ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-111

Inference in the FO(C) Modelling Language

Bogaerts Bart[1] and **Vennekens Joost**[1] and **Denecker Marc**[1] and **Van den Bussche Jan**[2]

Abstract. Recently, FO(C), the integration of C-LOG with classical logic, was introduced as a knowledge representation language. Up to this point, no systems exist that perform inference on FO(C), and very little is known about properties of inference in FO(C). In this paper, we study both of the above problems. We define normal forms for FO(C), one of which corresponds to FO(ID). We define transformations between these normal forms, and show that, using these transformations, several inference tasks for FO(C) can be reduced to inference tasks for FO(ID), for which solvers exist. We implemented this transformation and hence, created the first system that performs inference in FO(C). We also provide results about the complexity of reasoning in FO(C).

1 Introduction

Knowledge Representation and Reasoning is a subfield of Artificial Intelligence concerned with two tasks: defining modelling languages that allow intuitive, clear, representation of knowledge and developing inference tools to reason with this knowledge. Recently, C-LOG was introduced with a strong focus on the first of these two goals [3]. C-LOG has an expressive recursive syntax suitable for expressing various forms of non-monotonic reasoning: disjunctive information in the context of closed world assumptions, non-deterministic inductive constructions, causal processes, and ramifications. C-LOG allows for example nested occurrences of causal rules.

It is straightforward to integrate first-order logic (FO) with C-LOG, offering an expressive modelling language in which causal processes as well as assertional knowledge in the form of axioms and constraints can be naturally expressed. We call this integration FO(C).[3] FO(C) fits in the FO($\cdot$) research project [5], which aims at integrating expressive language constructs with a Tarskian model semantics in a unified language.

An example of a C-LOG expression is the following

$$\left\{\begin{array}{l}\mathbf{All}\, p[\mathrm{Apply}(p) \land \mathrm{PassedTest}(p)] : \mathrm{PermRes}(p).\\ (\mathbf{Select}\, p[\mathrm{Participate}(p)] : \mathrm{PermRes}(p)) \leftarrow \mathrm{Lott}.\end{array}\right\}$$

This describes that all persons who pass a naturalisation test obtain permanent residence in the U.S., and that one person who participates in the green card lottery also obtains residence. The person that is selected for the lottery can either be one of the persons that also passed the naturalisation test, or someone else. There are local closed world assumptions: in the example, the endogenous predicate $\mathrm{PermRes}$ only holds for the people passing the test and at most one extra person. We could add an FO constraint to this theory, for example $\forall p : \mathrm{Participate}(p) \Rightarrow \mathrm{Apply}(p)$. This results in a FO(C) theory; a structure is a model of this theory if it is a model of the C-LOG expression and no-one participates in the lottery without applying the normal way.

So far, very little is known about inference in FO(C). No systems exist to reason with FO(C), and complexity of inference in FO(C) has not been studied. This paper studies both of the above problems.

The rest of this paper is structured as follows: in Section 2, we repeat some preliminaries, including a very brief overview of the semantics of FO(C). In Section 3 we define normal forms on FO(C) and transformations between these normal forms. We also argue that one of these normal forms corresponds to FO(ID) [7] and hence, that IDP [4] can be seen as the first FO(C)-solver. In Section 4 we give an example that illustrates both the semantics of FO(C) and the transformations. Afterwards, in Section 5, we define inference tasks for FO(C) and study their complexity. We conclude in Section 6.

2 Preliminaries

We assume familiarity with basic concepts of FO. Vocabularies, formulas, and terms are defined as usual. A Σ-structure I interprets all symbols (including variable symbols) in Σ; D^I denotes the domain of I and σ^I, with σ a symbol in Σ, the interpretation of σ in I. We use $I[\sigma : v]$ for the structure J that equals I, except on σ: $\sigma^J = v$. *Domain atoms* are atoms of the form $P(\overline{d})$ where the d_i are domain elements. We use restricted quantifications. 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 some) 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 in C-LOG. 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.

Our logic has a standard, two-valued Tarskian semantics, which means that models represent possible states of affairs. Three-valued logic with partial domains is used as a technical device to express intermediate stages of causal processes. A truth-value is one of the following: $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, where $\mathbf{f}^{-1} = \mathbf{t}, \mathbf{t}^{-1} = \mathbf{f}$ and $\mathbf{u}^{-1} = \mathbf{u}$. Two partial orders are defined on truth values: the precision order $\leq_p$, given by $\mathbf{u} \leq_p \mathbf{t}$ and $\mathbf{u} \leq_p \mathbf{f}$ and the truth order $\mathbf{f} \leq \mathbf{u} \leq \mathbf{t}$. Let D be a set, a *partial set* $\mathcal{S}$ in D is a function from D to truth values. We identify a partial set with a tuple $(\mathcal{S}_{ct}, \mathcal{S}_{pt})$ of two sets, where the *certainly true set* $\mathcal{S}_{ct}$ is $\{x \mid \mathcal{S}(x) = \mathbf{t}\}$ and the *possibly true set* $\mathcal{S}_{pt}$ is $\{x \mid \mathcal{S}(x) \neq \mathbf{f}\}$. The union, intersection, and subset-relation of partial sets are defined pointwise. For a truth value v, we define the restriction of a partial set $\mathcal{S}$ to this truth-value, denoted $r(\mathcal{S}, v)$, as the partial set mapping every $x \in D$ to $\min_{\leq}(\mathcal{S}(x), v)$. Every set S is also a partial set, namely the tuple (S, S).

