9600004-5

Artificial Intelligence 82 (1996) 353–367

Research Note

On the relative expressiveness of description logics and predicate logics

Alex Borgida *

Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA

Received June 1995; revised January 1996

Abstract

It is natural to view concept and role definitions in description logics as expressing monadic and dyadic predicates in predicate calculus. We show that the descriptions built using the constructors usually considered in the DL literature are characterized exactly as the predicates definable by formulas in $\ddot{\mathcal{L}}^3$, the subset of first-order predicate calculus with monadic and dyadic predicates which allows only three variable symbols. In order to handle "number bounds", we allow numeric quantifiers, and for transitive closure of roles we use infinitary disjunction. Using previous results in the literature concerning languages with limited numbers of variables, we get as corollaries the existence of formulas of FOPC which cannot be expressed as descriptions. We also show that by omitting role composition, descriptions express exactly the formulas in $\ddot{\mathcal{L}}^2$, which is known to be decidable.

1. Overview

Description languages (DLs) are descendants of the KL-ONE [6] knowledge representation system, and have been the object of intensive theoretical study in the past decade, as well as forming the basis of several widely used implemented systems. A significant contribution of the original KL-ONE proposal of Brachman was the idea that in addition to primitive notions, such as "Person", one can also *define* concepts such as "Persons with at least three friends"—the definition providing both necessary and sufficient conditions for membership in this new concept.

It seems natural to wonder whether there are limits to the concepts that one could define in KL-ONE, or one of its descendants; for example, in a knowledge base dealing with persons and their relationships, is it possible to define the notion of "persons who have a clique of at least four friends" (i.e., persons who have four friends, who in turn are all friends of each other). Surprisingly, such a question has not been generally addressed in the knowledge representation literature (but

SSDI

ARTINT 1379

see [2]), although in databases there has been considerable work on comparing the "expressive power" of various query languages.

After establishing a formal framework that allows us to compare the "meaning" expressed by two different formalisms, we provide results that show, among others, that the DLs considered so far, even ones for which subsumption is undecidable, can express only (and all) those notions that can be expressed in (variants of) FOPC with three or fewer variables. Previous results in the literature then show that there are indeed things that one can say with $k+1$ variables that cannot be expressed in any way with just k variables. These limitations are all the more significant since, recently, there have been a number of proposals for using DLs as query languages for accessing data in databases (e.g., [5, 7]). It is in fact this practical question that motivated the present research.

The equivalences established in this paper also offer, as incidental corollaries, alternate proofs of decidability and undecidability for several subsets of description constructors, using previously known results about predicate calculus.

2. Descriptions

Descriptions are used to specify concepts (which group individuals) and roles (which relate pairs of individuals). For example, consider the description in Fig. 1. It is constructed from identifiers denoting binary relations (e.g., `venue`, `players`), individuals (e.g., `Toronto`, `1`) and other concepts (e.g., `GAME`, `STADIUM`) using description constructors **and**, **all**, **at-most**, **fills** and **one-of**. The description in Fig. 1 has as intended denotation "*Games which are held in a stadium* (their `venue` role's value must be an instance of concept `STADIUM`), *involving at most ten players* (the `players` role has at most ten values/fillers), *all of whom are from Toronto* (all `players` fillers are restricted to have value `Toronto` for attribute `hometown`)". Roles need not be atomic identifiers—there are composite descriptions denoting roles. For example, the term **compose**[`children`, **restrict**[`children`, `MALE`]] denotes "*grandsons*" in English, since **restrict** takes a binary relation such as `children` and derives the one where all elements in its range are instances of concept `MALE` (hence `sons`), and **compose** corresponds to binary relation composition.

☆ Preliminary versions of some results were presented, for a database audience, at the CIKM'94 Conference.

* Supported in part by NSF Grant IRI 91-19310. E-mail: borgida@cs.rutgers.edu.

and[
    all[`venue`, `STADIUM`]
    at-most[`10`, `players`]
    all[`players`, **fills**[`hometown`, `Toronto`]]

Fig. 1. Composite description for a concept.

2.1. The languages and logics of descriptions

The language of descriptions is obtained recursively by starting from a *schema* $S = (\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$ of names for concepts, roles, and individuals, and building from them more complex terms using description constructors.

Over the years, a considerable variety of DLs have been proposed, studied and implemented. Table 1 presents the language $\mathcal{DL}$, which contains a comprehensive list of the constructors based on recent survey papers [3, 19, 24]. (Clearly, the set of constructors in $\mathcal{DL}$ is not minimal; this is intentional, since considerable work in the field is devoted to finding subsets of constructors for which various decision problems have good computational properties.)

In the table, and elsewhere, we use the symbols A, B, C to range over concept descriptions, $p, q, \ldots$ to range over role descriptions, $a, b, \ldots$ for individual names, and D, E, F to denote descriptions in general.

The semantics of description terms is given denotationally, using the notion of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, (\cdot)^{\mathcal{I}} \rangle$, which starts with a domain (non-empty universe) of values $\Delta^{\mathcal{I}}$, and a mapping $(\cdot)^{\mathcal{I}}$ from concept descriptions to subsets of the domain, and role descriptions to sets of 2-tuples over the domain; the mapping also

Table 1
$\mathcal{DL}$: a compendium of concept and role constructors

Term	Interpretation
TOP-CONCEPT	$\Delta^{\mathcal{I}}$
NOTHING	$\emptyset$
and[C, D]	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
or[C, D]	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
not[C]	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
all[p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \subseteq C^{\mathcal{I}}\}$
some[p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}} \neq \emptyset\}$
at-least[n, p]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta)\rvert \geqslant n\}$
at-most[n, p]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta)\rvert \leqslant n\}$
at-least-c[n, p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}}\rvert \geqslant n\}$
at-most-c[n, p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}}\rvert \leqslant n\}$
same-as[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) = q^{\mathcal{I}}(\delta)\}$
subset[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \subseteq q^{\mathcal{I}}(\delta)\}$
not-same-as[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \neq q^{\mathcal{I}}(\delta)\}$
fills[p, b]	$\{\delta \in \mathrm{dom}^{\mathcal{I}} \mid b^{\mathcal{I}} \in p^{\mathcal{I}}(\delta)\}$
one-of[$b_1, \ldots, b_m$]	$\{b_1^{\mathcal{I}}, \ldots, b_m^{\mathcal{I}}\}$
TOP-ROLE	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
IDENTITY	$\{(\delta, \delta) \mid \delta \in \Delta^{\mathcal{I}}\}$
role-and[p, q]	$p^{\mathcal{I}} \cap q^{\mathcal{I}}$
role-or[p, q]	$p^{\mathcal{I}} \cup q^{\mathcal{I}}$
role-not[p]	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$
inverse[p]	$\{(\delta, \delta') \mid (\delta', \delta) \in R^{\mathcal{I}}\}$
restrict[p, C]	$\{(\delta, \delta') \in p^{\mathcal{I}} \mid \delta' \in C^{\mathcal{I}}\}$
compose[p, q]	$p^{\mathcal{I}} \circ q^{\mathcal{I}}$
product[C, D]	$C^{\mathcal{I}} \times D^{\mathcal{I}}$
trans[p]	$\bigcup_{n \geqslant 0} (p^{\mathcal{I}})^n$

