Extending the Role of Causality in Probabilistic Modeling

Joost Vennekens, Marc Denecker, and Maurice Bruynooghe
{joost, marcd, maurice}@cs.kuleuven.be
Dept. of Computer Science, K.U.Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium

Abstract

Causality plays an important role in probabilistic modeling. Often, a probability distribution can be naturally described as the outcome of a causal process, in which different random variables interact through a series of non-deterministic events. However, formal tools such as Bayesian networks do not directly represent such events, but focus instead on derivate concepts such as probabilistic independencies and conditional probabilities. In this paper, we present a logic, designed from fundamental causal principles, which has a representation of such non-deterministic, probabilistic events as its basic construct. We show that Bayesian networks can be described in this language and illustrate some of its interesting properties. We then relate this logic to a certain class of probabilistic logic programming languages. We show that our logic induces a semantics for disjunctive logic programs, in which these represent non-deterministic processes. We show that logic programs under the well-founded semantics can be seen as a language of deterministic causality, which we relate to McCain & Turner's causal theories.

Introduction

Causal information plays a crucial role in commonsense reasoning in general and probabilistic modeling in particular. The underlying assumption in work such as Pearl's influential treatment of causality (Pearl 2000) seems to be the assumption that the world can be viewed as consisting of a number of non-deterministic causal events. However, the formal tools that are typically used in probabilistic modeling—most notably Bayesian networks—do not directly represent such causal events. Instead, derivate kinds of information, such as probabilistic (in-)dependencies and conditional probability distributions, are represented.

In this paper, we propose a more direct approach, which is based on the following construct of a *conditional probabilistic event*, or *CP-event* for short: "If property φ is satisfied, then a probabilistic event will happen that causes at most one of propositions h_1, h_2, ..., h_m, where the probability of h_1 being caused is α_1, the probability of h_2 is α_2, ..., and the probability of h_m is α_m." We use the following syntax to represent a CP-event of the above form: $(h_1 : \alpha_1) \vee \cdots \vee (h_m : \alpha_m) \leftarrow \varphi$. In the next section, we will be more precise about which kinds of preconditions φ we allow. For now, we will consider the simple case of φ being a conjunction $b_1 \wedge \cdots \wedge b_n$ of propositions. The precondition φ may also be absent, in which case the event is called *unconditional*.

The language *Conditional Probabilistic Event Logic*, or *CP-logic* for short, now consists of sets of such CP-events. Such a set is called a *CP-theory*. By examining our intuitions about how different CP-events should interact with each other, the following two fundamental principles can be isolated. The first is that of *independent causation*. It states that every CP-event represents an independent causal event; in other words, the outcome of one event might affect whether or not some other event will happen, but does not have an influence on what the outcome of such an event will be, should it in fact happen. This principle is precisely what allows causality to act as the basis for a stable and modular view of the world. Moreover, it also implies that, to a certain extent, the order in which CP-events happen is irrelevant. The second principle, which we call the principle of *no deus ex machina effects*, is that nothing happens without a cause, i.e., everything remains false unless there is a cause for it to become true and something cannot cause itself. This is a fundamental principle of causal reasoning and will turn out to be especially vital in the presence of cyclic causal relations.

In the next secton, we will construct a semantics for CP-logic based on these two principles. In this semantics, a CP-theory constructively defines a unique probability distribution over interpretations of the propositions. At each step, this constructive process simulates a single CP-event. Such a simulation derives proposition h_i with probability α_i, but can only be performed if all the propositions $b_1, \ldots, b_n$ have already been derived. Moreover, each event can occur at most once. This process will start from the empty set, i.e., initially nothing has been derived yet, and will end once there are no more CP-events left to simulate. The probability of an interpretation, then, is the sum of the probabilities of all possible derivations of this interpretation. It can be shown that the precise order in which CP-events are simulated does not matter, i.e., all sequences

will construct the same distribution. This follows from the principle of independent causation, together with the fact that, by only considering preconditions that are conjunctions of propositions, we have ensured the *monotonicity* of such sequences of simulations, i.e., the fact that if, at a certain time, all preconditions to a CP-event are satisfied, they will remain satisfied. Moreover, the "no deus ex machina"-principle is clearly incorporated in this semantics, because a proposition is only derived if it is caused by a CP-event with satisfied preconditions.

To sketch some of the interesting properties of this language, we consider two ways in which a person might get infected by the HIV virus: sexual intercourse with an infected partner and blood transfusion. For concreteness, assume that the probability of contracting HIV from an infected partner is 0.6 and that the probability of contracting it through a blood transfusion is 0.01. For the case of two partners a and b, of which only a has received a blood transfusion, we can model this example by the following CP-theory: $\{(hiv(a) : 0.6) \leftarrow hiv(b).\ (hiv(b) : 0.6) \leftarrow hiv(a).\ (hiv(a) : 0.01).\}$

As this example shows, the principle of independent causation makes it easy to represent the relation between an effect and a number of independent causes for this effect in a compact, clear and modular way, with each possible cause corresponding to a single CP-event. Moreover, this principle also makes the representation elaboration tolerant, in the sense that adding (or removing) an additional cause simply corresponds to adding (removing) a single rule. For instance, if b undergoes a blood transfusion as well, we only need to add a rule $(hiv(b) : 0.01)$. Because of the "no deus ex machina"-principle, the cyclic causal relation between $hiv(a)$ and $hiv(b)$ can be represented in precisely the same way as an acyclic one. Indeed, the first two rules will act as one would expect from such a causal loop: if neither a nor b have been infected by an external cause, then both are not infected, i.e., by itself such a loop does not cause anything; if precisely one of a and b has been infected by an external cause, then the probability of the other also being infected is 0.6. Another useful consequence of the "no deus ex machina"-principle is that domains can be represented in a compact way, because cases in which some proposition is not caused can simply be ignored. Indeed, we do not need to mention that without either intercourse with an infected partner or blood transfusion, an HIV infection is impossible. Further on in this paper, we will compare CP-logic to Bayesian networks and address these issues in more detail.