[1] Department of Computer Science, KU Leuven, email: firstname.name@cs.kuleuven.be
[2] Hasselt University & transnational University of Limburg, email: jan.vandenbussche@uhasselt.be
[3] Previously, this language was called FO(C-LOG)

A *partial* Σ-*structure* I consists of 1) a *domain* D^I: a partial set of elements, and 2) a mapping associating a value to each symbol in Σ; for constants and variables, this value is in D^I_{ct}, for predicate symbols of arity n, this is a partial set P^I in $(D^I_{\mathrm{pt}})^n$. We often abuse notation and use the domain D as if it were a predicate. A partial structure I is *two-valued* if for all predicates P (including D), $P^I_{\mathrm{ct}} = P^I_{\mathrm{pt}}$. There is a one-to-one correspondence between two-valued partial structures and structures. If I and J are two partial structures with the same interpretation for constants, we call I more precise than J ($I \geq_p J$) if for all its predicates P (including D), $P^I_{\mathrm{ct}} \supseteq P^J_{\mathrm{ct}}$ and $P^I_{\mathrm{pt}} \subseteq P^J_{\mathrm{pt}}$.

Definition 2.1. *We define the value of an FO formula φ in a partial structure I inductively based on the Kleene truth tables.*

- $P(\bar{t})^I = P^I(\bar{t}^I)$,
- $(\neg\varphi)^I = ((\varphi)^I)^{-1}$
- $(\varphi \wedge \psi)^I = \min_{\leq}\left(\varphi^I, \psi^I\right)$
- $(\varphi \vee \psi)^I = \max_{\leq}\left(\varphi^I, \psi^I\right)$
- $(\forall x : \varphi)^I = \min_{\leq}\left\{\max(D^I(d)^{-1}, \varphi^{I[x:d]}) \mid d \in D^I_{\mathrm{pt}}\right\}$
- $(\exists x : \varphi)^I = \max_{\leq}\left\{\min(D^I(d), \varphi^{I[x:d]}) \mid d \in D^I_{\mathrm{pt}}\right\}$

In what follows we briefly repeat the syntax and formal semantics of C-LOG. For more details, an extensive overview of the informal semantics of CEEs, and examples of CEEs, we refer to [3].

2.1 Syntax of C-LOG

Definition 2.2. Causal effect expressions *(CEE) are defined inductively as follows:*

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

We call a CEE an *atom-* (respectively *rule-*, $\mathbf{And}$-, $\mathbf{Or}$-, $\mathbf{All}$-, $\mathbf{Select}$- or $\mathbf{New}$-*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. All other symbols are called *exogenous* in C. An occurrence of a variable x is *bound* in a CEE if it occurs in the scope of a quantification over that variable ($\forall x$, $\exists x$, $\mathbf{All}\, x$, $\mathbf{Select}\, x$, or $\mathbf{New}\, x$) 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 finite set of CEEs; the intended causal theory is the $\mathbf{And}$-conjunction of these CEEs. We often use Δ for a causal theory and C, C', C_1 and C_2 for its subexpressions. We stress that the connectives in CEEs differ from their FO counterparts. E.g., in the example in the introduction, the CEE expresses that there is a cause for several persons to become American (those who pass the test and maybe one extra lucky person). This implicitly also says that every person without cause for becoming American is not American. As such C-LOG-expressions are highly non-monotonic.

2.2 Semantics of C-LOG

Definition 2.3. *Let Δ be a causal theory; we associate a parse-tree with Δ. An* occurrence *of a CEE C in Δ is a node in the parse tree of Δ labelled with C. The* variable context *of an occurrence of a CEE C in Δ is the sequence of quantified variables as they occur on the path from Δ to C in the parse-tree of Δ. If $\overline{x}$ is the variable context of C in Δ, we denote C as $C\langle\overline{x}\rangle$ and the length of $\overline{x}$ as n_C.*

For example, the variable context of $P(x)$ in $\mathbf{Select}\, y[Q(y)] : \mathbf{All}\, x[Q(x)] : P(x)$ is $[y, x]$. *Instances* of an occurrence $C\langle\overline{x}\rangle$ correspond to assignments $\overline{d}$ of domain elements to $\overline{x}$.

Definition 2.4. *Let Δ be a causal theory and D a set. A Δ-*selection *ζ in D consists of*

- *for every occurrence C of a $\mathbf{Select}$-expression in Δ, a total function $\zeta^{sel}_C : D^{n_C} \to D$,*
- *for every occurrence C of a $\mathbf{Or}$-expression in Δ, a total function $\zeta^{or}_C : D^{n_C} \to \{1, 2\}$,*
- *for every occurrence C of a $\mathbf{New}$-expression in Δ, an injective partial function $\zeta^{new}_C : D^{n_C} \to D$.*

such that furthermore the images of all functions ζ^{new}_C are disjoint (i.e., such that every domain element can be created only once).

The initial elements *of ζ are those that do not occur as image of one of the ζ^{new}_C-functions: $\zeta^{in} = D \setminus \cup_C image(\zeta^{new}_C)$, where the union ranges over all occurrences of $\mathbf{New}$-expressions.*

The effect set of a CEE in a partial structure is a partial set: it contains information on everything that is caused and everything that might be caused. For defining the semantics a new, unary predicate $\mathcal{U}$ is used.

Definition 2.5. *Let Δ be a CEE and J a partial structure. Suppose ζ is a Δ-selection in a set $D \supseteq D^J_{\mathrm{pt}}$. Let C be an occurrence of a CEE in Δ. The* effect set *of C with respect to J and ζ is a partial set of domain atoms, defined recursively:*

- *If C is $P(\bar{t})$, then $\mathrm{eff}_{J,\zeta}(C) = \{P(\bar{t}^J)\}$,*
- *if C is C_1 $\mathbf{And}$ C_2, then $\mathrm{eff}_{J,\zeta}(C) = \mathrm{eff}_{J,\zeta}(C_1) \cup \mathrm{eff}_{J,\zeta}(C_2)$,*
- *if C is $C' \leftarrow \varphi$, then $\mathrm{eff}_{J,\zeta}(C) = r(\mathrm{eff}_{J,\zeta}(C'), \varphi^J)$,*
- *if C is $\mathbf{All}\, x[\varphi] : C'$, then*

