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

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

Benedict Wright and **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 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 of length 6 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 $\mathit{cost}(\mathit{move\text{-}right}) = x + 1$. 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 $1 + 2 + 3 + 4 + 5 = 15$.

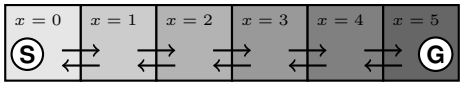

Figure 1: Corridor example. Initial position left, goal position right. The darker the grid cell, the more costly a movement out of this cell.

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 $\mathit{cost}(\mathit{move\text{-}right})(s^+) = 1$ from $0 \in s^+(x)$, but the next relaxed state will still include the value 3, because $2 \in s^+(x)$, meaning that we moved one cell to the right at

[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.

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 $x = 1$. This example can be generalized to show that 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. The key observation behind our proposed solution is that SDAC and CE are very closely related, as they can both 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 are used in various ways (Geißer, Keller, and Mattmüller 2015; 2016). The rest of the paper is concerned with how to formalize this idea and how to generalize it to arbitrary SDAC and CE.

[2]We use $\mathbb{N}$ instead of $\mathbb{Z}$ or even $\mathbb{Q}$, because having a well-founded set with a minimal element makes some later discussions a bit easier, and it is hardly a restriction of generality, since we often assume nonnegative action costs anyway, and fractional costs can still be approximated.

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* is a pair (v, d), where $v \in \mathcal{V}$ and $d \in \mathcal{D}_v$. We often refer to single facts as f and 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) $[e_1 \wedge \cdots \wedge e_n]_s = [e_1]_s \cup \cdots \cup [e_n]_s$.

(2) $[\varphi \rhd f]_s = \{f\}$ *if* $s \models \varphi$, *and* $[\varphi \rhd f]_s = \emptyset$, *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 $\mathcal{G} = (G, +, 0)$ consisting of a carrier set G, a binary operation $+$ on G, and an element $0 \in G$ such that $+$ is associative, and that 0 is the neutral element. Throughout the paper, we will also assume that the monoids we are concerned with are commutative.

Definition 2. *Let $\mathcal{G} = (G, +, 0)$ be a commutative monoid. An* EVMDD *over $\mathcal{G}$ and over $\mathcal{V}$ is a tuple $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$, where $\kappa \in G$ 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 $\mathcal{E}$, and $w_0, \ldots, w_k \in G$.*

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 $\mathcal{G}$ with carrier G and variables $\mathcal{V}$ denotes a function from the set of states S over $\mathcal{V}$ to G. 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 $\mathcal{G} = (G, +, 0)$ and $\mathcal{V}$ denotes the function $\kappa + f$ from the states over $\mathcal{V}$ to G, where f is the function denoted by $\mathbf{f}$. The terminal node $\mathbf{0}$ denotes the constant function 0, and $(v, \chi_0, \ldots, \chi_k, w_0, \ldots, w_k)$ 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 $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$, $\mathbf{f}$ 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 rectangular node labeled $\mathbf{0}$. Edge constraints d 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

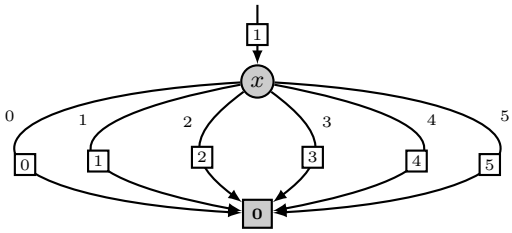

Figure 2: EVMDD over $\mathcal{N}$ for cost function $x + 1$.

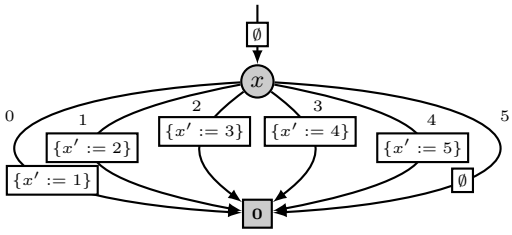

Figure 3: EVMDD over $\mathcal{F}$ for conditional effect $x' := x+1$.

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 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, depicted in 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 direct product $\mathcal{N} \otimes \mathcal{F}$ of $\mathcal{N}$ and $\mathcal{F}$. In this example, there is only one decision node with associated decision variable x in both diagrams, and therefore also just one product node.

