Noname manuscript No.
(will be inserted by the editor)

FO(ID) as an extension of DL with rules

Joost Vennekens · Marc Denecker · Maurice Bruynooghe

the date of receipt and acceptance should be inserted later

Abstract There are many interesting Knowledge Representation questions surrounding rule languages for the Semantic Web. The most basic one is of course: which kind of rules should be used and how do they integrate with existing Description Logics? Similar questions have already been addressed in the field of Logic Programming, where one particular answer has been provided by the language of FO(ID). FO(ID) is an extension of first-order logic with a rule-based representation for inductive definitions. By offering a general integration of first-order logic and Logic Programs, it also induces a particular way of extending Description Logics with rules. The goal of this paper is to investigate this integration and discover whether there are interesting extensions of DL with rules that can be arrived at by imposing appropriate restrictions on the highly expressive FO(ID).

1 Introduction

Over the past decades, Description Logics (DL) have emerged as an important Knowledge Representation (KR) technology. More recently, they have also had a significant impact on industry, most notably with the adoption of OWL as a W3C standard. In current research, we find a trend to investigate extensions of OWL with *rules* (e.g. [14]), and, in fact, the hierarchical Semantic Web architecture already prescribes a *rule layer* on top of the *ontology layer* formed by OWL. There are a number of interesting KR questions surrounding this topic.

– *Which kind of rules are to be used?* There are numerous kinds of rules known in the literature (inference rules, rewrite rules, . . .), with subtle differences between them.
– *What precisely do this kind of rules mean?* It should be possible to explain exactly the information content of such a rule and, obviously, this explanation should be consistent with the formal semantics of the rules.

Joost Vennekens
Campus De Nayer, Hogeschool voor Wetenschap & Kunst, Sint Katelijne Waver, Belgium

Marc Denecker · Maurice Bruynooghe
Dept. of Computer Science, K.U. Leuven, Belgium

– *How do the rules complement DL?* We should be able to clearly indicate how the rules extend the class of knowledge that can be represented by the logic.

In this paper, we will present one particular answer to these questions, based on the language of FO(ID), a general integration of classical first-order logic (FO) and Logic Programming (LP). Conceptually, FO(ID) is an extension of FO with *inductive definitions*. To explain, let us consider the following example of an inductive definition, taken from the wikipedia page on the topic[1].

> *The prime numbers can be defined as consisting of:*
> – *2, the smallest prime;*
> – *each positive integer which is not evenly divisible by any of the primes smaller than itself.*

FO(ID) offers a formal syntax for representing such an inductive definition as a set of *definitional rules*:

$$\left\{\begin{array}{c} Prime(2) \leftarrow \\ \forall x\ Prime(x) \leftarrow x > 2 \land \neg \exists y\ y < x \land Prime(y) \land Divisible(x, y) \end{array}\right\}$$

In natural language, an inductive definition consists of a set of cases in which the defined relation, *Prime* in this case, holds. Each of these cases corresponds to a definitional rule in the formal syntax of FO(ID). We remark that the '$\leftarrow$'-symbol in the above expression therefore conveys more meaning than the normal material implication of FO: it not only states that certain numbers are prime, but also that no number can be prime *unless* its "primeness" can be constructively derived by applying the two rules. This additional meaning of such a rule is formalized in FO(ID) using the (parameterized) *well-founded model* construction from Logic Programming [23]. It was argued in [5] that this formal construction correctly captures the common-sense meaning of such a definition, as it is understood throughout mathematics.

The "job" of an inductive definition is, obviously, to define certain relation(s) (e.g., $Prime/1$) in terms of some other relation(s) (e.g., $< /2$ and $Divisible/2$). In this sense, it has the same semantic status as an FO formula: it expresses a relation between a number of predicates, which may or may not be satisfied by a given interpretation for these predicates. Based on this observation, FO(ID) integrates inductive definitions into FO: a formula of FO(ID) is either a regular FO formula or an inductive definition. In this way, FO is extended with a particular form of rules, namely, definitional rules, whose meaning is that each such rule represent a "case" of a particular inductive definition. This enlarges the expressive power of the logic: it is well-known that inductive definitions, such as that of transitive closure, cannot be expressed in FO.

The integration of FO and LP offered by FO(ID) has a number of properties that are quite appealing from the point of view of extending DL with rules:

– as opposed to the many hybrid approaches that exist, FO(ID) offers a strong semantic integration of FO and LP;
– in this integration, both components have a clear knowledge representation "task": the LP component is to be used to *define* concepts, whereas the FO component can be used to assert additional properties of both the defined concepts and concepts for which no definition is provided;

[1] `http://wikipedia.org/wiki/Inductive_definition`

– FO(ID) is particularly natural from a DL perspective: the ability to properly define concepts is already considered to be an important and characteristic feature of DL[2], and FO(ID) essentially just extends it to definitions that cannot be represented by a normal FO equivalence.

These properties make FO(ID) an interesting source of inspiration for extending DL with rules. In particular, because it contains full FO and allows general FO formulas in the bodies of its definitional rules, it is well-suited to serve as an upperbound, from which we can derive meaningful extensions of DL with rules, by imposing appropriate restrictions on it. We believe that, in general, this is better than the opposite approach of gradually extending a small tractable language into a more expressive one, since it allows the trade-off between expressivity and complexity to be made more consciously and informedly, which reduces the risk of creating an *ad hoc* language, whose boundaries are decided more by coincidence than design. In the rest of this paper, we will therefore follow this methodology, by defining two fragments of FO(ID) that are interesting DL rule languages.

Part of this paper has been presented at the European Semantic Web Conference (ESWC) 2009 [24].

2 Preliminaries: FO(ID)

This section summarizes the definition of FO(ID) as found in [5]. A *definitional rule* is an expression of the form:

$$\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \varphi, \tag{1}$$

with

– P/n a predicate (where the notation P/n means that the arity of P is n);
– $\mathbf{x}$ a tuple of n variables, not necessarily distinct;
– φ an FO formula.

Let us emphasize that expression (1) is just a particular syntax for writing down a definitional rule with these three components. In particular, the symbol '$\leftarrow$', which we call *definitional implication*, is not the material implication of FO (which we denote as '$\Leftarrow$'); instead, it is a new symbol, which is used exclusively to write down definitional rules. Also the universal quantifier in expression (1) is simply part of the notation used for definitional rules.

For a definitional rule r of form (1), we refer to $P(\mathbf{x})$ as the *head* of r, denoted $head(r)$, and to the formula φ as its *body*, denoted $body(r)$.

Syntactically, a *definition* in FO(ID) is a finite set of definitional rules, enclosed by curly braces $\{\}$. An *FO(ID) formula* is any expression that can be formed by combining atoms and definitions, using the standard FO connectives and quantifiers. So, the syntactical status of a definition in FO(ID) is the same as that of a regular FO atom: it is itself a formula, and more complex formulas can be built from it. A definitional rule in FO(ID), on the other hand, has the same semantical status as a term in FO: even though it may already be a complex expression, it is itself not yet a formula.

[2] For instance, the DL handbook [1] has the following to say about DL's '$\equiv$'-connective: "This form of definition is much stronger than the ones used in other kinds of representations of knowledge, which typically impose only necessary conditions; the strength of this kind of declaration is usually considered a characteristic feature of DL knowledge bases."

A definitional rule may only appear as part of a definition (i.e., inside curly braces, together with zero or more other definitional rules), whereas a definition as a whole may appear wherever an FO atom is allowed, as long as it not inside another definition.

The meaning of the FO connectives is standard; e.g., a disjunction $\varDelta_1 \vee \varDelta_2$ of two definitions $\varDelta_1$ and $\varDelta_2$ holds if and only if $\varDelta_1$ holds or $\varDelta_2$ holds. We therefore only need to define the semantics of a definition (i.e., what does it mean, precisely, to say that a definition $\varDelta$ "holds"?) in order to define the semantics of FO(ID) in full.

Let us first recall some well-known semantical concepts. An *interpretation* $\mathcal{S}$ for a vocabulary $\varSigma$ consists of a non-empty domain D, a mapping from each function symbol f/n to an n-ary function $f^{\mathcal{S}}$ on D, and a mapping from each predicate symbol P/n to a relation $P^{\mathcal{S}} \subseteq D^n$. A *three-valued* interpretation ν is the same as a two-valued one, except that it maps each predicate symbol P/n to a function P^{ν} from D^n to the set of truth values $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. Such a ν assigns a truth value to each logical atom $P(\mathbf{c})$, namely $P^{\nu}(c_1^{\nu}, \ldots, c_n^{\nu})$. This assignment can be extended to an assignment $\nu(\varphi)$ of a truth value to each formula φ, using the standard Kleene truth tables for the logical connectives:

φ, ψ	$\mathbf{t}, \mathbf{t}$	$\mathbf{t}, \mathbf{f}$	$\mathbf{t}, \mathbf{u}$	$\mathbf{u}, \mathbf{f}$	$\mathbf{u}, \mathbf{u}$	$\mathbf{f}, \mathbf{f}$
$\varphi \vee \psi$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{u}$	$\mathbf{u}$	$\mathbf{f}$

φ	$\mathbf{t}$	$\mathbf{u}$	$\mathbf{f}$
$\neg\varphi$	$\mathbf{f}$	$\mathbf{u}$	$\mathbf{t}$

and so on. An existential quantification is of course similar to a disjunction, in the sense that $\exists x\ \varphi(x)$ is $\mathbf{t}$ if for some tuple $d \in D$, $\varphi(d)$ is $\mathbf{t}$, and otherwise it is $\mathbf{u}$ as soon as some $\varphi(d)$ is $\mathbf{u}$; therefore, $\exists x\ \varphi(x)$ is $\mathbf{f}$ only if all $\varphi(d)$ are $\mathbf{f}$.

The three truth values can be partially ordered according to *precision*:

$$\mathbf{u} \leq_p \mathbf{t} \text{ and } \mathbf{u} \leq_p \mathbf{f}.$$

This order induces a precision order $\leq_p$ on interpretations: $\nu \leq_p \nu'$ if for each predicate P/n and tuple $\mathbf{d} \in D^n$, $P^{\nu}(\mathbf{d}) \leq_p P^{\nu'}(\mathbf{d})$. For a predicate P/n, a tuple $\mathbf{d} \in D^n$ and a truth value $\mathbf{v} \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, we denote by $\nu[P(\mathbf{d})/\mathbf{v}]$ the three-valued interpretation ν' that coincides with ν on all symbols apart from P/n, and for which $P^{\nu'}$ maps $\mathbf{d}$ to $\mathbf{v}$ and all other tuples $\mathbf{d}'$ to $P^{\nu}(\mathbf{d}')$. We also extend this notation to $\nu[U/\mathbf{v}]$ with U a set of such pairs of predicates P_i and tuples $\mathbf{d}_i$.

