each variable $v \in \mathcal{V}$ a non-empty subset $s^+(v) \subseteq \mathcal{D}_v$ of its domain. A state $s \in S$ *is consistent with* s^+, in symbols $s \models s^+$, iff for all variables v, $s(v) \in s^+(v)$. An action $a = \langle p, e \rangle$ is *relaxed applicable* in s^+ iff $p(v) \in s^+(v)$ for all v for which p is defined. Generalizing Def. 1, we define the change set of an effect e of an action a with precondition p in a relaxed state s^+. However, instead of a set of facts, this will now be a set of pairs of facts and associated cost values.

Definition 4. *Let s^+ be a relaxed state and $a = \langle p, e\rangle$ be an action with effect e in ENF and cost function $c : S \to \mathbb{N}$. Then the* change set *of e in s^+ is defined as $[e]^c_{s^+} = \bigsqcup_{s\in S: s\models s^+} [\![e]\!]^c_s$, where*

(1) $[\![e_1 \wedge \cdots \wedge e_n]\!]^c_s = [\![e_1]\!]^c_s \cup \cdots \cup [\![e_n]\!]^c_s$,

(2) $[\![\varphi \rhd \mathsf{f}]\!]^c_s = \{(\mathsf{f}, c(s))\}$ *if* $s \models \varphi$, *and* $[\![\varphi \rhd \mathsf{f}]\!]^c_s = \emptyset$, *otherwise, and*

(3) $\bigsqcup_j E_j = \{(\mathsf{f}, n) \in \bigcup_j E_j \mid \forall (\mathsf{f}, \ell) \in \bigcup_j E_j : \ell \geq n\}$.

The change set $[e]^c_{s^+}$ consists of all those facts f that can be achieved using e in any state s with $s \models s^+$. With each such fact f, the change set associates the minimal cost at which f can be achieved among all s with $s \models s^+$. When defining $[e]^c_{s^+}$ by referring to all states s with $s \models s^+$, we do not have to distinguish between states where a is applicable and states where it is not. Since the precondition variables affect neither the costs nor the effect conditions, whenever we get a minimal cost value from a state where a is inapplicable, there must also be another state also consistent with s^+ where a is applicable and where it costs the same. In clause (1), we still use the regular union operation, which is justified since we assume that no fact occurs on two different right-hand sides of sub-effects. We might use the minimizing union $\bigsqcup$ just as well, leading to the equivalent phrasing $[\varphi_1 \rhd \mathsf{f}_1 \wedge \cdots \wedge \varphi_n \rhd \mathsf{f}_n]^c_{s^+} = \bigsqcup_{s\in S: s\models s^+} \bigsqcup_{i=1,\ldots,n} [\![\varphi_i \rhd \mathsf{f}_i]\!]^c_s$. For illustration, recall the introductory example and the relaxed state s^+ with $s^+(x) = \{0, 1, 2\}$. Let $c = \mathit{cost}(\mathit{move\text{-}right})$. Then we get $[x' := x + 1]^c_{s^+} = \{(x' := 1, 1), (x' := 2, 2), (x' := 3, 3)\}$.

EVMDD-Based Computation

Next, we show how we can compute change sets in relaxed states efficiently. The problem is that in Def. 4, we take the union over all unrelaxed states s with $s \models s^+$, in the worst case exponentially many in the number of state variables.[3] To avoid this exponentiality whenever possible, we use the product EVMDDs constructed above. We will describe a polynomial evaluation procedure for such EVMDDs $\mathcal{E}_{c,e}$ over costs and effects for relaxed states s^+ as input that returns $[e]^c_{s^+}$. Before describing the procedure, we want to point out that it *does not* simply consist of taking sums of costs and unions of effects independently (essentially the standard EVMDD evaluation from Def. 3), which would defeat the purpose of the construction.

Rather, our proposed evaluation procedure traverses $\mathcal{E}_{c,e}$, restricted to edges consistent with s^+, along a topological ordering from top to bottom. At each node $\mathbf{v}$, it keeps track of two pieces of information: (a) the set F of fact-cost pairs (f, n) for all achieved facts f at $\mathbf{v}$ along any incoming path, together with cheapest achievement costs n of f, and (b) the cost n of a cheapest path leading to $\mathbf{v}$. I. e., $\mathcal{E}_{c,e}(s^+)(\mathbf{v})$ will have the form (F, n). To formalize this procedure, let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be a topological ordering of $\mathcal{E}_{c,e}$, where $\mathbf{v}_n = \mathbf{0}$.