To define when a (reduced ordered) EVMDD is canonical, we still 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 $\mathbf{v}$ still have in common that could not be taken care of earlier in the decision diagram. For $\mathcal{N} = (\mathbb{N}, +, 0)$, this means that the minimum edge weight of any edge leaving $\mathbf{v}$ is zero (and hence, any excess weight has been pulled upward into the incoming edge weight). Similarly, for $\mathcal{F} = (2^F, \cup, \emptyset)$, it means that the intersection of the labels of the edges leaving $\mathbf{v}$ is empty (and hence, all partial effects that happen for all possible

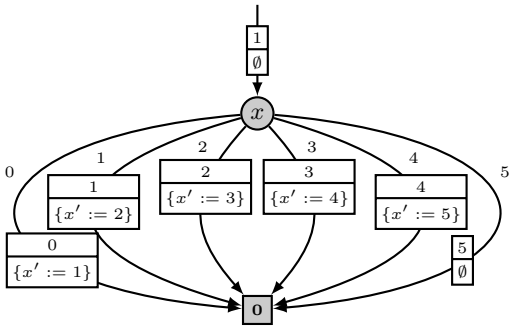

Figure 4: EVMDD over $\mathcal{N} \otimes \mathcal{F}$ for cost function $x + 1$ and conditional effect $x' := x + 1$ combined. Edge labels have their $\mathcal{N}$-part on top, and their $\mathcal{F}$-part at the bottom.

values of the current decision variable v are pulled upward into the incoming edge label). In general, it means that the EVMDDs have to respect a lattice order on their underlying monoid. A *meet-semilattice* is a partially ordered set $(G, \leq)$ which has a *greatest lower bound* for any nonempty finite subset $G' \subseteq G$, denoted by $\bigwedge G'$. A monoid $\mathcal{G} = (G, +, 0)$ is called *meet-semilattice ordered* if it comes with a partial order $\leq$ on G such that $(G, \leq)$ is a meet-semilattice, that the operation $+$ on G can be distributed over the greatest lower bound operator $\wedge$, and that $\bigwedge G = 0$. We will usually assume a meet-semilattice ordering implicitly without always mentioning it. It is easy to verify that both $\mathcal{N}$ with the natural order $\leq$ and the minimum operation $\min$ as greatest lower bound, and $\mathcal{F}$ with the subset relationship $\subseteq$ and the intersection operation $\cap$ as greatest lower bound are commutative meet-semilattice-ordered monoids.

With this, we can phrase the standard extra canonicity requirement for EVMDDs. Let $\mathbf{v} = (v, \chi_0, \ldots, \chi_k, w_0, \ldots, w_k)$ be a nonterminal node of an EVMDD over a meet-semilattice-ordered monoid $\mathcal{G} = (G, +, 0)$ with order $\leq$ and greatest lower bound $\bigwedge$. Then we require that $\bigwedge_{i=0,\ldots,k} w_i = 0$. In the following, we assume that all EVMDDs we deal with are canonical.

EVMDD Construction

In this section, we will discuss how EVMDDs over $\mathcal{N} \otimes \mathcal{F}$ 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.

Construction for State-Dependent Action Costs

The top-down EVMDD construction we sketch below is standard and known from the literature (Lai, Pedram, and Vrudhula 1996). It is basically the construction using repeated Shannon expansions into cofactors also known from BDDs (Bryant 1986), just with the additional requirement that the edge labels have to be set in the right way. Instead of describing the construction for arbitrary EVMDDs, we discuss it for EVMDDs over $\mathcal{N}$, and then explain how this generalizes to other monoids, in particular to $\mathcal{F}$. Let $c : S \to \mathbb{N}$ be the function we want to represent. For simplicity, let us also assume that c is a multivariate polynomial over the state variables given in canonical form as a linear combination of monomials, and that we re-establish this canonical form after each Shannon expansion. This allows us to easily identify whether two cofactors should be represented by the same decision node, which is the case iff the polynomials only differ in their constant subterm. Let $[c]$ be the equivalence class of all polynomials that differ from c only by an additive constant. We represent this class by c with the constant additive subterm set to zero, which we call $\tilde{c}$. Nodes in the decision diagram will represent such equivalence classes. Let $v_1, \ldots, v_n$ be the variable ordering. Then the EVMDD construction (ignoring possible Shannon reductions performed on the fly) given a polynomial c proceeds variable layer by variable layer. On each layer, we need decision nodes to represent a set of polynomials. For layers $i > 1$, this set will be determined by the previous layer(s). For layer $i = 1$, it is the singleton set $\{c\}$.