  $\mathrm{eff}_{J,\zeta}(C) = \bigcup\Big\{r\big(\mathrm{eff}_{J',\zeta}(C'), \min_{\leq}(D^J(d), \varphi^{J'})\big) \mid d \in D^J_{\mathrm{pt}}$ *and* $J' = J[x : d]\Big\}$

- *if $C\langle\overline{y}\rangle$ is C_1 $\mathbf{Or}$ C_2, then*
  - $\mathrm{eff}_{J,\zeta}(C) = \mathrm{eff}_{J,\zeta}(C_1)$ *if* $\zeta^{or}_C(\overline{y}^J) = 1$,
  - *and* $\mathrm{eff}_{J,\zeta}(C) = \mathrm{eff}_{J,\zeta}(C_2)$ *otherwise*
- *if $C\langle\overline{y}\rangle$ is $\mathbf{Select}\, x[\varphi] : C'$, let $e = \zeta^{sel}_C(\overline{y}^J)$, $J' = J[x : e]$ and $v = \min_{\leq}(D^J(e), \varphi^{J'})$. Then $\mathrm{eff}_{J,\zeta}(C) = r(\mathrm{eff}_{J,\zeta}(C'), v)$,*
- *if $C\langle\overline{y}\rangle$ is $\mathbf{New}\, x : C'$, then*
  - $\mathrm{eff}_{J,\zeta}(C) = \emptyset$ *if* $\zeta^{new}_C(\overline{y}^J)$ *does not denote,*
  - *and* $\mathrm{eff}_{J,\zeta}(C) = \{\mathcal{U}(\zeta^{new}_C(\overline{y}^J))\} \cup \mathrm{eff}_{J',\zeta}(C')$, *where* $J' = J[x : \zeta^{new}_C(\overline{y}^J)]$ *otherwise,*

An instance of an occurrence of a CEE in Δ is relevant *if it is encountered in the evaluation of $\mathrm{eff}_{I,\zeta}(\Delta)$. We say that C* succeeds[4] *with ζ in J if for all relevant occurrences $C\langle\overline{y}\rangle$ of $\mathbf{Select}$-expressions, $\zeta^{sel}_C(\overline{y}^J)$ satisfies the qualification of C and for all relevant instances $C\langle\overline{y}\rangle$ of $\mathbf{New}$-expressions, $\zeta^{new}_C(\overline{y}^J)$ denotes.*

[4] Previously, we did not say that C "succeeds", but that the effect set "is a possible effect set". We believe this new terminology is more clear.

Given a structure I (and a Δ-selection ζ), two lattices are defined: $L^{\Sigma}_{I,\zeta}$ denotes the set of all Σ-structures J with $\zeta^{in} \subseteq D^J \subseteq D^I$ such that for all exogenous symbols σ of arity n: $\sigma^J = \sigma^I \cap (D^J)^n$. This set is equipped with the truth order. And L^{Σ}_I denotes the sublattice of $L^{\Sigma}_{I,\zeta}$ consisting of all structures in $L^{\Sigma}_{I,\zeta}$ with domain equal to D^I.

A partial structure corresponds to an element of the bilattice $(L^{\Sigma}_{I,\zeta})^2$; the bilattice is equipped with the precision order.

Definition 2.6. *Let I be a structure and ζ a Δ-selection in D^I. The* partial immediate causality operator A_ζ *is the operator on $(L^{\Sigma}_{I,\zeta})^2$ that sends partial structure J to a partial structure J' such that*

- $D^{J'}(d) = \mathbf{t}$ *if* $d \in \zeta^{in}$ *and* $D^{J'}(d) = \mathrm{eff}_{J,\zeta}(\Delta)(\mathcal{U}(d))$ *otherwise*
- *for endogenous symbols P,* $P(\overline{d})^{J'} = \mathrm{eff}_{J,\zeta}(\Delta)(P(\overline{d}))$.

Such operators have been studied intensively in the field of Approximation Fixpoint Theory [6]; and for such operators, the well-founded fixpoint has been defined in [6]. The semantics of C-LOG is defined in terms of this well-founded fixpoint in [3]:

Definition 2.7. *Let Δ be a causal theory. We say that structure I is a* model *of Δ (notation $I \models \Delta$) if there exists a Δ-selection ζ such that (I,I) is the well-founded fixpoint of A_ζ, and Δ succeeds with ζ in I.*

FO(C) is the integration of FO and C-LOG. An FO(C) theory consists of a set of causal theories and FO sentences. A structure I is a model of an FO(C) theory if it is a model of all its causal theories and FO sentences. In this paper, we assume, without loss of generality, that an FO(C) theory $\mathcal{T}$ has exactly one causal theory.

3 A Transformation to DefF

In this section we present normal forms for FO(C) and transformations between these normal forms. The transformations we propose preserve equivalence modulo newly introduced predicates:

Definition 3.1. *Suppose $\Sigma \subseteq \Sigma'$ are vocabularies, $\mathcal{T}$ is an* FO(C) *theory over Σ and $\mathcal{T}'$ is an* FO(C) *theory over Σ'. We call $\mathcal{T}$ and $\mathcal{T}'$* Σ-equivalent *if each model of $\mathcal{T}$, can be extended to a model of $\mathcal{T}'$ and the restriction of each model of $\mathcal{T}'$ to Σ is a model of $\mathcal{T}$.*

From now on, we use $\mathbf{All}\,\overline{x}[\varphi] : C'$, where $\overline{x}$ is a tuple of variables as syntactic sugar for $\mathbf{All}\,x_1[\mathbf{t}] : \mathbf{All}\,x_2[\mathbf{t}] : \dots \mathbf{All}\,x_n[\varphi] : C'$, and similar for $\mathbf{Select}$-expressions. If $\overline{x}$ is a tuple of length 0, $\mathbf{All}\,\overline{x}[\varphi] : C'$ is an abbreviation for $C' \leftarrow \varphi$. It follows directly from the definitions that $\mathbf{And}$ and $\mathbf{Or}$ are associative, hence we use $C_1\,\mathbf{And}\,C_2\,\mathbf{And}\,C_3$ as an abbreviation for $(C_1\,\mathbf{And}\,C_2)\,\mathbf{And}\,C_3$ and for $C_1\,\mathbf{And}\,(C_2\,\mathbf{And}\,C_3)$, and similar for $\mathbf{Or}$-expressions.