Our goal is to define when a (two-valued) interpretation $\mathcal{S}$ is a model of a definition $\varDelta$. We call the predicates that appear in the head of a rule of $\varDelta$ its *defined predicates* and we denote the set of all these by $Def(\varDelta)$; all other symbols are called *open* and the set of open symbols is written $Op(\varDelta)$. The purpose of $\varDelta$ is now to define the predicates $Def(\varDelta)$ in terms of the symbols $Op(\varDelta)$, i.e., we should assume the interpretation of $Op(\varDelta)$ as given and try to construct a corresponding interpretation for $Def(\varDelta)$. Let O be the restriction $\mathcal{S}|_{Op(\varDelta)}$ of $\mathcal{S}$ to the open symbols. We are now going to construct sequences of three-valued interpretations $(\nu_{\alpha}^{O})_{0 \leq \alpha}$, each of which extends O; we will use the limit of these sequences to interpret $Def(\varDelta)$.

- ν_0^O assigns $P^O : D^n \to \{\mathbf{t}, \mathbf{f}\}$ to each $P \in Op(\varDelta)$ and the constant function that maps each tuple $\mathbf{d}$ to $\mathbf{u}$ to each $P \in Def(\varDelta)$;
- ν_{i+1}^O is related to ν_i^O in one of two ways:
  - either $\nu_{i+1}^O = \nu_i^O[P(\mathbf{d})/\mathbf{t}]$, such that $\varDelta$ contains a rule $\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \varphi(\mathbf{x})$ with $\nu_i^O(\varphi(\mathbf{d})) = \mathbf{t}$;
  - or $\nu_{i+1}^O = \nu_i^O[U/\mathbf{f}]$, where U is any *unfounded set*, meaning that it consists of pairs of predicates P/n and tuples $\mathbf{d} \in D^n$ for which $P^{\nu_i^O}(\mathbf{d}) = \mathbf{u}$, and for each rule $\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \varphi(\mathbf{x})$, we have that $\nu_{i+1}^O(\varphi(\mathbf{d})) = \mathbf{f}$.

- For each limit ordinal λ, ν^O_λ is the least upper bound w.r.t. $\leq_p$ of all ν^O_δ for which $\delta < \lambda$.

The intuition behind the concept of an unfounded set U is that we can safely conclude that all $P(\mathbf{d}) \in U$ are false, because doing so will falsify all conditions under which we would be able to derive that one of them is true. We call such a sequence $(\nu^O_\alpha)_{0\leq\alpha}$ a *well-founded induction* of Δ in O. Each such sequence eventually reaches a limit ν^O_β. It was shown in [5] that all sequences reach the same limit, and that this limit is precisely the well-founded model of Δ given O [23]. It is now this ν^O_β that tell us how to interpret the defined predicates. To be more precise, we define that, for each two-valued interpretation $\mathcal{S}$:

$$\mathcal{S} \models \Delta \text{ iff } \mathcal{S}|_{Def(\Delta)} = \nu^O_\beta|_{Def(\Delta)}, \text{ with } O = \mathcal{S}|_{Op(\Delta)}.$$

This tells us when a definition Δ is satisfied in a structure $\mathcal{S}$. The semantics of full FO(ID), in which definitions and atoms can be combined using the standard FO connectives, is now simply that which is obtained by adding the usual recursive cases for the connectives (e.g., $\mathcal{S} \models \varphi \wedge \psi$ if $\mathcal{S} \models \varphi$ and $\mathcal{S} \models \psi$, and so). Note that if there is some predicate P/n and $\mathbf{d} \in D^n$ such that $P^{\nu_\beta}(\mathbf{d})$ is still $\mathbf{u}$, then the definition has no models extending O. Intuitively, this means that, for this particular interpretation of its open symbols, Δ does not manage to unambiguously define the predicates $Def(\Delta)$, due to some non well-founded use of negation.

In the rest of this paper, we will only consider relational vocabularies, that is, there will be no function symbols of arity > 0.

3 From FO(ID) to DL(ID)

As a language that extends full FO, FO(ID) is too expressive to serve as a DL itself. In this section, we discuss the problem of restricting FO(ID) to a smaller, more DL-like language. In doing this, it is important to be clear about the goals we are trying to achieve. DLs themselves are fragments of FO that are interesting for essentially two reasons:

- they are decidable;
- they are tailored towards a concept-centric modeling style, which they support by means of an intuitive syntax sugar that hides away many of the complexities of FO.

The first of these properties is about computation, whereas the second concerns knowledge representation. To a certain extent, they are therefore orthogonal concerns.

Most of our attention in this paper will go to the knowledge representation issue, simply because we feel that it *comes first*. If a logic is to be used at all, for whatever purpose, then theories will have to be written in it. If this is difficult, for instance because the language is too complex or because the meaning of its statements is unclear, then it is unlikely the language will enjoy enduring success, no matter how efficient certain inference task for it might be. Therefore, a primary goal of designing a rule language for DL should be to ensure that it provides a clean and coherent integration of rules into DL, which is easy to understand for those with a working knowledge of DL.

This does not mean that decidability is unimportant, but in the end it is of secondary nature, since also undecidable languages can have interesting uses. Indeed, decidability is, by definition, only relevant if the goal is to perform *deductive inference* (i.e., to decide whether, for formulas φ and ψ, it is the case that $\varphi \models \psi$, in the sense that each model of φ is also a model of ψ) in an *unknown* or *infinite* domain. Or, to put things more formally:

– given an FO theory T and an FO formula φ, it is undecidable whether for all structures S, it holds that $S \models T$ implies $S \models \varphi$;
– however, given *a finite set* D, an FO theory T and an FO formula φ, it is decidable (with data complexity in co-NP and combined complexity in PSPACE) whether for all structures S *such that the domain of* S *is* D, it holds that $S \models T$ implies $S \models \varphi$;

When we go from FO to FO(ID), both of these theorems still hold. This is important, because the second situation arises in numerous interesting applications. For instance, when reasoning in the context of a particular database, we only want to consider the domain D consisting of all the objects in the database, as DB querying systems indeed typically do. Similarly, if the goal is to query some data from Semantic Web sites, we often only want to consider those objects about which we are able to retrieve information; for instance, if we are looking for the cheapest flight from Athens to Rome, it does us little good to be told that an unknown carrier might be offering a flight for \$1 that is not advertised anywhere. A second example is that for many combinatorial problems, the domain is given as part of the specific problem instance that is to be solved; e.g., in graph colouring, D would be the set of nodes, which is fixed once we get a specific graph to colour. Moreover, even when the domain is not known, there might still be approximate algorithms, that are able to derive useful information without guarantees of completeness.

We will therefore now first present a fragment of FO(ID) that allows a DL-like syntactic sugar and offers a DL-like modeling methodology. Afterwards, we will turn to the topic of inference.

4 A fragment of FO(ID): $\mathcal{ALCI}$(ID)

The goal of this section is to present a fragment of FO(ID), that supports a DL-like modeling style. This will be an extension of the Description Logic $\mathcal{ALCI}$. More expressive logics, such as $\mathcal{SHOIN}$(D) which underlies OWL-DL, can be extended in the same way; we restrict to $\mathcal{ALCI}$ merely for simplicity.

Let us first briefly recall $\mathcal{ALCI}$. We start from a set of unary predicates $\{A_1, A_2, \ldots\}$ called (atomic) *concepts* and a set of binary predicates $\{B_1, B_2, \ldots\}$ called (atomic) *roles*. From the atomic concepts, we build more complex ones using the connectives $\sqcup$, $\sqcap$, $\neg$, $\exists$ and $\forall$; their meaning can be inductively defined by the mapping to FO given in Fig. 1. A role is either an atomic role B or its inverse B^-, as in Fig. 2. We will use the naming convention that, for a role R, $R(x, y)$ is to be read as "x is an R of y". This is reflected in the cases for $\exists$ and $\forall$ in Fig. 1, which express that x belongs to the concept $\exists R.C$ or $\forall R.C$, respectively, if *there exists a* y *that is an* R *of* x *such that* y *belongs to* C or if *for each* y *that is an* R *of* x, *it holds that* y *belongs to* C. We will write $\bot$ and $\top$ to denote the empty and universal concept, respectively.

A *TBox* then consists of a set of inclusions and equivalences as in Fig. 3.

C	$\langle C\rangle(x)$	
A	$A(x)$	with A an atomic concept
$C_1 \sqcap C_2$	$\langle C_1\rangle(x) \wedge \langle C_2\rangle(x)$	with C_1, C_2 concepts
$C_1 \sqcup C_2$	$\langle C_1\rangle(x) \vee \langle C_2\rangle(x)$	with C_1, C_2 concepts
$\neg C$	$\neg\langle C\rangle(x)$	with C a concept
$\forall R.C$	$\forall y\ \langle R\rangle(y,x) \Rightarrow \langle C\rangle(y)$	with R a role and C a concept
$\exists R.C$	$\exists y\ \langle R\rangle(y,x) \wedge \langle C\rangle(y)$	with R a role and C a concept

Fig. 1 A concept C of $\mathcal{ALCI}$ represents a unary formula $\langle C\rangle(x)$ of FO.

R	$\langle R\rangle(x,y)$	
B	$B(x,y)$	with B an atomic role
B^-	$B(y,x)$	with B an atomic role

Fig. 2 A role R of $\mathcal{ALCI}$ represents a binary formula $\langle R\rangle(x,y)$ of FO.

φ	$\langle\varphi\rangle$	
$C_1 \sqsubseteq C_2$	$\forall x\ \langle C_1\rangle(x) \Rightarrow \langle C_2\rangle(x)$	with C_1, C_2 concepts
$C_1 \equiv C_2$	$\forall x\ \langle C_1\rangle(x) \Leftrightarrow \langle C_2\rangle(x)$	with C_1, C_2 concepts

Fig. 3 A statement φ in a TBox correspond to an FO sentence $\langle\varphi\rangle$.

R	$\langle R\rangle(x,y)$	
$R_1 \sqcap R_2$	$\langle R_1\rangle(x,y) \wedge \langle R_2\rangle(x,y)$	with R_1, R_2 roles
$R_1 \sqcup R_2$	$\langle R_1\rangle(x,y) \vee \langle R_2\rangle(x,y)$	with R_1, R_2 roles
$\neg R$	$\neg\langle R\rangle(x,y)$	with R a role
$R_1.R_2$	$\exists z\ \langle R_1\rangle(x,z) \wedge \langle R_2\rangle(z,y)$	with R_1, R_2 roles
$C_1 \times C_2$	$C_1(x) \wedge C_2(y)$	with C_1, C_2 concepts

Fig. 4 Additional ways of constructing roles.

We now form the language $\mathcal{ALCI}$(ID) by extending $\mathcal{ALCI}$. First, we add the connectives in Fig. 4 for constructing more complex roles: the first three are the obvious analogues of the connectives for concepts, the fourth takes the join of two roles, and the last one takes the Cartesian product of two concepts. Second, we add two new connectives for representing inductive definitions. The first is the symbol '$\doteq$', which is conceptually the same as '$\equiv$', with the difference that it also works for inductive definitions. Formally, it is defined as an abbreviation for an FO(ID) definition containing a single rule, as shown in the first two entries of Fig. 5.