On layer $1 \leq i \leq n$, let $N = \{c_1, \ldots, c_n\}$ be the set of functions to be represented. First, we partition N into equivalence classes $\{\tilde{c}_1, \ldots, \tilde{c}_n\}$. For each c_i, let $\mathit{offset}_i = c_i - \tilde{c}_i$ be the additive constant that got dropped when representing c_i as $\tilde{c}_i$. For each representative $\tilde{c}$, we construct a decision node $\mathbf{v}(\tilde{c})$ associated with the current variable $v = v_i$ in the variable ordering. In order to obtain sub-EVMDDs for all outgoing (v, d)-branches at all nodes, at each node $\mathbf{v}(\tilde{c})$, we consider all *cofactors* $\tilde{c}|_{v=d}$ of $\tilde{c}$ for $d \in \mathcal{D}_v$, where $\tilde{c}|_{v=d}$ is obtained from $\tilde{c}$ by substituting value d for variable v and simplifying. Call those cofactors $\tilde{c}_0, \ldots, \tilde{c}_k$. On the next layer $i + 1$, we will have to construct sub-EVMDDs for the functions in the set $\bigcup_{c \in N} \{\tilde{c}_0, \ldots, \tilde{c}_k\}$, i. e., for all functions from the previous layer with possible values of the variable v plugged in. That recursive construction will return, for each cofactor $\tilde{c}_i$, $i = 0, \ldots, k$, of each function $c \in N$, a dangling edge with some weight w_i pointing to some successor node χ_i. Now, for each node $\mathbf{v}(\tilde{c})$, we make the outgoing edges point to successors χ_i, $i = 0, \ldots, k$. Each corresponding successor weight w_i gets replaced with $w_i - w$, where $w = \min_{i=0,\ldots,k} w_i$, to ensure that the minimal successor weight is zero. Finally, we have to pull excess weight upward. To do that, for each function $c_i \in N$, we return a new dangling edge pointing to node $\mathbf{v}(\tilde{c}_i)$ and carrying weight $w + \mathit{offset}_i$, i. e., the minimal weight w we had to pull upward from the children plus the weight representing the error we introduced by replacing c_i with its representative $\tilde{c}_i$. On the terminal layer $n + 1$, after all variables have been branched on, necessarily $c = \kappa$ is a constant. Therefore, we return the EVMDD whose dangling incoming edge immediately leads to the terminal node and carries label κ.

Notice that in this construction, we perform isomorphism reductions along the way. For Shannon reductions, all we have to do is skip branching on variables on which the current cofactor does not depend any more, or, equivalently, skip a decision node if all its children carry the same weight and lead to the same successor node.

Example 1. *To illustrate the construction, in Fig. 5 we depict an EVMDD over $\mathcal{N}$ with variable ordering x, y, z rep-*

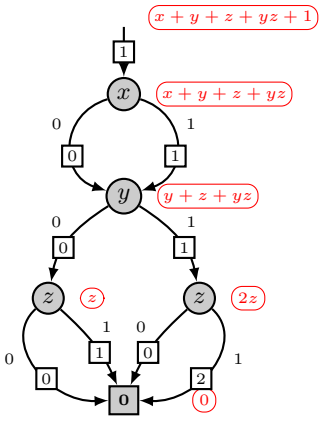

Figure 5: EVMDD for $x + y + z + yz + 1$.

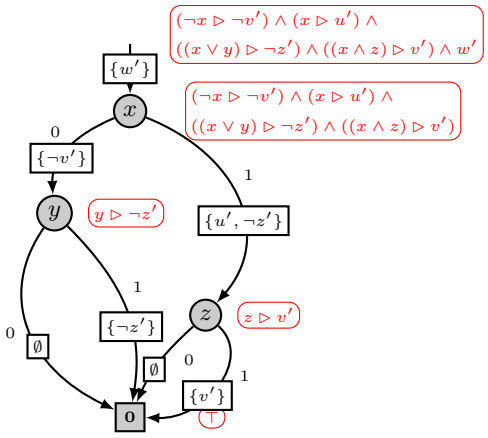

Figure 6: EVMDD for $(\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge ((x \wedge z) \rhd v') \wedge w'$.

resenting the function $c(x, y, z) = x+y+z+yz+1$, where the domains of all variables are binary. Importantly, the red annotations at all decision nodes (and in the beginning) are the respective cofactor representatives of the nodes. Following the unique path through the EVMDD corresponding to a given state s, say a state with $s(x) = s(y) = s(z) = 1$, and summing up the edge weights along the way, results in the correct function value, in this case $c(x, y, z) = 5$.