3.1 Normal Forms

Definition 3.2. *Let C be an occurrence of a CEE in C'. The* nesting depth *of C in C' is the depth of C in the parse-tree of C'. In particular, the nesting depth of C' in C' is always* 0*. The* height *of C' is the maximal nesting depth of occurrences of CEEs in C'. In particular, the height of atom-expressions is always* 0*.*

Example 3.3. Let Δ be $A\,\mathbf{And}\,((\mathbf{All}\,x[P(x)] : Q(x))\,\mathbf{Or}\,B)$. The nesting depth of B in Δ is 2 and the height of Δ is 3.

Definition 3.4. *A* C-LOG *theory is* creation-free *if it does not contain any $\mathbf{New}$-expressions, it is* deterministic *if it is creation-free and it does not contain any $\mathbf{Select}$ or $\mathbf{Or}$-expressions. An* FO(C) *is* creation-free *(resp.* deterministic*) if its (unique)* C-LOG *theory is.*

Definition 3.5. *A* C-LOG *theory is in* Nesting Normal Form (NestNF) *if it is of the form $C_1\,\mathbf{And}\,C_2\,\mathbf{And}\,C_3\,\mathbf{And}\,\dots$ where each of the C_i is of the form $\mathbf{All}\,\overline{x}[\varphi_i] : C'_i$ and each of the C'_i has height at most one. A* C-LOG *theory Δ is in* Definition Form (DefF) *if it is in NestNF and each of the C'_i have height zero, i.e., they are atom-expressions. An* FO(C) *theory is NestNF (respectively DefF) if its corresponding* C-LOG *theory is.*

Theorem 3.6. *Every* FO(C) *theory over Σ is Σ-equivalent with an* FO(C) *theory in DefF.*

We will prove this result in 3 parts: in Section 3.4, we show that every FO(C) theory can be transformed to NestNF, in Section 3.3, we show that every theory in NestNF can be transformed into a deterministic theory and in Section 3.2, we show that every deterministic theory can be transformed to DefF. The FO sentences in an FO(C) theory do not matter for the normal forms, hence most results focus on the C-LOG part of FO(C) theories.

3.2 From Deterministic FO(C) to DefF

Lemma 3.7. *Let Δ be a* C-LOG *theory. Suppose C is an occurrence of an expression $\mathbf{All}\,\overline{x}[\varphi] : C_1\,\mathbf{And}\,C_2$. Let Δ' be the causal theory obtained from Δ by replacing C with $(\mathbf{All}\,\overline{x}[\varphi] : C_1)\,\mathbf{And}\,(\mathbf{All}\,\overline{x}[\varphi] : C_2)$. Then Δ and Δ' are equivalent.*

Proof. It is clear that Δ and Δ' have the same selection functions. Furthermore, it follows directly from the definitions that given such a selection, the defined operators are equal. □

Repeated applications of the above lemma yield:

Lemma 3.8. *Every deterministic* FO(C) *theory is equivalent with an* FO(C) *theory in DefF.*

3.3 From NestNF to Deterministic FO(C)

Lemma 3.9. *If $\mathcal{T}$ is an* FO(C) *theory in NestNF over Σ, then $\mathcal{T}$ is Σ-equivalent with a deterministic* FO(C) *theory.*

We will prove Lemma 3.9 using a strategy that replaces a Δ-selection by an interpretation of new predicates (one per occurrence of a non-deterministic CEE). The most important obstacle for this transformation are $\mathbf{New}$-expressions. In deterministic C-LOG, no constructs influence the domain. This has as a consequence that the immediate causality operator for a deterministic C-LOG theory is defined in a lattice of structures with fixed domain, while in general, the operator is defined in a lattice with variable domains. In order to bridge this gap, we use two predicates to describe the domain, $\mathcal{S}$ are the initial elements and $\mathcal{U}$ are the created, the union of the two is the domain. Suppose a C-LOG theory Δ over vocabulary Σ is given.

Definition 3.10. *We define the* Δ-selection vocabulary Σ^s_Δ *as the vocabulary consisting of:*

- *a unary predicate $\mathcal{S}$,*
- *for every occurrence C of a $\mathbf{Or}$-expression in Δ, a new n_C-ary predicate* $\mathrm{Choose1}_C$*,*
- *for every occurrence C of a $\mathbf{Select}$-expression in Δ, a new (n_C+1)-ary predicate* Sel_C*,*
- *for every occurrence C of a $\mathbf{New}$-expression in Δ, a new (n_C+1)-ary predicate* Create_C*,*

Intuitively, a Σ^s_Δ-structure corresponds to a Δ-selection: $\mathcal{S}$ correspond to ζ^{in}, $\mathrm{Choose1}_C$ to ζ^{or}_C, Sel_C to ζ^{sel}_C and Create_C to ζ^{new}_C.

Lemma 3.11. *There exists an FO theory* S_Δ *over* Σ^s_Δ *such that there is a one-to-one correspondence between* Δ*-selections in* D *and models of* S_Δ *with domain* D.