The contributions of this paper are the following. We introduce the concept of a CP-event and the principles of independent causation and no deus ex machina effects. We then use these to define the language of CP-logic. We show how this logic relates to Bayesian networks and illustrate some advantages of making causal events explicit. We relate our logic to logic programming based approaches to probabilistic modeling. This result provide additional motivation for these approaches and helps to clarify their knowledge representation methodology. It also allows us to consider a causal interpretation for disjunctive and normal logic progams. We compare this to McCain & Turner's causal theories.

Conditional Probabilistic Event Logic

The full syntax of CP-logic extends the one presented in the introduction in two ways. Firstly, it allows variables to be used to represent a set of CP-events by a single rule. Secondly, arbitrary first-order logic formulas can be used as preconditions to CP-events. More concretely, a CP-theory consists of a set of rules of the form: $(A_1 : \alpha_1) \vee \cdots \vee (A_n : \alpha_n) \leftarrow \varphi$. Here, the A_i's are atoms which may contain variables, φ is a first-order formula, and the α_i's are numbers between 0 and 1, s.t. $\sum_{i=1}^{n} \alpha_i \leq 1$. We distinguish between two different kinds of variables: those that are bound by a quantifier in φ and those that are free. The free variables $free(r)$ of a rule r are treated as place holders for ground terms. Concretely, we will consider a rule r as an abbreviation for the *grounding* (w.r.t. the Herbrand universe) of this rule, i.e., the set of all rules r' that can be derived from r by replacing all the variables $free(r)$ by ground terms. We reserve the term CP-event to refer to a rule in which no free variables appear. As such, a rule with free variables is simply a convenient way of representing a set of similarly structured CP-events. In formal discussions, we will always assume that CP-theories have already been grounded, i.e., we restrict attention to sets of rules of the form:

$$(h_1 : \alpha_1) \vee \cdots \vee (h_n : \alpha_n) \leftarrow \varphi. \qquad (1)$$

where the h_i's are ground atoms and φ is a formula with $free(\varphi) = \{\}$. For a rule r of form (1), we use $Body(r)$ to refer to the formula φ. We use $Body_+(r)$ to denote the set of all atoms that occur only positively (i.e., within the scope of an even number of negations) in φ and $Body_-(r)$ to denote all atoms that occur negatively (in the scope of an odd number of negations). By $body_+(r)$ and $body_-(r)$ we mean the set of all ground atoms belonging to the grounding of an atom in, respectively, $Body_+(r)$ and $Body_-(r)$. The set of pairs $\{(h_1, \alpha_1), \ldots, (h_n, \alpha_n)\}$ will be denoted by $head(r)$. By $head_{At}(r)$ we mean the set $\{h_1, \ldots, h_n\}$ of all atoms appearing in the head of r. Rules of the form $(h : 1) \leftarrow \varphi$ are called *deterministic*. We also write such a rule as $h \leftarrow \varphi$.

We now discuss how the definition of the semantics of CP-logic outlined in the introduction can be adapted to cope with arbitrary formulas as preconditions. The main difficulty here lies in the presence of negation. Indeed, this causes the previously mentioned monotonicity property to be lost, i.e., it will no longer be the case that if the preconditions to a CP-event are satisfied at a certain point in time, they are guaranteed to remain satisfied. As a consequence of this, the order in which CP-events are executed might matter. Consider, for instance, the following example: $\{p \leftarrow \neg q.\ \ q.\}$ If the first

event is executed first, then p will hold; if the second event is executed first, then p will not. So, negation introduces an ambiguity into the language and this needs to be resolved. Moreover, to avoid making the language too cumbersome, we would like to resolve it without forcing the user to explicitly specify an order in which events must happen. The most reasonable convention that can be assumed to solve this problem is that a literal $\neg r$ refers to the truth of r *after* all executable events that might cause r have been executed. As such, in our semantics, the simulation of such an event will be delayed as long as there are still executable events left that might cause r.

To ensure that this is indeed possible, we require CP-theories to be *stratified*, meaning that there has to exist a way of assigning to each ground atom p a level $\lambda(p) \in \mathbb{N}$, s.t. for each rule r and $h \in head_{At}(r)$, for all $b \in body_+(r)$, $\lambda(h) \geq \lambda(b)$, while for all $b \in body_-(r)$, $\lambda(h) > \lambda(b)$. The level of a rule r is defined as the minimum of the levels of the atoms in $head_{At}(r)$. Now, by executing CP-events with a lower level first, we can make sure that, by the time we need to decide whether a precondition $\neg r$ of some event holds, all executable events that might cause r have already been executed. From now on, we will restrict our attention to CP-theories which admit such a stratification.

