FO(C): *A Knowledge Representation Language of Causality*

Bart Bogaerts, Joost Vennekens, Marc Denecker
Department of Computer Science, KU Leuven
E-Mail: firstname.lastname@cs.kuleuven.be

Jan Van den Bussche
Hasselt University & transnational University of Limburg
E-Mail: jan.vandenbussche@uhasselt.be

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Cause-effect relations are an important part of human knowledge. In real life, humans often reason about complex causes linked to complex effects. By comparison, existing formalisms for representing knowledge about causal relations are quite limited in the kind of specifications of causes and effects they allow. In this paper, we present the new language C-Log, which offers a significantly more expressive representation of effects, including such features as the creation of new objects. We show how C-Log integrates with first-order logic, resulting in the language FO(C). We also compare FO(C) with several related languages and paradigms, including inductive definitions, disjunctive logic programming, business rules and extensions of Datalog.

1 Introduction

Cause-effect relations are an important part of human knowledge. There exist a number of knowledge representation languages (McCain and Turner 1996; Vennekens et al. 2009; Cabalar 2012) in which logic programming style rules are used to represent such relations. The basic idea in all these approaches is that the head of such a rule represents an effect that is caused by its body. In this paper, we are particularly concerned with CP-logic (Vennekens et al. 2009). More specifically, we consider the variant of CP-logic without probabilities, and we will extend this language with three features: dynamic non-determinsitic choice; object creation; and recursive nesting of cause-effect relations. We call the resulting language C-Log. In this paper, we present C-Log and its informal semantics. For the formal semantics, we refer to an accompanying technical report (Bogaerts et al. 2014a). We also present the integration of C-Log with first-order logic, and thus show that C-Log fits in the FO($\cdot$) Knowledge Base System project (Denecker 2012).

Let us begin by recalling the guiding principles behind CP-logic. When compared to predecessors, such as the causal logic of McCain and Turner (1996), one of the important contributions of this languages is to add two modelling principles that are common in causal modelling. The first is the distinction between *endogenous* and *exogenous* properties, i.e., those whose value is determined by the causal laws in the model and those whose value is not, respectively (Pearl 2000). The second is the *default-deviant* assumption, used also by, e.g., Hall (2004) and Hitchcock (2007). The idea here is to assume that each endogenous property of the domain has some "natural" state, that it will be in whenever nothing is acting upon it. For ease of notation, CP-logic identifies the default state with falsity, and the deviant state with truth. For example, consider the following simplified model of a bicycle, in which a pair of gear wheels can be put in motion by pedalling:

$$\text{Turn(BigGear)} \leftarrow \text{Pedal}. \tag{1}$$
$$\text{Turn(BigGear)} \leftarrow \text{Turn(SmallGear)}. \tag{2}$$
$$\text{Turn(SmallGear)} \leftarrow \text{Turn(BigGear)}. \tag{3}$$

Here, Pedal is exogenous, while Turn(BigGear) and Turn(SmallGear) are endogenous. The semantics of this causal model is given by a straightforward "execution" of the rules. The domain starts out in an initial state, in which all endogenous atoms have their default value *false* and the exogenous atom Pedal has some fixed value. If Pedal is true, then the first rule is applicable and may be fired ("Pedal causes Turn(BigGear)") to produce a new state of the domain in which Turn(BigGear) now has its deviant value *true*. In this way, we construct the following sequence of states (we abbreviate symbols by their first letter):

$$\{\text{P}\} \overset{(1)}{\to} \{\text{P}, \text{T(B)}\} \overset{(3)}{\to} \{\text{P}, \text{T(B)}, \text{T(S)}\} \overset{(2)}{\to} \{\text{P}, \text{T(B)}, \text{T(S)}\} \tag{4}$$

Note that firing rule (2) does not change the state of the world, because its effect is already true. Moreover, it is obvious that this will always be the case, so this rule may seem redundant. However, many interesting applications of causal models require the use of interventions (Pearl 2000), e.g., to evaluate counterfactuals or to predict the effects of actions. As shown by Vennekens et al. (2010), rule (2) allows CP-logic to represent this example in a way that produces the correct results for all conceivable interventions in a manner that is more modular and more concise than, among others, Pearl's structural models (Pearl 2000).