associates with every individual name $\mathcal{IN}$ some distinct value in $\Delta^{\mathcal{I}}$. (The reason for distinctness is the Unique Name Assumption, which is normally made in knowledge representation.) The interpretation function $(\cdot)^{\mathcal{I}}$ is extended recursively to composite descriptions in Table 1, where the interpretation of roles is viewed, in the obvious ways, as a function from an individual to the set of individuals related to it (i.e., as $\mathcal{RN} \rightarrow (\Delta^{\mathcal{I}} \rightarrow 2^{\Delta^{\mathcal{I}}})$).

The *meaning* of a description D can then be thought of as a mapping from interpretations $\mathcal{I}$ to extents $D^{\mathcal{I}}$, and a variety of logical judgments are defined on its basis:

- description E *subsumes* D, written $D \Longrightarrow E$, iff for every interpretation $\mathcal{I}$, $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$;
- description D is coherent/satisfiable, if there is at least one $\mathcal{I}$ such that $D^{\mathcal{I}} \neq \emptyset$;
- descriptions E and D are disjoint iff for every interpretation $\mathcal{I}$, $D^{\mathcal{I}} \cap E^{\mathcal{I}} = \emptyset$.

3. Relating descriptions to predicate calculus

There is an obvious similarity between concepts (respectively roles) in DLs, and monadic (respectively dyadic) predicates in predicate calculus. Such a similarity was already exploited by Schmolze and Israel [22] to give a semantics for the original KL-ONE language using the λ-calculus. Our aim is to compare the "expressive power" of various sublanguages of descriptions and predicate calculus. To do so we need to set up a common framework for the two formalisms.

3.1. Predicate calculus

We start, as usual, from a set of names for predicates and individual constants. Since almost all the work on DLs has been carried out in the framework of binary (as opposed to n-ary) roles/relationships, we will side-step the problem of dealing with n-ary predicates by restricting the arity of predicates to 1 or 2. We therefore start again from a schema $S = (\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$ of monadic and dyadic predicates, and constant symbols. From these, as well as the equality predicate and variable symbols in the set $\mathcal{VN}$ (which is assumed to have a lexicographic ordering on it), atomic and composite formulas are built, as usual, with connectives $\neg$, $\wedge$, $\exists$, the other connectives being defined as macros. The notion of "free variable" is defined as usual, and we use the notation $\Psi(x, y)$ to refer to a formula that has as free variables, x and y, and only them.

The semantics of $\mathcal{FOPC}$ is also based on an interpretation $\mathcal{I}$ of $\mathcal{CN}$, $\mathcal{RN}$ and $\mathcal{IN}$, but in this case one traditionally also has to deal with variable symbols; this can be done by means of a partial function $\mu : \mathcal{VN} \rightarrow \Delta^{\mathcal{I}}$ which provides a substitution for some of the variables. An interpretation $\mathcal{I}$ and a substitution μ define a partial function $[\![\cdot]\!]^{\mathcal{I},\mu}$ from formulas to truth values $\{True, False\}$ in the usual way; for example, for the formula $P(x, a)$, we have $[\![P(x, a)]\!]^{\mathcal{I},\mu} = True$ iff $(\mu(x), a^{\mathcal{I}}) \in P^{\mathcal{I}}$.

In order to be able to compare formulas with free variables on equal footing with descriptions, we extend the interpretation function $(\cdot)^{\mathcal{I}}$ to formulas of the predicate calculus as done in the field of databases: the meaning $\Psi^{\mathcal{I}}$ of a formula Ψ is a set of k-tuples, where k is the number of distinct free variables in Ψ:

$$\Psi(x_1,\ldots,x_k)^{\mathcal{I}} = \{(\alpha_1,\ldots,\alpha_k) \mid \alpha_i \in \Delta^{\mathcal{I}}, [\![\Psi(x_1,\ldots,x_k)]\!]^{\mathcal{I},\mu} = \mathit{True} \text{ for the substitution } \mu \text{ mapping } x_i \text{ to } \alpha_i\}.$$

Since the tuples are ordered, in order to obtain a unique meaning we require the free variables $x_1, x_2, \ldots$ to appear in lexicographic order.

The *meaning of a formula* Ψ is then defined as the function $\lambda\mathcal{I}.\Psi^{\mathcal{I}}$, where $\Psi^{\mathcal{I}}$ is defined as above if Ψ has at least one free variable, and is otherwise a truth value. Note that by the semantic definition of $\forall$ and $\Leftrightarrow$, for any formulas $\Psi(x_1,\ldots)$ and $\Phi(x_1,\ldots)$

Ψ and Φ have the same meaning
iff $\forall x_1,\ldots.\Psi(x_1,\ldots) \Leftrightarrow \Phi(x_1,\ldots)$ is a theorem.

3.2. Comparing the meaning of formulas in different languages

We now have "meanings" for sentences in both languages defined uniformly as mappings from interpretations to sets of tuples over $\Delta^{\mathcal{I}}$. We can therefore say that description D has the same meaning as formula $\Psi(x)$ iff $\lambda\mathcal{I}.D^{\mathcal{I}} = \lambda\mathcal{I}.\Psi(x)^{\mathcal{I}}$, where $\mathcal{I}$ are interpretations over the same schema S. [1] We can also create hybrid sentences, which mix the two kinds of formulas: On the one hand, we can allow descriptions to appear as monadic and dyadic predicates in FOPC formulas, interpreting $[\![D(a)]\!]^{\mathcal{I},\mu}$, where D is a concept description, as $a^{\mathcal{I}} \in D^{\mathcal{I}}$; in this formulation, $\Psi(x)$ expresses the meaning of D iff $\forall x.D(x) \Leftrightarrow \Psi(x)$ is a theorem. On the other hand, we can treat $\Psi(x)$ as a description by interpreting it using $(\cdot)^{\mathcal{I}}$ whenever we encounter it, in which case $\Psi(x)$ expresses the meaning of D iff $D \Longrightarrow \Psi(x)$ and $\Psi(x) \Longrightarrow D$. Similar hybrid formulas can be set up for the other kinds of judgments one is normally interested in for DLs.