For instance, we can define the concept of an uncle as the brother of a parent:

$$Uncle \doteq Brother.Parent$$

This abbreviates the FO(ID) definition:

$$\{\forall x, y\ Uncle(x,y) \leftarrow \exists z\ Brother(x,z) \wedge Parent(z,y)\}$$

Such a non-inductive definition (i.e., the role/concept on the left-hand side does not appear in the right-hand side) is equivalent to a regular FO equivalence:

$$\forall x, y\ Uncle(x,y) \Leftrightarrow \exists z\ Brother(x,z) \wedge Parent(z,y).$$

However, '$\doteq$' can also correctly represent inductive definitions. For instance, we can define the role *Ancestor* as the transitive closure of *Parent*, by saying that an *Ancestor* is either a *Parent* or the *Parent* of an *Ancestor*:

$$Ancestor \doteq Parent \sqcup Parent.Ancestor$$

φ	$\langle\varphi\rangle$	
$R_1 \doteq R_2$	$\{\forall x, y\ \langle R_1\rangle(x,y) \leftarrow \langle R_2\rangle(x,y)\}$	with R_1, R_2 roles
$C_1 \doteq C_2$	$\{\forall x\ \langle C_1\rangle(x) \leftarrow \langle C_2\rangle(x)\}$	with C_1, C_2 concepts
$\{\varphi_1, \ldots, \varphi_n\}$	$\{\langle\varphi_1\rangle, \ldots, \langle\varphi_n\rangle\}$	with $\varphi_1, \ldots, \varphi_n$ definitional rules (Fig.6)

Fig. 5 A definition φ in $\mathcal{ALCI}$(ID) corresponds to a definition $\langle\varphi\rangle$ in FO(ID).

φ	$\langle\varphi\rangle$	
$R_1 \leftarrow R_2$	$\forall x, y\ \langle R_1\rangle(x,y) \leftarrow \langle R_2\rangle(x,y)$	with R_1, R_2 roles
$C_1 \leftarrow C_2$	$\forall x\ \langle C_1\rangle(x) \leftarrow \langle C_2\rangle(x)$	with C_1, C_2 concepts

Fig. 6 A definitional rule φ in $\mathcal{ALCI}$(ID) corresponds to a definitional rule $\langle\varphi\rangle$ in FO(ID).

Because of the translation to FO(ID), this too has the correct semantics. Therefore, '$\doteq$' can be used instead of a transitive closure construct such as $\cdot^+$ or the reflexive-transitive closure $\cdot^*$ of e.g. [16]; it can also replace non-nested uses of the explicit least fixpoint operator μ.

Whereas $\doteq$ defines a concept by a single rule, it is convenient to also be able to define a concept by a set of rules. For this purpose we introduce a new connective $\leftarrow$ to represent a definitional rule and allow sets of such rules, again enclosed in curly braces, as statements in our language; this is shown in Fig. 5 and 6. For instance, extending the notion of an uncle to also include uncles-by-marriage, we could then define this concept by saying that an uncle is

– either a brother of a parent;
– or a husband of an aunt.

We can write this down in $\mathcal{ALCI}$(ID) as:

$$\left\{\begin{array}{l} \mathit{Uncle} \leftarrow \mathit{Brother}.\mathit{Parent} \\ \mathit{Uncle} \leftarrow \mathit{Husband}.\mathit{Aunt} \end{array}\right\}$$

This abbreviates the following definition in FO(ID):

$$\left\{\begin{array}{l} \forall x, y\ \mathit{Uncle}(x,y) \leftarrow \exists z\ \mathit{Brother}(x,z) \wedge \mathit{Parent}(z,y) \\ \forall x, y\ \mathit{Uncle}(x,y) \leftarrow \exists z\ \mathit{Husband}(x,z) \wedge \mathit{Aunt}(z,y) \end{array}\right\}$$

In general, if a definition contains multiple rules with the same predicate in the head, these can always be replaced by a single rule whose body is the disjunction of the bodies of the original rules. Therefore, we could rephrase the above definition of *Uncle* as

$$\mathit{Uncle} \doteq \mathit{Brother}.\mathit{Parent} \sqcup \mathit{Husband}.\mathit{Aunt}$$

However, this loses the natural case-based structure of the definition, i.e., it makes it more difficult to see that there are actually two separate sufficient conditions, which together also form a necessary condition. Moreover, the rule-based format is also more elaboration tolerant, since rules can more easily be added or removed. For instance, a bank might define the class of persons eligible for a loan as consisting of people with a large income, people who own a house and people with a good credit history; each time the bank now tightens or relaxes its policy, certain rules would have to be removed or added to this definition.

The rule-based representation is also more general than '$\doteq$', since it allows definitions by *simultaneous* induction as well. For instance, given a two-player game whose

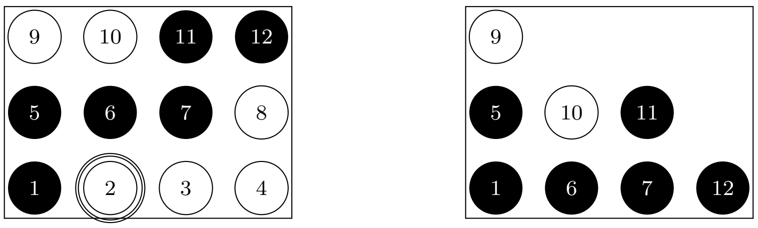


Fig. 7 The result of selecting ball 2.

move tree is described by the role *Parent*, we can define the nodes in which I move and the nodes in which my opponent moves by the following simultaneous induction (assuming I start):

$$\left\{\begin{aligned} HisMove \leftarrow & \exists Parent.MyMove \\ MyMove \leftarrow & \exists Parent.HisMove \\ MyMove \leftarrow & \forall Parent.\bot \end{aligned}\right\}$$

5 An Example

In this section, we will illustrate the new features offered by $\mathcal{ALCI}$(ID). Our example concerns a simple game, in which the player is presented a grid of coloured balls. He makes a move by selecting one of these balls. The effect of this is that the entire colour-group to which the ball belongs disappears; the remaining balls then fall down, yielding the next position of the game, as depicted in Fig. 7. The goal of the game is to remove all balls from the grid in such a way as to score as many points as possible. What we will do here is construct an $\mathcal{ALCI}$(ID) theory that describes the effect of a single move of this game.

To make things more concrete, let us fix a representation for a state of the game. We represent the grid by roles *Up* and *Left*, both with the obvious meaning. The starting grid in Fig. 7, for instance, would correspond to the following interpretations:

$$\begin{aligned} Up = & \{(5,1),(9,5),(6,2),(10,6),(7,3), \\ & (11,7),(8,4),(12,8)\}; \\ Left = & \{(1,2),(2,3),(3,4),(5,6),(6,7), \\ & (7,8),(9,10),(10,11),(11,12)\}. \end{aligned}$$

We represent the player's move by a concept *Chosen*; the move made in Fig. 7 would correspond to $Chosen = \{2\}$. Our goal is now to define roles Up' and $Left'$, representing the next state of the game.

We first define some useful auxiliary roles and concepts. We begin by defining the role *Above* as the transitive closure of *Up*. One ball is above another if it is either directly on top of it, or on top of a ball that is already above it:

$$Above \doteq Up \sqcup Up.Above$$

A ball is next to another ball —the role *NextTo*— if it is either to the left, to the right, underneath, or on top of it:

$$NextTo \doteq Left \sqcup Right \sqcup Up \sqcup Down$$

Of course, the roles *Right* and *Down* are the inverses of, respectively $Left$ and Up:

$$\mathit{Right} \doteq Left^{-} \qquad \mathit{Down} \doteq Up^{-}$$

We remark that of these four definitions, *Above* is defined inductively, while the others are not. However, in $\mathcal{ALCI}$(ID) this difference hardly matters: all four definitions have the same "look and feel".

The concept *Disappears* describes the balls that disappear after the move. These are the chosen ball itself and all balls belonging to the same colour-group:

$$\mathit{Disappears} \doteq \mathit{Chosen} \sqcup \exists \mathit{InColourGroup}.\mathit{Chosen}$$

The role *InColourGroup* expresses that balls are in the same colour group, i.e., they are connected through a sequence of balls of the same colour:

$$\left\{ \begin{array}{l} \mathit{InColourGroup} \leftarrow \mathit{SameColour} \sqcap \mathit{NextTo} \\ \mathit{InColourGroup} \leftarrow \mathit{InColourGroup} \sqcap \mathit{NextTo} \end{array} \right\}$$

Here, we use the $\leftarrow$ connective to separate the base case and the inductive step of this definition; the same could be done with our earlier definition of *Above*. Both forms are equivalent, so it is up to the modeler to decide whether he prefers to use multiple rules or a single rule with a disjunction in the body. The obvious guideline is to consider how you would write the definition in a natural language text: if you would be inclined to use a bulleted list, then use multiple rules; if you would write it as running text instead (using an "either ... or ..."), use a single rule.

Two balls have the same colour if the colour of one is also that of the other:

$$\mathit{SameColour} \doteq \mathit{HasColour}.\mathit{HasColour}^{-}$$

Having now defined which balls disappear, we define in the concept *Remains* the remaining balls as the complement thereof:

$$\mathit{Remains} \doteq \neg \mathit{Disappears}$$

We now define the role *Above*$'$, i.e., the "above"-relation as it will be in the next state. This will hold for any two remaining balls that were originally above each other:

$$\mathit{Above}' \doteq \mathit{Above} \sqcap (\mathit{Remains} \times \mathit{Remains})$$

We can now define the role *Up*$'$ as the intransitive relation of which *Above*$'$ is the transitive closure:

$$\mathit{Up}' \doteq \mathit{Above}' \sqcap \neg(\mathit{Above}'.\mathit{Above}')$$

We define an auxiliary concept *OnGround*$'$ as consisting of those balls that will form the bottom row in the new situation:

$$\mathit{OnGround}' \doteq \mathit{Remains} \sqcap \neg\exists \mathit{Above}^{-}.\mathit{Remains}$$

All that remains now is to define the role $Left'$. Let us first define the role $InLeftColumn$ which describes when a ball x is in the column to the left of some other ball y. Regardless of which of the two columns is higher, the following three cases cover all such situations:

– x is to the left y;

– x is to the left of a ball $z \neq y$ that is in the same column as y;
– x is in the same column as a ball $z \neq y$ that is to the left of y.

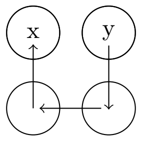
Even though this definition is not inductive, it is still a nice fit with the case-based structure of definitions in $\mathcal{ALCI}$(ID).

$$\left\{\begin{array}{l} InLeftColumn \leftarrow Left \\ InLeftColumn \leftarrow Left.(Above \sqcup Above^{-}) \\ InLeftColumn \leftarrow (Above \sqcup Above^{-}).Left \end{array}\right\}$$

We now define $Left'$ as consisting of all pairs of balls such that (1) one ball is in the column to the left of the other, and (2) both are on the same height:

$$Left' \doteq \underbrace{InLeftColumn}_{(1)} \sqcap \underbrace{((OnGround' \times OnGround') \sqcup Up'.(Left'.Up'^{-}))}_{(2)}$$

Note that this too is an inductive definition, which proceeds along the rows of the grid: the base case is the bottom row, whereas the inductive case defines that x is to the left of y in the following situation:



This concludes our representation of the game. As this example shows, $\mathcal{ALCI}$(ID) can represent many different kinds of definitions (case-based definitions, non-inductive definitions, monotone inductive definitions, non-monotone inductive definitions) in a uniform way in, leading to a clean and coherent language.