After rules (1), (3) and (2) have all fired, there are no more rules left whose body is satisfied and that have not yet fired. At this point, the process is at an end and the domain has reached a final state. It is this final state, rather than the details of the intermediate process, that we are really interested in. One of the most important properties of CP-logic is that, while there may be any number of different processes derived from a causal theory, the final state that is eventually reached is unique for any given interpretation for the exogenous predicates—at least, for examples such as this one. In general, CP-logic also allows rules with a *non-deterministic* effect, such as:

$$(\text{Turn(SmallGear)} : 0.99)\ \textbf{Or}\ (\text{ChainBreaks} : 0.01) \leftarrow \text{Turn(BigGear)}.$$

Now, the cause Turn(BigGear) produces one of two possible effects, and there is an associated probability distribution over these two possibilities. The effect on the semantics is that, instead of a linear progression of states as in (4), we get a tree structure in which each firing of a non-deterministic rule introduces a branching of possibilities. When considering also the probabilities associated to non-deterministic choices, the tree defines a probability distribution over its leaves, i.e., over the final states that may be reached. It was shown in (Vennekens et al. 2009) that, given a specific interpretation for the exogenous atoms, this distribution is unique, even though there may exists many probability trees that produce it.

In many circumstances, the precise values of the probabilities are not of interest. In such cases, a non-probabilistic variant of CP-logic may be used, in which these are omitted. The head of a rule is then simply a disjunction:

$$\text{Turn}(\text{SmallGear})\ \mathbf{Or}\ \text{ChainBreaks} \leftarrow \text{Turn}(\text{BigGear}).$$

The trees then no longer produce a probability distribution over final states, but they describe the set of all final states that may be reached. In other words, this formalism has a possible world semantics. It is this non-probabilistic variant that concerns us in this paper.

Like other rule-based approaches to causality, CP-logic uses a very simple way of specifying the possible effects of some cause, namely, as a disjunction of ground atoms. Clearly, this does not—or, at least, not directly—cover many interesting phenomena that may occur in practice:

- A robot enters a room, opens some of the doors in this room, and then leaves by one of the doors that are open. The robot's leaving corresponds to a non-deterministic choice between a *dynamic* set of alternatives, which is determined by the robot's own actions, and therefore cannot be hard-coded into the head of a rule. A language construct for representing such choices is present in P-log (Baral et al. 2004).
- A stallion and a mare that are put in the same field may cause the birth of a foal. Therefore, not only the properties of these horses are governed by causal laws, but also their very existence.
- A horse being the parent of a foal is itself a cause for its own height to have a causal link to the height of the foal. Therefore, causal laws may be nested, in the sense that an effect can itself again consist of an entire causal law.

The goal of this paper is to develop an expressive knowledge representation language that is able to represent these more complex effects, and others like them, in a direct way. Moreover, we want to do this in a way that extends the approach of CP-logic. To summarise, the formal semantics of the language should consist of a set of possible worlds, each of which can be constructed by a non-deterministic causal process. This process will take place in the context of a fixed interpretation for the exogenous atoms. It will start from an initial state in which each of the endogenous atoms is at its default value false. The causal laws of our language will then "fire" and flip atoms to their deviant value, until no more such flips are possible. Whereas in CP-logic these flips happen one atom at a time, our extended language will flip *sets* of atoms at the same time. Moreover, our logic will present syntax and semantics for object-creation, as is needed in the second of the above examples.

