The Frame Problem and Bayesian Network Action Representations

Craig Boutilier[1] and Moisés Goldszmidt[2]

[1] Dept. of Computer Science, University of British Columbia,
Vancouver, BC V6T 1Z4, CANADA
cebly@cs.ubc.ca
[2] Rockwell Science Center, 444 High Street, Palo Alto, CA 94301, U.S.A.
moises@rpal.rockwell.com

Abstract. We examine a number of techniques for representing actions with stochastic effects using Bayesian networks and influence diagrams. We compare these techniques according to ease of specification and size of the representation required for the complete specification of the dynamics of a particular system, paying particular attention the role of *persistence relationships*. We precisely characterize two components of the *frame problem for Bayes nets* and stochastic actions, propose several ways to deal with these problems, and compare our solutions with Reiter's solution to the frame problem for the situation calculus. The result is a set of techniques that permit both ease of specification and compact representation of probabilistic system dynamics that is of comparable size (and timbre) to Reiter's representation (i.e., with no explicit frame axioms).

1 Introduction

Reasoning about action has been a central problem in artificial intelligence since its inception. Since the earliest attempts to formalize this problem, the straightforward encoding of actions and their effects has been fraught with difficulties, such as the frame, qualification and ramification problems. Representations such as the situation calculus [20] and STRIPS [10], as well as various methodologies for using these systems (e.g., [16, 28, 1, 25, 14]) have been proposed for dealing with such issues. However, such problems are exacerbated by considerations such as nondeterministic or stochastic action effects, the occurrence of exogenous events, incomplete or uncertain knowledge, imprecise observations, and so on.

Increasing interest in stochastic and decision theoretic planning [8, 9], with the objective of incorporating the above considerations into planning systems, requires that attention be paid to the natural and effective representation of actions with stochastic effects. A number of researchers have adopted for this purpose a very efficient approach to representing and reasoning with probability distributions, namely *Bayesian networks* (BNs) [21]. BNs provide a formal, graphical way of decomposing a state of belief by exploiting probabilistic independence relationships. BNs can also be augmented to represent *actions*, for

instance, using the methods of *influence diagrams* (IDs) [27, 21],[3] or representations such as *two-stage* or *dynamic* BNs [7]. However, though considerable effort has been spent in characterizing the representational power of BNs in general, and developing good probabilistic inference algorithms that exploit the factored representations they provide, relatively little effort has been devoted to the study of the special features of action representations, especially with respect to classical problems such as the frame problem.

In this paper, we examine in detail the representation of actions in stochastic settings with respect to issues such as the frame and ramification problems (focusing primarily on the frame problem), providing some insight into how uncertain knowledge impacts the effort required to specify and represent actions. In particular, we provide a definition of (various aspects of) the frame problem in the context of dynamic BNs or IDs, proposing this as a standard against which future proposed solutions to the frame problem in stochastic environments can be measured. We also propose a methodology for representing actions in BNs in a very economical way, suggesting methods in which BNs and IDs can be augmented to exploit additional independencies (and better deal with ramifications) based on the rule structure that is taken for granted in nonprobabilistic representations. This bridges a wide gap between traditional probabilistic and nonprobabilistic approaches to action representation.

Our goal is to provide a detailed comparison of probabilistic and nonprobabilistic representations of actions, attempting to identify the key similarities and differences between these methods, and show the extent to which these different approaches can borrow techniques from one another. Space precludes a detailed survey and discussion of the work in this area and a number of interesting issues. We defer such discussion to a longer version of this paper [6], though we will point out some of these issues in the concluding section. In this paper, we concentrate on the issues of naturalness and compactness of action specification, and the frame problem in particular, focusing solely on the the situation calculus as the classical action representation, and the relatively elegant treatment of the frame problem proposed by Reiter [25]; from the probabilistic side, we deal exclusively with dynamic BNs and IDs. We emphasize several ways for augmenting dynamic BNs so that the size of representation and effort to specify the effects of actions in stochastic domains is essentially equivalent to that of Reiter's method.[4]

[3] IDs are representational tools used for optimal decision making in decision analysis. Actions are usually referred to as *decisions*, but for our purposes the two can be considered equivalent.

[4] There are a number of other representational methods that deserve analysis (e.g., the *event calculus* [17], the $\mathcal{A}$ language of [12] and its variants, probabilistic STRIPS rules [18, 2], probabilistic Horn rules [24]) which unfortunately we cannot provide here; but see the full paper [6].