6 Inference in $\mathcal{ALCI}$(ID)

In this section, we discuss some issues related to inference in $\mathcal{ALCI}$(ID). Our first result is that this language is, in general, undecidable.

Theorem 1 *$\mathcal{ALCI}$(ID) is undecidable. It is also not even semi-decidable.*

Proof We will prove this result by showing that $\mathcal{ALCI}$(ID) can express *tiling problems*: a tiling problem consists of a set of tiles D and relations $H, V \subseteq D \times D$ describing which tiles match horizontally and vertically, respectively. It is well-known that it is not even semi-decidable whether the first quadrant of the plane can be tiled by a given set of tiles (i.e., whether it is possible to assign to each point $(i, j) \in \mathbb{N}^2$ one of the tiles in D such that all horizontally/vertically adjacent tiles match, as dictated by the sets H and V, respectively) [4]. Therefore, it suffices to show that for each tiling problem (D, H, V), we can construct an $\mathcal{ALCI}$(ID) theory T, such that T is satisfiable if and only if (D, H, V) can indeed tile this quadrant. We construct T as follows.

For each tile $i \in D$, we have a role $Tile_i$, which is intended to hold for all (x, y) such that square (x, y) is tiled by i. We use the role $Succ$ to represent the successor relation of the natural numbers. The theory then is:

$$\top \sqsubseteq \exists Succ.\top \qquad \text{(each number has a successor)}$$

$$\left\{\begin{array}{l} Greater \leftarrow Succ \\ Greater \leftarrow Succ.Greater \end{array}\right\} \qquad \text{(transitive closure of } Succ\text{)}$$

$$\bot \times \bot \sqsupseteq Greater^{-} \sqcap Greater \qquad (Greater \text{ is anti-symmetric})$$

$$\top \times \top \sqsubseteq \sqcup_i Tile_i \qquad \text{(the tiles cover all positions)}$$

$$Tile_i \sqcap Tile_j \sqsubseteq \bot \times \bot, \text{ for all } i \neq j \qquad \text{(at most one tile per point)}$$

$$Tile_i.Succ \sqsubseteq \sqcup_{(i,j)\in V} Tile_j \qquad \text{(tiles match vertically)}$$

$$Tile_i^{-}.Succ \sqsubseteq \sqcup_{(i,j)\in H} Tile_j^{-} \qquad \text{(tiles match horizontally)}$$

However, the undecidability of $\mathcal{ALCI}$(ID) does not imply that the logic is unsuited for all computational applications. In particular, we want to call attention to the following interesting inference task of model expansion.

6.1 Model expansion

[17] considered the inference task of *model expansion* for FO(ID): given an interpretation $\mathcal{S}$ for some subset Σ_0 of the alphabet Σ of a theory T, extend $\mathcal{S}$ with an interpretation for the remaining symbols $\Sigma \setminus \Sigma_0$, such that the resulting interpretation is a model of T. Here, $\mathcal{S}$ is required to have a finite domain. Note that, in general, a model expansion problem may have many different solutions, because even though $\mathcal{S}$ has to fix the domain of discourse, it does not fix the interpretation of $\Sigma \setminus \Sigma_0$ in this domain, but only restrains it to satisfy T. This form of inference is decidable for full FO(ID) and in fact captures the complexity class NP.

To illustrate the usefulness of this inference task, let us consider again the example of the previous section. Here, we defined the next state of a game ($Left'$ and Up') in terms of its old state ($Left$ and Up) and a given move ($Chosen$). We can therefore compute a new state of the game by performing model expansion on the structure $\mathcal{S}$ for the alphabet $\Sigma_0 = \{Left, Up, Chosen\}$. Because our representation of the game defines all of its predicates except the ones in Σ_0, this computation can actually be done in polynomial time.

The IDP-system[3] implements the task of model expansion for FO(ID). A program transforming $\mathcal{ALCI}(ID)$ syntax to input for this system is available.[4] Together, these two programs are able to use our formalization of Section 5 to compute state transitions for this game. It is not hard to extend this to a method of determining whether a given game has a solution (i.e., a sequence of moves that removes all balls). However, this is only possible because there is an upperbound on the number of moves that can be made—since every move removes at least one ball, the possible length of a game is limited by the number of balls. For games that might last infinitely long, such as the fifteen puzzle, model expansion cannot determine whether a solution exists (at least, not unless clever tricks are used); it can only say whether, for some fixed n, a solution exists in n steps or less. In this respect, the model expansion system is similar to a *lightweight verification* system such as Alloy.[5] To perform inference in

[3] `http://www.cs.kuleuven.be/~dtai/krr/software/idp.html`
[4] `http://www.cs.kuleuven.be/~joost/alc_id.tar.gz`
[5] `http://alloy.mit.edu/`

unbounded domains, as a heavyweight verification method would do, more general forms of deduction are needed. The next section presents a fragment of $\mathcal{ALCI}$(ID) in which general deduction is decidable.

6.2 Guarded $\mathcal{ALCI}$(ID)

As discussed, there are interesting problems that can be solved by reasoning in a fixed, finite domain, and for which the undecidability of FO(ID) is therefore not a problem. However, since we would not want to claim that this covers all potential applications, this section will develop a decidable fragment of $\mathcal{ALCI}$(ID).

This fragment will be based on the *guarded fragment* of FO. We recall that an FO formula ψ is *guarded* if every one of its quantified subformulas is either of the form $\exists \mathbf{x}\ G(\mathbf{x},\mathbf{y}) \wedge \varphi(\mathbf{x},\mathbf{y})$ or $\forall \mathbf{x}\ G(\mathbf{x},\mathbf{y}) \Rightarrow \varphi(\mathbf{x},\mathbf{y})$, where $G(\mathbf{x},\mathbf{y})$ is an atom, called the *guard*, such that the free variables $free(\varphi(\mathbf{x},\mathbf{y}))$ are a subset of $free(G(\mathbf{x},\mathbf{y}))$. There also exists a *loosely* guarded fragment, which allows the guards $G(\mathbf{x},\mathbf{y})$ to be conjunctions of atoms, rather than simply atoms, on condition that each quantified variable $\mathbf{x}$ appears together with every other variable of $\mathbf{x} \cup \mathbf{y}$ in at least one of the atoms.

In [10], the loosely guarded fragment of FO was extended to FO(LFP). We recall that FO(LFP) extends FO with a least-fixpoint construct that can be used to represent monotone induction. This works as follows.

In general, suppose that $\varphi(\mathbf{x})$ is a formula with n free variables $\mathbf{x}$. Together with an interpretation I for its vocabulary, such a formula defines a set $Sat(\varphi, I)$ of tuples from $dom(I)$ that satisfy it:

$$Sat(\varphi, I) = \{\mathbf{d} \in dom(I)^n \mid I \models \varphi(\mathbf{d})\}.$$

Now, let us suppose that φ contains some predicate P of arity precisely n, which is not interpreted by I. By supplementing I with an interpretation J for the predicate P, we can obtain the set of tuples $Sat(\varphi, I \cup J)$. This set of tuples can now serve as a new interpretation J' for the predicate P. That is, φ and I define the following operator $F^I_{\varphi,P}$ on interpretations J of the predicate P:

$$F^I_{\varphi,P}(J) = Sat(\varphi, I \cup J).$$

If the predicate P appears only positively in the formula φ, this operator is monotone and, therefore, has a least fixpoint.

The logic FO(LFP) is now obtained by extending FO with the following kind of *fixpoint expression*:

$$[\mathbf{lfp}_P(\varphi)](\mathbf{t}). \tag{2}$$

Here, P is a predicate of arity n, φ is a formula with n free variables, and $\mathbf{t}$ is a tuple of n terms; the predicate P is only allowed to appear positively in the formula φ. The semantics of such an expression is that it is satisfied by an interpretation I and variable assignment σ if and only if $\mathbf{t}^{I,\sigma}$ belongs to the least fixpoint of the operator $F^I_{\varphi,P}$.

For instance, FO(LFP) allows us to express that a pair (A, B) belongs to the transitive closure of a relation R as:

$$[\mathbf{lfp}_{T(x,y)} R(x,y) \vee \exists z\ T(x,z) \wedge T(z,y)](A,B)$$

To better bring out the similarity to FO(ID), we will also write a fixpoint expression of form (2) in the following syntax:

$$[\mathbf{lfp}\{\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \varphi\}](\mathbf{t}).$$

Grädel et al. define the following (loosely) guarded fragment of FO(LFP).

Definition 1 (paraphrased from [10]) The (loosely) guarded fragment of FO(LFP) is defined by extending the (loosely) guarded fragment of FO with the following rule for constructing (loosely) guarded fixpoint formulas: a fixpoint formula $[\mathbf{lfp}_{P(\mathbf{x})}\varphi(\mathbf{x})](\mathbf{t})$ is (loosely) guarded if and only if φ is a (loosely) guarded formula such that P does not appear in any of its guards.

They then go on to prove the following result:

Theorem 2 ([10]) *The satisfiability problem for (loosely) guarded fixpoint logic is 2*EXPTIME*-complete.*

We now define a similar (loosely) guarded fragment of FO(ID). First, we define the concept of a (loosely) guarded definition, as an analogue of a (loosely) guarded fixpoint formula. We will restrict attention to definitions that are *total*, i.e., for which the limit of each induction sequence is always two-valued (regardless of the interpretation of the open predicates). There is an easy syntactic criterion that suffices to ensure totality. Let us say that a predicate P *depends on* a predicate Q in definition Δ if there exists a sequence of predicates $P_1, \ldots, P_n$ with $P = P_1$ and $P_n = Q$, such that for each i, Δ contains a rule with P_i in the head and P_{i+1} in the body; we say that P depends *negatively* on Q if for at least one i, P_{i+1} appears negatively in the body of the rule for P_i. As long as there are now no two predicates P and Q (where it is allowed that $P = Q$) that depend negatively on each other, the definition Δ is guaranteed to be total. In practice, it is quite rare to encounter definitions that do not satisfy this criterion.

Definition 2 A definition Δ of FO(ID) is *(loosely) guarded* if:

- for each of its rules $\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \psi$, it holds that ψ is a (loosely) guarded formula;
- none of the defined predicates $Def(\Delta)$ are used as guards;
- no defined predicates depend negatively on each other (as defined above).