[3] Computing change sets from Def. 4 is an NP-hard problem, even for unit-cost tasks, as a simple reduction from SAT shows: a propositional formula φ is satisfiable iff $(\mathsf{f}, 1) \in [\varphi \rhd \mathsf{f}]^c_{s^+}$, where $c(s) = 1$ for all s, and $s^+(v) = \mathcal{D}_v$ for all $v \in \mathcal{V}$.

Base case for $i = 1$: Node $\mathbf{v}_1$ only has the dangling incoming edge with label $\kappa = (n, F')$. We let $\mathcal{E}_{c,e}(s^+)(\mathbf{v}) = (\mathsf{F}, \mathsf{n})$ with $\mathsf{F} = \{(\mathsf{f}, n) \mid \mathsf{f} \in F'\}$ and $\mathsf{n} = n$.

Inductive case for $i > 1$: Let $\mathbf{v} = \mathbf{v}_i$ be an interior node of $\mathcal{E}_{c,e}$. To determine F for $\mathbf{v}$, we collect all facts F^{old} inherited from parent nodes of incoming edges (consistent with s^+), with their costs increased by the incoming edge cost. To those, we add all facts F^{new} achieved on incoming edges, with cost of achieving them there; the resulting set of fact-cost pairs is filtered so that we only associate the *cheapest* cost with each fact. Formally, let us denote incoming edges of $\mathbf{v}$ as tuples consisting of a parent node $\mathbf{v}_j$ with associated decision variable v_j and edge constraint $v_j = d_j$, and edge label (n_j, F_j), consisting of partial costs n_j and partial effects F_j. Index the incoming edges with $j = 1, \ldots, M$. Let $(\mathsf{F}_j, \mathsf{n}_j) = \mathcal{E}_{c,e}(s^+)(\mathbf{v}_j)$ be the evaluation result associated with parent node $\mathbf{v}_j$. Then we define $\mathsf{F}^{\text{old}}_j = \{(\mathsf{f}, n + n_j) \mid (\mathsf{f}, n) \in \mathsf{F}_j\}$, $\mathsf{F}^{\text{new}}_j = \{(\mathsf{f}, \mathsf{n}_j + n_j) \mid \mathsf{f} \in F_j\}$, $\mathsf{F}^{\text{old}} = \bigsqcup_{j=1,\ldots,M} \mathsf{F}^{\text{old}}_j$, $\mathsf{F}^{\text{new}} = \bigsqcup_{j=1,\ldots,M} \mathsf{F}^{\text{new}}_j$, and $\mathsf{F} = \mathsf{F}^{\text{old}} \sqcup \mathsf{F}^{\text{new}}$. Notice that for old facts, we still need to take the respective edge costs into account, even after the facts have already been achieved. To determine n for $\mathbf{v}$, we set $\mathsf{n} = \min_{j=1,\ldots,M}(\mathsf{n}_j + n_j)$. Then, $\mathcal{E}_{c,e}(s^+)(\mathbf{v}_i) = (\mathsf{F}, \mathsf{n})$.

Finally, we let $\mathcal{E}_{c,e}(s^+)$ denote the first component of the value $\mathcal{E}_{c,e}(s^+)(\mathbf{0})$, discarding the reachability cost of $\mathbf{0}$, and only keeping the reached facts with their associated costs.

Proposition 4. *Let s^+ be a relaxed state and $a = \langle p, e\rangle$ an action with effect e in ENF and cost function $c : S \to \mathbb{N}$. Let $\mathcal{E}_{c,e}$ be the product of an $\mathcal{E}_c$ encoding c and $\mathcal{E}_e$ encoding e. Let the evaluation procedure of $\mathcal{E}_{c,e}$ for relaxed states be as described above. Then $[e]^c_{s^+} = \mathcal{E}_{c,e}(s^+)$.*