We will now formally define the semantics of a CP-theory C. We use the mathematical structure of a *probabilistic transition system*. This is a tree structure $\mathcal{T}$, in which every edge is labeled with a probability. To each node c, we associate a Herbrand interpretation $I(c)$ for the alphabet of C. Formally, such an interpretation is simply a set of ground atoms. A node c *executes* a rule r of form (1) if c has as its children precisely nodes $c_0, c_1, \ldots, c_n$, where $I(c_0) = I(c)$ and, for all $i > 0$, $I(c_i) = I(c) \cup \{h_i\}$; the probability of the edge (c, c_0) is the probability with which none of the h_i's is caused, i.e., $1 - \sum_{1 \leq i \leq n} \alpha_i$, and, for $i > 0$, the probability of (c, c_i) is α_i. A rule r is *executable* in a node c if $body(r)$ holds in $I(c)$ and no ancestor of c already executes r. A probabilistic transition system $\mathcal{T}$ *runs* a CP-theory C iff:

- For the root r of $\mathcal{T}$, $I(r)$ is $\{\}$;
- For every node c of $\mathcal{T}$, either c executes an executable rule r, s.t. no executable rule r' has a lower level than that of r, or no rules are executable in c and c is a leaf;

A system $\mathcal{T}$ defines a probability distribution over its leaves: the probability of a leaf c is the product of the probabilities of all edges in the path from the root to c. From this, a probability distribution $\pi_{\mathcal{T}}$ over interpretations can be derived, by defining the probability $\pi_{\mathcal{T}}(I)$ of an interpretation I as the sum of the probabilities of all leaves c for which $I(c) = I$.

It can be shown that every system $\mathcal{T}$ that runs a CP-theory C defines the same probability distribution, i.e., for all such $\mathcal{T}$ and $\mathcal{T}'$, $\pi_{\mathcal{T}} = \pi_{\mathcal{T}'}$. While we do not have space to present a formal proof of this, the following informal argument should show that, if in a certain node n there are a number of different rules that might be executed, it does not matter which one is chosen first. If some r is executed first, then the existence of the stratification for C ensures that, for each of the children n' of n, all other rules r' that were executable in n are still executable in n', and, in fact, will remain this way until they are executed. Because every executable rule must eventually be executed, this shows that in every branch originating from n, every r' that was originally executable in n will actually be executed. Now, let us consider the sum of the probabilities of all branches going through n, in which an atom h, with $(h, \alpha) \in head(r')$, is derived as a result of executing r'. It can be checked that, because $\mathcal{T}$ is constructed in such a way that the sum of the labels of all edges that leave a node is always equal to 1, this probability will be α times the sum of the probabilities of all branches going through n, which is precisely what it would have been if we had chosen to execute r' in n. A formal proof can be found in (Vennekens *et al.* 2006).

The formal semantics of a CP-theory C is now defined as precisely this unique distribution, which we denote as π_C.

Bayesian networks in CP-logic

CP-logic can express the same kind of knowledge as expressed in a Bayesian network. The semantics of Bayesian networks (Pearl 1988) states that a probability distribution is a model of a network with graph $\langle N, E \rangle$ iff (1) the conditional probabilities it determines are the same as the appropriate entries in the various tables and (23) the value of a node n is probabilistically independent of the value of all nodes m, s.t. there is no path from n to m in E, given a value for the parents of n in E. If a distribution π satisfies condition (2) w.r.t. some binary relation E, we say that π is *Bayesian w.r.t.* E.

We now investigate whether a similar relation exists for the semantics π_C of a CP-theory C. In other words, we want to answer the question: "When does observing the truth value of a ground atom p give us direct information about the truth of some other atom q?" By the word "direct" in this sentence, we mean that this information is not mediated by some other atom r. It turns out that this happens in two cases.

- An atom p has a *direct causal influence* on an atom q if p is a precondition to a CP-event that might cause q, i.e., if $\exists r \in C, q \in head_{At}(r)$ and $p \in body(r)$. In this case, whether p holds affects whether one of the CP-events that might cause q is executed and, therefore, learning the truth of p clearly might affect the probability of q.
- Atoms p and q are *alternatives* iff p and q appear in the head of the same rule, i.e., iff $\exists r \in C : p, q \in head_{At}(r)$ and $p \neq q$. In this case, p and q are alternative outcomes of the same CP-event. As such,

learning that p holds will decrease the probability that q holds, and vice versa.

Let us now say that an atom p *directly affects* an atom q if one of these conditions holds, i.e., if $\exists r \in C : q \in head_{At}(r)$ and $p \in body(r) \cup (head_{At}(r) \setminus \{p\})$. We have shown in a recent technical report [reference omitted for anonymity] that the semantics π_C of a CP-theory is Bayesian w.r.t. this "directly affects"-relation. From this, a way of representing Bayesian networks in CP-logic can be derived. We first show how to represent the following Bayesian network:



	B,E	B,¬E	¬B,E	¬B,¬E
A	0.9	0.8	0.8	0.1

E	0.2
B	0.1

This can be modeled by the following CP-theory.

$$(bg : 0.1). \qquad (eq : 0.2).$$
$$(al : 0.9) \leftarrow bg \wedge eq. \quad (al : 0.8) \leftarrow \neg bg \wedge eq.$$
$$(al : 0.8) \leftarrow bg \wedge \neg eq. \quad (al : 0.1) \leftarrow \neg bg \wedge \neg eq.$$

Because this network contains only boolean nodes, we can translate each random variable into a single propositional symbol. As a result, in the CP-theory, all rules have just a single atom in their head; in the terminology defined above, no atoms are alternatives. The structure of the Bayesian network is now mirrored by the structure of the rules: the bodies of the rules for *earthquake* and *burglary* are empty, while the rules for *alarm* have both *burglary* and *earthquake* in their body. As such, this CP-theory expresses the same probabilistic independencies as the Bayesian network.