We will then say that some language $\mathcal{L}_2$ *is as expressive as* language $\mathcal{L}_1$, if there is a total function *transl* from $\mathcal{L}_1$ to $\mathcal{L}_2$ such that for every sentence L in $\mathcal{L}_1$, *transl*(L) expresses the meaning of L. Two languages are *equally expressive* if each is as expressive as the other.

For example, Schmolze and Israel [22] show that FOPC is as expressive as $\mathcal{DL} - \{\mathbf{trans}\}$ by essentially defining a translation function $\tau\langle\cdot\rangle$, which maps concepts to formulas with free variable x, and roles to formulas with free variables x and y. For example, $\tau\langle \mathbf{all}[\mathtt{p}, \mathtt{C}]\rangle$ is "$\lambda x.\forall w.p(x,w) \Rightarrow C(w)$", while the translation of $\mathbf{compose}[\mathtt{p}, \mathtt{q}]$ is $\lambda x, y.\exists z.p(x,z) \wedge q(z,y)$.

4. DLs and FOPC with limited variables

Note that the translation from descriptions to FOPC mentioned in the previous section must introduce new variables whenever a new quantifier appears, in order to avoid spurious capture of variables. For example, without this precaution, the

[1] For alternate techniques for comparing the "expressive power" of languages, see [2, 23].

Table 2

Term $\mathtt{C}$	$\mathcal{T}^x\langle\mathtt{C}\rangle$	$\mathcal{T}^y\langle\mathtt{C}\rangle$
TOP-CONCEPT	$x = x$	$y = y$
NOTHING	$\neg(x = x)$	$\neg(y = y)$
and$[\mathtt{C},\mathtt{D}]$	$\mathcal{T}^x\langle\mathtt{C}\rangle \wedge \mathcal{T}^x\langle\mathtt{D}\rangle$	$\mathcal{T}^y\langle\mathtt{C}\rangle \wedge \mathcal{T}^y\langle\mathtt{D}\rangle$
or$[\mathtt{C},\mathtt{D}]$	$\mathcal{T}^x\langle\mathtt{C}\rangle \vee \mathcal{T}^x\langle\mathtt{D}\rangle$	$\mathcal{T}^y\langle\mathtt{C}\rangle \vee \mathcal{T}^y\langle\mathtt{D}\rangle$
not$[\mathtt{C}]$	$\neg\mathcal{T}^x\langle\mathtt{C}\rangle$	$\neg\mathcal{T}^y\langle\mathtt{C}\rangle$
all$[\mathtt{p},\mathtt{C}]$	$\forall y.\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \Rightarrow \mathcal{T}^y\langle\mathtt{C}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle\mathtt{p}\rangle \Rightarrow \mathcal{T}^x\langle\mathtt{C}\rangle$
some$[\mathtt{p},\mathtt{C}]$	$\exists y.\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \wedge \mathcal{T}^y\langle\mathtt{C}\rangle$	$\exists x.\mathcal{T}^{y,x}\langle\mathtt{p}\rangle \wedge \mathcal{T}^x\langle\mathtt{C}\rangle$
subset$[\mathtt{p},\mathtt{q}]$	$\forall y.\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \Rightarrow \mathcal{T}^{x,y}\langle\mathtt{q}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle\mathtt{p}\rangle \Rightarrow \mathcal{T}^{y,x}\langle\mathtt{q}\rangle$
same-as$[\mathtt{p},\mathtt{q}]$	$\forall y.\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{x,y}\langle\mathtt{q}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle\mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{y,x}\langle\mathtt{q}\rangle$
not-same-as$[\mathtt{p},\mathtt{q}]$	$\exists y.\neg(\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{x,y}\langle\mathtt{q}\rangle)$	$\exists x.\neg(\mathcal{T}^{y,x}\langle\mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{y,x}\langle\mathtt{q}\rangle)$
fills$[\mathtt{p},\mathtt{b}]$	$\exists y.(y = b) \wedge \mathcal{T}^{x,y}\langle\mathtt{p}\rangle$	$\exists x.(x = b) \wedge \mathcal{T}^{y,x}\langle\mathtt{p}\rangle$
one-of$[\mathtt{b}_1,\ldots,\mathtt{b}_m]$	$x = b_1 \vee \cdots \vee x = b_m$	$y = b_1 \vee \cdots \vee y = b_m$

translation of **compose**$(\mathtt{p}, $ **compose**$(\mathtt{p},\mathtt{p}))$ would yield $\exists z.p(x,z) \wedge \exists z.(p(z,z) \wedge p(z,y))$, which is clearly wrong—one wants $\exists z_2.p(x,z_2) \wedge \exists z_1.(p(z_2,z_1) \wedge p(z_1,y))$.

We now show that in fact one does not need to introduce new variables, thus relating DLs to FOPC with limited number of variables. So let $\mathcal{L}^k$ be the set of all FOPC formulas with equality which can be expressed using at most k variables. Note that $\mathcal{L}^k$ does not limit the number of nested quantifiers in a formula since the same variable may be reused in nested subformulas, as in $\forall x,y.P(x,y) \Rightarrow \exists x.Q(y,x)$. Properties of such language families have been studied, among others, in [8, 11, 16]. In our case, since we are dealing with roles and concepts, we will be interested only in those formulas that (i) have one or two free variables (though they may have closed subformulas), and (ii) have only monadic and dyadic predicates. Henceforth, we will use $\ddot{\mathcal{L}}^k$ to refer to this sublanguage.

Our first result shows that almost everything that can be said with DLs, can be said with just a few variables.

Theorem 1. *The language* $\ddot{\mathcal{L}}^3$ *is as expressive as* $\mathcal{DL}-\{$**trans**, **at-least**, **at-most**$\}$. *The language* $\ddot{\mathcal{L}}^2$ *is as expressive as* $\mathcal{DL}-\{$**compose**, **trans**, **at-least**, **at-most**$\}$.