The rest of this paper is structured as follows: we start by introducing causal effect expressions (CEEs) and their informal semantics in Section 2. In Section 3, we explain how C-Log is integrated with first-order logic, resulting in FO(C), a member of the FO($\cdot$) family of extensions of first-order logic. We conclude in Section 4 by comparing C-Log with various other paradigms, including inductive definitions (Denecker and Ternovska 2008), disjunctive logic programming with existential quantifications (You et al. 2013), Business Rules systems (Business Rules Group 2000) and Datalog extensions (Green et al. 2012).

2 Syntax and Informal Semantics

We assume familiarity with basic concepts of first-order logic (FO). Vocabularies, formulas, and terms are defined as usual. A Σ-structure $\mathcal{I}$ interprets all symbols (including variable symbols) in a vocabulary Σ; $D^{\mathcal{I}}$ denotes the domain of $\mathcal{I}$ and $\sigma^{\mathcal{I}}$, with σ a symbol in Σ, denotes the interpretation of σ in $\mathcal{I}$. We use $\mathcal{I}[\sigma : v]$ for the structure $\mathcal{J}$ that equals $\mathcal{I}$, except on σ: $\sigma^{\mathcal{J}} = v$. *Domain atoms* are atoms of the form $P(\overline{d})$ where the d_i are domain elements. We use restricted quantifications (Preyer and Peter 2002), e.g., in FO, these are formulas of the form $\forall x[\psi] : \varphi$ or $\exists x[\psi] : \varphi$, meaning that φ holds for all (resp. for a) x such that ψ holds. The above expressions are syntactic sugar for $\forall x : \psi \Rightarrow \varphi$ and $\exists x : \psi \land \varphi$, but such a reduction is not possible for other restricted quantifiers that we will define below. We call ψ the *qualification* and φ the *assertion* of the restricted quantifications. From now on, let Σ be a relational vocabulary, i.e., Σ consists only of predicate, constant and variable symbols.

2.1 Syntax

Definition 2.1
Causal effect expressions (CEE) are defined inductively as follows:

- if $P(\overline{t})$ is an atom, then $P(\overline{t})$ is a CEE,
- if φ is a first-order formula and C' is a CEE, then $C' \leftarrow \varphi$ is a CEE,
- if C_1 and C_2 are CEEs, then C_1 **And** C_2 is a CEE,
- if C_1 and C_2 are CEEs, then C_1 **Or** C_2 is a CEE,
- if x is a variable, φ is an FO formula and C' is a CEE, then **All** $x[\varphi] : C'$ is a CEE,
- if x is a variable, φ an FO formula and C' a CEE, then **Select** $x[\varphi] : C'$ is a CEE,
- if x is a variable and C' is a CEE, then **New** $x : C'$ is a CEE.

We call a CEE an *atom-expression* (respectively *rule-*, **And**-, **Or**-, **All**-, **Select**- or **New**-*expression*) if it is of the corresponding form. We call a predicate symbol P *endogenous* in C if P occurs as the symbol of a (possibly nested) atom-expression in C, i.e., if P occurs in C but not only in first-order formulas, i.e., not only in qualifications of restricted C-Log quantifications (**All** and **Select**) or conditions of rule-expressions. All other symbols are called *exogenous* in C. This is a straightforward generalisation of the same notions in CP-logic. An occurrence of a variable x is *bound* in a CEE if it occurs in the scope of a quantification over that variable ($\forall x$, $\exists x$, **All** x, **Select** x, or **New** x) and *free* otherwise. A variable is *free* in a CEE if it has free occurrences. A *causal theory*, or C-Log *theory* is a CEE without free variables. By abuse of notation, we often represent a causal theory as a set of CEEs; the intended causal theory is the **And**-conjunction of these CEEs. We often use Δ for a causal theory and C, C', C_1 and C_2 for its subexpressions.