2 Actions: Semantics and Basic Representations

2.1 Semantics

Before presenting various representations of actions, we present the semantic model underlying these representations, namely that of *discrete transition systems*, a view common in dynamical systems and control theory, as well as computer science.[5] A transition system consists of a set of *states* $\mathcal{S}$, a set of *actions* $\mathcal{A}$, and a *transition relation* T. Intuitively, actions can occur (or be executed) at certain system states, causing the state to change as described by the transition relation. The exact nature of the transition relation varies with the type of system (or our knowledge of it).

A *deterministic transition system* is one where T is a (possibly partial) function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. If $T(s, a) = t$, then t is the *outcome* of action a applied at s; and if $T(s, a)$ is undefined, then we take a to be impossible at s . If T is a relation over $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ then the system is *nondeterministic*: the *possible outcomes* of a at s are those states t such that $T(s, a, t)$ holds (if the set of outcomes is empty, a is not possible at s). Finally, a *stochastic transition function* is a function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$; the probability of outcome of state t resulting when a is applied at s is $Pr(s, a, t) \equiv_{\text{df}} T(s, a, t)$, the only requirement being that $\sum_t Pr(s, a, t) = 1$ for each state s, and action a applicable at s. If a is not applicable at s, we take $Pr(s, a, t) = 0$ for all t. We assume below that all actions can be applied, or attempted, at all states (perhaps with trivial effects). We note that this formulation assumes that the system is *Markovian*: the probability of moving to state t given a depends only on the current state s, not on past history.

The representation of actions in a transition system is relatively straightforward. In a deterministic system, each action a requires a tabular representation associating an outcome state with each state in $\mathcal{S}$. A nondeterministic action can be represented in a 0-1 matrix of size $|\mathcal{S}| \times |\mathcal{S}|$, where a 1 in entry (i, j) indicates that state s_j is a possible outcome of a at s_i. A stochastic action can be represented by a similar *stochastic matrix*, where entry (i, j) is the probability of transition from s_i to s_j.[6]

One difficulty with the direct semantic view of actions, from the point of view of problem specification and representation, is that AI problems (e.g., planning problems) are rarely described in terms of an explicit state space. Rather one imagines a set of propositions, or predicates and domain objects, or random variables, that describe the system under investigation; and actions are viewed in terms of their effects on these propositions. This view underlies almost all work in action representation in AI. We assume that a set of propositional atoms $\mathbf{P}$ characterize the system. The set of states induced by this language consists of

[5] Many of the ideas discussed here can be extended to continuous time, continuous state systems (see, e.g., [23] for continuous time extensions of the situation calculus, or [7] for continuous time action networks).

[6] Clearly if the branching factor b of nondeterministic or stochastic actions is small, sparse matrix methods can be used, requiring size $O(|\mathcal{S}|b)$ representations.

the set of truth assignments to $\mathbf{P}$, each a possible configuration of the system.[7] A state space that can be factored in this way will often permit compact representation of actions, as we now explore.

2.2 Situation Calculus

The situation calculus (SC) was among the first logical formalisms for representing (deterministic) actions adopted in AI [20] and continues to be the focus of much research [16, 28, 1, 25, 14]. We adopt a somewhat simplified version of SC here. SC is a typed first-order language with two classes of domain objects, *states* and *actions*, a function symbol *do* mapping state-action pairs into states, and a set of unary predicate symbols, or *fluents* corresponding to the propositions of the underlying problem, that take state arguments.[8] We write $do(a, s)$ to denote the *successor state* of state s when action a is performed, and write $F(s)$ to denote that fluent F is true in state s.

SC can be used to describe the effects of actions quite compactly, in a way that exploits regularities in the effects actions have on particular propositions. A typical *effect axiom* is:

$$\forall s \;\; holding(s) \wedge fragile(s) \supset broken(do(drop, s)) \wedge \neg holding(do(drop, s)) \quad (1)$$

which states that *broken* holds (e.g., of some object of interest), and *holding* doesn't, in the state that results from performing the action *drop* if it was held in state s and is fragile. Because of the Markovian assumption in our semantics, we assume that the only state term occurring in the antecedent is a unique state variable (e.g., s) and that each state term in the consequent has the form $do(a, s)$ for some action term a. Note that Axiom (1) describes a property of a large number of state transitions quite concisely; however, it does not uniquely determine the transitions induced by the action *drop*, a point to which we return below. Furthermore, it is a natural description of (some of) the effects of dropping an object.