For nodes with a domain of more than two possible values, the situation is more complex. Indeed, such a node no longer corresponds to a single proposition, but rather to set of propositions. For instance, let us suppose that the alarm can be in three states: off (f), on (n), or disabled (d). We would then get rules of the following form:

$$(al(n) : 0.8) \vee (al(f) : 0.1) \vee (al(d) : 0.1) \leftarrow bg \wedge eq.$$
$$(al(n) : 0.7) \vee (al(f) : 0.1) \vee (al(d) : 0.1) \leftarrow bg \wedge \neg eq.$$
$$(al(n) : 0.7) \vee (al(f) : 0.1) \vee (al(d) : 0) \leftarrow \neg bg \wedge eq.$$
$$(al(n) : 0.2) \vee (al(f) : 0.8) \vee (al(d) : 0) \leftarrow \neg bg \wedge \neg eq.$$

This principle generalizes to a method of translating arbitrary networks to CP-logic.

The Role of Causality in CP-logic

In this section, we examine the differences between Bayesian networks and CP-logic in more detail.

Independent Causes

As already discussed, CP-logic incorporates the same kind of probabilistic independencies as Bayesian networks do. However, the principle of independent causation also allows a different kind of independence to be expressed, namely that between different causes for the same effect. In the introduction, we illustrated this by considering a number of different causes for HIV infection. We return to that example in the next section, where it will be used to illustrate the fact that our methodology for representing independent causation also applies when cyclic causal relations are involved. In this section, we focus on an example containing only acyclic causality:

Example 1. *Consider a game of Russian roulette with two guns, one in the player's right hand and one in his left. Each of the guns is loaded with a single bullet. What is the probability of the player dying?*

Firing a gun causes death with probability $\frac{1}{6}$. In CP-logic, we write: $(death : \frac{1}{6}) \leftarrow fire(Gun)$.This rule is all that is needed to model the operation of the guns. Indeed, if our alphabet also includes constants $left_gun$ and $right_gun$, then, after grounding, we get the following rules: $\{(death : \frac{1}{6}) \leftarrow fire(left_gun).\ (death : \frac{1}{6}) \leftarrow fire(right_gun).\}$. In words, "firing the left gun" and "firing the right gun" are two independent causes for death and each has a probability of $\frac{1}{6}$ of actually causing death. In a Bayesian network, this relation would typically be expressed as follows (the numbers can be computed by applying *noisy-or*[1] to the multiset of the probabilities with which the guns that are fired each cause death):

	l,r	¬l,r	l,¬r	¬l,¬r
death	11/36	1/6	1/6	0

In CP-logic, unlike in Bayesian networks, independence between different causal mechanisms is a *structural* property, rather than a quantitative one. From a knowledge representation point of view, there are a number of different reasons why this is useful. Firstly, it makes such properties more obvious. Indeed, the fact that $fire(left_gun)$ and $fire(right_gun)$ are independent possible causes of *death* is clear from the fact that these atoms do not appear in the body of the same rule together. Secondly, independence between causes is more robust to changes in the specification of a problem than quantitative knowledge usually is. For instance, if we were to find out that one of the guns has a mechanical defect, making the probability of the bullet being in front of the hammer not $\frac{1}{6}$ precisely, but $\frac{11}{60}$ instead, then this would not affect the independence between the two possible causes for death. Thirdly, a large part of Pearl's book (Pearl 2000) is concerned with various manipulations of causal models. These are important, for instance, to deal with counterfactuals or concepts such as the "actual cause" of some effect. Typically, such a manipulation consists of preempting the normal relation between a node and its parents and replacing it by an entirely new relation. With the more fine-grained structure imposed by CP-logic, a number of new, interesting manipulations become possible. For instance,

[1] The *noisy-or* maps a multiset S of probabilities to $1 - \prod_{p \in S}(1-p)$.

one could now consider the situation that would result by only inhibiting one of the causal mechanisms that might cause a particular atom or by adding a new such causal mechanism. This also improves the *elaboration tolerance* of the representation. For instance, if it is also possible that the player dies of a heart attack, we can simply include a CP-event "$(death : 0.2)$.", which might lead to death without firing any guns. Finally, like other qualitative properties, knowledge about these independencies often originates from an expert's background knowledge about the domain, instead of being derived from a dataset. For instance, when performing parameter learning in a Machine Learning setting, this is an important consideration.

Cyclic Causal Relations

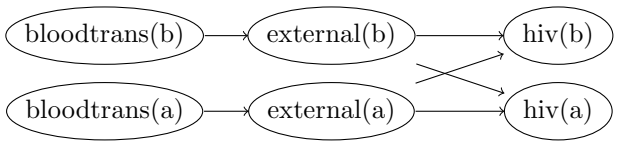
We now revisit the following example from the introduction:

Example 2. *There are two ways of getting infected by the HIV virus. Firstly, there is a probability of 0.01 of contracting the virus through a blood transfusion. Secondly, one might get infected by an already infected sexual partner. The probability of an infected person infecting his/her partner is 0.6.*

$$(hiv(X) : 0.6) \leftarrow hiv(Y) \wedge intercourse(X, Y).$$
$$(hiv(X) : 0.01) \leftarrow blood_transfusion(X).$$

Because the *intercourse*-relation is symmetric, this example leads naturally to cyclic causal relations. Because of the "no deus ex machina"-principle of CP-logic, cyclic relations can be modeled in the same way as acyclic ones. To make the discussion more concrete, suppose that a and b are the only persons we wish to consider and that these have intercourse. For simplicity, we ignore the *intercourse* and $blood_transfusion$ predicates and simply assume that our grounding process knows who have intercourse and who have received blood transfusions. The first rule of the CP-theory will lead to the grounding $\{(hiv(a) : 0.6) \leftarrow hiv(b).\ (hiv(b) : 0.6) \leftarrow hiv(a).\}$. Let us examine how these rules act when part of a larger theory. Firstly, if these two rules were to constitute the entire theory, then, because of the "no deus ex machina"-principle, neither partner would be infected. Now, if partner a undergoes a blood transfusion, the rule "$(hiv(a) : 0.01)$." would also appear, adding this as an additional cause for $hiv(a)$. In this case, a has a non-zero probability (namely 0.01) of being infected by an external cause and, therefore, b also has a non-zero probability (0.01×0.6) of infection. If both a and b undergo a blood transfusion, then the probability of, for instance, $hiv(a)$ will be higher still, because there are now two independent causes for $hiv(a)$: a could have gotten infected by a transfusion, but also because b was first infected by a transfusion and infected a in turn.