Proof. This theory contains sentences that express that Sel_C is functional, and that Create_C is a partial function. It is straightforward to do this in FO (with among others, constraints such as $\forall \overline{x} : \exists y : \mathrm{Sel}_C(\overline{x}, y)$). Furthermore, it is also easy to express that the Create_C functions are injective, and that different **New**-expressions create different elements. Finally, this theory relates $\mathcal{S}$ to the Create_C expressions: $\forall y : \mathcal{S}(y) \Leftrightarrow \neg \bigvee_C (\exists \overline{x} : \mathrm{Create}_C(\overline{x}, y))$ where the disjunction ranges over all occurrences C of **New**-expressions. □

The condition that a causal theory succeeds can also be expressed as an FO theory. For that, we need one more definition.

Definition 3.12. *Let* Δ *be a causal theory in NestNF and let* C *be one of the* C'_i *in definition 3.5, then we call* φ_i *(again, from definition 3.5) the* relevance condition *of* C *and denote it* Rel_C.

In what follows, we define one more extended vocabulary. First, we use it to express the constraints that Δ succeeds and afterwards, for the actual transformation.

Definition 3.13. *The* Δ-transformed vocabulary Σ^t_Δ *is the disjoint union of* Σ *and* Σ^s_Δ *extended with the unary predicate symbol* $\mathcal{U}$.

Lemma 3.14. *Suppose* Δ *is a causal theory in NestNF, and* ζ *is a* Δ*-selection with corresponding* Σ^s_Δ*-structure* M. *There exists an FO theory* Succ_Δ *such that for every (two-valued) structure* I *with* $I|_{\Sigma^s_\Delta} = M$, Δ *succeeds with respect to* I *and* ζ *iff* $I \models \mathrm{Succ}_\Delta$.

Proof. Δ is in NestNF; for every of the C'_i (as in Definition 3.5), $\mathrm{Rel}_{C'_i}$ is true in I if and only if C'_i is relevant. Hence, for Succ_Δ we can take the FO theory consisting of the following sentences:

- $\forall \overline{x} : \mathrm{Rel}_C \Rightarrow \exists y : \mathrm{Create}_C(\overline{x}, y)$, for all **New**-expressions $C\langle\overline{x}\rangle$ in Δ,
- $\forall \overline{x} : \mathrm{Rel}_C \Rightarrow \exists y : (\mathrm{Sel}_C(\overline{x}, y) \wedge \psi)$, for all **Select**-expressions $C\langle\overline{x}\rangle$ of the form $\mathbf{Select}\, y[\psi] : C'$ in Δ. □

Now we describe the actual transformation: we translate every quantification into a relativised version, make explicit that a **New**-expression causes an atom $\mathcal{U}(d)$, and eliminate all non-determinism using the predicates in Σ^s_Δ.

Definition 3.15. *Let* Δ *be a* C-LOG *theory over* Σ *in NestNF. The* transformed theory Δ^t *is the theory obtained from* Δ *by applying the following transformation:*

- *first replacing all quantifications* $\alpha x[\psi] : \chi$, *where* $\alpha \in \{\forall, \exists, \mathbf{Select}, \mathbf{All}\}$ *by* $\alpha x[(\mathcal{U}(x) \vee \mathcal{S}(x)) \wedge \psi] : \chi$
- *subsequently replacing each occurrence* $C\langle\overline{x}\rangle$ *of an expression* $\mathbf{New}\, y : C'$ *by* $\mathbf{All}\, y[\mathrm{Create}_C(\overline{x}, y)] : \mathcal{U}(y)\ \mathbf{And}\ C'$,
- *replacing every occurrence* $C\langle\overline{x}\rangle$ *of an expression* $C_1\ \mathbf{Or}\ C_2$ *by* $(C_1 \leftarrow \mathrm{Choose1}_C(\overline{x}))\mathbf{And}(C_2 \leftarrow \neg\mathrm{Choose1}_C(\overline{x}))$,
- *and replacing every occurrence* $C\langle\overline{x}\rangle$ *of an expression* $\mathbf{Select}\, y[\varphi] : C'$ *by* $\mathbf{All}\, y[\varphi \wedge \mathrm{Sel}_C(\overline{x}, y)] : C'$.

Given a structure I and a Δ-selection ζ, there is an obvious lattice morphism $m_\zeta : L^\Sigma_{I,\zeta} \to L^{\Sigma^t_\Delta}_I$ mapping a structure J to the structure J' with domain $D^{J'} = D^I$ interpreting all symbols in Σ^s_Δ according to ζ (as in Lemma 3.11), all symbols in Σ (except for the domain) the same as I and interpreting $\mathcal{U}$ as $D^J \setminus \mathcal{S}^{J'}$. m_ζ can straightforwardly be extended to a bilattice morphism.

Lemma 3.16. *Let* ζ *be a* Δ*-selection for* Δ *and* A_ζ *and* A *be the partial immediate causality operators of* Δ *and* Δ^t *respectively. Let* J *be any partial structure in* $(L^\Sigma_{I,\zeta})^2$. *Then* $m_\zeta(A_\zeta(J)) = A(m_\zeta(J))$.

Idea of the proof. **New**-expressions $\mathbf{New}\, y : C'$ in Δ have been replaced by **All** expressions causing two subexpressions: $\mathcal{U}(y)$ and the C' for exactly the y's that are created according to ζ. Furthermore, the relativisation of all other quantifications guarantees that we correctly evaluate all quantifications with respect to the domain of J, encoded in $\mathcal{S} \cup \mathcal{U}$.

Furthermore, all non-deterministic expressions have been changed into **All**-expressions that are conditionalised by the Δ-selection; this does not change the effect set; thus, the operators correspond. □

Lemma 3.17. *Let* ζ, A_ζ *and* A *be as in lemma 3.16. If* I *is the well-founded model of* A_ζ, $m_\zeta(I)$ *is the well-founded model of* A.