2.3 Dynamic Bayesian Networks and Influence Diagrams

In a stochastic setting, the effect of an action a at a given state s_i determines a probability distribution over possible resulting states. With respect to the representation discussed in Section 2.1, row i of the stochastic matrix for a is the (conditional) distribution over the resulting states when action a is executed, given that s_i was the initial state. Given that states can be factored propositionally, and this distribution is in fact a joint distribution over $\mathbf{P}$, we would like to employ a representation that takes advantage of this factorization. Bayesian networks (BNs) [21] are one such representation.

[7] Multi-valued random variables are treated similarly.

[8] The restriction to unary predicates means that the underlying domain is described using propositions rather than predicates itself. We adopt this merely for simplicity of exposition — rarely is the assumption made in practice.

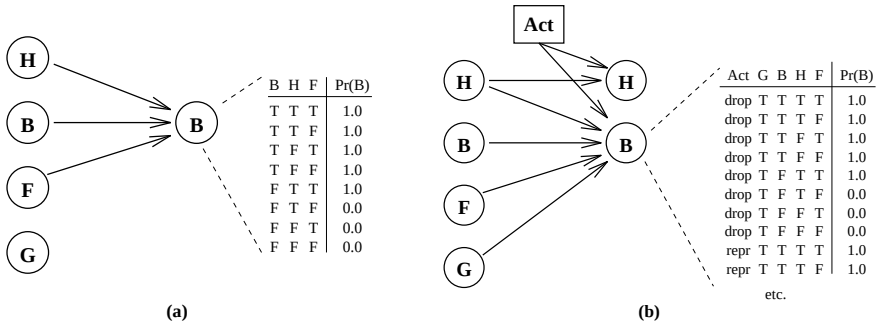

Fig. 1. Bayes Nets: (a) Without Action Node; (b) With Action Node

A BN is a directed acyclic graph (dag), where nodes represent the random variables of interest (in this case the fluents) and the arcs represent direct influences or dependencies between random variables. A BN encodes the following assumption of probabilistic independence: any node in a BN is probabilistically independent of its non-descendants, given the state of its parents in the network.[9] A BN can compactly capture a distribution P by representing independence relationships in its graphical structure. To represent the distribution, we annotate each node in the network with a *conditional probability table* (CPT) denoting the conditional distribution of each variable given its parents in the network. Given the assumptions of independence, any state of belief can be simply computed by multiplying entries in these tables.

The BN (or fragment of a BN) corresponding to Axiom (1) is pictured in Figure 1(a). In general, to represent the effect of action a, we have a set of variables corresponding to certain fluents in the state prior to the performance of the action, and variables corresponding to fluents after the action. In our example, the B on the left denotes the proposition $broken(s)$, while the B on the right denotes $broken(do(drop, s))$. The arcs indicate that $Pr(broken(do(drop, s)))$ depends on the truth or falsity of *broken*, *holding* and *fragile* in state s, but does not depend on the value of a fourth fluent G ($hasglue$), nor on the values of fluents in state $do(drop, s))$ given that one knows their values in s. The CPT on the right denotes the magnitude of this influence on *broken*: for each assignment to its parents, the probability that $broken(do(drop, s))$ is true must be specified (here the effect is deterministic).

Several remarks are in order at this point. First, we do not require prior probabilities on the "pre-action" nodes in the network. If they existed, we could use standard techniques to determine the probability of any post-action node. Thus, this network does not represent a complete probability distribution. The intent is to represent the stochastic matrix for a given action; therefore the network is schematic in the sense that it describes the effect of the action for

[9] The details of algorithms for testing independence is beyond the scope of this paper. We refer to [21][Chapter 3] for details.

any state, or assignment to pre-action variables. This coincides with the classical view of action representation. Second, such a network is sometimes called a *dynamic BN* or *two-stage BN* [7], since it should be viewed as schematic across time. The restriction to two stages (corresponding to states s and $do(a, s)$) is appropriate given our semantics. We also point out that, as with our description of the situation calculus, such a BN does not uniquely specify a transition matrix for action a (more in the next section).

Finally, *influence diagrams* (IDs) [13, 27] have been used in probabilistic inference and decision analysis to represent decision problems. Similar in structure to BNs, they have additional types of variables, represented by *value nodes* and *decision nodes*; we are only interested in decision nodes here. A decision node (or *action node*) is a random variable denoting the action that was executed causing the state transition. This is denoted by the square node in Figure 1(b), a variable *Act* that can take on possible actions as values (in this case, *repair*, *drop*). Since the state of a fluent after an action is executed will depend on the action chosen, the additional dependencies between *Act* and the other fluents must be represented. We examine the impact of these additional influences below.