This causal loop can be represented in a Bayesian network by introducing, for every $hiv(x)$ in this loop, a new node $external(x)$ to represent the event that x has gotten infected by some external (i.e., not part of the loop) cause. These are then connected as follows:



	bt(a)	¬bt(a)
e(a)	0.01	0

	bt(b)	¬bt(b)
e(b)	0.01	0

	e(a),e(b)	e(a),¬e(b)	¬e(a),e(b)	¬e(a),¬e(b)
hiv(a)	1	1	0.6	0

	e(a),e(b)	e(a),¬e(b)	¬e(a),e(b)	¬e(a),¬e(b)
hiv(b)	1	0.6	1	0

The Absence of Causes

Another consequence of the "no deus ex machina"-principle is that CP-logic does not require cases in which an effect is not caused to be mentioned at all. Obviously, this can make representations more compact. This feature is made more powerful by the fact that CP-theories may contain negation, which allows the falsity of an atom, i.e., the absence of a cause for this atom, to act as a cause for other atoms. To illustrate, we consider the well-known dice game of craps. Similar phenomena occur, e.g., when representing *inertia axioms*, i.e., rules stating that some property persists unless these is a cause for it not to.

Example 3. *In craps, one keeps on rolling a pair of dice until one either wins or loses. In the first round, one immediately wins by rolling* 7 *or* 11 *and immediately loses by rolling* 2,3, *or* 12. *If any other number is rolled, this becomes the player's "box point". The game then continues until either the player wins by rolling the box point again or loses by rolling* 7.

$$(roll(T+1, 2) : 1/36) \vee \cdots \vee (roll(T+1, 12) : 1/36) \leftarrow \neg(win(T) \vee lose(T)).$$
$$win(1) \leftarrow roll(1, 7) \vee roll(1, 11).$$
$$lose(1) \leftarrow roll(1, 2) \vee roll(1, 3) \vee roll(1, 12).$$
$$boxpoint(X) \leftarrow roll(1, X) \wedge \neg win(1) \wedge \neg lose(1).$$
$$win(T) \leftarrow boxpoint(X) \wedge roll(T, X) \wedge T > 1.$$
$$lose(T) \leftarrow roll(T, 7) \wedge T > 1.$$

(In principle, this CP-theory would lead to an infinite grounding. While it is possible to define a semantics for CP-logic that is able to correctly handle such theories, this is beyond the scope of this paper. Because this issue is orthogonal to the points we want to discuss here, we simply assume that a certain maximum number of throws, i.e., an upper bound for the variable T, has been fixed up front and will be taken into account when performing the grounding of this theory.)

In this CP-theory, we only specify when the game is won or lost and use negation to express that, as long as neither happens, the game carries on. In Bayesian networks, there is no real way of ignoring irrelevant cases. Instead, there will be a probability of zero in the conditional probability table. For this game, we could use

variables $roll_t$, representing the outcome of a certain roll (with domain 2 through 12), and bp representing the box point (with possible values 4,5,6,7,8,9 or 10), that influence the state s_t of the game at time t as follows:

	$(bp, roll_t)$						
s_t	$(4,2)$	$(4,3)$	$(4,4)$	$(4,5)$	$(4,6)$	$(4,7)$	$\cdots$
win	0	0	1	0	0	0	$\cdots$
lose	0	0	0	0	0	1	$\cdots$

Logic Programs with Annotated Disjunctions

Logic Programs with Annotated Disjunctions (LPADs) are a probabilistic logic programming language, that was conceived in (Vennekens, Verbaeten, & Bruynooghe 2004) as a straightforward extension of logic programs with probability. In this section, we relate LPADs to CP-logic. This achieves the following goals:

- We can clarify the position of CP-logic among related work, such as Poole's Independent Choice Logic and McCain and Turner's causal theories.
- We gain additional insight into a number of probabilistic logic programming languages, by showing that theories in these languages can be seen as descriptions of causal processes. Moreover, as we will discuss in the next section, this also leads to an interesting way of looking at normal and disjunctive logic programs.
- Probabilistic logic programming languages are usually motivated in a bottom-up way, i.e., along the following lines: "Logic programs are a good way of representing knowledge about relational domains, probability is a good way of representing knowledge about uncertainty; therefore, a combination of both should be useful for modeling uncertainty in a relational domain." Our results provide an additional top-down motivation, by showing that these languages are a natural way of representing causal processes.

We first recall the formal definition of LPADs from (Vennekens, Verbaeten, & Bruynooghe 2004). An LPAD is a set of rules $(h_1 : \alpha_1) \vee \cdots \vee (h_n : \alpha_n) \leftarrow l_1 \wedge \cdots \wedge l_n$, where the h_i are atoms and the l_j literals. As such, LPADs are a syntactic sublogic of CP-logic. However, their semantics is defined quite differently. Every rule of the above form represents a probability distribution over the set of logic programming rules $\{\text{“}h_i \leftarrow l_1 \wedge \cdots \wedge l_n\text{”} \mid 1 \leq i \leq n\}$. From these distributions, a probability distribution over logic programs is then derived. To formally define this distribution, we introduce the following concept of a *selection*. In this definition, we use the notation $head^*(r)$ to denote the set of pairs $head(r) \cup \{(\emptyset, 1 - \sum_{(h:\alpha)\in head(r)} \alpha)\}$, where $\emptyset$ represents the possibility that none of the h_i's are caused by the rule r.