Proof. The proof relies on a more careful encoding of the constructors into predicate calculus, where the same variable is reused as much as possible. We will present the translation function in several variants that behaves as follows: $\mathcal{T}^x\langle\rangle$ makes x be the free variable of the monadic predicate it will produce for its argument concept, while $\mathcal{T}^y\langle\rangle$ makes the free variable be y. So, for a concept $\mathtt{C}$ in $\mathcal{CN}$, $\mathcal{T}^x\langle\mathtt{C}\rangle = C(x)$, while $\mathcal{T}^y\langle\mathtt{C}\rangle = C(y)$. In the case of roles $\mathtt{R}$, $\mathcal{T}^{x,y}\langle\mathtt{R}\rangle$ produces a predicate $R(x,y)$, while $\mathcal{T}^{y,x}\langle\mathtt{R}\rangle$ produces predicate $R(y,x)$. The translation functions $\mathcal{T}^x\langle\rangle$, $\mathcal{T}^y\langle\rangle$, and $\mathcal{T}^{x,y}\langle\rangle$ are presented in Tables 2 and 3. $\mathcal{T}^{y,x}\langle\rangle$ is obtained from $\mathcal{T}^{x,y}\langle\rangle$ by simultaneously exchanging *all* occurrences of x and y (whether free or bound).

The key ideas in the above translation are the alternating use of $\mathcal{T}^x\langle\rangle$ and $\mathcal{T}^y\langle\rangle$ in nested concept descriptions (such as **all** or **some**), and the use of the equalities $x = z$ and $y = z$ in the translation of **compose**, which make it unnecessary to introduce new variables during the translation process.

The translation function $\mathcal{T}\langle\rangle$ can now be defined simply as $\mathcal{T}\langle\mathtt{C}\rangle = \mathcal{T}^x\langle\mathtt{C}\rangle$ for

Table 3

Term $\mathtt{R}$	Translation $\mathcal{T}^{x,y}\langle\mathtt{R}\rangle$
TOP-ROLE	$x = x \wedge y = y$
IDENTITY	$x = y$
role-and[p, q]	$\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \wedge \mathcal{T}^{x,y}\langle\mathtt{q}\rangle$
role-or[p, q]	$\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \vee \mathcal{T}^{x,y}\langle\mathtt{q}\rangle$
role-not[p]	$\neg\mathcal{T}^{x,y}\langle\mathtt{p}\rangle$
inverse[p]	$\mathcal{T}^{y,x}\langle\mathtt{p}\rangle$
restrict[p, C]	$\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \wedge \mathcal{T}^{y}\langle\mathtt{C}\rangle$
compose[p, q]	$\exists z.(\exists y.y = z \wedge \mathcal{T}^{x,y}\langle\mathtt{p}\rangle) \wedge (\exists x.x = z \wedge \mathcal{T}^{x,y}\langle\mathtt{q}\rangle)$
product[C, D]	$\mathcal{T}^{x}\langle\mathtt{C}\rangle \wedge \mathcal{T}^{y}\langle\mathtt{D}\rangle$

Table 4

$\Upsilon(x)$	D_Υ
$C(x)$, $C \in \mathcal{CN}$	C
$P(x,b)$, $P \in \mathcal{RN}$	**fills**[P, b]
$P(b,x)$	**fills**[**inverse**[P], b]
$P(x,x)$	**some**[**role-and**[P, IDENTITY], TOP-CONCEPT]
$x = b$	**one-of**[b]
$x = x$	TOP-CONCEPT
$\neg\Psi_1(x)$	**not**[D_{Ψ_1}]
$\Psi() \wedge \Phi(x)$	**and**[D_Ψ, D_Φ]
$\Psi(x) \wedge \Phi(x)$	**and**[D_Ψ, D_Φ]
$\exists y.\Psi(x,y)$	**some**[R_Ψ, TOP-CONCEPT]
$\exists y.\Psi(x)$	D_Ψ

concept descriptions C, and $\mathcal{T}\langle\mathtt{r}\rangle = \mathcal{T}^{x,y}\langle\mathtt{r}\rangle$ for role descriptions r. Once again, a straightforward proof by induction shows that for every description D and interpretation $\mathcal{I}$, $\mathtt{D}^\mathcal{I} = \mathcal{T}\langle\mathtt{D}\rangle^\mathcal{I}$. $\square$

It turns out that the converses of the above results also hold. To begin with, we have:

Theorem 2. *The description language with concept constructors* {TOP-CONCEPT, NOTHING, **and**, **not**, **some**, **fills**, **one-of**} *and role constructors* {**role-and**, **role-not**, **product**, **inverse**} *is as expressive as* $\ddot{\mathcal{L}}^2$.

Proof. Suppose the two variables we can use in $\ddot{\mathcal{L}}^2$ are x and y. We shall proceed by structural recursion on the syntax of formulas with up to two free variables.

Table 4 lists the various kinds of formulas $\Upsilon(x)$ that have a single free variable x, and shows how each kind is translated into a concept description D_Υ:

The translation of formulas with a single free variable y is identical, except for the case when $\Upsilon(y)$ is of the form $\exists x.\Psi(x,y)$, when we need to invert the relationship represented by Ψ, so it is translated as **some**[**inverse**[D_Ψ], TOP-CONCEPT].

Formulas of the form $\Psi(x,y)$ are translated to roles relating x to y according to Table 5.