2.2 Informal Semantics of CEEs

We now present the informal semantics of CEEs, due to space restrictions, the formalisation of this semantics is lacking in this paper. For a complete description of the formal semantics, we refer to an accompanying technical report (Bogaerts et al. 2014a). A CEE is a description of a set of causal laws. In the context of a state of affairs—which we represent, as usual, by a structure—a CEE non-deterministically describes a set of effects, i.e., a set of events that take place and change the state of affairs. We call such a set the *effect set* of the CEE. From a CEE C, we can derive causal processes similar to (4); a causal process is a sequence of intermediate states, starting from the default state, such that, at each state, the effects described by C take place. The process ends if the effects no longer cause changes to the state. A structure is a model of a CEE if it is the final result of such a process. We now explain in a compositional way what the effect set of a CEE is in a given state of affairs.

The effect of an atom-expression A is that A is flipped to its deviant state. A conditional effect, i.e., a rule expression, causes the effect set of its head if its body is satisfied in the current state, and nothing otherwise. The effect set described by an **And**-expression is the union of the effect sets of its two subexpressions; an **All**-expression $\mathbf{All}\, x[\varphi] : C'$ causes the union of all effect sets of $C'(x)$ for those x's that satisfy φ. An expression $C_1\, \mathbf{Or}\, C_2$ non-deterministically causes either the effect set of C_1 or the effect set of C_2; a **Select**-expression $\mathbf{Select}\, x[\varphi] : C'$ causes the effect set of C' for a non-deterministically chosen x that satisfies φ. An object-creating CEE $\mathbf{New}\, x : C'$ causes the creation of a new domain element n and the effect set of $C'(n)$.

Example 2.2
Permanent residence in the United States can be obtained in several ways. One way is passing the naturalisation test. Another way is by playing the "Green Card Lottery", where each year a number of lucky winners are randomly selected and granted permanent residence. We model this as follows:

$$\mathbf{All}\, p[\mathrm{Applied}(p) \land \mathrm{PassedTest}(p)] : \mathrm{PermRes}(p)$$
$$(\mathbf{Select}\, p[\mathrm{Applied}(p)] : \mathrm{PermRes}(p)) \leftarrow \mathrm{Lottery}.$$

The first CEE describes the "normal" way to obtain permanent residence; the second rule expresses that one winner is selected among everyone who applies. If $\mathcal{I}$ is a structure in which Lottery holds, due to the non-determinism, there are many possible effect sets of the above CEE, namely all sets $\{\mathrm{PermRes}(p) \mid p \in D^{\mathcal{I}} \land p \in \mathrm{Applied}^{\mathcal{I}} \land \mathrm{PassedTest}(p)^{\mathcal{I}}\}$ $\cup$ $\{\mathrm{PermRes}(d)\}$ for some $d \in \mathrm{Applied}^{\mathcal{I}}$. The two CEEs are considered independent: the winner could be one of the people that obtained it through standard application, as well as someone else. Note that in the above, there is a great asymmetry between Applied(p), which occurs as a qualification of **Select**-expression, and PermRes(p), which occurs as a caused atom, in the sense that the effect will never cause atoms of the form Applied(p), but only atoms of the form PermRes(p). This is one of the cases where the qualification of an expression cannot simply be eliminated.

Example 2.3
Hitting the "send" button in your mail application causes the creation of a new package containing a specific mail. That package is put on a channel and will be received some

(unknown) time later. As long as the package is not received, it stays on the channel. In C-Log, we model this as follows:

$$\begin{aligned}&\mathbf{All}\, m,t[\mathrm{Mail}(m) \land \mathrm{HitSend}(m,t)] : \mathbf{New}\, p : \mathrm{Pack}(p)\, \mathbf{And}\, \mathrm{Cont}(p,m)\, \mathbf{And}\, \mathrm{OnCh}(p,t+1)\\ &\quad \mathbf{And}\, \mathbf{Select}\, d[d>0] : \mathrm{Received}(p,t+d)\\ &\mathbf{All}\, p,t[\mathrm{Pack}(p) \land \mathrm{OnCh}(p,t) \land \neg\mathrm{Received}(p,t)] : \mathrm{OnCh}(p,t+1)\end{aligned}$$