Definition 1 (C-selection). Let C be an LPAD. A *C-selection* is a function σ from C to $\bigcup_{r\in C} head^*(r)$, such that for all $r \in C$, $\sigma(r) \in head^*(r)$. By $\sigma^h(r)$ and $\sigma^\alpha(r)$ we denote, respectively, the first and second element of the pair $\sigma(r)$.

The probability $\pi(\sigma)$ of a selection σ is now defined as $\prod_{r\in C} \sigma^\alpha(r)$. By C^σ we denote the logic program $\{\text{“}\sigma^h(r) \leftarrow body(r)\text{”} \mid r \in C \text{ and } \sigma^h(r) \neq \emptyset\}$. Such a C^σ is called an *instance* of C. These instances are interpreted according to the well-founded model semantics (Van Gelder, Ross, & Schlipf 1991). In general, the well-founded model $wfm(P)$ of a program P is a pair (I, J) of interpretations, where I contains all atoms that are certainly true and J contains atoms that might possibly be true. If $I = J$, the model is said to be *two-valued*. Intuitively, if $wfm(P)$ is two-valued, then the truth of all atoms can be decided, i.e., everything that is not false can be derived. In the semantics of LPADs, we want to ensure that all uncertainty is expressed by means of the annotated disjunctions. In other words, given a specific selection, there should no longer be any uncertainty. We impose the following criterion.

Definition 2 (Soundness). An LPAD C is *sound* iff all instances of C have a two-valued well-founded model.

For such LPADs, the following semantics can now be defined.

Definition 3 (Instance based semantics μ_C). Let C be a sound LPAD. For an interpretation I, we denote by $W(I)$ the set of all C-selections σ for which $wfm(C^\sigma) = (I, I)$. The *instance based semantics* μ_C of C is the probability distribution on interpretations, that assigns to each I the probability $\sum_{\sigma\in W(I)} \pi(\sigma)$.

Now, the key result of this section is that this instance based semantics coincides with the semantics defined previously for CP-logic.

Theorem 1. Let C be a stratified CP-theory. Then C is also a sound LPAD and, moreover, for each interpretation J, $\mu_C(J) = \pi_C(J)$.

We remark that it is not the case that every sound LPAD is also a valid CP-theory. In other words, there are some sound LPADs that do not seem to represent a causal process.

In (Vennekens, Verbaeten, & Bruynooghe 2004), LPADs are compared to a number of different probabilistic logic programming formalisms. For instance, it was shown that this logic is very closely related to Poole's Independent Choice Logic. Because of the above theorem, these comparisons carry over to CP-logic.

CP-logic and Logic Programming

In this section, we examine some consequences of the results of the previous section from a logic programming point-of-view.

Disjunctive logic programs

In probabilistic modeling, it is often useful to consider the structure of a theory separately from its probabilistic parameters. Indeed, for instance, in machine learning, the problems of structure learning and parameter learning are two very different tasks. If we consider only the structure of a CP-theory, then, syntactically speaking, we end up with a *disjunctive logic program*[2], i.e., a set of rules $h_1 \vee \cdots \vee h_n \leftarrow \varphi$. Let us now consider the class of all CP-theories C that result from adding probabilities α_i to each rule, in such a way that, for every rule, $\sum \alpha_i = 1$. Every probability distribution π_C defined by such a C induces a possible world semantics, namely the set of interpretations I for which $\pi_C(I) > 0$. This set of possible worlds does not depend on the precise values of the α_i, i.e., it is the same for all CP-theories C in this class. As such, it captures precisely the structural information in such a CP-theory.

From the point of view of disjunctive logic programming, this set of possible worlds can be seen as an alternative semantics for such a program. Under this semantics, the intuitive reading of a rule should be: "if φ holds, there will be a non-deterministic event, that causes precisely one of $h_1, \ldots, h_n$." Clearly, this is a different informal reading than is used in the standard stable model semantics for disjunctive programs (Przymusinski 1991). Indeed, under our reading, a rule corresponds to a causal event, whereas, under the stable model reading, it is supposed to describe an aspect of the reasoning behaviour of a rational agent. This difference also manifests itself in the resulting formal semantics. Consider, for instance, the disjunctive program $\{p \vee q.\ p.\}$. To us, this program describes a set of two non-deterministic events: One event causes either p or q and another event always causes p. This might, for instance, correspond to the following story: "Someone is going to shoot a gun at Bob and this will either cause Bob's death or a hole in the wall behind Bob. Bob has also just ingested a lethal dose of poison and this is going to cause Bob's death." Our formal semantics reflects this interpretation, by considering both the interpretation $\{p\}$ (Bob is dead and there is no hole in the wall) and $\{p, q\}$ (Bob is dead and there is a hole in the wall) to be possible. Under the stable model semantics, these rules describe beliefs of a rational agent: The agent believes either p or q and the agents believes q. This interpretation might correspond to the following story: "I know that someone was going to shoot a gun at Bob, which would either result in Bob's death or in a hole in the wall. Moreover, I also learn that Bob is dead." In this case, I would have no reason to believe there might be a hole in the wall. Indeed, the only stable model is $\{p\}$.