3 Single Actions: Structure and the Frame Problem

As alluded to above, neither the SC nor BN description of action effects will generally provide a complete and unambiguous specification of a transition system. We focus first on the SC formulation, and compare the usual BN methods.

3.1 Frame Axioms in the Situation Calculus

There are two difficulties with Axiom (1) as a specification of the action *drop*. The first is that, while it describes the effect of *drop* on the fluents *holding* and *broken* (under some conditions), it fails to describe its effect on other fluents. For example, to completely specify the transition function for *drop*, one must also assert the effect *drop* has on other fluents in the domain (such as *hasglue*). Unfortunately, there are a typically a great many fluents that are completely unaffected by any given action, and that we do not consider part of the natural specification of an action's effects. Intuitively, we would like the user to specify how the action influences affected fluents, and assume that other fluents persist. We call this the problem of *persistence of unaffected fluents* (PUF). A second difficulty is that while Axiom (1) describes the effect of *drop* on *holding* and *broken* under the condition $holding(s) \wedge fragile(s)$, it fails to specify what happens when this condition is false. Once again, it is usually taken to be desirable not to force the user to have to say a fluent is unaffected in other circumstances, leaving it as a tacit assumption. We call this the problem of *persistence of affected fluents* (PAF).

The *frame problem* [20] is that of easing the user from the burden of having to specify conditions under which an action does not affect a fluent: PUF and PAF are two instances of this problem. One possible solution is to provide a means

to automatically derive explicit *frame axioms* given the user's specification. For example, given the input

$$\forall s \;\; holding(s) \wedge fragile(s) \supset broken(do(drop, s)) \wedge \neg holding(do(drop, s)) \quad (2)$$

$$\forall s \;\; holding(s) \wedge \neg fragile(s) \supset \neg holding(do(drop, s)) \quad (3)$$

one could, under the assumption that this describes *all* effects of *drop*, generate axioms such as

$$\forall s \;\; (\neg holding(s) \vee \neg fragile(s)) \wedge broken(s) \supset broken(do(drop, s)) \quad (4)$$

$$\forall s \;\; (\neg holding(s) \vee \neg fragile(s)) \wedge \neg broken(s) \supset \neg broken(do(drop, s)) \quad (5)$$

$$\forall s \;\; hasglue(s) \supset hasglue(do(drop, s)) \quad (6)$$

$$\forall s \;\; \neg hasglue(s) \supset \neg hasglue(do(drop, s)) \quad (7)$$

Axioms (4) and (5) deal with the PAF problem, while axioms (6) and (7) handle PUF. In general, we require $2|\mathbf{P}|$ such frame axioms, describing the lack of effect of a on each fluent in $\mathbf{P}$; or $2|\mathbf{P}||\mathcal{A}|$ such axioms for the entire set of action $\mathcal{A}$.

Other approaches deal not just with the specification problem, but also with the sheer number of axioms required. One example is the solution proposed by Reiter [25], extending the work of Pednault [22] and Schubert [26]. The aim is to directly encode the "assumption" that all conditions under which an action affects a fluent have been listed. This is accomplished by building a disjunction of all the conditions under which an action A affects a fluent F, asserting the F changes as dictated when these conditions hold, and that it retains its value otherwise. More precisely, let $\gamma^{+}_{F,A}(s)$ (resp., $\gamma^{-}_{F,A}(s)$) denote the disjunction of the antecedents of the effect axioms for action A in which F appears positively (resp., negatively) in the consequent. We know that

$$\forall s \;\; \gamma^{+}_{F,A}(s) \supset F(do(A, s)) \quad \text{and} \quad \forall s \;\; \gamma^{-}_{F,A}(s) \supset \neg F(do(A, s))$$

both follow from the action specification. Under natural conditions, we can ensure the persistence of F under all other conditions by writing:

$$\forall s \;\; F(do(A, s)) \equiv \gamma^{+}_{F,A}(s) \vee (F(s) \wedge \neg\gamma^{-}_{F,A}(s))$$

If we assert one such axiom for each fluent F, it is not hard to see that we uniquely determine a (deterministic) transition function for action A over the state space. In our example, these *closure axioms* for action *drop* (with some simplification) include:

$$\forall s \;\; holding(do(drop, s)) \equiv \bot \quad (8)$$

$$\forall s \;\; broken(do(drop, s)) \equiv [(holding(s) \wedge fragile(s)) \vee broken(s)] \quad (9)$$

$$\forall s \;\; hasglue(do(drop, s)) \equiv hasglue(s) \quad (10)$$

We require $|\mathbf{P}|$ axioms to characterize an action in this way. This is not a substantial saving over the use of explicit frame axioms, for we require $|\mathcal{A}|$ such axiom sets (one per action). However, as we see below, Reiter's method