Our method for proving the decidability of (loosely) guarded FO(ID) will be to present a translation from FO(ID) to FO(LFP) and then show that it maps the (loosely) guarded fragment of the former into the (loosely) guarded fragment of the latter. Because it will ease notation, we will actually present our transformation using *simultaneous* least-fixpoint logic FO(SLFP). As shown in, e.g., [7], FO(SLFP) can be reduced to regular FO(LFP), by increasing the arities of the fixpoint predicate(s). Again, for consistency with FO(ID) notation, we will use a rule-based form to write such simultaneous fixpoint formulas, as is also done in, e.g., [15]. To be more concrete, a simultaneous least fixpoint expression is of the form:

$$[\mathbf{lfp}_{P_i} S](\mathbf{t}) \tag{3}$$

where S is now a set of fixpoint rules:

$$S = \left\{ \begin{array}{c} \forall \mathbf{x_1}\ P_1(\mathbf{x_1}) \leftarrow \varphi_1 \\ \cdots \\ \forall \mathbf{x_n}\ P_n(\mathbf{x_n}) \leftarrow \varphi_n \end{array} \right\} \tag{4}$$

Here, each φ_i is a formula with the tuple $\mathbf{x_i}$ as free variables, such that no predicate P_j (including P_i) appears negatively in it. Such an expression induces an operator that now no longer works on interpretations of a single predicate P, but rather on interpretations of all the P_i simultaneously. That is, in each interpretation I with domain D that interprets (at least) all predicates of the φ_i that are not any of the P_j, such a set of rules defines a fixpoint operator $\mathcal{F}_S^I$ that takes an "old" tuple of interpretations

$$(P_1^J, \ldots, P_n^J) \in 2^{D^{|\mathbf{x}_1|}} \times \cdots \times 2^{D^{|\mathbf{x}_n|}}$$

of the fixpoint predicates and maps it to a new such tuple $(P_1^{J'}, \ldots, P_n^{J'})$ in the following way.

Definition 3 For a set S of fixpoint equations of form (4) and an interpretation I for (at least) all the predicates of S different from the fixpoint predicates P_j, the operator $\mathcal{F}_S^I$ maps each interpretation J for the predicates P_j to the interpretation J' that interprets each P_j by $Sat(\varphi_j, I \cup J)$.

Ultimately, however, it is only the predicate P_i that we are interested in: the formula (3) is satisfied if and only if $\mathbf{t}$ belongs to $P_i^{J^\infty}$, where J^∞ is the least fixpoint of this operator $\mathcal{F}_S^I$.

In our version of simultaneous least-fixpoint logic FO(SLFP), we will also allow *positively nested* occurrences of the least-fixpoint operator, i.e., the formulas φ_i may themselves again contain **lfp**-expressions, as long as these do not appear in the scope of a negation. This too does not increase the expressivity of the logic, and removing such nestings is again simply a matter of increasing the arities of the fixpoint predicates [18].

We will call a simultaneous least-fixpoint expression *guarded* if and only if all of the formulas φ_i and predicates P_j satisfy the condition of Def. 1, i.e., if the φ_i are guarded formulas in which the P_j do not appear as guards. Translating a guarded simultaneous fixpoint formula into FO(LFP) preserves its guardedness. Therefore, it now suffices to show that we can translate guarded FO(ID) to guarded FO(SLFP). The crux of this transformation lies in the following characterization of the limit of a well-founded induction sequence as the least-fixpoint of a certain operator.

It was shown in [5] that defining the semantics of a definition Δ in FO(ID) as the limit of an induction sequence is equivalent to using the well-founded model of Δ [23]. Let us briefly recall how the well-founded model is usually defined. There is a strong duality between three-valued interpretations and pairs (I, J) of two-valued interpretations for which $I \leq J$, in the sense that $P^I \subseteq P^J$ for each predicate P. To be more concrete, to such a pair (I, J) we associate the following three-valued interpretation $\tau(I, J)$.

– If both $P^I(\mathbf{a}) = \mathbf{t}$ and $P^J(\mathbf{a}) = \mathbf{t}$ then $P^{\tau(I,J)}$ also maps $\mathbf{a}$ to $\mathbf{t}$.
– Similarly, if both $P^I(\mathbf{a}) = \mathbf{f}$ and $P^J(\mathbf{a}) = \mathbf{f}$ then $P^{\tau(I,J)}$ also maps $\mathbf{a}$ to $\mathbf{f}$.
– If $P^I(\mathbf{a}) = \mathbf{f}$ and $P^J(\mathbf{a}) = \mathbf{t}$ then $P^{\tau(I,J)}$ maps $\mathbf{a}$ to $\mathbf{u}$.

We will refer to pairs (I, J) for which $I \leq J$ as *consistent* pairs. τ is a one-to-one mapping between consistent pairs and three-valued interpretations.

For a definition Δ and an interpretation O for Δ's open symbols, let us now define a function U_Δ^O that takes as arguments a consistent pair I, J of two-valued interpretations such that $I|_{Op(\Delta)} = O = J|_{Op(\Delta)}$. In fact, whenever we superscript a function with such an interpretation O, we will implicitly assume that the domain of this function is restricted to interpretations I for which $I|_{Op(\Delta)} = O$, or to tuples thereof. We define U^O_Δ as mapping (I, J) to the following two-valued interpretation K: $P^K(\mathbf{a}) = \mathbf{t}$ if and only if Δ contains a rule $\forall \mathbf{x}\ P(\mathbf{x}) \leftarrow \varphi(\mathbf{x})$ such that $\nu(\varphi(\mathbf{a})) = \mathbf{t}$ with $\nu = \tau(I, J)$.

This function U^O_Δ is monotone in its first argument, and anti-monotone in its second. Let us now consider the operator $U^O_\Delta(\cdot, J)$ that we obtain by keeping the second argument J of U^O_Δ fixed, i.e., $U^O_\Delta(\cdot, J)$ maps each interpretation I to $U^O_\Delta(I, J)$. Because U^O_Δ is monotone in its first argument, each operator $U^O_\Delta(\cdot, J)$ is monotone. Tarski's fixpoint theorem implies that each monotone operator — and therefore every operator $U^O_\Delta(\cdot, J)$ — has a least fixpoint. For each J, let us denote this least fixpoint by $ST^O_\Delta(J)$, i.e.,

$$ST^O_\Delta(J) = \mathbf{lfp}(U^O_\Delta(\cdot, J)).$$

The function ST^O_Δ is now itself again an operator on the lattice of interpretations extending O. Because U^O_Δ is anti-monotone in its second argument, this operator ST^O_Δ is anti-monotone, i.e., for all $J \leq J'$,

$$ST^O_\Delta(J) = \mathbf{lfp}(U^O_\Delta(\cdot, J)) \geq \mathbf{lfp}(U^O_\Delta(\cdot, J')) = ST^O_\Delta(J').$$

This anti-monotonicity implies that if we take a consistent pair (I, J), i.e., one such that $I \leq J$, then the following pair of interpretations

$$(ST^O_\Delta(J), ST^O_\Delta(I)) \tag{5}$$

is again consistent. Let us denote by $\mathcal{S}^O_\Delta$ the operator that maps each consistent pair of interpretations (I, J) to this new consistent pair (5). The one-to-one mapping τ allows us to equivalently view $\mathcal{S}^O_\Delta$ as an operator on three-valued interpretations. It can be shown that $\mathcal{S}^O_\Delta$ is monotone w.r.t. the precision order $\leq_p$ on three-valued interpretations (to be more precise, whenever (I, J) and (I', J') are such that $\tau(I, J) \leq_p \tau(I', J')$, then $\tau(\mathcal{S}^O_\Delta(I, J)) \leq_p \tau(\mathcal{S}^O_\Delta(I, J))$). As a consequence, Tarksi's fixpoint theorem implies that it has a least fixpoint (V, W). The corresponding three-valued interpretation $\tau(V, W)$ is called the *well-founded model* of Δ given O. [5] showed that the limit of each induction sequence of Δ in O is precisely the well-founded model of Δ given O.

If Δ is total, then, by definition, it must be the case that $V = W$. It can easily be seen that in this case V (and W) is also the least fixpoint of the square $(ST^O_\Delta)^2$ that maps each I to $ST^O_\Delta(ST^O_\Delta(I))$. Our transformation of FO(ID) to FO(SLFP) now exploits precisely this fact, by constructing a **lfp**-expression that produces this least fixpoint $\mathbf{lfp}((ST^O_\Delta)^2)$. Essentially, this transformation therefore provides an encoding of the alternation fixpoint construct of [23]. It proceeds as follows.

Let Δ be an FO(ID)-definition of the following form:

$$\left\{ \begin{array}{c} \forall \mathbf{x_1}\ P_1(\mathbf{x_1}) \leftarrow \varphi_1 \\ \dots \\ \forall \mathbf{x_n}\ P_n(\mathbf{x_n}) \leftarrow \varphi_n \end{array} \right\}$$

in which every predicate is defined by a single rule, i.e., $P_i \neq P_j$ for all $i \neq j$. (Recall that a set of rules with the same predicate in the head can always be replaced by a single rule whose body is the disjunction of the bodies of the original rules.)

For each P_i, let P'_i be a new predicate symbol. For each i, let φ'_i be the result of replacing all *positive* occurrences of an atom $P_j(\mathbf{t})$ in φ_i by $P'_j(\mathbf{t})$. Let S be the following system of fixpoint equations:

$$\left\{\begin{array}{c} \forall \mathbf{x_1}\ P'_1(\mathbf{x_1}) \leftarrow \varphi'_1 \\ \dots \\ \forall \mathbf{x_n}\ P'_n(\mathbf{x_n}) \leftarrow \varphi'_n \end{array}\right\}$$

The following correspondence between this system of equations and the operator ST^O_Δ now follows directly from its definition.

Lemma 1 *Let I be an interpretation for $Def(\Delta) \cup Op(\Delta)$. With S the above system of equations, we have that:*

$$I \models \bigwedge_{i=1}^{n} \forall \mathbf{x}\ P_i(\mathbf{x}_i) \Leftrightarrow [\mathbf{lfp}_{P'_i} S](\mathbf{x}_i) \text{ if and only if } I|_{Def(\Delta)} = ST_\Delta^{I|_{Op(\Delta)}}(I).$$

Proof Let O be $I|_{Op(\Delta)}$. It suffices to show that $U^O_\Delta(\cdot, I)$ is identical to $\mathcal{F}^I_S$, because this will obviously imply that also $\mathbf{lfp}(U^O_\Delta(\cdot, I)) = ST^O_\Delta(I)$ is equal to $\mathbf{lfp}(\mathcal{F}^I_S)$. Let J be an interpretation for $Def(\Delta)$. The operator $U^O_\Delta(J, I)$ now produces a new interpretation for $Def(\Delta)$ by evaluating the bodies of the rules of Δ as follows.

- Open predicates are interpreted by $O = I|_{Op(\Delta)}$.
- Negative occurrences of defined predicates are interpreted by I.
- Positive occurrences of defined predicates are interpreted by J.

Let us compare this to $\mathcal{F}^I_S(J)$, which produces a new interpretation for the fixpoint predicates P' of S by evaluating:

- the non-fixpoint predicates by I;
- the fixpoint predicates P'_i by J.

Because the transformation has replaced precisely the positive occurrences of a defined predicate P_i by P'_i, it is now obvious that these two operators do the same thing.