CP-logic treats disjunction in a fundamentally different way than the stable semantics. Interestingly, the *possible model semantics* (Sakama & Inoue 1994) for disjunctive programs is very similar to our treatment. Indeed, it consists of the stable models of instances of a program. Because, as shown in the previous section, the semantics of CP-logic considers the well-founded models of instances, these two semantics are very closely related. Indeed, for a large class of programs, including all stratified ones, they coincide completely.

[2]In most of the literature, the bodies of the rules of a disjunctive logic program must be conjunctions of literals. For our purposes, however, this restriction is not relevant.

Normal logic programs

A normal logic program P is a set of rules $h \leftarrow \varphi$, with h an atom and φ a formula. If P is stratified, then, at least syntactically, it is also a CP-theory. Its semantics π_P assigns a probability of 1 to a single interpretation and 0 to all other interpretations. Moreover, the results from the previous section tell us that the interpretation with probability 1 will be precisely the well-founded model of P. As such, a logic program under the well-founded semantics can be viewed as a description of causal information about a deterministic process. Concretely, we can read a rule $h \leftarrow \varphi$ as: "if φ holds, there will be a deterministic event, that causes h."

This observation exposes an interesting connection between logic programming under the well-founded semantics and causality. Such a connection helps to explain, for instance, the usefulness of this semantics in dealing with recursive ramifications when reasoning about actions (Denecker, Theseider-Dupré, & Belleghem 1998). Moreover, there is also an interesting link here to the language of *ID-logic* (Denecker & Ternovska 2004). This is an extension of classical logic, that uses logic programs under the well-founded semantics to represent inductive definitions. Inductive definitions are a well-known mathematical construct, where a concept is defined in terms of itself. In mathematical texts, such a definition should be accompanied by a well-founded order, over which the induction happens, e.g., the well-known inductive definition of the satisfaction relation $\models$ of classical logic is a definition over the length of formulas. One of the key observations that underlie ID-logic is the fact that if such an order is not explicitly given, one can still be derived from the rule-based structure of a definition. This derived order is precisely the order imposed by the well-founded semantics. There is an obvious parallel here to the role of time in CP-logic: a complete description of a process should specify when events happen; however, if this information is not explicitly given, the order of events can still be derived from the rule-based structure of a CP-theory. It is interesting that the same mathematical construct of the well-founded semantics can be used to derive both the well-founded order for an inductive definition and the temporal order for a set of CP-events. This observation seems to imply that an inductive definition is nothing more than a representation of a causal process, that takes place in the domain of mathematical objects.

McCain and Turner's causal theories

In this section, we compare the treatment of causality in CP-logic to McCain and Turner's *causal theories* (McCain & Turner 1996). A causal theory is a set of rules of the form $\varphi \Leftarrow \psi$, where φ and ψ are propositional formulas. The semantics of such a theory T is defined as follows. An interpretation I is a model of T iff I is the *unique* classical model of the theory $T^I = \{\varphi \mid$ there exists a rule $\varphi \Leftarrow \psi$ in T such that $I \models \psi\}$. This semantics is based on the principle of *universal causation*, which states that: "every fact that obtains is caused" (McCain & Turner 1996). We now compare this language to deterministic CP-logic, i.e., CP-logic in which every CP-event causes one atom with probability 1. The most obvious difference concerns the fundamental knowledge representation methodology of these logics. In CP-logic, a proposition represents a property that is false unless there is a cause for it to be true. For McCain and Turner, however, truth and falsity are completely symmetric, i.e., not only is a property not true unless there is a cause for it to be true, but a property is also not false unless there is a cause for it to be false. It is up to the user to make sure there is always a cause for either falsity or truth. For instance, the CP-theory $\{p \leftarrow \neg q\}$ has $\{p\}$ as its model, while the causal theory $\{p \Leftarrow \neg q\}$ has no models, because neither q nor $\neg q$ is caused. The CP-logic view that falsity is the natural state of an atom can be simulated in causal theories, by adding a rule $\neg p \Leftarrow \neg p$ for every atom p. Essentially, this says that $\neg p$ is in itself reason enough for $\neg p$. Let C' be the result of adding such rules to some original CP-theory C. As shown in (McCain 1997), the models of C' are all interpretations I that consist of all heads of rules $r \in C$, for which $I \models body(r)$. In logic programming terms, these are the *supported models* of C, i.e., fixpoints of the immediate consequence operator T_C.

The difference such a CP-theory C and its corresponding causal theory C' is, therefore, precisely the difference between the well-founded model semantics and supported model semantics. It is well-known that this lies in the treatment of loops. In our context, it can be traced back to the fundamental principles of these logics. McCain and Turner's principle of "universal causation" states that everything that holds must have a cause. This is a weaker principle than our principle of no deus ex machina effects, which states that every true proposition must have a cause *and* that something cannot cause itself. Indeed, the CP-theory $\{p \leftarrow p\}$ has $\{\}$ as its model, whereas the causal theory $\{p \Leftarrow p\}$ has $\{p\}$ as its model. In other words, in McCain and Turner's theories, it can be stated that a certain atom might be true "on its own", i.e., without any additional causal explanation being required. This can be useful to incorporate exogenous actions into a theory, i.e., actions that can simply happen, without any part of the model describing why they happen. These currently cannot be represented in CP-logic. On the other hand, McCain and Turner's approach to self-causation does not allow them to directly represent cyclic causal relations of the kind appearing in our HIV example.

Related work

The correspondence to LPADs establishes a relation between CP-logic and probabilistic logic programming formalisms. In (**?**), a detailed comparison is made between LAPDs and a number of such approaches. At the formal level, these comparisions carry over to CP-logic. Here, we briefly discuss some of these formalisms, with a focus on the causal interpretation of CP-logic.