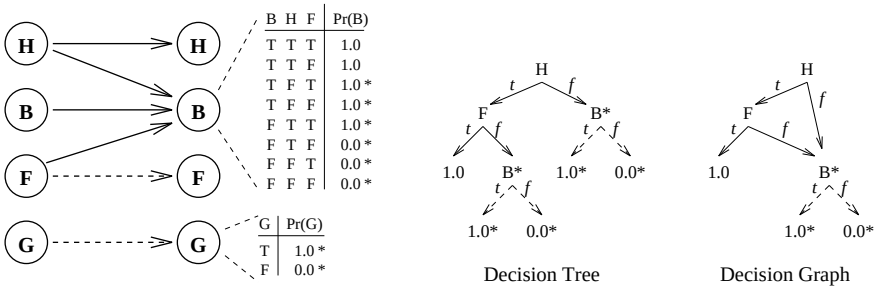

Fig. 2. (a) A Dynamic BN, and (b) Compact Representation of the CPT

avoids the repetition of these axioms in multiple action settings. The size of the axioms is also of interest. Imagine some (presumably small) number f of fluents is affected by a, that the average action condition has c conjuncts, and that affected fluents appear in an average of e effect axioms. Then we can expect the f closure axioms for affected fluents to be roughly of size ce, while the remaining $|\mathbf{P}| \perp f$ axioms are of constant size.

3.2 The Frame Problem in Bayesian Networks

The BNs shown in Figure 1 do not completely characterize a transition system for the same reasons described for SC above. Unlike the work in classical reasoning about action, the use of BNs for representing dynamical systems typically assumes that a complete specification of the action's effects, on all variables and under all conditions, is given. A (complete) dynamic BN for the action *drop* is shown in Figure 2(a) (we have left out CPTs for H, F).[10]

Most BN models of action assume this network is explicitly specified by the user; however we have used broken arcs in Figure 2(a) to indicate persistence relations among unaffected fluents (the CPT for node G shows persistence explicitly). Furthermore, the starred entries in the CPT for node B denote the persistence of fluent *broken* under the condition $\neg holding \vee \neg fragile$. The fact that a user must specify these persistence relationships explicitly is the obvious counterpart of the PUF and PAF problems in the situation calculus. Therefore, we can take the *frame problem for Bayesian networks* to be exactly the need to make these relationships explicit.

As described above, there are two possible perspectives on what constitutes a solution to this problem: relieving the user of this burden, and minimizing the size of the representation of such persistence. The first type of solution is not especially hard to deal with in this representation. A rather simple idea is to have the user specify only the unbroken arcs in the network and only those unhighlighted probabilities in the CPTs. It is a simple matter to then automatically add persistence relationships.

[10] Although not used yet, arcs are allowed between post-action variables (see below, on ramifications).

3.3 Adding Further Structure to Bayesian Networks

The size of the dynamic BN for an action (regardless whether the persistence relations are generated automatically) is then comparable to that of Reiter's solution (with one substantial proviso). Again, assume that f fluents are affected by a, and that the average fluent is influenced by c preaction fluents. The CPTs for each of the f affected fluents will be of size 2^c, whereas the remaining $|\mathbf{P}| \perp f$ CPTs are of constant size. The important factor in the comparison to the Reiter's solution is the difference in the affected fluent representation, with this method requiring a representation of size roughly $f \cdot 2^k$, while SC requires a representation of size fce. Here k is the number of *relevant* preaction fluents for a typical affected fluent F, those that are part of *some* condition that influences the action's effect on F (i.e., the number of parents of F in the BN). Note that $k \leq ce$. The exponential term for the BN formulation is due to the fact that CPTs require a distribution for the affected fluent for *each assignment to its parents*. For instance, since B, F and H are relevant to predicting the effect of *drop* on *broken*, we must specify this effect for all eight assignments to $\{B, F, H\}$.

Axiom (2), and more significantly the closure Axiom (9), are much more compact, requiring only that the single positive effect condition be specified, with persistence under other circumstances being verified automatically. CPTs in BNs are *unstructured*, and fail to capture the regularities in the action effects that fall out naturally in SC. Only recently there is work attempting to represent regularities in BN matrices [11], and in particular, structures such as logical formulae, decision trees and rooted decision graphs have been explored as compact representations for matrices [3, 5]. Examples of representations that capture this regularity are the decision tree and decision graph shown in Figure 2(b), corresponding to the original CPT for variable B. The broken arrows indicate the persistence relationships that can be added automatically when left unspecified by the user.