Suppose an interpretation $\mathrm{HitSend}^{\mathcal{I}} = \{(\mathrm{MyMail},0)\}$ is given. A causal process then unfolds as follows: it starts in the initial state, where all endogenous predicates are false. The effect set of the above causal effect in that state consists of 1) the creation of one new domain element, say $\_p$, and 2) the caused atoms $\mathrm{Pack}(\_p)$, $\mathrm{Cont}(\_p, \mathrm{MyMail})$, $\mathrm{OnCh}(\_p, 1)$ and $\mathrm{Received}(\_p, 7)$, where instead of 7, we could have chosen any number greater than zero. Next, it continues, and in every step t, before receiving the package, an extra atom $\mathrm{OnCh}(p, t+1)$ is caused. Finally, in the seventh step, no more atoms are caused; the causal process ends. The final state is a model of the causal theory.

3 FO(C): Integrating FO and C-Log

First-order logic and C-Log have a straightforward integration, FO(C). Theories in this logic are sets of FO sentences and CEEs. A model of such a theory is a structure that satisfies each of its expressions (each of its CEEs and formulas). An illustration is the mail protocol from Example 2.3, which we can extend with the "observation" that at at some time point, two packages are on the channel: $\exists t, p_1, p_2 : \mathrm{OnCh}(p_1, t) \land \mathrm{OnCh}(p_2, t) \land p_1 \neq p_2$. Models of this theory represent states of affairs where at least once two packages are on the channel simultaneously. This entirely differs from **And**-conjoining our CEE with

$$\mathbf{Select}\, t, p_1, p_2[p_1 \neq p_2] : \mathrm{OnCh}(p_1, t)\, \mathbf{And}\, \mathrm{OnCh}(p_2, t).$$

The resulting CEE would have unintended models in which two packages suddenly appear on the channel for no reason.

In FO(C), **New**-expressions can be simulated with **Select**-expressions together with FO axioms expressing the unicity of the newly "created" objects. E.g.,

$$\mathbf{New}\, x : P(x,a)\, \mathbf{And}\, \mathbf{New}\, x : Q(x)$$

is simulated by introducing auxiliary unary predicates N_1 and N_2 that identify the objects created by the expressions and writing:

$$\begin{aligned}&\{(\mathbf{Select}\, x[\mathbf{t}] : (N_1(x)\, \mathbf{And}\, P(x,a)))\, \mathbf{And}\, \mathbf{Select}\, x[\mathbf{t}] : (N_2(x)\, \mathbf{And}\, Q(x))\}\\ &\forall x : \neg(N_1(x) \land N_2(x))\end{aligned}$$

It is clear that **New**-expressions are more natural and more modular than this simulation.

Despite the syntactical correspondence between CEEs and FO formulas (**And** corresponds to $\land$, **All** to $\forall$, ...), it is obvious that they have an entirely different meaning, and that both are useful. This is why we chose to introduce new connectives rather than overloading the ones of FO. The logic FO(C) has further interesting extensions, e.g., by adding aggregates in FO formulas, including in qualifications and conditions of CEEs.

4 Comparison and Future Work

In this section, we compare FO(C) to other existing paradigms. This comparison is only an initial study. By the time of publishing, a more extended version of a comparison between FO(C) and other paradigms has appeared (Bogaerts et al. 2014b).