To construct the least fixpoint of the square $(ST^O_\Delta)^2$, we need a fixpoint expression that it is slightly more complicated still. For each i, we define φ''_i as the result of replacing each *negative* occurrence of an atom $P_j(\mathbf{t})$ in φ_i by the expression $[\mathbf{lfp}_{P'_j} S](\mathbf{t})$. Let S' be the following system of fixpoint equations:

$$\left\{\begin{array}{c} \forall \mathbf{x_1}\ P_1(\mathbf{x_1}) \leftarrow \varphi''_1 \\ \dots \\ \forall \mathbf{x_n}\ P_n(\mathbf{x_n}) \leftarrow \varphi''_n \end{array}\right\}$$

Theorem 3 *Assume that Δ is a total definition. Let I be an interpretation for $Def(\Delta) \cup Op(\Delta)$. With S' the above system of equations, we have that:*

$$I \models \bigwedge_{i=1}^{n} \forall \mathbf{x}\ P_i(\mathbf{x}_i) \Leftrightarrow [\mathbf{lfp}_{P_i} S'](\mathbf{x}_i) \text{ if and only if } I \models \Delta$$

Before proving this theorem, let us examine a small example.

Example 1 We consider the following FO(ID) definition.

$$\left\{\begin{array}{l}\forall x\ P(x) \leftarrow \neg Q(x).\\ \forall x\ Q(x) \leftarrow Q(x).\end{array}\right\}$$

This translates to:

$$\begin{array}{r}\forall x\ P(x) \Leftrightarrow \left([\mathbf{lfp}_P \left\{\begin{array}{l} P(x) \leftarrow \neg[\mathbf{lfp}_{Q'} \left\{\begin{array}{l}P'(x) \leftarrow \neg Q(x)\\ Q'(x) \leftarrow Q'(x)\end{array}\right\}](x)\\ Q(x) \leftarrow Q(x)\end{array}\right\}](x)\right)\\ \wedge\ \forall x\ Q(x) \Leftrightarrow \left([\mathbf{lfp}_Q \left\{\begin{array}{l} P(x) \leftarrow \neg[\mathbf{lfp}_{Q'} \left\{\begin{array}{l}P'(x) \leftarrow \neg Q(x)\\ Q'(x) \leftarrow Q'(x)\end{array}\right\}](x)\\ Q(x) \leftarrow Q(x)\end{array}\right\}](x)\right)\end{array}$$

Taking a domain consisting of a single object A, the well-founded model of this example in that domain says that $P(A)$ is $\mathbf{t}$, while $Q(A)$ is $\mathbf{f}$. Intuitively, this conclusion can be reached in the following way.

1. If we assume that $P(A)$ and $Q(A)$ are both false, we are sure to be *under*estimating their actual truth value. By applying this assumption to only the *negative* occurrences of these atoms in a formula, we are sure to be *over*estimating the effect that their actual truth value would have on this formula. For instance, assuming that $Q(A)$ is false will lead us to believe that $\neg Q(A)$ is true.
2. By applying this assumption to all negative occurrences of atoms in the bodies of the rules of a definition, we end up with a positive definition (i.e., one which now no longer has any negative occurrence of atoms, and of which we can therefore simple construct a least fixpoint) that is sure to *over*estimate the truth of all atoms. In this case, it will be
$$\left\{\begin{array}{l}P(A) \leftarrow \neg\mathbf{f}\\ Q(A) \leftarrow Q(A)\end{array}\right\}$$
and its least fixpoint says that $P(A)$ is true and $Q(A)$ is false.
3. If we now apply this *over*estimate to the negative occurrences of atoms in rule bodies of the original definition, we end up with an *under*estimating positive definition. In this case, the overestimate says that $P(A)$ is true and $Q(A)$ is false, so the underestimating definition is that
$$\left\{\begin{array}{l}P(A) \leftarrow \neg\mathbf{f}\\ Q(A) \leftarrow Q(A)\end{array}\right\}$$
4. The least fixpoint of this underestimating definition now provides a new, greater underestimate for the truth of the atoms, which we can plug in again in step 1, and repeat until a fixpoint is reached. In this case, this greater underestimate says that $P(A)$ is true and $Q(A)$ is false, which is the same as the overestimate we already had in point 2 and hence the fixpoint.

The nested **lfp** expressions above now simulates this process as follows.

1. The first iteration of the outer **lfp** starts by assuming that its fixpoint predicates P and Q have an empty interpretation, i.e., that $P(A)$ and $Q(A)$ are false, as in:
$$\left\{\begin{array}{l} P(A) \leftarrow \neg[\mathbf{lfp}_{Q'} \left\{\begin{array}{l} P'(A) \leftarrow \neg\mathbf{f} \\ Q'(A) \leftarrow Q'(A) \end{array}\right\}](A) \\ Q(A) \leftarrow \mathbf{f} \end{array}\right\}$$
2. Under this interpretation for P and Q, the inner **lfp** expression then construct the least fixpoint of P' and Q', which says that $P'(A)$ is true and $Q'(A)$ is false;
3. The body of the first rule of the outer **lfp** expression then singles out Q' from this least fixpoint computed by the inner expression and check whether A belongs to it. It does not, so the body of this rule evaluates to true:
$$\left\{\begin{array}{l} P(A) \leftarrow \neg\mathbf{f} \\ Q(A) \leftarrow \mathbf{f} \end{array}\right\}$$
4. Therefore, this first iteration produces that $P(A)$ is true and $Q(A)$ is false, which is then plugged in as an interpretation for P and Q to start the second iteration of the outer expression. Again, the reader can verify that this second iteration will again produce the same result, which is therefore the least fixpoint.

The following proof now formally establishes the correctness of this translation.

Proof (Proof (of Theorem 3)) It follows from Lemma 1 and the construction of S' that, in each structure O, the expression $[\mathbf{lfp}_{P_i} S']$ is in fact constructing the least fixpoint of the operator — let us call it Γ^O — that maps each I to $\Gamma^O(I) = U^O_\Delta(I, ST^O_\Delta(I))$. We now show that this coincides with the least fixpoint of the square $(ST^O_\Delta)^2$, which, under the assumption of totality, is known to be equal to the well-founded model of Δ given O. Therefore, this will suffice to prove the desired equivalence.

First, we remark that all fixpoints I of $(ST^O_\Delta)^2$ are also fixpoints of Γ^O: because each such I is by definition the least fixpoint of $U(\cdot, ST^O_\Delta(I))$, it is *a fortiori* also a fixpoint, which means that $I = U(I, ST^O_\Delta(I))$. Let I be $\mathbf{lfp}((ST^O_\Delta)^2)$ and let J be $\mathbf{lfp}(\Gamma^O)$. We will show that $I = J$. Being a fixpoint of $(ST^O_\Delta)^2$, I is also a fixpoint of Γ^O, so $I \geq J$.

On the other hand, J is by construction a fixpoint of Γ^O, so $J = \Gamma^O(J) = U^O_\Delta(J, ST^O_\Delta(J))$. Therefore, J is also a fixpoint of the operator $U^O_\Delta(\cdot, ST^O_\Delta(J))$ and, hence, must be greater than the least fixpoint of this operator, i.e., $J \geq \mathbf{lfp}(U^O_\Delta(\cdot, ST^O_\Delta(J)) = (ST^O_\Delta)^2(J)$, or in other words, J is a prefixpoint of $(ST^O_\Delta)^2$. Because Tarski's theorem implies that the least fixpoint of a monotone operator is also its least prefixpoint, we know that for the least fixpoint I of $(ST^O_\Delta)^2$, it is the case that $I \leq J$. Having shown in the previous paragraph that also $I \geq J$, we conclude $I = J$.

This theorem shows how we can, in general, transform FO(ID) into FO(SLFP). However, if we look at the formulas that are used in this translation:
$$\forall \mathbf{x}\ P_i(\mathbf{x}_i) \Leftrightarrow [\mathbf{lfp}_{P_i} S'](\mathbf{x}_i)$$
we see that these are not guarded (even if all formulas in S' are guarded). Therefore, they are unsuitable for our purposes. Our solution to this problem will be to avoid having to assert the equivalence between the defined predicate symbols P_i and their definitions, by simply *replacing* all of the predicate symbols P_i by their definition. However, this can only work if there are no cyclic dependencies between predicates defined in different definitions.

Definition 4 An FO(ID) theory T is *acyclic* if the definitions of T can be ordered as $(\Delta_1, \ldots, \Delta_n)$, such that none of the defined predicates of Δ_i appear in a definition Δ_j for which $j < i$.

Note that this condition implies that each predicate is defined by at most one definition. Moreover, it ensures that we can recursively replace defined predicates P by their unique definition without potentially running into infinite loops.

We also remark that this condition applies to predicates defined by *different* definitions, whereas the condition of no negative dependency-cycles that we imposed in Def. 2 to ensure totality applies within a *single* definition.

Putting all of this together, we now define the guarded fragment of FO(ID) as follows.

Definition 5 An FO(ID) theory T is (loosely) guarded if all the following conditions are satisfied.

- Each definition in T is (loosely) guarded (Def. 2).
- Each FO formula in T is (loosely) guarded.
- No defined predicates (i.e., predicates that belong to $Def(\Delta)$ of some definition Δ of T) are used as guards.
- T is acyclic (Def. 4).

We now have all the elements necessary for our transformation. Recall that, above, we presented a translation of an FO(ID) definition Δ into a set $eq(\Delta)$ of fixpoint equations. In more detail, for each Δ of the form:

$$\left\{ \begin{array}{c} \forall \mathbf{x_1}\ P_1(\mathbf{x_1}) \leftarrow \varphi_1 \\ \cdots \\ \forall \mathbf{x_n}\ P_n(\mathbf{x_n}) \leftarrow \varphi_n \end{array} \right\}$$

we constructed $eq(\Delta)$ as:

$$\left\{ \begin{array}{c} \forall \mathbf{x_1}\ P_1(\mathbf{x_1}) \leftarrow \varphi_1'' \\ \cdots \\ \forall \mathbf{x_n}\ P_n(\mathbf{x_n}) \leftarrow \varphi_n'' \end{array} \right\}$$

where each φ_i'' was the result of replacing each *negative* occurrence of an atom $P_j(\mathbf{t})$ in φ_i by:

$$[\mathbf{lfp}_{P_j'} \left\{ \begin{array}{c} \forall \mathbf{x_1}\ P_1'(\mathbf{x_1}) \leftarrow \varphi_1' \\ \cdots \\ \forall \mathbf{x_n}\ P_n'(\mathbf{x_n}) \leftarrow \varphi_n' \end{array} \right\}](\mathbf{t})$$

where each φ_i' is the result of replacing all *positive* occurrences of an atom $P_j(\mathbf{t})$ in φ_i by $P_j'(\mathbf{t})$.

We now translate a (loosely) guarded FO(ID) theory T into an FO(SLFP) theory $s(T)$ as follows.