A formula $\Upsilon()$ without free variables occurs only as a conjunct, and the number of free variables (one or two) in its context determines its translation: a concept

Table 5

$\Upsilon(x,y)$	R_Υ
$P(x,y)$, $P \in \mathcal{RN}$	P
$P(y,x)$	**inverse**[P]
$x = y$	IDENTITY
$\neg\Psi(x,y)$	**role-not**[$R_{\Psi(x,y)}$]
$\Psi(x) \wedge \Phi(y)$	**product**[D_Ψ, D_Φ]
$\Psi(x,y) \wedge \Phi()$	**role-and**[R_Ψ, R_Φ]
$\Psi(x,y) \wedge \Phi(x)$	**role-and**[R_Ψ, **product**[D_Φ, TOP-CONCEPT]]
$\Psi(x,y) \wedge \Phi(y)$	**role-and**[R_Ψ, **product**[TOP-CONCEPT, D_Φ]]
$\Psi(x,y) \wedge \Phi(x,y)$	**role-and**[R_Φ, R_Ψ]

Table 6

$\Upsilon()$	D_Υ
$C(b)$	**all**[**product**[TOP-CONCEPT, **oneof**[b]], C]
$P(b,b)$	**all**[**product**[TOP-CONCEPT, **oneof**[b]], **fills**[P, b]]
$P(a,b)$	**all**[**product**[TOP-CONCEPT, **oneof**[a]], **fills**[P, b]]
$b = b$	TOP-CONCEPT
$a = b$	NOTHING
$\neg\Psi()$	**not**[$D_{\Psi()}$]
$\Psi() \wedge \Phi()$	**and**[$D_{\Psi()}, D_{\Phi()}$]
$\exists x.\Psi(x)$	**some**[TOP-ROLE, D_Ψ]
$\exists y.\Psi(y)$	**some**[TOP-ROLE, D_Ψ]
$\exists x.\Psi()$	D_Ψ

or a role. For the case when a concept is desired, we need a description D_Υ with the property that for any interpretation $\mathcal{I}$, if $\Upsilon()^\mathcal{I} = True$ then $D_\Upsilon^\mathcal{I} = \Delta^\mathcal{I}$, and if $\Upsilon()^\mathcal{I} = False$ then $D_\Upsilon^\mathcal{I} = \emptyset$. This essentially "gates" the meaning of the other conjunct. Table 6 provides such translations.

In contexts where we require roles, the translation is just

$$R_{\Upsilon()} = \mathbf{product}[D_{\Upsilon()}, D_{\Upsilon()}]. \qquad \square$$

More generally, we have:

Theorem 3. *The description language* $\mathcal{DL} - \{\mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$ *is as expressive as* $\ddot{\mathcal{L}}^3$.

Proof. In this case we deal with formulas having possibly three variables: x, y, and z. Once again we define a recursive procedure for translating formulas into descriptions. Formulas Ψ with one or two free variables are translated, as before, into concept descriptions D_Ψ or role descriptions R_Ψ. The translation of formulas of the form $\Psi(x,z)$ or $\Psi(y,z)$ is carried out by the same procedure as for formulas $\Psi(x,y)$. Similarly, the translation of monadic formulas of the form $\Psi(y)$ and $\Psi(z)$ is handled by the same procedure as for formulas $\Psi(x)$.

The main novelty therefore lies in the translation of subformulas $\Psi(x,y,z)$ with three free variables. Since the final formula can have no more than two free variables, every subformula $\Psi(x, y, z)$ must in fact be part of a larger subformula of the form $\exists\gamma.\Phi(x, y, z)$, where γ is either x, y or z. Without loss of generality, suppose it is $\exists z.\Phi(x, y, z)$. Because all atomic formulas involve only monadic and dyadic predicates, we will be able to prove that it is sufficient to consider the case when $\exists z.\Phi(x, y, z)$ is of the form $\exists z.\Phi_1(x, z) \wedge \Phi_2(y, z)$, which can be translated as the role description **compose**$[R_{\Phi_1}, \textbf{inverse}[R_{\Phi_2}]]$.

It therefore remains to show that every formula of the form $\exists z.\Phi(x, y, z)$ can be reduced to the form $\exists z.\Phi_1(x, z) \wedge \Phi_2(y, z)$. This is accomplished by massaging $\Phi(x, y, z)$ into a normal form that allows the quantifier to be moved in. Specifically, let Φ_i be the *maximal subformulas* of $\Phi(x, y, z)$ that have at most two free variables. Since there are only three variables and all predicates are of arity at most 2, $\Phi(x, y, z)$ must be the boolean combination of these Φ_i—any intervening quantifier would reduce the number of variables to 2, and hence would be part of some Φ_j. Using de Morgan's laws, it is therefore possible to rewrite $\Phi(x, y, z)$ into disjunctive normal form $\bigvee_j(\bigwedge_k \Phi_{j,k})$, where each $\Phi_{j,k}$ is either some Φ_i or its negation. Since an existential quantifier can be moved in past disjunctions $(\exists z.(\alpha \vee \beta) \equiv (\exists z.\alpha) \vee (\exists z.\beta))$, $\exists z.\Phi(x, y, z)$ is therefore logically equivalent to $\bigvee_j \Theta_j$ where $\Theta_j = \exists z.(\bigwedge_k \Phi_{j,k})$, and note that each Θ_j has at most two free variables (z is being quantified over).

Therefore we need only consider formulas $\exists z.\Phi(x, y, z)$ where $\Phi(x, y, z)$ is the conjunction of subformulas $\Phi_{j,k}$, each with at most two free variables. By associativity of conjunction, group together the subformulas that have the same free variables, thereby obtaining that $\Phi(x, y, z)$ is in the most general case of the form $\Psi_0() \wedge \Psi_1(x) \wedge \Psi_2(y) \wedge \Psi_3(z) \wedge \Psi_4(x, y) \wedge \Psi_5(x, z) \wedge \Psi_6(y, z)$. But then we can move the subformulas not containing z outside the quantifier, rewriting $\exists z.\Phi(x, y, z)$ in the form $\beta(x, y) \wedge \exists z.((\Psi_3(z) \wedge \Psi_5(x, z)) \wedge \Psi_6(y, z))$. Therefore, in the end the formula in the scope of $\exists z$ does have the desired restricted form $\exists z.\Psi_7(x, z) \wedge \Psi_5(y, z)$, establishing our claim. (Note that if subformulas Ψ_5 or Ψ_6 are empty then we are left inside the scope of $\exists z$ with a formula with at most two free variables, whose translation had already been provided earlier.) □

Combining the preceding theorems we get

Corollary 4.
(i) *The description language* $\mathcal{DL} - \{\textbf{trans}, \textbf{compose}, \textbf{at-least}, \textbf{at-most}\}$ *and* $\ddot{\mathcal{L}}^2$ *are equally expressive.*
(ii) *The description language* $\mathcal{DL} - \{\textbf{trans}, \textbf{at-least}, \textbf{at-most}\}$ *and* $\ddot{\mathcal{L}}^3$ *are equally expressive.*

4.1. Consequences concerning DLs

Several potentially interesting corollaries follow from the above theorems and existing results in the logic and computer science literature. In each case, we associate the corollary with the reference where the key result comes from.