Due to its simple recursive syntax, FO(C) is a very general logic that generalises several existing logics and shows overlaps with many others in different areas of computational logic. C-Log is an extension of (the non-probabilistic version of) CP-logic. FO(C) is an extension of the logic FO(ID) (Denecker and Ternovska 2008). An FO(ID) theory is a set of FO sentences and inductive definitions (ID), which are sets of rules of the form

$$\forall \overline{x} : P(\overline{t}) \leftarrow \varphi,$$

where φ is an FO formula. Such a rule corresponds to a CEE

$$\textbf{All}\, \overline{x}[\varphi] : P(\overline{t})$$

or equivalently,

$$\textbf{All}\, \overline{x}[\mathbf{t}] : (P(\overline{t}) \leftarrow \varphi)$$

and a definition corresponds to the **And**-conjunction of its rules. The semantics of FO(ID) corresponds exactly to the semantics of the corresponding FO(C) theory (Bogaerts et al. 2014a). Denecker et al. (1998) already pointed to the correspondence between causality and inductive definitions and exploited it for solving the causal *ramification problem* of temporal reasoning (McCarthy and Hayes 1969). The CEEs presented here can be seen as a non-deterministic extension of inductive definitions with an informal semantics based on causal processes.

FO(C) shows similarity to extensions of disjunctive logic programming (DLP) such as DLP with existential quantification in rule heads (You et al. 2013) and the stable semantics for FO as defined by Ferraris et al. (2011). Here constraints correspond to FO sentences in FO(C) and other rules correspond to C-Log expressions. However, there is an important semantical difference. Suppose we want to express Example 2.2, where all people passing a test and one random person are given permanent residence in the United States. The E-disjunctive program

$$\exists X : permres(X) \text{:-}\, lottery$$
$$\forall X : permres(X) \text{:-}\, passtest(X)$$

is similar to

$$(\textbf{Select}\, x[\mathbf{t}] : permres(x)) \leftarrow lottery$$
$$\textbf{And}\, \textbf{All}\, x[passtest(x)] : permres(x)$$

Semantically, the E-disjunctive program imposes a minimality condition: the lottery is always won by a person succeeding the test, if there exists one. On the other hand, in FO(C) the two rules execute independently, and models might not be minimal. In this example, it is the latter that is intended. We believe that one advantage of C-Log is its clear causal informal semantics. On the other hand, there are ways to simulate the causal semantics and the **New** operator of C-Log in E-disjunctive programs while it follows from complexity arguments that not all E-disjunctive programs can be expressed in FO(C) (Bogaerts et al. 2014c).

Other semantics than the stable semantics for DLP have been developed. For example, Brass and Dix (1996) defined D-WFS, a well-founded semantics for DLP. This semantics has the property that if a program contains two identical lines, one of them can be removed. However, in our context, a duplicate effect means that a same causal effect happens twice (maybe for different reasons), independently, and hence different choices might be made in each of these rules.

The logic of cause and change (McCain and Turner 1996) differs from C-Log in several important aspects; in McCain & Turner's logic both true and false atoms need a cause. In C-Log on the other hand, endogenous predicates can be false (the default value) without reason but can only be true (the deviant value) if caused. Moreover, we rule out unfounded "cyclic" causation. For instance, if Pedal is false, in C-Log, Turn(BigGear) and Turn(SmallGear) are false but in McCain and Turner's logic they may be true and caused by each other. We call this "spontaneous generation" and do not admit it in C-Log.

We find operators similar to those of C-Log in several other formalisms. For example, **Select**-, **All**-, **Or**- and rule-expressions are present in the subformalism of the language Event-B that serves to specify effects of actions (Abrial 2010). The **New** operator is found in various other rule based paradigms, for example in Business Rules systems (Business Rules Group 2000). The JBoss manual (Browne 2009) contains the following rule:

```
when Order( customer == null ) then  insertLogical(new
   ValidationResult( validation.customer.missing ));
```

meaning that if an order is created without customer, a new *ValidationResult* is created with the message that the customer is missing. This can be translated to C-Log as follows:

$$\textbf{All}\, y[\text{Order}(y) \wedge \text{NoCustumer}(y)] : \textbf{New}\, x : \text{ValidationR}(x)\, \textbf{And}\, \text{Message}(x, \text{“}\ldots\text{”}).$$

Another field in which related language constructions have been developed is the field of deductive databases. Abiteboul and Vianu (1991) considered various extensions of Datalog, resulting in non-deterministic semantics for queries and updates. One of the studied extensions is object creation. Such an extension is present in the LogicBlox system (Green et al. 2012). An example from the latter paper is the rule: $President(p), presidentOf[c] = p \leftarrow Country(c)$ which means that for every country c, a new (anonymous) "derived entity" of type *President* is created. Of course, this president is not a new person, but it is new with respect to a given database. Such rules with implicit existentially quantified head variables correspond with **New**-expressions in C-Log.