Proof. Follows directly from lemma 3.16: the mapping $J \mapsto m_\zeta(J)$ is an isomorphism between $L^\Sigma_{I,\zeta}$ and the sublattice of $L^{\Sigma^t_\Delta}_{I,\zeta'}$ consisting of those structures such that the interpretations of $\mathcal{S}$ and $\mathcal{U}$ have an empty intersection. As this isomorphism maps A_ζ to A, their well-founded models must agree. □

Lemma 3.18. *Let* Δ *be a causal theory in NestNF,* ζ *a* Δ*-selection for* Δ *and* I *a* Σ*-structure. Then* $I \models \Delta$ *if and only if* $m_\zeta(I) \models \Delta^t$ *and* $m_\zeta(I) \models \mathrm{S}_\Delta$ *and* $m_\zeta(I) \models \mathrm{Succ}_\Delta$.

Proof. Follows directly from Lemmas 3.17, 3.11 and 3.14. □

Proof of Lemma 3.9. Let Δ be the C-LOG theory in $\mathcal{T}$. We can now take as deterministic theory the theory consisting of Δ^t, all FO sentences in $\mathcal{T}$, and the sentence $\mathrm{S}_\Delta \wedge \mathrm{Succ}_\Delta \wedge \forall x : \mathcal{S}(x) \Leftrightarrow \neg\mathcal{U}(x)$, where the last formula excludes all structures not of the form $m_\zeta(I)$ for some I (the created elements $\mathcal{U}$ and the initial elements $\mathcal{S}$ should form a partition of the domain). □

3.4 From General FO(C) to NestNF

In the following definition we use $\Delta[C'/C]$ for the causal theory obtained from Δ by replacing the occurrence of a CEE C by C'.

Definition 3.19. *Suppose* $C\langle\overline{x}\rangle$ *is an occurrence of a CEE in* Δ. *With* $\mathtt{Unnest}(\Delta, C)$ *we denote the causal theory* $\Delta[P(\overline{x})/C]\ \mathbf{And}\ \mathbf{All}\, \overline{x}[P(\overline{x})] : C$ *where* P *is a new predicate symbol.*

Lemma 3.20. *Every* FO(C) *theory is* Σ*-equivalent with an* FO(C) *theory in NestNF.*

Proof. First, we claim that for every C-LOG theory over Σ, Δ and $\mathtt{Unnest}(\Delta, C)$ are Σ-equivalent. It is easy to see that the two theories have the same Δ-selections. Furthermore, the operator for $\mathtt{Unnest}(\Delta, C)$ is a part-to-whole monotone fixpoint extension[5] (as defined in [8]) of the operator for Δ. In [8] it is shown that in this case, their well-founded models agree, which proves our claim. The lemma now follows by repeated applications of the claim. □

Proof of Theorem 3.6. Follows directly by combining lemmas 3.20, 3.9 and 3.8. For transformations only defined on C-LOG theories, the extra FO part remains unchanged. □

[5] Intuitively, a part-to-whole fixpoint extension means that all predicates only depend positively on the newly introduced predicates

3.5 FO(C) and FO(ID)

An inductive definition (ID) [7] is a set of rules of the form $\forall \overline{x} : P(\bar{t}) \leftarrow \varphi$, an FO($ID$) theory is a set of FO sentences and IDs, and an $\exists$SO(ID) theory is a theory of the form $\exists \bar{P} : \mathcal{T}$, where $\mathcal{T}$ is an FO(ID) theory. A causal theory in DefF corresponds exactly to an ID: the CEE $\mathbf{All}\,\overline{x}[\varphi] : P(\bar{t})$ corresponds to the above rule and the **And**-conjunction of such CEEs to the set of corresponding rules. The partial immediate consequence operator for IDs defined in [7] is exactly the partial immediate causality operator for the corresponding C-LOG theory. Combining this with Theorem 3.6, we find (with $\bar{P}$ the introduced symbols):

Theorem 3.21. *Every* FO(C) *theory is equivalent with an* $\exists$SO(ID) *formula of the form* $\exists \bar{P} : \{\Delta, \mathcal{T}\}$, *where* Δ *is an ID and* $\mathcal{T}$ *is an FO sentence.*

Theorem 3.21 implies that we can use reasoning engines for FO(ID) in order to reason with FO(C), as long as we are careful with the newly introduced predicates. We implemented a prototype of this transformation in the IDP system [4], it can be found at [2].

4 Example: Natural Numbers

Example 4.1. Let Σ be a vocabulary consisting of predicates $\mathrm{Nat}/1, \mathrm{Succ}/2$ and $\mathrm{Zero}/1$ and suppose $\mathcal{T}$ is the following theory:

$$\left\{ \begin{array}{l} \mathbf{New}\, x : \mathrm{Nat}(x)\ \mathbf{And}\ \mathrm{Zero}(x) \\ \mathbf{All}\, x[\mathrm{Nat}(x)] : \mathbf{New}\, y : \mathrm{Nat}(y)\ \mathbf{And}\ \mathrm{Succ}(x,y) \end{array} \right\}$$

This theory defines a process creating the natural numbers. Transforming it to NestNF yields:

$$\left\{ \begin{array}{l} \mathbf{New}\, x : T_1(x) \\ \mathbf{All}\, x[T_1(x)] : \mathrm{Nat}(x) \\ \mathbf{All}\, x[T_1(x)] : \mathrm{Zero}(x) \\ \mathbf{All}\, x[\mathrm{Nat}(x)] : \mathbf{New}\, y : T_2(x,y) \\ \mathbf{All}\, x,y[T_2(x,y)] : \mathrm{Nat}(y) \\ \mathbf{All}\, x,y[T_2(x,y)] : \mathrm{Succ}(x,y), \end{array} \right\}$$

where T_1 and T_2 are auxiliary symbols. Transforming the resulting theory into deterministic C-LOG requires the addition of more auxiliary symbols $\mathcal{S}/1, \mathcal{U}/1, \mathrm{Create}_1/1$ and $\mathrm{Create}_2/2$ and results in the following C-LOG theory (together with a set of FO-constraints):

$$\left\{ \begin{array}{l} \mathbf{All}\, x[\mathrm{Create}_1(x)] : \mathcal{U}(x)\ \mathbf{And}\ T_1(x) \\ \mathbf{All}\, x[(\mathcal{U}(x) \lor \mathcal{S}(x)) \land T_1(x)] : \mathrm{Nat}(x) \\ \ldots \end{array} \right\}$$