Corollary 5 (Immerman). *There exists a "conjunctive" FOPC formula $\Psi(x)$ (one involving only logical connectives $\wedge$ and $\exists$) which cannot be expressed by any des-*

cription in $\mathcal{DL} - \{\mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$.

This is true for any conjunctive formula that is expressible in FOL but not in $\mathcal{L}^3$. For example, Immerman [11] shows, among others, that for the schema $\langle\{NODE\}, \{Edge\}, \emptyset\rangle$, which can be used to represent a graph, $\ddot{\mathcal{L}}^{k-1}$ does not allow the expression of such graph-theoretic properties as the existence of a k-subclique. In other words, there are pairs of graphs that differ in having the property but that cannot be distinguished by any formula of $\ddot{\mathcal{L}}^{k-1}$. Therefore, for the following simple formula $\Psi(y)$ from $\ddot{\mathcal{L}}^4$

$$(\exists x_1, x_2, x_3)\left(NODE(y) \bigwedge_i Edge(y, x_i) \bigwedge_{i \neq j} Edge(x_i, x_j)\right)$$

there is no concept in $\mathcal{DL} - \{\mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$ which has the same meaning.

Immerman's results also allow us to prove, among others, that **compose** provides an increase in expressive power, which is not entirely obvious since in some situations **compose** can be eliminated; for example **all**[**compose**[p, q], C] is equivalent to **all**[p, **all**[q, C]].

Corollary 6. *The constructor* **compose** *is independent of the other ones in* $\mathcal{DL} - \{\mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$ *in the sense that it cannot be eliminated by finding an equivalent description without it.*

This is obtained simply from the fact that the formula

$$(\exists x_1, x_2)(NODE(y) \bigwedge_i Edge(y, x_i) \bigwedge_{i \neq j} Edge(x_i, x_j)),$$

which is in $\ddot{\mathcal{L}}^3$, is not in $\ddot{\mathcal{L}}^2$ for the schema above. (An alternate proof is provided by the decidability results below.)

As pointed out by Franz Baader, the previous results also have consequences concerning the decidability of subsumption for various classes of connectives.

Corollary 7.
(Mortimer) *Subsumption is decidable for* $\mathcal{DL} - \{\mathbf{compose}, \mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$. ■
(Lewis) *Subsumption is undecidable for* $\mathcal{DL} - \{\mathbf{trans}, \mathbf{at\text{-}least}, \mathbf{at\text{-}most}\}$.

The first result follows from the decidability of validity in the logic $\mathcal{L}^2$ with equality, proven in [16]: the subsumption problem $D \Longrightarrow E$ can be posed as the validity of the formula $\forall x.\mathcal{T}^x\langle \mathtt{D}\rangle \Rightarrow \mathcal{T}^x\langle \mathtt{E}\rangle$, which by Theorem 1, is in $\ddot{\mathcal{L}}^2$.

The second result follows from the undecidability of validity for the class of formulas with quantifier prefix $\forall\exists\forall$, shown in [14]: the formula exhibited in Lewis' proof only has monadic and dyadic predicates, and given a closed formula $\Psi()$ in $\ddot{\mathcal{L}}^3$, its validity can be determined by verifying that the formula $\Psi'(x)$, defined as $(x = x) \wedge \Psi$ is true for every substitution for x; by Theorem 3, $\Psi'(x)$ can then be expressed by a description $D_{\Psi'}$, and Ψ is valid if and only if the subsumption TOP-CONCEPT$\Longrightarrow D_{\Psi'}$ holds. (We note that the second result is not novel—

various subsets of $\mathcal{DL}$ are known to be undecidable (e.g., [18])—but the proof is novel.)

4.2. Dealing with terminological axioms

Most description logics provide the ability to specify a knowledge base of additional assertions, which restrict the set of interpretations that are considered in making judgments.

The simplest such axioms provide definitions for some concept and role names, in the form $\mathtt{C}\doteq\mathtt{D}$ where $\mathtt{C} \in \mathcal{CN}$ and $\mathtt{D}$ is a concept description. If such definitions do not produce cycles (i.e., there is no recursion), then they only provide abbreviations and can be easily expanded out.

More generally, axioms of the form $\mathtt{D}\sqsubseteq\mathtt{E}$, for arbitrary descriptions $\mathtt{D}$ and $\mathtt{E}$, constrain the meaning of descriptions to consider only those interpretations $\mathcal{I}$ which are "models" of the axioms, in the sense that $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. (In this case definitions $\mathtt{C}\doteq\mathtt{D}$, even ones involving cycles, can be replaced by axioms $\mathtt{C}\sqsubseteq\mathtt{E}$ and $\mathtt{E}\sqsubseteq\mathtt{C}$, if we are content to capture the so-called "descriptive semantics" for recursion [17].) Therefore, given a knowledge base KB of axioms $\{E_i\sqsubseteq F_i\}$, the meaning of a description is now redefined as $\mathcal{M}[D, \mathrm{KB}](\mathcal{I})=D^{\mathcal{I}}$ for interpretation $\mathcal{I}$ if for every i, $E_i^{\mathcal{I}} \subseteq F_i^{\mathcal{I}}$, and is $\emptyset$ otherwise.

In this case, the translation of descriptions into predicate calculus offered in Theorem 1 must be extended to also take into account the context of the KB. The following translation function

$$\mathcal{T}\langle \mathtt{D}, \mathtt{KB}\rangle = \mathcal{T}\langle \mathtt{D}\rangle \wedge \bigwedge_i (\forall x. \mathcal{T}^x\langle E_i\rangle \Rightarrow \mathcal{T}^x\langle F_i\rangle)$$

provides a formula which has the same meaning as D in the context of the KB. Significantly, $\mathcal{T}\langle \mathtt{D}, \mathtt{KB}\rangle$ is in $\ddot{\mathcal{L}}^3$ (or $\ddot{\mathcal{L}}^2$ respectively) just in case the translation of the descriptions D, E_i and F_i are themselves in this category. Therefore judgments such as subsumption and disjointness continue to translate to logical questions about $\ddot{\mathcal{L}}^2$ or $\ddot{\mathcal{L}}^3$.

Parallel arguments apply in case the knowledge base contains additional kinds of axioms, dealing for example with the disjointness of certain descriptions.