Other Datalog extensions with other forms of object creation exist. For example Van den Bussche and Paredaens (1995) discuss a version with creation of sets and compare its expressivity with simple object creation.

Non-deterministic choices have been studied intensively in the context of deductive databases. Krishnamurthy and Naqvi (1988) introduced a non-deterministic choice in Datalog. This choice was *static*: choice models are constructed in three steps. First, models are calculated while ignoring choices (choosing everything); second, this model is used to select a number of choices for all occurrences of *choice goals* and third, models are recalculated with respect to these choice goals. In other work (Saccà and Zaniolo 1990; Giannotti et al. 1991), it is argued that static choices do not behave well in the presence of recursion; hence *dynamic* choices were introduced. Saccà and Zaniolo (1990) use stable models to provide a model-theoretic description of these dynamic choices. Weidong and Jinghong (1996) introduced an alternative choice principle on predicates P. There, the values in certain argument positions in the tuples of P are chosen non-deterministically in function of the values at the other argument positions. The semantics of that logic is based on the well-founded semantics; this choice principle is very different from the principle in C-Log. Compared to these, C-Log resembles most the language of Saccà and Zaniolo (1990); the difference is that C-Log supports a recursive syntax and is based on the well-founded semantics, whereas Saccà and Zaniolo (1990) use stable semantics.

The above similarities suggest that FO(C) is a promising language to study and unify many existing logical paradigms and to provide a clear informal semantics for them. An in-depth semantical analysis of the exact relationship between FO(C) and the languages described above is an interesting topic for future work. Another research challenge is extending FO(C) with types, function symbols, arithmetic, etc. in order to make it useful as a KR-language. We need to study the complexity of various inference tasks in FO(C), and develop and implement algorithms for these various tasks. By the time of publication, a first study of complexity and inference in FO(C) has appeared (Bogaerts et al. 2014c). Another research question is to add probabilities to C-Log to obtain an extension of the probabilistic CP-logic, and possibly also of other related logics such as BLOG (Milch et al. 2005) and P-Log (Baral et al. 2004).

References

Abiteboul, S. and Vianu, V. 1991. Datalog extensions for database queries and updates. *J. Comput. Syst. Sci. 43,* 1, 62–124.

Abrial, J.-R. 2010. *Modeling in Event-B - System and Software Engineering.* Cambridge University Press.

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

Bogaerts, B., Vennekens, J., Denecker, M., and Van den Bussche, J. 2014a. C-Log: A knowledge representation language of causality. Tech. Rep. CW 656, Departement of Computer Science, Katholieke Universiteit Leuven.

Bogaerts, B., Vennekens, J., Denecker, M., and Van den Bussche, J. (in press) 2014b. FO(C) and related modelling paradigms. In *Proceedings of the Fifteenth International Workshop on Non-Monotonic Reasoning, NMR 2014, Vienna, Austria, September 17-19.*

Bogaerts, B., Vennekens, J., Denecker, M., and Van den Bussche, J. (in press) 2014c. Inference in the FO(C) modelling language. In *ECAI 2014 - 21th European Conference on Artificial Intelligence, Prague, Czech Republic, August 18-22, 2014, Proceedings.*

Brass, S. and Dix, J. 1996. Characterizing D-WFS: Confluence and iterated GCWA. In *Logics in Artificial Intelligence*, J. J. Alferes, L. M. Pereira, and E. Orlowska, Eds. Lecture Notes in Computer Science, vol. 1126. Springer Berlin Heidelberg, 268–283.