Proof sketch. Both sides of the equality are by definition functional sets of fact-cost pairs (f, n) where each fact f occurs at most once. Functionality follows from the use of the minimizing union operator $\sqcup$ in both cases. We first argue that the sets of facts occurring in $[e]^c_{s^+}$ and $\mathcal{E}_{c,e}(s^+)$ are identical. This is easy to see: by definition, a fact f occurs in $[e]^c_{s^+}$ iff there is an unrelaxed state s with $s \models s^+$ such that the effect condition for f is satisfied in s. This is the same as saying that $\mathsf{f} \in [e]_s$ for some such s. This is equivalent to $\mathsf{f} \in \mathcal{E}_e(s)$ for such an s according to Prop. 2, which, according to the EVMDD product construction, is equivalent to f appearing as part of some edge label in $\mathcal{E}_{c,e}$ for an edge on a path corresponding to s. This, finally, is equivalent to f occurring in $\mathcal{E}_{c,e}(s^+)$, since during the evaluation procedure of $\mathcal{E}_{c,e}$, exactly the edges on paths corresponding to some s with $s \models s^+$ are traversed, and all visited effect edge labels are collected along the way and no fact is ever discarded.

Now that we know that the same facts are mentioned in $[e]^c_{s^+}$ and $\mathcal{E}_{c,e}(s^+)$, we still have to show that they are associated with the same costs in both. In $[e]^c_{s^+}$, for fact f, by definition this is the minimal cost $c(s)$ at which f can be achieved in any state s with $s \models s^+$. Let s be such a minimizer. We have to show that f is associated with the same cost in $\mathcal{E}_{c,e}(s^+)$. We know that $c(s)$ is the sum of edge labels in the cost EVMDD $\mathcal{E}_c$ for the path corresponding to s. By definition of the product construction, the same labels (and

therefore the same sum of labels) is also present for s in the product EVMDD $\mathcal{E}_{c,e}$.

Moreover, in the evaluation of $\mathcal{E}_{c,e}$, that path will also be traversed. The point at which f appears as an edge label may be anywhere on the path, not just on the last edge before the terminal node. The cost associated with f in $\mathcal{E}_{c,e}$ along that path is first determined after the edge where f appears as a label, and there it is the cost of the prefix of the path corresponding to s ending in the node after f has been set. By construction, for the prefix, the sum of costs is the same as the partial sum of costs in $c(s)$. From there, when f gets propagated further along the path suffix corresponding to s, the associated cost is always incremented accordingly, by adding n_j in the definition of $\mathsf{F}_j^{\text{old}}$. Also, the cost coming from s never disappears in a minimizing union operation, since s itself is a minimizer. This shows that $\mathcal{E}_{c,e}(s^+)(\mathsf{f}) \leq [e]^c_{s^+}(\mathsf{f})$. For the opposite direction, it suffices to note that if $\mathcal{E}_{c,e}(s^+)(\mathsf{f})$ were strictly smaller, then there would have to be a state s responsible for this, which would also have to be taken into account in $[e]^c_{s^+}(\mathsf{f})$, a contradiction. □

Since we never associate more fact-cost pairs to a node than there are facts, the evaluation procedure is clearly polynomial in the size of the planning task and the product EVMDD. To illustrate the evaluation, notice that the EVMDD from Fig. 4 is such a product EVMDD. Evaluating it for relaxed state s^+ with $s^+(x) = \{0, 1, 2\}$ means removing all arcs with a constraint on x inconsistent with s^+, i. e., the arcs for $x = 3$, $x = 4$, and $x = 5$. Then, at the decision node for x, we get the intermediate result $(\emptyset, 1)$. At the terminal node, we get $(\emptyset \sqcup \{(x' := 1, 1)\} \sqcup \{(x' := 2, 2)\} \sqcup \{(x' := 3, 3)\}, 1)$. Its first component is the same as $\{(x' := 1, 1), (x' := 2, 2), (x' := 3, 3)\} = [x' := x + 1]^c_{s^+}$.

Notice that, for a definition of optimal relaxed plans, we will have to associate costs to facts in relaxed states as well, and adapt Def. 4 accordingly.

Empirical Evaluation

While a complete analysis of h^+ and its approximations in the SDAC/CE setting is beyond the scope of this paper, we will discuss how the current state of the art for delete relaxation heuristics in this setting compares to our approach on a conceptual level, and give a few insights about their relation for some domains.