5. Numeric quantifiers and transitive closure

The translation of descriptions involving counting, such as **at-least**$[7, \mathtt{players}]$ into standard FOPC would seem to require seven distinct variables, which would put us outside the bounds of languages with limited variables. We proceed however by extending the syntax of FOPC to allow numeric/counting quantifiers, as in [12]; for example, $\exists_7 y.\mathtt{players}(x, y)$ predicates the existence of seven distinct values for which the formula is satisfied. Note that this is not treated as an abbreviation because we will wish to say that the above formula has only two variables, x and y!

Let us extend the languages $\ddot{\mathcal{L}}^k$ with counting quantifiers $\exists_n$, for every positive integer n, obtaining the languages $\ddot{\mathcal{C}}^k$—FOPC with monadic and dyadic predicates with counting quantifiers. In fact, we will take a restricted subset of such

Table 7

Υ	D_Υ
$\exists_n y.\Psi(x,y)$	**at-least**[n, R_Ψ]
$\exists_n y.\Psi(x)$	D_Ψ and **at-least**[n, TOP-ROLE]
$\exists_n x.\Psi(x)$	**at-least-c**[n, TOP-ROLE, D_Ψ]
$\exists_n y.\Psi(y)$	**at-least-c**[n, TOP-ROLE, D_Ψ]

languages—ones where in any subformula $\exists_n.\Psi$, Ψ has no more than two free variables. For the case of two and three variables, let us call these languages $\ddot{\mathcal{L}}^2_{CNT}$ and $\ddot{\mathcal{L}}^3_{CNT}$.

Corresponding to Theorems 1–3, we then have:

Theorem 8.

(i) $\ddot{\mathcal{L}}^2_{CNT}$ *and the description language* $\mathcal{DL} - \{\textbf{trans}, \textbf{compose}\}$ *are equally expressive.*

(ii) $\ddot{\mathcal{L}}^3_{CNT}$ *and the description language* $\mathcal{DL} - \{\textbf{trans}\}$ *are equally expressive.*

Proof. For (i), in one direction $\mathcal{T}^x\langle\textbf{at-least-c}[\mathtt{n},\mathtt{p},\mathtt{C}]\rangle$ is translated as

$$\exists_n y.(\mathcal{T}^{x,y}\langle\mathtt{p}\rangle \wedge \mathcal{T}^y\langle\mathtt{C}\rangle),$$

while **at-most-c**[n, p, C] is translated as

$$\textbf{not}[\textbf{at-least-c}[\mathtt{n+1},\mathtt{p},\mathtt{C}]].$$

In the other direction, we need to give the translations given in Table 7 for formulas with zero, one or two free variables.

The proof of part (ii) follows immediately from part (i) and earlier proofs because we have restricted counting quantifiers (as opposed to the ordinary existential quantifier) so they can have in their scope only subformulas with two or fewer free variables. □

We leave it as an open problem whether the constructors in $\mathcal{DL}$ can be used to "simulate" formulas of the form $\exists_n z.\Psi(x,y,z)$, particularly $\exists_n z.\Phi_1(x,z) \wedge \Phi_2(y,z)$, or whether one would need to introduce a new version of **compose** for it. In the former case, $\ddot{\mathcal{L}}^3_{CNT}$ could be replaced by the more general $\ddot{\mathcal{C}}^3$ in the statement of the preceding theorem.

The importance of this characterization lies in the fact that in a more recent paper [8], Cai et al. present examples of the expressive limitations of $\ddot{\mathcal{C}}^k$. In particular, they describe pairs of graphs that cannot be distinguished using formulas of $\ddot{\mathcal{C}}^k$. These graphs can again be easily distinguished using only existential quantifiers and conjunction in FOPC. We therefore get once again corollaries about the expressive limitations of DLs, in this case even with number restrictions.

There are many ways in which one can attempt to deal with transitive closure. In order to obtain the similar kinds of results about the expressive limitation of DLs, we follow Kolaitis et al. in using infinitary disjunction, thus translating **trans** as

$$\mathcal{T}^{x,y}\langle\textbf{trans}[\mathtt{p}]\rangle = \bigvee_{n=0}^{\infty} \mathcal{T}^{x,y}\langle\textbf{compose}^n[\mathtt{p},\mathtt{p}]\rangle.$$

Infinitary disjunction leads to the FOPC variant $\mathcal{L}^k_{\infty,\omega}(\mathbf{COUNT})$. In this case we get the equivalent of Theorem 1, but not the converse, since there are of course many formulas involving infinite disjunction that cannot be expressed as transitive closure.

The inexpressibility results in [8, 11] are obtained using variants of Ehrenfeucht–Fraïssé pebbling games, which are shown to characterize $\ddot{\mathcal{L}}^k$ and $\ddot{\mathcal{C}}^k$. Kolaitis and Vardi [13] present a modified pebbling game that characterizes $\ddot{\mathcal{C}}^k$ extended with infinitary disjunction, and the proofs in [8] go through for this case, establishing that transitive closure will not be helpful in expressing the relevant formulas in [8]. As a result, we continue to have formulas in the FOPC with only existential quantification and conjunction that cannot be expressed as descriptions in the full $\mathcal{DL}$.

6. Conclusions

We have compared the "expressive power" of two different kinds of knowledge representation languages: description logics and predicate calculus. For this we have used a formal framework in which the meaning of a formula is defined to be a mapping from interpretations (over a fixed schema of identifiers) to sets of tuples over the domain of interpretation—a framework that preserves the logics of the two approaches. Our results characterize various subsets of the "universal" description language $\mathcal{DL}$ as having exactly the expressive power of certain subsets of FOPC with two or three variables, possibly augmented by counting quantifiers.

While we have obtained our results essentially by "direct simulation", it has been pointed out that several other results in the logical literature are relevant, and can be used to obtain alternative proofs in some cases. First, Tarski and Givant [23] have investigated a variable-free algebra for binary relations, for which they prove that it is "equipolent" with $\mathcal{L}^3$. Since the equations of this algebra can be expressed as axioms for descriptions in $\mathcal{DL}$, this provides an alternative path to proving that all of $\mathcal{L}^3$ can be expressed by $\mathcal{DL}$ (see [21]). The same referee points to a connection between description logics and $\mathcal{L}^2$ via results concerning modal logics: the description logic $\mathcal{ALC}$ is a notational variant of modal logic [20]; modal logic can be expressed in $\mathcal{L}^2$ [1]; $\mathcal{L}^2$ can be expressed by an appropriate modal logic [10]. [2]

In addition to their intrinsic interest, the results presented here have several relevant consequences for research on DLs and their application, because, as corollaries of previous results from the logical literature on $\ddot{\mathcal{L}}^k$, we have pointed to the existence of certain FOPC formulas, built only with conjunction and existential quantification, which cannot be expressed as descriptions in $\mathcal{DL}$.