In general, it will be hard to compare the relative sizes of different representations, and the logic-based method of Reiter, since this depends crucially on exact logical form of the action conditions involved. For instance, decision trees can be used to represent certain logical distinctions very compactly, but others can require trees of size exponentially greater than a corresponding set of logical formulae. However, one can also use graphs, logical formulae and other representations for CPTs—each has particular advantages and disadvantages with respect to size and speed of inference [5]. For example, Poole [24] has used Horn rules with probabilities as a way of representing Bayes nets. In general, we can see that appropriate representations of CPTs can exploit the same regularities as logical formulae; thus BNs augmented in this way are of comparable size to Reiter's model.

4 Multiple Actions: The Frame and Ramification Problems

4.1 Compact Solutions to the Frame Problem

While the proposals above for individual actions relieve the user from explicitly specifying persistence, it did little to reduce the size of the action representation (compared to having explicit frame axioms). In Reiter's method, each action requires $|\mathbf{P}|$ axioms, with f axioms having size ce and $|\mathbf{P}| \perp f$ axioms having constant size. To represent a transition system with action set $\mathcal{A}$ thus requires $|\mathcal{A}||\mathbf{P}|$ axioms. This is only a factor of 2 better than using explicit frame axioms.

Fortunately, Reiter's solution is designed with multiple actions in mind. Reiter exploits the fact that, since they are terms in SC, one can quantify over actions. His procedure will (under reasonable conditions) produce one axiom of the form

$$\forall s, a \quad F(do(a,s)) \equiv \gamma_F^+(a,s) \vee (F(s) \wedge \neg\gamma_F^-(a,s))$$

for each fluent F. Here $\gamma_F^+(a,s)$ denotes the disjunction of the formulae $a = A \wedge \gamma_{F,A}^+(s)$ for each specific action A which affects fluent F positively (similar remarks apply to $\gamma_F^-(a,s)$). Thus, we see instead of having $|\mathcal{A}||\mathbf{P}|$ axioms, we have only $|\mathbf{P}|$ axioms, and the axiom for fluent F contains only reference to actions that influence it. If each fluent is affected by n actions (presumably n is much smaller than $|\mathcal{A}|$), each action condition has c conjuncts, and each affected fluents appear in e effect axioms for any action, then we expect this specification of the transition system to be of size $|\mathbf{P}|nce$. In our example, the axiom for fluent *broken* is:

$$\begin{aligned}\forall s \quad & broken(do(a,s)) \equiv [(a = drop \wedge holding(s) \wedge fragile(s)) \\ & \vee (\neg(a = repair \wedge holding(s) \wedge fragile(s)) \wedge broken(s))] \end{aligned} \tag{11}$$

Notice that actions like *paint* and *move* that have no influence (positive or negative) on *broken* are not mentioned.

Similar problems plague BNs when we consider the size of the representation of all actions in a transition system. Having a separate network for each action will require a representation of size $|\mathcal{A}|(2(|\mathbf{P}| \perp f) + f \cdot 2^k)$ (where f is the expected number of fluents affected by an action, k the number of conditions relevant to each affected fluent). The usual way to represent a transition system is to use an action node and condition each post-action variable on the action variable (we assume every variable can be influenced by some action). Figure 3 shows our example in this format.

The difficulty with this representation is the fact that, since (most or) every fluent is affected by some action, the action node becomes a parent of each post-action node, increasing the size of each CPT by a factor of $|\mathcal{A}|$ (as if we had a separate network for each action). Indeed, this representation, standard in decision analysis, is in fact worse, because any preaction variable that influences a post-action variable under *any* action must be a parent. Since the size of the representation is exponential in the number of parents, this will virtually