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 $c(s) = \mathcal{E}_c(s)$.* □

The construction works for any type of functions over states and corresponding sets of edge labels as long as we can (a) determine cofactors for given variable-value pairs $v = d$, (b) determine the edge labels, and (c) determine whether two decision nodes represent the same function. All this is simple for multivariate polynomials as above. Generally, every arithmetic function from states to natural numbers can be represented as a reduced EVMDD, even uniquely for a fixed variable ordering.

Construction for Conditional Effects

For CE, using the monoid $\mathcal{F} = (2^F, \cup, \emptyset)$, the construction works in the same manner. Let $e = (\varphi_1 \rhd e_1) \wedge \cdots \wedge (\varphi_n \rhd e_n)$ be an effect in ENF, and let $v = d$ be a fact. Then the cofactor $e|_{v=d}$ of e with respect to $v = d$ is e with truth ($\top$) substituted for all occurrences of $v = d$ and falsity ($\bot$) substituted for all occurrences of $v = d'$ for any $d' \neq d$ in any effect condition φ_i, $i = 1, \ldots, n$; and simplified. To obtain the edge labels, a sub-effect $w' := d'$ is moved into an edge label as soon as it becomes unconditional, and gets removed from the remaining cofactor. This is in analogy with the construction for cost functions, since we can consider two effects to be equivalent in the sense that they should be represented by the same decision node iff they only differ in their unconditional effects. Determining whether this is the case amounts to a syntactic comparison of the remaining cofactors, which is simple if the effects are in ENF and all conditions φ_i are also appropriately normalized. In summary, this means that every conditional effect can be represented as an EVMDD over $\mathcal{F}$.

Example 2. *Consider the conditional effect in ENF $e = (\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge ((x \wedge z) \rhd v') \wedge w'$. 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 primed variables in the edge labels (partial effects) help to distinguish them from their unprimed counterparts in the decision nodes (conditions). Fig. 6 depicts an EVMDD over $\mathcal{F}$ with variable ordering x, y, z, u, v, w representing the effect e. Again, the red annotations are the cofactor representatives of the nodes. Following the unique path through the EVMDD corresponding to a given state s, say a state with $s(x) = s(y) = s(z) = 1$, and taking the union of the edge labels along the way, results in the effect $\{w', u', \neg z', v'\}$.*

For CE, the semantics of an effect applied to a state is its change set $[e]_s$. Therefore, the analogue to Prop. 1 for CE 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 $[e]_s = \mathcal{E}_e(s)$.*

Proof sketch. The proof is by induction on the variable ordering $v_1, \ldots, v_n$, showing that on each level $i = 0, \ldots, n$ of $\mathcal{E}_e$, the partial union $\mathcal{E}_e^i(s)$ of edge labels following state s up to level i is the same as the partial change set $[e]_s^i$ up to level i. The partial change set $[e]_s^i$ is defined with clause (1) as in Def. 1, and with clause (2) replaced by clause (2′) $[\varphi \rhd f]_s^i = f$ if $\varphi|_{v_1=s(v_1),\ldots,v_i=s(v_i)}$ is a tautology, and $[\varphi \rhd f]_s^i = \emptyset$, otherwise. Note that $\varphi|_{v_1=s(v_1),\ldots,v_i=s(v_i)}$ is φ with the values that s assigns to the first i variables plugged in. This formula is a tautology iff it is already clear that the effect $\varphi \rhd f$ will fire after the first i variables in s have been evaluated. We show inductively that for all $i = 0, \ldots, n$, we have $[e]_s^i = \mathcal{E}_e^i(s)$. In the base case, both $[e]_s^0$ and $\mathcal{E}_e^0(s)$