It has been shown in (**?**) that LPADs are very closely to the *Independent Choice Logic (ICL)* (Poole 1997). This language is based on abductive logic programs under the stable model semantics and was developed within the framework of decision theory. In (Finzi & Lukasiewicz 2003), a connection was made between ICL and Pearl's structural model approach to causality. One of the motivations for studying this relation is that it allows concepts such as "actual cause" and "explanation", that have been investigated by Halpern and Pearl in the context of structural models (Halpern & Pearl 2001), to be used in ICL. By linking ICL to CP-logic, we show that ICL can also be seen as a logic that incorporates our more finegrained concept of causality, based on causal events. This raises the question of whether there are meaningful adaptations of Halpern and Pearl's definitions that can take into account this additional structure. This is an interesting avenue for future research.

Bayesian Logic Programs (BLPs) (Kersting & Raedt 2000) and *Relational Bayesian Networks (RBNs)* (Jaeger 1997) are two formalisms that aim at lifting the propositional formalism of Bayesian networks to a first-order representation. Both these language allow arbitrary functions to be used to compute certain probabilities. By using a *noisy-or*, certain properties of CP-logic can be simulated. For instance, in a Relational Bayesian Network, one would model the Russian roulette example by the probability formula $P(death) = \textit{noisy-or}(\{1/6 \cdot fire(x) \mid x\})$. However, neither language offers a way of dealing with cyclic causal relations, other than an encoding similar to that for Bayesian networks.

Baral et al. introduced *P-log* (Baral, Gelfond, & Rushton 2004), a probabilistic extension of A-prolog. This language seems to be quite similar to CP-logic, even though it is somewhat broader in scope, being aimed at combining probabilistic and logical reasoning, rather than simply representing a probability distribution. As far as the representation of probability knowledge is concerned, P-log appears to be closer to Bayesian networks in the sense that it does not share CP-logic's focus on independent causation; instead, in every situation that might arise, there has to be precisely one statement that defines the probability of a certain effect in terms of all its possible causes. An interesting feature of P-log is that it allows random selections from a dynamic range of alternatives, which is

more flexible than the static enumerations of possible outcomes used in CP-logic.

Conclusions

We have investigated the role of causality in modeling probabilistic processes. To this end, we introduced the concept of a CP-event as a formal representation of the intuitive notion of a causal event. We presented a semantics for the language consisting of sets of such CP-events. This is based on two fundamental principles, that govern the interaction between different CP-events. The first is the principle of independent causation, which establishes the basic modularity of our causal view of the world. The second principle, that of no deus ex machina effects, captures the intuition that nothing happens without a cause, even in the presence of cyclic causal relations.

The direct representation of causal events in CP-logic turns out to have a number of interesting properties when compared to a Bayesian network style representation in terms of conditional probability. In particular, a new kind of independence, namely that between different causes for the same effect, emerges as a structural property, improving the elaboration tolerance of the representation. Moreover, cyclic causal relations can also be represented in a natural way and do not require any special treatment.

We have related CP-logic to a class of existing probalistic logic programming approaches. This shows that these languages can also be seen as representations of causal events. Moreover, this also shows that CP-logic induces a possible world semantics for disjunctive logic programs, that is quite diferent from the standard stable model semantics, but very similar to the possible model semantics. Another consequence of this results is that normal logic programs under the well-founded semantics can be seen as a logic of deterministic causality, which points towards an interesting relation between causality and inductive definitions. We have compared this way of handling causality to the McCain and Turner's causal theories.

References

Baral, C.; Gelfond, M.; and Rushton, N. 2004. Probabilistic reasoning with answer sets. In *Proc. Logic Programming and Non Monotonic Reasoning, LPNMR'04*, 21–33. Springer-Verlag.

Denecker, M., and Ternovska, E. 2004. A logic of non-monotone inductive definitions and its modularity properties. In *Proc. 7th LPNMR*, volume 2923 of *LNCS*.

Denecker, M.; Theseider-Dupré, D.; and Belleghem, K. V. 1998. An inductive definition approach to ramifications. *Linköping EACIS* 3(7):1–43.

Finzi, A., and Lukasiewicz, T. 2003. Structure-based causes and explanations in the independent choice logic. In *Proc. Uncertainty in Artificial Intelligence (UAI)*.

Halpern, J., and Pearl, J. 2001. Causes and explanations: A structural model approach – part I: Causes. In *Proc. Uncertainty in Artificial Intelligence (UAI)*.

Jaeger, M. 1997. Relational bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*.

Kersting, K., and Raedt, L. D. 2000. Bayesian logic programs. In Cussens, J., and Frisch, A., eds., *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 138–155.

McCain, N., and Turner, H. 1996. Causal theories of action and change. In *Proc. 13th AAAI/8th IAAI*.

McCain, N. 1997. *Causality in Commonsense Reasoning about Actions*. Ph.D. Dissertation, University of Texas at Austin.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Poole, D. 1997. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence* 94(1-2):7–56.

Przymusinski, T. C. 1991. Stable semantics for disjunctive programs. *New Generation Computing* 3/4:401–424.

Sakama, C., and Inoue, K. 1994. An alternative approach to the semantics of disjunctive logic programs and deductive databases. *Journal of automated reasoning* 13(1):145–172.

Van Gelder, A.; Ross, K.; and Schlipf, J. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38(3):620–650.

Vennekens, J.; Denecker, M.; ; and Bruynoogh, M. 2006. On the equivalence of Logic Programs with Annotated Disjunctions and CP-logic. Technical report, K.U. Leuven.

Vennekens, J.; Verbaeten, S.; and Bruynooghe, M. 2004. Logic programs with annotated disjunctions. In *Proc. 20th ICLP*, volume 3132 of *LNCS*. Springer.