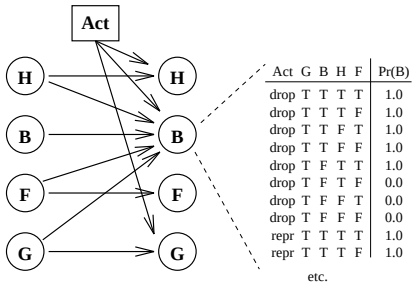

Fig. 3. A Dynamic BN with Action Node

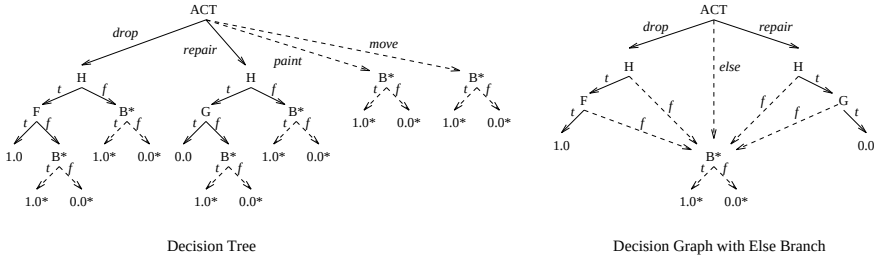

Fig. 4. Compact Representation of *broken* CPT

always be a substantially less compact representation than that required by a set of $|\mathcal{A}|$ action networks. In general, it will have size $|\mathcal{A}||\mathbf{P}|2^m$, where m is the expected number of fluents that are relevant to a post-action node under *any* action (typically much larger than k above). In this example, G is a parent of B (because it is relevant for *repair*), and its value must be considered even when we specify the effect of *drop* (for which it is actually not relevant).

In order to alleviate this problem we can use compact representations such as decision trees or decision graphs as shown in Figure 4. The broken arcs and marked nodes represent persistence relations that can be generated automatically if unspecified by the user. The tree representation provides some savings, but requires that each fluent have a separate subtree for each possible action, failing to alleviate the $|\mathbf{P}||\mathcal{A}|$ factor in the size of the representation. However, a decision graph can provide considerable savings, as shown, especially if we allow an "else" branch to exit the Act node (see Figure 4). This technique allows one to specify only the actions that actually influence a fluent, and only the conditions under which that effect is nontrivial, leaving the persistence relation to be filled in automatically on the "else" branch (all other actions) and for unspecified conditions for the mentioned actions. Using a graph rather than a tree means that all unfilled branches will connect to the (single) persistence subgraph. It is

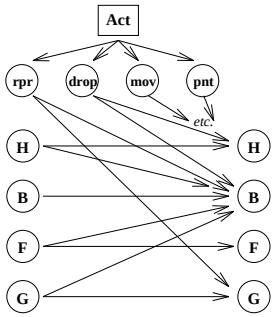

Fig. 5. The Use of Occurrence Variables

not hard to see that (assuming conjunctive action conditions) that the size of this representation will be exactly the same as that proposed by Reiter.

Another compact representation of multiple actions that does not require the use of else branches involves the use of proxy nodes that we called "occurrence variables". Occurrence variables are new nodes corresponding to the proposition that a given action has taken place (see Figure 5). Instead to being directly connected to the action node, each fluent has as parents the occurrence variables for actions that directly affect them. The occurrence variables signify that a particular A has "occurred" and are children of the action (or decision) node. Assuming that exactly one occurrence variable is true at any time, this representation permits CPTs to be of roughly the same size those using graphs with else branches, and more compact than graphs without else branches.

4.2 The Ramification Problem

To this point, we have ignored the issue of ramifications. Space precludes a full discussion of this problem (see [6]), but we mention the main issues. Ramifications or *domain constraints* are *synchronic* constraints on possible state configurations (as opposed to *diachronic* constraints – of the type we've discussed – that relate features of one state to features of its successor). When representing multiple actions, domain constraints allows compact specification of regular effects that are independent of the action being performed. For instance, if the location of the contents of a briefcase are always the same as the location of the briefcase itself, a statement to this effect relieves one from explicitly stating that every action a that changes the location of the briefcase also changes the location of it contents. We would like this effect (or ramification) of moving, dropping, throwing or otherwise dislocating a briefcase to be derived from the effect these actions has on the briefcase itself.

In SC, domain constraints can easily be expressed; for instance:

$$\forall s \quad in(x, b, s)) \supset loc(x, s) = loc(b, x) \tag{12}$$

The ramification problem has to do with the interaction of such constraints with possible solutions to the frame problem. Solutions have been proposed by Lin and Reiter [19], Kartha and Lifschitz [15] among others.

A domain constraint is represented in a BN as an arc between two post-action variables, representing the dependency between two fluents in a single state. Note that the constraints imposed by the limited language in a BN, plus the restriction on the acyclicity of the underlying graphs limits some of the problems of including ramifications. In particular we only have to worry about modifying the specification of an action whenever a synchronic fluent becomes a new parent. Still the problem is similar to the case of SC: how to specify such domain constraints independently of a particular action, and then impose these relationships on the action network(s) in such a way that automatically derived persistence relationships account for these ramifications directly. The general ideas used for SC solutions (especially those based on "compiling" ramifications into effect axioms automatically [19]) can be applied to the BN case and are discussed in the full paper.