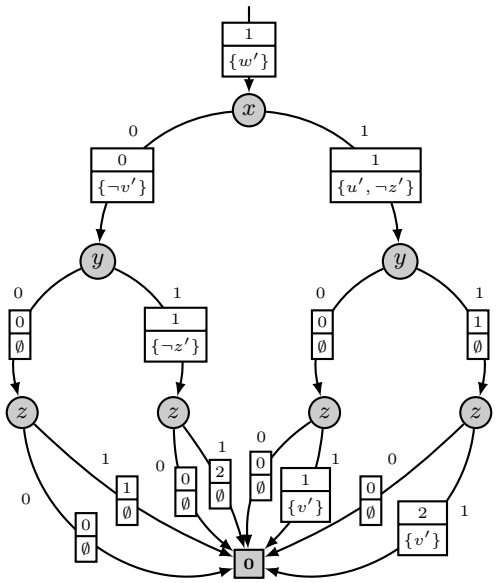

Figure 7: EVMDD for $x + y + z + yz + 1$ and $(\neg x \rhd \neg v') \wedge (x \rhd u') \wedge ((x \vee y) \rhd \neg z') \wedge ((x \wedge z) \rhd v') \wedge w'$ combined.

are the sets of unconditional effects of e; for $[e]^0_s$, because nothing gets substituted in φ, and for $\mathcal{E}^0_e(s)$, since this is the label of the dangling incoming edge of $\mathcal{E}_e$. In the inductive case, when going from $[e]^i_s$ to $[e]^{i+1}_s$ and from $\mathcal{E}^i_e(s)$ to $\mathcal{E}^{i+1}_e(s)$, in both cases exactly those facts are added that become unconditional when also setting v_{i+1} to $s(v_{i+1})$. In conclusion, since $[e]_s = [e]^n_s$ and $\mathcal{E}_e(s) = \mathcal{E}^n_e(s)$ (both obvious by definition), and $[e]^i_s = \mathcal{E}^i_e(s)$ for all $i = 0, \ldots, n$ (by the induction above), we also get $[e]_s = \mathcal{E}_e(s)$. □

Product EVMDDs

Before giving the general construction rule for product EVMDDs, let us have a look at an example.

Example 3. *Consider the two EVMDDs from Ex. 1 and 2. Their product is depicted in Fig. 7. The diagram contains a full binary tree over x, y, and z, 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.*

The roadmap for our description of the product construction will be as follows: we will first express both the cost EVMDD and the effect EVMDD as EVMDDs over the direct product $\mathcal{N} \otimes \mathcal{F}$. In the augmented cost EVMDD, each edge carries its old weight together with the empty set of effects, and similarly, in the augmented effect EVMDD, each edge carries zero weight together with its old partial effects. These two augmented EVMDDs can then be combined into a product using the general *apply* procedure (Lai, Pedram, and Vrudhula 1996) with the compound operator $(+, \cup)$ that adds first components and takes unions of second components. In other words, this application of $(+, \cup)$ means that we independently add costs (actual costs coming from one EVMDD, only zeroes coming from the other) and take unions of effects (only empty effects coming from one EVMDD, actual effects coming from the other).

Let us understand the *apply* procedure. For that, we assume that a state space S and a variable ordering are fixed. Let $\mathcal{G} = (G, +_G, 0_G)$ be a commutative monoid, and let $\circ$ be an operator on G, possibly different from $+_G$. Assume further that g_1 and g_2 are two functions from S to G, and that, by slight abuse of notation, we view $\circ$ also as an operator on functions from S to G in the obvious way via $(g_1 \circ g_2)(s) = g_1(s) \circ g_2(s)$. Furthermore, let $\mathcal{E}_{(\cdot)}$ be the construction that turns a function $g : S \to G$ into the reduced ordered EVMDD $\mathcal{E}_g$ representing it. We would like to have an operator $\circ_\mathcal{E}$ on EVMDDs over $\mathcal{G}$ (taking as input and returning such EVMDDs) that mimics the behavior of $\circ$ on EVMDDs, i. e., such that $\mathcal{E}_{(g_1 \circ g_2)} = \mathcal{E}_{g_1} \circ_\mathcal{E} \mathcal{E}_{g_2}$. The *apply* procedure does exactly that. In the literature, the application of $\circ$ on the EVMDD level, $\mathcal{E}_{g_1} \circ_\mathcal{E} \mathcal{E}_{g_2}$, is usually written as $\mathit{apply}(\circ, \mathcal{E}_{g_1}, \mathcal{E}_{g_2})$. Algorithmically, the *apply* procedure traverses both input EVMDDs $\mathcal{E}_{g_1}$ and $\mathcal{E}_{g_2}$ 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 again when the recursive computation has terminated. In the base case, when both EVMDDs only represent constant functions encoded in their bottom-most edge labels w_1 and w_2, those get combined into the new edge label $w_1 \circ w_2$. 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_G before proceeding.