This example shows that the proposed transformation is in fact too complex. E.g., here, almost all occurrences of $\mathcal{U}(x) \lor \mathcal{S}(x)$ are not needed. This kind of redundancies can be eliminated by executing the three transformations (from Sections 3.2, 3.3 and 3.4) simultaneously. In that case, we would get the simpler deterministic theory:

$$\left\{ \begin{array}{l} \mathbf{All}\, x[\mathrm{Create}_1(x)] : \mathrm{Nat}(x)\ \mathbf{And}\ \mathrm{Zero}(x)\ \mathbf{And}\, \mathcal{U}(x) \\ \mathbf{All}\, x,y[(\mathcal{U}(x) \lor \mathcal{S}(x)) \land \mathrm{Nat}(x) \land \mathrm{Create}_2(x,y)] : \\ \quad \mathrm{Nat}(y)\ \mathbf{And}\ \mathrm{Succ}(x,y)\ \mathbf{And}\, \mathcal{U}(y) \end{array} \right\}$$
$$\forall x : \mathcal{U}(x) \Leftrightarrow \neg \mathcal{S}(x)$$
$$\forall y : \mathcal{S}(y) \Leftrightarrow \neg(\mathrm{Create}_1(y) \lor \exists x : \mathrm{Create}_2(x,y)).$$
$$\exists x : \mathrm{Create}_1(x).$$
$$\ldots$$

These sentences express the well-known constraints on $\mathbb{N}$: there is at least one natural number (identified by Create_1), and every number has a successor. Furthermore the initial element and the successor elements are unique, and all are different. Natural numbers are defined as zero and all elements reachable from zero by the successor relation. The theory we started from is much more compact and much more readable than any FO(ID) theory defining natural numbers. This shows the Knowledge Representation power of C-LOG.

5 Complexity Results

In this section, we provide complexity results. We focus on the C-LOG fragment of FO(C) here, since complexity for FO is well-studied. First, we formally define the inference methods of interest.

5.1 Inference Tasks

Definition 5.1. *The* model checking *inference takes as input a* C-LOG *theory* Δ *and a finite (two-valued) structure* I. *It returns true if* $I \models \Delta$ *and false otherwise.*

Definition 5.2. *The* model expansion *inference takes as input a* C-LOG *theory* Δ *and a partial structure* I *with finite two-valued domain. It returns a model of* Δ *more precise than* I *if one exists and "unsat" otherwise.*

Definition 5.3. *The* endogenous model expansion *inference is a special case of model expansion where* I *is two-valued on exogenous symbols of* Δ *and completely unknown on endogenous symbols.*

The next inference is related to database applications. In the database world, languages with object creation have also been defined [1]. A query in such a language can create extra objects, but the interpretation of exogenous symbols (tables in the database) is fixed, i.e., exogenous symbols are always false on newly created elements.

Definition 5.4. *The* unbounded query *inference takes as input a* C-LOG *theory* Δ, *a partial structure* I *with finite two-valued domain such that* I *is two-valued on exogenous symbols of* Δ *and completely unknown on endogenous symbols of* Δ, *and a propositional atom* P. *This inference returns true if there exist i) a structure* J, *with* $D^J \supseteq D^I$, $\sigma^J = \sigma^I$ *for exogenous symbols* σ, *and* $P^J = \mathbf{t}$ *and ii) a* Δ*-selection* ζ *in* D^J *with* $\zeta^{in} = D^I$, *such that* J *is a model of* Δ *with* Δ*-selection* ζ. *It returns false otherwise.*

5.2 Complexity of Inference Tasks

In this section, we study the datacomplexity of the above inference tasks, i.e., the complexity for fixed Δ.

Lemma 5.5. *For a finite structure* I, *computing* $A_\zeta(I)$ *is polynomial in the size of* I *and* ζ.

Proof. In order to compute $A_\zeta(I)$, we need to evaluate a fixed number of FO-formulas a polynomial number of times (with exponent in the nesting depth of Δ). As evaluating a fixed FO formula in the context of a partial structure is polynomial, the result follows. ☐

Theorem 5.6. *For a finite structure* I, *the task of computing the* A_ζ*-well-founded model of* Δ *in the lattice* $L^\Sigma_{I,\zeta}$ *is polynomial in the size of* I *and* ζ.

Proof. Calculating the well-founded model of an approximator can be done with a polynomial number of applications of the approximator. Furthermore, Lemma 5.5 guarantees that each of these applications is polynomial as well. □

Theorem 5.7. *Model expansion for* C-LOG *is NP-complete.*

Proof. After guessing a model and a Δ-selection, Theorem 5.6 guarantees that checking that this is the well-founded model is polynomial. Lemma 3.14 shows that checking whether Δ succeeds is polynomial as well. Thus, model expansion is in NP.

NP-hardness follows from the fact that model expansion for inductive definitions is NP-hard and inductive definitions are shown to be a subclass of C-LOG theories, as argued in Section 3.5. □

Example 5.8. We show how the SAT-problem can be encoded as model checking for C-LOG. Consider a vocabulary Σ_{IN}^{SAT} with unary predicates Cl and PS and with binary predicates Pos and Neg. Every SAT-problem can be encoded as a Σ_{IN}^{SAT}-structure: Cl and PS are interpreted as the sets of clauses and propositional symbols respectively, $\mathrm{Pos}(c,p)$ (respectively $\mathrm{Neg}(c,p)$) holds if clause c contains the literal p (respectively $\neg p$).