Browne, P. 2009. *JBoss Drools Business Rules.* From technologies to solutions. Packt Publishing, Limited.

Business Rules Group. 2000. Defining Business Rules ∼ What Are They Really? Tech. rep.

Cabalar, P. 2012. Causal logic programming. In *Correct Reasoning*, E. Erdem, J. Lee, Y. Lierler, and D. Pearce, Eds. Lecture Notes in Computer Science, vol. 7265. Springer, 102–116.

Denecker, M. 2012. The FO(·) knowledge base system project: An integration project (invited talk). In *ASPOCP*.

Denecker, M. and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL) 9,* 2 (Apr.), 14:1–14:52.

Denecker, M., Theseider-Dupré, D., and Van Belleghem, K. 1998. An inductive definition approach to ramifications. *Linkoping Electronic Articles in Computer and Information Science 3,* 7 (Jan.), 1–43.

Ferraris, P., Lee, J., and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence 175*, 236–263.

Giannotti, F., Pedreschi, D., Saccà, D., and Zaniolo, C. 1991. Non-determinism in deductive databases. In *Deductive and Object-Oriented Databases*, C. Delobel, M. Kifer, and Y. Masunaga, Eds. Lecture Notes in Computer Science, vol. 566. Springer Berlin Heidelberg, 129–146.

Green, T. J., Aref, M., and Karvounarakis, G. 2012. Logicblox, platform and language: A tutorial. In *Datalog*, P. Barceló and R. Pichler, Eds. LNCS, vol. 7494. Springer, 1–8.

Hall, N. 2004. Two concepts of causation. In *Causation and Counterfactuals.*

Hitchcock, C. 2007. Prevention, preemption, and the principle of sufficient reason. *Philosophical review 116,* 4.

Krishnamurthy, R. and Naqvi, S. A. 1988. Non-deterministic choice in datalog. In *JCDKB* (2002-01-03). 416–424.

McCain, N. and Turner, H. 1996. Causal theories of action and change. In *AAAI/IAAI.* AAAI Press, 460–465.

McCarthy, J. and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, 463–502.

Milch, B., Marthi, B., Russell, S. J., Sontag, D., Ong, D. L., and Kolobov, A. 2005. Blog: Probabilistic models with unknown objects. In *IJCAI*, L. P. Kaelbling and A. Saffiotti, Eds. Professional Book Center, 1352–1359.

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

Preyer, G. and Peter, G. 2002. *Logical Form and Language.* Clarendon Press.

Saccà, D. and Zaniolo, C. 1990. Stable models and non-determinism in logic programs with negation. In *Proceedings of the Ninth ACM Symposium on Principles of Database Systems.* ACM Press, 205–217.

Van den Bussche, J. and Paredaens, J. 1995. The expressive power of complex values in object-based data models. *Information and Computation 120*, 220–236.

Vennekens, J., Bruynooghe, M., and Denecker, M. 2010. Embracing events in causal modelling: Interventions and counterfactuals in CP-logic. In *Logics in Artificial Intelligence*, T. Janhunen and I. Niemelä, Eds. Lecture Notes in Computer Science, vol. 6341. Springer Berlin Heidelberg, 313–325.

Vennekens, J., Denecker, M., and Bruynooghe, M. 2009. CP-logic: A language of causal probabilistic events and its relation to logic programming. *Theory and Practice of Logic Programming 9,* 3, 245–308.

Weidong, C. and Jinghong, Z. 1996. Nondeterminism through well-founded choice. *The Journal of Logic Programming 26,* 3, 285–309.

You, J.-H., Zhang, H., and Zhang, Y. 2013. Disjunctive logic programs with existential quantification in rule heads. *Theory and Practice of Logic Programming 13*, 563–578.