Let $\mathcal{H} = (H, +_H, 0_H)$ be another commutative monoid, and assume that we want to take the product of two EVMDDs $\mathcal{E}_\mathcal{G}$ over $\mathcal{G}$ and $\mathcal{E}_\mathcal{H}$ over $\mathcal{H}$. Let us call $\mathcal{E}'_\mathcal{G} = \mathit{expr}_{1,\mathcal{G}\otimes\mathcal{H}}(\mathcal{E}_\mathcal{G})$ and $\mathcal{E}'_\mathcal{H} = \mathit{expr}_{2,\mathcal{G}\otimes\mathcal{H}}(\mathcal{E}_\mathcal{H})$ the results of expressing both EVMDDs over the direct product $\mathcal{G} \otimes \mathcal{H}$. The operations $\mathit{expr}_{i,\mathcal{G}\otimes\mathcal{H}}$ take their input EVMDD and leave its entire structure intact, but replace each edge label w, including the dangling incoming edge label κ, with the pair $(w, 0_H)$ for $i = 1$, or with the pair $(0_G, w)$ for $i = 2$, respectively. To take the product of $\mathcal{E}_\mathcal{G}$ and $\mathcal{E}_\mathcal{H}$, it now suffices to express both over $\mathcal{G} \otimes \mathcal{H}$, and then to apply the operation $+_{G\times H} = (+_G, +_H)$ to them:

$$\mathcal{E}_\mathcal{G} \otimes \mathcal{E}_\mathcal{H} := \mathit{expr}_{1,\mathcal{G}\otimes\mathcal{H}}(\mathcal{E}_\mathcal{G}) \; +_{G\times H} \; \mathit{expr}_{2,\mathcal{G}\otimes\mathcal{H}}(\mathcal{E}_\mathcal{H}).$$

The definition of product EVMDDs along with the observation that product EVMDDs can be constructed effectively allows us, in the following, to assume that we always have access to EVMDDs talking about SDAC and CE in one common structure. Next, we want to argue that this product construction does the right thing. For that, we show that by evaluating the product EVMDD for a state s, we still get the correct cost values and change sets back via projection.

Proposition 3. *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 $\mathcal{E}_{c,e}(s) = (\mathcal{E}_c(s), \mathcal{E}_e(s)) = (c(s), [e]_s)$.

Proof sketch. The second part of the equation immediately follows from Props. 1 and 2. For the first part, we assume without loss of generality that all EVMDDs in this proof are *quasi-reduced*, which means that every variable appears on every path through the EVMDD. This can always be achieved by undoing all Shannon reductions and inserting tests for all variables such that all outgoing arcs of those unnecessary decision nodes point to the same successor and carry the neutral element as edge labels. This only leads to a polynomial blowup and does not change the semantics.

Let $s \in S$ be an arbitrary state over the common state variables. Let $\mathcal{E}_c$, $\mathcal{E}_e$, and $\mathcal{E}_{c,e}$ be as stated in the proposition, but already quasi reduced. Let $\mathcal{E}'_c = \mathit{expr}_{1,\mathcal{N}\otimes\mathcal{F}}(\mathcal{E}_c)$ and $\mathcal{E}'_e = \mathit{expr}_{2,\mathcal{N}\otimes\mathcal{F}}(\mathcal{E}_e)$. When we construct $\mathcal{E}_{c,e} = \mathcal{E}'_c \otimes \mathcal{E}'_e$, the *apply* procedure recursively combines corresponding edge labels. Let $\mathbf{v}$ be a decision node constructed during the *apply* procedure, and let $\mathbf{v}_c$ and $\mathbf{v}_e$ be the original nodes in $\mathcal{E}'_c$ and $\mathcal{E}'_e$ from which it was constructed. Let $(w_c^0, \emptyset), \ldots, (w_c^k, \emptyset)$ and $(0, w_e^0), \ldots, (0, w_e^k)$ the outgoing edge weights of $\mathbf{v}_c$ and $\mathbf{v}_e$, respectively. By construction, they carry all neutral elements as one component. Then the outgoing edge labels of $\mathbf{v}$ will be $(w_c^0, w_e^0), \ldots, (w_c^k, w_e^k)$. No labels will be permanently shifted up or down during the procedure, since $\bigwedge_{i=0,\ldots,k}(w_c^i, w_e^i) = (\bigwedge_{i=0,\ldots,k} w_c^i, \bigwedge_{i=0,\ldots,k} w_e^i) = (\min_{i=0,\ldots,k} w_c^i, \bigcap_{i=0,\ldots,k} w_e^i) = (0, \emptyset)$. This holds, since we assumed that the original EVMDDs were canonical. The successor nodes of $\mathbf{v}$ for pairs of decision variable v and value d will also be the respective pairs, i. e., if $\mathbf{v}_c$ has children $\chi_c^0, \ldots, \chi_c^k$, and $\mathbf{v}_e$ has children $\chi_e^0, \ldots, \chi_e^k$, then $\mathbf{v}$ has children $(\chi_c^0, \chi_e^0), \ldots, (\chi_c^k, \chi_e^k)$.