Definition 6 For a (loosely) guarded FO(ID) theory T, we define the corresponding FO(SLF) theory $s(T)$ as the result of:

- recursively replacing all atoms $P(\mathbf{t})$, such that P is defined by the (unique) definition Δ, by the least fixpoint expression $[\mathbf{lfp}_P eq(\Delta)](\mathbf{t})$;

– subsequently removing all definitions from T.

We now obtain the following result.

Theorem 4 *The (loosely) guarded fragment of FO(ID) is decidable.*

Proof It follows from Theorem 3 and the fact that the transformation steps of Definition 6 are equivalence preserving that a (loosely) guarded FO(ID) theory T is equivalent to the (loosely)guarded FO(LFP) theory $s(T)$. The known results for FO(LFP) [10] therefore imply that (loosely) guarded FO(ID) is indeed decidable.

The 2EXPTIME upperbound of the (loosely) guarded fragment of FO(LFP) does not directly carry over, however, because the transformation s may exponentially increase the size of the theory. This proof therefore only allows us to say that the complexity is at most triple exponential—an upperbound which may or may not be tight.

Having found a decidable fragment of FO(ID), we now examine how this gives rise to a corresponding decidable fragment of $\mathcal{ALCI}$(ID).

Definition 7 We say that a role R (or concept C) is *defined* in an $\mathcal{ALCI}$(ID) theory T if T contains a formula of the form $R \doteq \varphi$ (or $C \doteq \varphi$) or a definition in which a rule $R \leftarrow \varphi$ (or $C \leftarrow \varphi$) appears. An $\mathcal{ALCI}$(ID) theory is *loosely guarded* if it is the case that:

– for every construct $\exists R.C$ that appears in T, R is either an atomic role that is not defined, or a conjunction of roles $R_1 \sqcap \cdots \sqcap R_n$ such that at least one of the R_i is an atomic role that is not defined;
– for every construct $\forall R.C$ that appears in T, R is an atomic role that is not defined;
– for every construct $R.S$ that appears in T, both R and S must be atomic roles that are not defined, or the inverse of such a role;
– for every inclusion $C_1 \sqsubseteq C_2$, C_1 must be an atomic concept that is not defined, or a conjunction of such concepts;
– there are no definitions in which predicates depend negatively on each other;
– the theory is acyclic (Def. 4).

It is easy to see that, if all these conditions are satisfied, the corresponding FO(ID) theory, as defined in Section 4, is loosely guarded.

Lemma 2 *For every loosely guarded $\mathcal{ALCI}$(ID) theory T, it holds that the translation $s(\langle T\rangle)$ of T into FO(LFP) is a loosely guarded theory that is equivalent to T.*

Proof A loosely guarded $\mathcal{ALCI}$(ID) theory T satisfies the five conditions enumerated in Definition 7. The last item in this list ensures that we can actually perform the transformation of replacing each defined predicate by its unique definition. Therefore, we can effectively construct $s(\langle T\rangle)$. Moreover, the fourth condition ensures the equivalence to the original theory T, as shown in Theorem 3.

Therefore, it suffices to show that all quantifiers that appear in the FO(ID) theory $\langle T\rangle$ are indeed properly guarded. Consulting the relevant tables in section 4 reveals that quantifiers can be introduced in the following ways.

– Each concept inclusion axiom $C \sqsubseteq D$ introduces a single universal quantifier; the concept C is the only potential guard for this quantifier.

- The $\forall R.C$ and $\exists R.C$ constructs introduce a universal and existential quantifier, respectively, that could potentially be guarded by R.
- The $R_1.R_2$ construct introduces an existential quantifier that can be loosely guarded by the conjunction of R_1 and R_2.

Recall that the guarded fragment of FO(LFP) allows all predicates to serve as guards apart from fixpoint predicates. This means that, for each of these cases, if the "potential guard", as we called it, is not a defined predicate, it will actually be a proper guard of the formula. The first three conditions in the list of Definition 7 now ensure precisely this fact.

Consequently, we obtain the following result.

Theorem 5 *The guarded fragment of $\mathcal{ALCI}$(ID) is decidable.*

The guarded fragment of $\mathcal{ALCI}$(ID) is a straightforward analogue of the guarded fragment of FO(LFP) and is, therefore, one of the most obvious candidates for developing a guarded fragment of $\mathcal{ALCI}$(ID). However, it remains to be seen how useful it is in practice.

For instance, our representation of the game in Section 5 falls outside this fragment. The main problem here lies in formulas such as:

$$Disappears \doteq Chosen \sqcup \exists InColourGroup.Chosen$$

The role $InColourGroup$ is not a valid guard here, because our theory also contains a definition for this role; moreover, because $InColourGroup$ is meant to represent a "reachability"-relation, we really have no alternative but to include an inductive definition for it.

This seems to suggest that, maybe, this guarded fragment is not the decidable fragment that is the most suited for capturing the interesting features of FO(ID). In a currently ongoing research project, other decidable fragments of FO(ID) are being constructed. In particular, efforts are being made to derive such fragments from Büchi's decidability result for monadic second order logic in the natural numbers [3]. Once completed, this research will induce other decidable fragments for $\mathcal{ALCI}$(ID), which might provide a better match with the typical modeling style it inherits from FO(ID).

Nevertheless, the next section presents an example that is covered by the guarded fragment we have developed here.

7 A Second Example: Conference management

While the example given in Section 5 illustrates the new features offered by our language, it is perhaps not very evocative from a Semantic Web perspective. [8] uses a running example that could easily be envisioned as part of an online conference management system such as EasyChair. We will briefly outline how this example can be represented in our language.

There are a number of papers (belonging to a concept *Paper*), which are assigned keywords (the role $AssignedTo(Keyword, Paper)$). The keywords are clustered into sets of similar keywords ($Similar(Keyword, Keyword)$). There are also people, who can be experts in certain areas ($ExpertIn(Person, Area)$). We assume to have some

DL-knowledge base which provides information about these concepts and roles. This may be just a list of facts such as

$$ExpertIn(IanHorrocks, SemanticWeb),$$

but it may also derive the knowledge from, say, a publication database.

[8] then proceeds to build an ASP-layer on top of the existing DL theory. In our case, we achieve the same effect by adding some useful $\mathcal{ALCI}$(ID) definitions to the knowledge base.

First, we add the assumption that whenever some keyword (e.g., "OWL") is assigned to a paper, all similar keywords (e.g., "Web Ontology Language") also pertain ($PertainsTo(Keyword, Paper)$) to it.

$$\left\{\begin{array}{l} PertainsTo \leftarrow AssignedTo \\ PertainsTo \leftarrow Similar.PertainsTo \end{array}\right\} \tag{6}$$

Without making this more precise, [8] also assumes that the knowledge base somehow assigns a paper to a specific area ($InArea(Paper, Area)$) based on the keywords that pertain to it. In our case, we can imagine this being done, for instance, using a predicate $KeywordFor(Keyword, Area)$ and the following definition:

$$InArea \doteq PertainsTo^{-}.KeywordFor \tag{7}$$

We can then define those people who are good candidate reviewers for a paper as those who are experts in the area of the paper:

$$GoodCandidateFor \doteq ExpertIn.InArea^{-} \tag{8}$$

The goal of reviewer assignment is to assign a good candidate to each paper ($Assigned(Paper, Reviewer)$). A PC chair might therefore be interested in querying those problematic papers that have not yet been assigned a good candidate:

$$Problematic \doteq Paper \sqcap \neg\exists(Assigned \sqcap GoodCandidateFor).Person \tag{9}$$

This theory does not fall in the guarded fragment of $\mathcal{ALCI}$(ID), because defined predicates appear as guards in various places. Again, this is not necessarily problematic: if we have a fixed set of n papers that is to be divided among m reviewers, then this provides a fixed and finite domain context, in which we can easily compute the set of problematic papers for a given assignment. By adding also the following restriction to the theory:

$$Problematic \sqsubseteq \bot \tag{10}$$

we can compute whether an assignment without problematic papers is possible. This is again a model expansion task, where $AssignedTo(Keyword, Paper)$, $Similar(Keyword, Keyword)$, $KeywordFor(Keyword, Area)$ and $ExpertIn(Person, Area)$ are given, and $Assigned(Paper, Reviewer)$ needs to be computed.

In addition to these closed world reasoning tasks, a conference management system might also need to perform open world reasoning. For instance, while submissions are still in progress and reviewers are still being added to the system, we might very well be interested to know which papers are already certainly (un-)problematic given a current partial set of assignments. Unfortunately, the guarded fragment of $\mathcal{ALCI}$(ID) does not provide us with any guarantees that this task will be decidable.

A possible work-around is to introduce for each paper p a separate concept $AssignedTo_p$ that contains just those keywords that were assigned to p, a separate concept $Pertains_p$ that contains only those keywords pertaining to p, and so on.

$$\left\{\begin{array}{l} Keyword_p \leftarrow \exists Similar.Keyword_p \\ Keyword_p \leftarrow AssignedTo_p \end{array}\right\}$$

$$Area_p \doteq \exists KeywordFor.Keyword_p$$

$$Candidate_p \doteq \exists ExpertIn.Area_p$$

This theory now is guarded, with *Similar*, *KeywordFor* and *ExpertIn* serving as guards. To figure out whether p is a problematic paper, we can then add also:

$$Candidate_p \sqcap Assigned_p \sqsubseteq \bot$$

In this way, we will be able to reason with the existence of unknown reviewers, though not with unknown papers, since each paper requires its own set of concepts. Of course, even here the guardedness still depends on the assumption that the rest of the theory does not define any of the concepts *Similar*, *KeywordFor* or *ExpertIn*.

8 Related Work

In this section, we discuss several related approaches. For clarity, we divide them into a number of different categories, the first being that of hybrid languages.

8.1 Hybrid languages

In [6], a combination of DL and LP under the well-founded semantics is investigated, while dl-programs [8] combine DL and Answer Set Programming (ASP). Unlike FO(ID)'s semantic integration, these two approaches foster a strong separation between the LP and DL components: essentially, they allow a logic program to pose queries to a description logic theory, with the latter acting as a black box towards the former. In contrast, FO(ID) provides a full integration, in which both rules and description logic axioms are first-class citizens.

While hybrid systems offer flexibility and are often also easy to implement using off-the-shelf technology, they are not particularly satisfactory from a knowledge representation point of view. They do not shed much light on the semantical relation between the two languages or on how they might complement each other. Moreover, since the languages are never truly integrated, a hybrid theory cannot be thought of as a *single* piece of knowledge—a single mental model of some reality—but must always be considered as two different pieces of knowledge that interact according to some specific interface. One illustration of this is that, in [8], disjunctive information from a DL knowledge base is lost to the ASP program. For instance, a hybrid theory containing DL-statements $Person \sqsubseteq Woman \sqcup Man$ and $Person(Alex)$, together with an ASP program:

$$\forall x\ P(x) \leftarrow DL[Woman](x).$$
$$\forall x\ P(x) \leftarrow DL[Man](x).$$

does not actually imply $P(Alex)$, because the DL knowledge base cannot derive either $Woman(Alex)$ or $Man(Alex)$.