We now extend Σ_{IN}^{SAT} to a vocabulary Σ_{ALL}^{SAT} with unary predicates Tr and Fa and a propositional symbol Sol. Tr and Fa encode an assignment of values (true or false) to propositional symbols, Sol means that the encoded assignment is a solution to the SAT problem. Let Δ_{SAT} be the following causal theory:

$$\mathbf{All}\, p[\mathrm{PS}(p)] : \mathrm{Tr}(p)\, \mathbf{Or}\, \mathrm{Fa}(p)$$
$$\mathrm{Sol} \leftarrow \forall c[\mathrm{Cl}(c)] : \exists p : (\mathrm{Pos}(c,p) \wedge \mathrm{Tr}(p) \vee (\mathrm{Neg}(c,p) \wedge \mathrm{Fa}(p))$$

The first rules guesses an assignment. The second rule says that Sol holds if every clause has at least one true literal. Model expansion of that theory with a structure interpreting Σ_{IN}^{SAT} according to a SAT problem and interpreting Sol as true, is equivalent with solving that SAT problem, hence model expansion is NP-hard (which we already knew). In order to show that model *checking* is NP-hard, we add the following CEE to the theory Δ_{SAT}.

$$(\mathbf{All}\, p[\mathrm{PS}(p)] : \mathrm{Tr}(p)\, \mathbf{And}\, \mathrm{Fa}(p)) \leftarrow \mathrm{Sol}$$

Basically, this rules tells us to forget the assignment once we have derived that it is a model (i.e., we hide the witness of the NP problem). Now, the original SAT problem has a solution if and only if the structure interpreting symbols in Σ_{IN}^{SAT} according to a SAT problem and interpreting all other symbols as constant true is a model of the extended theory. Hence:

Theorem 5.9. *Model checking for* C-LOG *is NP-complete.*

Model checking might be a hard task but in certain cases (including for Δ_{SAT}) endogenous model expansion is not. The results in Theorem 5.6 can sometimes be used to generate models, if we have guarantees to end in a state where Δ succeeds.

Theorem 5.10. *If Δ is a total*[6] *causal theory without* $\mathbf{New}$ *and* $\mathbf{Select}$*-expressions, endogenous model expansion is in P.*

Note that Theorem 5.10 does not contradict Example 5.8 since in that example, Sol is interpreted as true in the input structure, i.e., the performed inference is not endogenous model expansion. It is future work to generalise Theorem 5.10, i.e., to research which are sufficient restrictions on Δ such that model expansion is in P.

[6] A causal theory is *total* if for every Δ-selection ζ, $w(A_\zeta)$ is two-valued, i.e., roughly, if it does not contain relevant loops over negation.

It is a well-known result in database theory that query languages combining recursion and object-creation are computationally complete [1]; C-LOG can be seen as such a language.

Theorem 5.11. *Unbounded querying can simulate the language* $\mathrm{while}_{\mathrm{new}}$ *from [1].*

Proof. We already showed that we can create the natural numbers in C-LOG. Once we have natural numbers and the successor function Succ, we add one extra argument to every symbol (this argument represents time). Now, we encode the looping construct from $\mathrm{while}_{\mathrm{new}}$ as follows. An expression of the form `while P do s` corresponds to the CEE: $\mathbf{All}\, t[P(t)] : C$, where C is the translation of the expression s. An expression `P = new Q` corresponds to a CEE (where the variable t should be bound by a surrounding while).

$$\mathbf{All}\, \overline{x}, t'[\mathrm{Succ}(t,t')] : \mathbf{New}\, y : P(\overline{x}, y, t') \leftarrow Q(\overline{x}, t). \qquad \Box$$

Now, it follows immediately from [1] that

Corollary 5.12. *For every decidable class $\mathcal{S}$ of finite structures closed under isomorphism, there exists a Δ such that unbounded exogenous model generation returns true with input I iff $I \in \mathcal{S}$.*

6 Conclusion

In this paper we presented several normal forms for FO(C). We showed that every FO(C) theory can be transformed to a Σ-equivalent deterministic FO(C) theory and to a Σ-equivalent FO(C) theory in NestNF or in DefF. Furthermore, as FO(C) theories in DefF correspond exactly to FO(ID), these transformations reduce inference for FO(C) to FO(ID). We implemented a prototype of this above transformation, resulting in the first FO(C) solver. We also gave several complexity results for inference in C-LOG. All of these results are valuable from a theoretical point of view, as they help to characterise FO(C), but also from a practical point of view, as they provide more insight in FO(C).

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[2] Bart Bogaerts. IDP-CLog. http://dtai.cs.kuleuven.be/krr/files/software/various/idp-clog.tar.gz, 2014.

[3] Bart Bogaerts, Joost Vennekens, Marc Denecker, and Jan Van den Bussche, 'FO(C): A knowledge representation language of causality', *TPLP*, (Online-Supplement, Technical Communication ICLP14), ((in press) 2014).

[4] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker, 'Predicate logic as a modelling language: The IDP system', *CoRR*, **abs/1401.6312**, (2014).

[5] Marc Denecker, 'The FO(·) knowledge base system project: An integration project (invited talk)', in *ASPOCP*, (2012).

[6] Marc Denecker, Maurice Bruynooghe, and Joost Vennekens, 'Approximation fixpoint theory and the semantics of logic and answers set programs', in *Correct Reasoning*, eds., Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, volume 7265 of *LNCS*, Springer, (2012).

[7] Marc Denecker and Eugenia Ternovska, 'A logic of nonmonotone inductive definitions', *ACM Trans. Comput. Log.*, **9**(2), 14:1–14:52, (April 2008).

[8] Joost Vennekens, Maarten Mariën, Johan Wittocx, and Marc Denecker, 'Predicate introduction for logics with a fixpoint semantics. Part I: Logic programming', *Fundamenta Informaticae*, **79**(1-2), 187–208, (September 2007).