This implies that, when computing $\mathcal{E}_{c,e}(s)$, we trace a sequence of edge labels $(w_c^0, w_e^0), \ldots, (w_c^n, w_e^n)$, from top to bottom such that the corresponding sequences we trace when computing $\mathcal{E}_c(s)$ and $\mathcal{E}_e(s)$ are $w_c^0, \ldots, w_c^n$, and $w_e^0, \ldots, w_e^n$, respectively. For $\mathcal{E}_{c,e}(s)$ we use the neutral element $(0, \emptyset)$ and component-wise addition $(+, \cup)$, so that we arrive at $\mathcal{E}_{c,e}(s) = (\sum_{j=0}^n w_c^j, \bigcup_{j=0}^n w_e^j)$, which is the same as $(\mathcal{E}_c(s), \mathcal{E}_e(s))$. □

We could alternatively have constructed the product EVMDD directly in the product space, taking the two types of cofactors for cost terms and conditional effects side by side in each step, fixing partial costs and partial effects in edge labels independently as soon as they are guaranteed to occur, and only identifying two nodes on the same level if both their remaining cost terms and their remaining effects are identical. Since this would also correctly encode costs and effects, the resulting EVMDD would be identical to $\mathcal{E}_{c,e}$. We opted for the product construction for clarity.

The size of a product EVMDD $\mathcal{E}_\mathcal{G} \otimes \mathcal{E}_\mathcal{H}$ is always bounded by the product of the sizes of the factors $\mathcal{E}_\mathcal{G}$ and $\mathcal{E}_\mathcal{H}$. Moreover, there are two special cases where the product construction incurs no blowup whatsoever. First, if $\mathcal{E}_\mathcal{G}$ and $\mathcal{E}_\mathcal{H}$ 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 $V_\mathcal{G}$ on which $\mathcal{E}_\mathcal{G}$ depends is disjoint from the set of variables $V_\mathcal{H}$ on which $\mathcal{E}_\mathcal{H}$ depends, and if $V_\mathcal{G}$ and $V_\mathcal{H}$ are *not* interleaved in the variable ordering, then $\mathcal{E}_\mathcal{G} \otimes \mathcal{E}_\mathcal{H}$ will essentially be $\mathcal{E}_\mathcal{G}$ and $\mathcal{E}_\mathcal{H}$ sequentially "glued together" in one way or the other, 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 $\mathcal{N} \otimes \mathcal{F}$. 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. Now, 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) $[\![\varphi \rhd f]\!]^c_s = \{(f, c(s))\}$ *if* $s \models \varphi$, *and* $[\![\varphi \rhd f]\!]^c_s = \emptyset$, *otherwise, and*
(3) $\bigsqcup_j E_j = \{(f, n) \in \bigcup_j E_j \mid \forall (f, \ell) \in \bigcup_j E_j : \ell \geq n\}$.