8.2 Layered integrations

This section discusses the language of r-hybrid rules [21], which was later extended to DL+log [22], and to g-hybrid knowledge bases [12], which uses a transformation to guarded open answer set programs [13] to prove decidability.

Unlike e.g. dl-programs, r-hybrid rules offer a semantic integration of DL and LP. However, even though there is no longer a strict, syntactical separation between the two components, the approach is, at the semantical level, still essentially a layered one, in which an ASP program is added on top of a DL knowledge base. This can clearly be seen in the semantics of the language, which proceeds according to a two-step process. First, the predicates that appear in the DL part of the theory are considered in isolation; afterwards, they are "projected" out of the theory, leaving only an ASP program. To be more precise, an interpretation I is an *NM-model* of a combined theory (K, P) if:

1. the restriction of I to the DL predicates is a model (in the standard sense) of the DL knowledge base K;
2. the restriction of I to the rule predicates is an answer set of the program that results from replacing in P all DL predicates by their truth value according to I.

This language is therefore less hybrid than dl-programs, in the sense that the integration is done directly on the semantical level, rather than by having one component query the other. However, it is still a layered approach, in which a crisp distinction is made between DL-predicates and rule predicates. For an example of what this means, consider the following two rules:

$$\begin{aligned} Person(x) &\leftarrow Man(x) \\ Person(x) &\leftarrow Woman(x) \end{aligned}$$

In a regular logic program (as well as in an inductive definition), these two rules would imply, in the absence of any other rules with *Person* in their head, that each person is either a man or a woman. In DL+log, they will mean the same thing, *but only* if *Person* does not appear in the DL part of the theory. If, on the other hand, there is for instance a fact *Person*(*Bob*) in the DL ABox, then *Person* will be interpreted as a DL predicate and the meaning of these two rules will be different: the theory will then have a model in which *Bob* is a person but neither a man nor a woman.

By contrast, FO(ID) and $\mathcal{ALCI}$(ID) do not make any distinction between DL and rule predicates; a definition

$$\left\{ \begin{array}{l} Person \leftarrow Man \\ Person \leftarrow Woman \end{array} \right\}$$

always implies that a person must be either a man or a woman, regardless of how or where the predicates that appear in it are used in the rest of the theory.

8.3 Full integrations

One of the most widely known languages to combine DL with rules is that of SWRL [14], which offers a full (non-hybrid and non-layered) integration of the two. SWRL extends OWL with Horn clauses under the regular FO semantics. One of the limitations of SWRL is that its rules can only be used to state sufficient conditions, and not necessary ones. As a consequence, SWRL rules cannot be used to define concepts. Because definitions are such an important feature of DL (see footnote on p.3), this is a curious lack. $\mathcal{ALCI}$(ID), by contrast, also allows (in addition to the regular first-order implication $\sqsupseteq$) the definitional rule construct $\leftarrow$, which can be used to define concepts, even inductively. Moreover, it also allows more than just Horn clauses, since the body of a rule may contain negation and quantifiers.

While SWRL itself is undecidable, many decidable fragments exists. For instance, there is the fragment of *(strongly) safe* rules [20]: a SWRL-rule (i.e., Horn clause) is safe if every one of its variables appears in an atom whose predicate is not used anywhere in the TBox (i.e., it may only appear in other rules or in the ABox). *Description logic programs* [11], which were actually developed prior to SWRL itself, form another decidable fragment of SWRL.

There also exists a number of full integrations that, like $\mathcal{ALCI}$(ID), offer more expressive languages. In particular, [19] uses an embedding into MKNF to achieve this, while [2] uses the first-order version of auto-epistemic logic FO-AEL. These approaches are similar to ours in that they also start from a very expressive language and then consider various interesting fragments of these languages. Moreover, like FO(ID), both MKNF and FO-AEL are extensions of classical propositional/first-order logic.

The main difference to our approach lies in the nature of this extension: FO(ID) extends classical logic with inductive definitions, whereas MKNF and FO-AEL both extend classical logic with modal operator(s). Modal operators represent statements about knowledge or possibility, which inherently refer to a reference class of many possible worlds, existing side-by-side with the actual world. The semantics of these languages is therefore naturally defined in terms of sets of interpretations. Inductive definitions, on the other hand, do not have a modal component and do not refer to multiple possible worlds. All they do is define the meaning of some predicate(s) in terms of some other predicate(s), by means of a set of rules that serve as a recipe to construct one from the other. The formal semantics of FO(ID) is therefore defined in terms of this constructive process, as it is envisaged by mathematicians writing down such definitions, and [5] attempts to demonstrate that the formal objects that FO(ID) uses for this purpose indeed correctly capture the underlying intuitions. FO(ID) and MKNF/FO-AEL therefore try to achieve different goals by means of different mathematical constructions. These goals are not mutually exclusive and might even be complementary: there is nothing *a priori* impossible or undesirable about developing a language FO(ID, K, NF) that extends classical logic with both inductive definitions and modal operators.

This goes back to a point that we already touched upon in the introduction to this paper; namely, that the question of how to extend DL with rules is not one which has a single right answer. It all depends on what this extension is meant to achieve. People who are of the opinion that a key capability missing from DL is that of explicitly reasoning about knowledge would do well to look into the papers on MKNF or FO-AEL. People who would like the ability to include more expressive forms of definitions in a DL theory are hopefully well-served by this paper.

For our part, we have attempted throughout this paper to argue that inductive definitions are a useful addition to DL. Inductive definitions are well-known to mathematicians and computer scientists, and judging from the abundance of inductively defined concepts in research papers (e.g., reachability, transitive closure, the satisfaction relation $\models$ of first-order logic), they indeed pop up quite often. Inductively defined relations such as the ancestors of a person or the Google PageRank of a website cannot be defined by typical DLs—and the same also holds for certain non-inductive concepts like being someone's uncle. In [5], a mathematical theory was developed with the sole purpose of demonstrating that FO(ID) offers a semantically correct and syntactically convenient representation of such inductive definitions. Borrowing from this, we therefore believe that $\mathcal{ALCI}$(ID) is a useful extension of DL with rules. While the MKNF and FO-AEL based approaches may very well also be useful extensions of DL with rules, they ultimately try to a achieve a different goal, which makes them a potential complement rather than an alternative to our approach. Whether it is possible and/or desirable to combine them into a joint language that can express both modalities and inductive definitions is a potential topic for future research.

9 Conclusions and future work

In this paper, we have investigated the extension of DL with rules that is induced by FO(ID). We first argued that this is quite natural for the following reasons:

– there is an appealing match between the intuitive notion of a "case" in an inductive definition and the formal construct of a definitional rule in FO(ID);
– since non-inductive definitions are already a key feature of DL, it makes sense to exploit this match by adding definitional rules to DL, in order to extend the class of definitions that can be represented.

Motivated by these arguments, we have defined a fragment $\mathcal{ALCI}$(ID) of FO(ID), which offers a DL-like syntax for representing (inductive) definitions. For FO(ID), and therefore also for $\mathcal{ALCI}$(ID), there exist useful inference tasks that can be performed efficiently; in Section 6, we discussed the task of model expansion, which can be used to compute the next state of the game we modeled in Section 5. This is despite the fact that the language as a whole is undecidable for deductive inference. We have defined a decidable guarded fragment of $\mathcal{ALCI}$(ID). A more comprehensive analysis of this and other decidable fragments of FO(ID) is left for future work. The goal of the current paper is not to present a single DL rule-language that is suitable for all purposes, but rather to point towards FO(ID) in general as an interesting foundation from which such languages can be derived.

Acknowledgements Part of this work was carried out while Joost Vennekens was a postdoctoral researcher of the FWO Vlaanderen. Joost Vennekens is an affiliated researcher of the Dept. of Computer Science at the KU. Leuven (Belgium).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Theory, Implementation and Applications. Cambridge University Press (2002)
2. Bruijn, J.D., Pearce, D., Polleres, A., Valverde, A.: Quantified equilibrium logic and hybrid rules. In: International Conference on Web Reasoning and Rule Systems (RR) (2007)
3. Büchi, J.R.: Weak second order arithmetic and finite automata. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **6**, 66–92 (1960)
4. Börger, E., Grädel, E., Gurevich, Y.: The classical Decision Problem. Perspectives in Mathematical Logic. Springer-Verlag (1997)

5. Denecker, M., Vennekens, J.: Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In: C. Baral, G. Brewka, J. Schlipf (eds.) Ninth International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR, *Lecture Notes in Artificial Intelligence*, vol. LNAI 4483, pp. 84–96. Springer (2007)
6. Drabent, W., Henriksson, J., Maluszynski, J.: HD-rules: A hybrid system interfacing Prolog with DL-reasoners. In: 2nd International Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services (2007)
7. Ebbinghaus, H.D., Flum, J.: Finite Model Theory. Perspectives in Mathematical Logic. Springer-Verlag (1995)
8. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: In Proceedings of the International Conference of Knowledge Representation and Reasoning (KR) (2004)
9. Grädel, E.: On the restraining power of guards. Journal of Symbolic Logic **64**(4), 1719–1742 (1988)
10. Grädel, E., Walukiewicz, I.: Guarded fixed point logic. In: LICS (Logic in Computer Science), pp. 45–55 (1999)
11. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: Proceedings of the 12th international conference on World Wide Web (2003)
12. Heymans, S., de Bruijn, J., Predoiu, L., Feier, C., Van Nieuwenborgh, D.: Guarded hybrid knowledge bases. Theory and Practice of Logic Programming (TPLP) **8**(3), 411–429 (2008)
13. Heymans, S., Van Nieuwenborgh, D., Vermeir, D.: Open answer set programming with guarded programs. ACM TOCL **9**(4) (2008)
14. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantics web rule language combining OWL and RuleML (2004). W3C Submission, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
15. Kreutzer, S.: Expressive equivalence of least and inflationary fixed-point logic. In: Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS) (2002)
16. Lenzerini, M.: Tbox and Abox reasoning in expressive description logics. In: In Proc. of KR-96, pp. 316–327. Morgan Kaufmann (1996)
17. Mitchell, D., Ternovska, E.: A framework for representing and solving NP search problems. In: AAAI'05, pp. 430–435. AAAI Press/MIT Press (2005)
18. Moschovakis, Y.N.: Elementary Induction on Abstract Structures. North-Holland Publishing Company, Amsterdam- New York (1974)
19. Motik, B., Rosati, R.: Reconciling description logics and rules. In: Journal of the ACM (JACM) **57**(5) (2010)
20. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. In: Proc. of the 3rd International Semantic Web Conference (ISWC) (2004)
21. Rosati, R.: On the decidability and complexity of integrating ontologies and rules. Journal of Web Semantics **3**, 61–73 (2005)
22. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 68–78 (2006)
23. Van Gelder, A.: The alternating fixpoint of logic programs with negation. Journal of Computer and System Sciences **47**(1), 185–221 (1993)
24. Vennekens, J., Denecker, M.: FO(ID) as an extension of DL with rules. In: European Semantic Web Conference, *Lecture Notes in Computer Science*, vol. 5554, pp. 384–398 (2009)