Currently, planning with delete-relaxation heuristics for domains incorporating state-dependent action costs and conditional effects in a planning system such as Fast Downward (Helmert 2006) consists of two steps: first, compile SDAC away (either with exponential overhead, or with a compilation based on EVMDDs (Geißer, Keller, and Mattmüller 2015)). Second, solve the resulting task (with constant costs and conditional effects) with forward search and delete-relaxation heuristics. Note that in order to compute delete relaxation heuristics for tasks with CE, Fast Downward internally performs another compilation (only used to compute the heuristic, not for the actual search): For each conditional effect $\varphi_i \rhd \mathsf{f}_i$, there is an action $\langle p \wedge \varphi_i, \mathsf{f}_i \rangle$, where p is the action precondition. We can therefore understand the overall process, where costs and conditional effects are considered separately (which leads to inaccurate heuristic values), as a two-step compilation; SDAC compilation is linear in the size of $\mathcal{E}_c$, and effect compilation is linear in the number of conditional effects. On the other hand, an approach which incorporates SDAC and CE based on their product EVMDD is bounded by the product of the sizes of $\mathcal{E}_c$ and $\mathcal{E}_e$. Thus, we want to analyze the size of effect EVMDDs compared to the number of conditional effects for practical planning domains. We have developed a tool to generate EVMDDs over arbitrary meet-semilattice ordered CMMs and analyzed the benchmark set of the International Planning Competition 2014. However, most conditional effects are removed by Fast Downward during preprocessing. Only the *Cave Diving* and *Citycar* domains have remaining conditional effects, but even for these there are less than 5 effects for each instance. Therefore, we consider two domains which incorporate not only conditional effects, but also state-dependent action costs.

ASTERIX	$\mathcal{E}_c$	$\mathcal{E}_e$	$\mathcal{E}_{c,e}$	PSR	$\mathcal{E}_c$	$\mathcal{E}_e$	$\mathcal{E}_{c,e}$
$\lvert\mathcal{E}\rvert$	18.78	41.44	57.89		22	22	44
$\lvert\mathcal{V}_\mathcal{E}\rvert$	2.67	1.78	3.11		11	11	22
$\lvert T\rvert$	1380.98	124.00	1380.98		2048	2048	4194304

Table 1: Average EVMDD sizes for two exemplary instances of *Asterix* and *Power Supply Restoration*.

The *Asterix* domain is a toy example, similar to our corridor example. Asterix has to gather the Edelweiss from a mountain; climbing the mountain depends on Asterix' current location. If Asterix does not have the magic potion, climbing gets harder at the peak, therefore the climb action has conditional effects as well as state-dependent action costs. In the second domain, *Power Supply Restoration*, the goal is to resupply a number of lines in a faulty electricity network (Thiébaux and Cordier 2001). The PDDL version of this domain only considers conditional effects, we augment the actions with state-dependent action costs, where an action costs less, if more lines are supplied with power. This corresponds to minimizing breakdown costs, as described in the original problem description.

We first compare the number of conditional effects to the size (number of nodes) of the largest effect EVMDD. For the *Asterix* domain, EVMDD size grows linearly to the number of conditional effects. For *Power Supply Restoration* we consider the wait action which has one conditional effect for each device. In this case the EVMDD size is equal to the number of conditional effects. Additionally, Tab. 1 reports the average size of the EVMDDs for c, e and their product, as well as the number of variables the functions depend on. We also provide a comparison if one would use a naïve representation instead of EVMDDs, i.e. the size of a simple lookup table storing the results, denoted as $|T|$ (which is exponential in $|\mathcal{V}_\mathcal{E}|$, as there is an entry for each state). For Asterix, we consider the hardest instance, results are of similar fashion for the other instances. While the largest effect EVMDD has nearly 800 nodes, the average size is still relatively small, despite variables having a domain size of 15. In PSR, costs and conditional effects depend on many more

(binary) variables. However, the structure of their functions results in linearly growing EVMDDs, which in turn results in a small and compact representation. This holds for all instances of PSR.