Traditionally, work in the DL area has attempted to identify *subsets* of description constructors from $\mathcal{DL}$, or limited forms thereof, for which subsumption is at least decidable, maybe even tractable, and which are expressively adequate for some particular family of applications. Our results imply that this strategy will not work for hybrid systems in which the terminological component needs to express the

[2] The modal logics involved in these three results are not identical.

above kinds of formulas. In such cases, one must look for entirely new kinds of concept constructors.

Second, in applications where DLs are used as query languages for existing data or knowledge bases (see [4] for a survey), one is very likely going to need an extended query language, since even the full $\mathcal{DL}$ cannot express the so-called "conjunctive queries"—the least powerful query languages considered in the relational database literature, and for which, incidentally, subsumption is definitely decidable. The obvious route to follow in this direction is to add formulas with variables, or their equivalent (e.g., relational algebra expressions), into description languages. For example, in Loom [15] FOPC formulas may be given as arguments to the **:satisfies** concept constructor. Another approach is to integrate descriptions and Horn formulas, as in [9]. Alternatively, one can consider ways of presenting queries that can create new objects or relationships (like relational algebra in databases).

Finally, an open question remaining in this work is to examine the expressive power of DLs with *recursive* concept definitions, where recursion is defined by some fixed point semantics. It is known that such forms of recursion enhance the power of database query languages, so we may expect similar kinds of results for DLs.

Acknowledgment

This work was prompted by a remark of Moshe Vardi on the possible connection to $\mathcal{L}^2$. I am deeply indebted to Franz Baader for extensive comments on this work and its presentation, and I have benefited from the insightful comments of the referees, as well as discussions with W. Nutt, R. van der Meyden, M. Yannakakis, and others. M. Vardi, A. Mendelzon, P. Kolaitis, and an anonymous referee provided very useful references, for which I am grateful.

References

[1] H. Andreka, J. van Benthem and I. Nemeti, Back and forth between modal logic and classical logic, *J. Int. Gp. Pure Appl. Logic* **3** (5) (1995) 685–720.

[2] F. Baader, A formal definition for expressive power of Knowledge Representation Languages, in: *Proceedings ECAI-90*, Stockholm (1990) 53–58.

[3] F. Baader, H-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt and H. Profitlich, Terminological knowledge representation: a proposal for a terminological logic, DFKI Report, DFKI, Saarbrücken (1992).

[4] A. Borgida, Description logics in data management, *IEEE Trans. Knowledge Data Eng.* **7** (1995) 671–682.

[5] A. Borgida and R.J. Brachman, Loading data into description reasoners, in: *Proceedings ACM SIGMOD Conference on Data Management*, Washington, DC (1993) 217–226.

[6] R.J. Brachman, A structural paradigm for representing knowledge, Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, Cambridge, MA (1977).

[7] M. Buchheit, M. Jeusfeld, W. Nutt and M. Staudt, Subsumption between queries in object-oriented databases, *Inf. Syst.* **19** (1) (1994) 33–54.

[8] J. Cai, M. Fürer and N. Immerman, An optimal lower bound on the number of variables for graph identification, in: *Proceedings 30th IEEE Symposium on Foundations of Computer Science* (1989) 612–617.

[9] F.M. Donini, M. Lenzerini, D. Nardi and A. Schaerf, A hybrid system with Datalog and concept languages, in: *Trends in Artificial Intelligence*, Lecture Notes in Artificial Intelligence **549** (Springer, Berlin, 1991) 88–97.
[10] D. Gabbay, Expressive functional completeness in tense logic, in: U. Mönnich, ed., *Aspects of Philosophical Logic* (Reidel, Dordrecht, Netherlands, 1981) 91–117.
[11] N. Immerman, Upper and lower bounds for first-order expressibility, *J. Comput. Syst. Sci.* **25** (1982) 76–98.
[12] N. Immerman, Relational queries computable in polynomial time, *Inf. Control* **68** (1986) 86–104.
[13] P. Kolaitis and M. Vardi, On the expressive power of Datalog: tools and a case study, in: *Proceedings 9th ACM Symposium on Principles of Database Systems*, Nashville, TN (1990) 61–71.
[14] H.R. Lewis, *Unsolvable Classes of Quantificational Formulas* (Addison-Wesley, Reading, MA, 1979).
[15] R.M. MacGregor, A deductive pattern matcher, in: *Proceedings AAAI-87*, Seattle, WA (1987) 403–408.
[16] M. Mortimer, On languages with two variables, *Z. Math. Logik Grundl. Math.* **21** (1975) 135–140.
[17] B. Nebel, Terminological cycles: semantics and computational properties, in: J. Sowa, ed., *Principles of Semantic Networks* (Morgan Kaufmann, Los Altos, CA, 1991) 331–362.
[18] P.F. Patel-Schneider, Undecidability of subsumption in NIKL, *Artif. Intell.* **39** (1989) 263–272.
[19] P.F. Patel-Schneider and B. Swartout, Description logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort, AT&T Bell Laboratories Report, Murray Hill, NJ (1994).
[20] K. Schild, A correspondence theory for terminologic logics, in: *Proceedings IJCAI-91*, Sydney, Australia (1991).
[21] K. Schild, The role of domain-closure axioms and complete knowledge in enabling decidable and tractable reasoning in description logics, in: K.R. Apt and E. Marchiori, eds., *Deliverables for Year 3 of ESPRIT BRA Compulog 2* (1995).
[22] J.G. Schmolze and D. Israel, KL-ONE: semantics and classification, in: *Research in Knowledge Representation for NL Understanding*, Tech. Rept. 5421, BBN Laboratories (1983).
[23] A. Tarski and S. Givant, *A Formalization of Set Theory without Variables*, AMS Colloquium Publications (American Mathematical Society, Providence, RI, 1987).
[24] W.A. Woods and J.G. Schmolze, The KL-ONE family, *Comput. Math. Appl.* **23** (2–5) Special Issue on Semantic Networks in Artificial Intelligence.