5 Concluding Remarks

In this paper we have taken a close look at representational issues in encoding actions in stochastic domains using Bayesian networks. In the process we defined two aspects of the frame problem for BNs, proposed possible solutions, and compared these to Reiter's method for SC. We have demonstrated that the usual models of stochastic actions have not provided the compact or natural specification methodology provided in the classical/logical setting; but that the use of "automatic" filling in of persistence relationships and the use of the compact CPT representations recently adopted in [3, 5] allow solutions to the frame problem of similar size, structure and timbre to Reiter's. In this sense, we have proposed a starting point for a methodology for the natural, compact representation of actions in BNs. The advantages of BNs as representations for probabilistic actions will be enhanced by the incorporation of such techniques.

An important issue that deserves further attention is that of nondeterminism and correlated effects. BNs are designed to take exploit independence in action effects, since this is the only way to have compact representation when effects are possibly correlated, as they can be when any sort of nondeterminism comes into play. Thus, while representing ramifications as synchronic constraints are a convenience in deterministic settings (one can do without these at the expense of additional axioms), they *must* be used in any nondeterministic representation where correlated effects are possible. Thus, the methodology reflected in BNs has an important role to play in informing the extension of classical representations such as SC to handle nondeterministic actions.[11] We discuss this further in [6].

Finally, we shouldn't forget that BNs are designed to facilitate efficient inference. The use of dynamic BNs in inference is very flexible—standard BN

[11] Nondeterministic actions are addressed in, e.g., [15]; the difficulties with correlations are addressed in [4].

algorithms can be used to answer queries with respect to temporal projection and explanation, for action sequences of arbitrary (finite) length, and can be used for plan generation [3, 27]. In this regard, an important question is whether the compact representations of actions proposed here can enhance computation for probabilistic queries about the effects of actions, temporal projection, and planning in stochastic domains. Certain aspects of this question are investigated in [5] (in the context of BNs without actions) and [3] (with dynamic BNs).

Acknowledgements

Thanks to David Poole, Nir Friedman and especially all the participants of the 1995 AAAI Spring Symposium on Extending Theories of Action for their spirited and enjoyable discussion of these topics. Craig Boutilier acknowledges the support of NSERC Research Grant OGP0121843 and NCE IRIS-II Program IC-7. The work of Moisés Goldszmidt was funded in part by ARPA contract F30602-95-C-0251.

References

1. Andrew B. Baker. Nonmonotonic reasoning in the framework of the situation calculus. *Artificial Intelligence*, 49:5–23, 1991.
2. Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, Seattle, 1994.
3. Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, Montreal, 1995.
4. Craig Boutilier and Nir Friedman. Nondeterministic actions and the frame problem. In *AAAI Spring Symposium on Extending Theories of Action: Formal Theory and Practical Applications*, pages 39–44, Stanford, 1995.
5. Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. (manuscript), 1996.
6. Craig Boutilier and Moisés Goldszmidt. A comparison of probabilistic and non-probabilistic action representations. (manuscript), 1995.
7. Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.
8. Thomas Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, San Mateo, 1991.
9. Steve Hanks (ed.). Decision theoretic planning: Proceedings of the aaai spring symposium. Technical Report SS-94-06, AAAI Press, Menlo Park, 1994.
10. Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.
11. Dan Geiger and David Heckerman. Advances in probabilistic reasoning. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 118–126, Los Angeles, 1991.

12. Michael Gelfond and Vladimir Lifschitz. Representing actions in extended logic programming. In K. Apt, editor, *Proceedings of the Tenth Conference on Logic Programming*, pages 559–573, 1992.
13. Ronald A. Howard and James E. Matheson, editors. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, Menlo Park, CA, 1984.
14. G. Neelakantan Kartha. Two counterexamples related to Baker's approach to the frame problem. *Artificial Intelligence*, 69:379–392, 1994.
15. G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 341–350, Bonn, 1994.
16. Henry A. Kautz. The logic of persistence. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 401–405, Philadelphia, 1986.
17. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
18. Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic least-commitment planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1073–1078, Seattle, 1994.
19. Fangzhen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.
20. John McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.
21. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.
22. Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 324–332, Toronto, 1989.
23. Javier A. Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, University of Toronto, 1994.
24. David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.
25. Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy)*, pages 359–380. Academic Press, San Diego, 1991.
26. Lenhart K. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, Boston, 1990.
27. Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 33(6):871–882, 1986.
28. Yoav Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, 1988.

This article was processed using the LaTeX macro package with LLNCS style