Discussion

In the literature, SDAC and CE were only discussed separately so far. In this paper, we demonstrated that they are, in fact, just two sides of the same coin. Whereas using EVMDDs to encode conditional effects works well for us, since besides often being compact, EVMDDs appear to be "heuristic-friendly", too, we have to point out, though, that EVMDDs are not the magic bullet for dealing with conditional effects. E. g., an EVMDD-based compilation of conditional effects can, in the worst case, become exponentially larger than Nebel's compilation (Nebel 2000). To see this, consider a conditional effect of the form $\varphi \rhd \mathsf{f}$, where φ is a propositional formula with an exponentially large decision diagram representation. Then, an EVMDD-based compilation will be exponential, whereas Nebel's compilation will be of constant size, since it only branches on the entire formula φ once, whereas EVMDDs may only branch on single variables in each step.

EVMDDs defined over structures other than $\mathbb{N}$ are not completely novel; EVMDDs defined over groups, instead of over CMMs, were already applied to model checking (Roux and Siminiceanu 2010). The attentive reader familiar with the *successor generator* (SG) in the Fast Downward planner will have noticed that EVMDDs over $\mathcal{F}$ are basically edge-valued SGs without don't-care branches.

Finally, when combining the decision diagrams for SDAC and CE, one has to carefully choose a common variable ordering. In particular, one should avoid interleaving variables that only occur in the cost function with variables that only occur in the effect conditions.

Conclusion

We defined a relaxed operator semantics in the presence of SDAC and CE that is closer to the unrelaxed semantics than an alternative naïve semantics where costs and effects are handled separately. Whereas the new semantics refers to exponentially many unrelaxed states, we proposed an EVMDD-based way of computing it that avoids this exponentiality in many practical cases.

We intend to build upon this work to derive informative relaxation heuristics, such as generalizations of the additive (Bonet, Loerincs, and Geffner 1997) or the FF (Hoffmann and Nebel 2001) heuristic. We believe that our encoding will also prove useful in the definition of abstraction heuristics, similarly as in previous work (Geißer, Keller, and Mattmüller 2016). Moreover, we will define and analyze action compilations based on product EVMDDs in a way similar to previous work on SDAC (Geißer, Keller, and Mattmüller 2015). Finally, we will also investigate admissible ways of keeping our EVMDDs small, possibly at the cost of some precision.

Acknowledgments. This work was partly supported by BrainLinks-BrainTools, Cluster of Excellence funded by the German Research Foundation (DFG, grant number EXC 1086). We thank the the anonymous reviewers for their insightful comments. We thank Malte Helmert and Florian Pommerening for answering our questions regarding Fast Downward.

References

Amer, K. 1984. Equationally complete classes of commutative monoids with monus. *Algebra Univers.* 18(1):129–131.

Ball, T.; Podelski, A.; and Rajamani, S. K. 2001. Boolean and cartesian abstraction for model checking C programs. In *Proc. TACAS 2001*, 268–283.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proc. AAAI 1997*, 714–719.

Ciardo, G., and Siminiceanu, R. 2002. Using edge-valued decision diagrams for symbolic generation of shortest paths. In *Proc. FMCAD 2002*, 256–273.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete relaxations for planning with state-dependent action costs. In *Proc. IJCAI 2015*, 1573–1579.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for planning with state-dependent action costs. In *Proc. ICAPS 2016*, 140–148.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J. 2005. Where "ignoring delete lists" works: Local search topology in planning benchmarks. *(JAIR)* 24:685–758.

Lai, Y.; Pedram, M.; and Vrudhula, S. B. K. 1996. Formal verification using edge-valued binary decision diagrams. *IEEE Transactions on Computers* 45(2):247–255.

Mattmüller, R.; Geißer, F.; Wright, B.; and Nebel, B. 2017. On the relationship between state-dependent action costs and conditional effects in planning. In *Proc. HSDIP 2017*.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *JAIR* 12:271–315.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In *Proc. ICAPS 2003*, 185–194.

Roux, P., and Siminiceanu, R. I. 2010. Model checking with Edge-valued Decision Diagrams. In *Proc. NFM 2010*, 222–226.

Seipp, J., and Helmert, M. 2013. Counterexample-guided Cartesian abstraction refinement. In *Proc. ICAPS 2013*, 347–351.

Thiébaux, S., and Cordier, M.-O. 2001. Supply restoration in power distribution systems — a benchmark for planning under uncertainty. In *Proc. ECP 2001*, 196–202.