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 $[\varphi_1 \rhd f_1 \wedge \cdots \wedge \varphi_n \rhd f_n]^c_{s^+} = \bigsqcup_{s \in S : s \models s^+} \bigsqcup_{i=1,\ldots,n} [\![\varphi_i \rhd f_i]\!]^c_s$. 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. We would like to avoid this exponentiality whenever possible. This is where EVMDDs come into play. Below, we will describe a polynomial evaluation procedure for product EVMDDs $\mathcal{E}_{c,e}$ over costs and effects for relaxed states s^+ as input that returns $[e]^c_{s^+}$. The main complication behind the evaluation is that, in computing costs and effects for a relaxed state s^+, for costs we "minimize" ($\min$) over all s with $s \models s^+$ to get the cheapest costs, whereas for effects, we "maximize" ($\bigcup$) over all s with $s \models s^+$ to get all possible effects. This combination makes sense in a relaxed setting to retain all behavior from the unrelaxed setting at no higher costs. It also means, however, that we have to come up with a custom evaluation procedure for EVMDDs over $\mathcal{N} \otimes \mathcal{F}$ and relaxed states to reflect the described intuition.

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 $\mathbf{v}_n = \mathbf{0}$.

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 $\mathsf{F} = \{(f, n) \mid f \in F'\}$ 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 $\mathsf{F}^{\text{old}}_j = \{(f, n + n_j) \mid (f, n) \in \mathsf{F}_j\}$, $\mathsf{F}^{\text{new}}_j = \{(f, \mathsf{n}_j + n_j) \mid f \in F_j\}$, $\mathsf{F}^{\text{old}} = \bigsqcup_{j=1,\ldots,M} \mathsf{F}^{\text{old}}_j$, $\mathsf{F}^{\text{new}} = \bigsqcup_{j=1,\ldots,M} \mathsf{F}^{\text{new}}_j$, 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 $\mathcal{E}_{c,e} = \mathcal{E}_c \otimes \mathcal{E}_e$ be the product EVMDD of an EVMDD $\mathcal{E}_c$ encoding c and an EVMDD $\mathcal{E}_e$ 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 $f \in [e]_s$ for some such s. This is equivalent to $f \in \mathcal{E}_e(s)$ 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 at which f can be achieved in any state s with $s \models s^+$, i. e., $\min_{s \in S : s \models s^+ \text{ and } s \models \varphi} c(s)$ where φ is the effect condition of f. 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 weights in the cost EVMDD $\mathcal{E}_c$ for the path corresponding to s. By definition of the product construction, the same weights (and therefore the same sum of weights) 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. It is clear by construction that 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 $\mathsf{F}^{\text{old}}_j$. Also, the cost coming from s never disappears in a minimizing union operation, since s itself is a minimizer. This shows that $\mathcal{E}_{c,e}(s^+)(f) \leq [e]^c_{s^+}(f)$. For the opposite direction, it suffices to note that if $\mathcal{E}_{c,e}(s^+)(f)$ 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 $[e]^c_{s^+}(f)$, 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. A complete analysis of h^+ and its approximations in the SDAC/CE setting is beyond the scope of this paper and left for future work.

Discussion

In the literature, SDAC and CE were only discussed separately. In this paper, we demonstrated that they are, in fact, just two sides of the same coin. This makes us conjecture that, since EVMDDs seem to be an appropriate data structure to represent both, these decision diagrams might allow us to handle all kinds of state-dependent aspects of actions in a uniform way. We also have to point out, though, that EVMDDs are not the magic bullet for dealing with conditional effects. E. g., it is easy to see that 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 $\varphi \rhd w' := d'$, 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.

The attentive reader familiar with the *successor generator* (SG) in the Fast Downward planner (Helmert 2006) 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, making them compatible not only means making both edge-valued, but also making sure both use the same variable ordering. Practically, this implies that such an ordering needs to be chosen carefully. 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 EVMDD encoding will also prove useful in the definition of Cartesian abstraction heuristics, similarly as in previous work (Geißer, Keller, and Mattmüller 2016).

Moreover, we will define and analyze action compilations based on product EVMDDs. Similar to previous work on SDAC (Geißer, Keller, and Mattmüller 2015), those compilations will turn decision diagram edges into auxiliary actions with costs corresponding to the edge costs, and partial effects corresponding to the edge's effect label, additionally keeping track of the current node in the diagram and of the original preconditions and original effects, with a clean-up action in the end that copies the content of primed variables back to their unprimed counterparts. Finally, we will also investigate admissible ways of keeping our EVMDDs small, possibly at the cost of some precision.

Acknowledgements. This work was partly supported by BrainLinks-BrainTools, Cluster of Excellence funded by the German Research Foundation (DFG, grant number EXC 1086).

References

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.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

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.

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.

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.

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