D2.5.8 Integrating RDF and OWL with Other Reasoning Paradigms

Jos de Bruijn, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
Thomas Eiter, Institut für Informationssysteme 184/3 Technische Universität Wien, Austria
Fausto Giunchiglia, University of Trento, Italy
Stijn Heymans, Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria
Pascal Hitzler, Institute AIFB, Universität Karlsruhe, Germany
Tobias Matzner, Institute AIFB, Universität Karlsruhe, Germany
Jeff Z. Pan, University of Aberdeen, UK
Axel Polleres, Digital Enterprise Research Institute, Galway, Ireland
Rob Shearer, Oxford University Computing Laboratory, Oxford, UK
Hans Tompits, Institut für Informationssysteme 184/3 Technische Universität Wien, Austria
Rodney Topor, Griffith University, Australia
Kewen Wang, Griffith University, Australia
Yuting Zhao, Griffith University, Australia

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB Deliverable D2.5.8 (WP2.5)

Keyword list: ontology language, RDF, OWL, semantics, C-OWL, Prolog, rules, logic,

Document Identifier	KWEB/2007/D2.5.8/v1.0
Project	KWEB EU-IST-2004-507482
Version	1.0
Date	31 December, 2007
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer
Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas /Informatics and Telematics Institute
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
University of Manchester
University of Trento
Vrije Universiteit Amsterdam
University of Innsbruck

Changes

Version	Date	Author(s)	Changes
0.1	2007.10.15	Rob Shearer	creation
1.0	2007.12.31	Rob Shearer	Response to reviewer comments

Executive Summary

RDF and OWL together provide a family of ontology languages with syntax and semantics officially endorsed by the World Wide Web Consortium (W3C). Deliverables 2.5.1, 2.5.2, and 2.5.3 addressed rule and query languages compatible with the established RDF and OWL semantics. Deliverable 2.5.4 analyzed perceived shortcomings of the existing RDF and OWL recommendations; deliverable 2.5.5 provided a comprehensive proposal for an extension to OWL which overcomes many of the identified issues while maintaining backwards compatibility with the current language semantics, and deliverable 2.5.6 detailed a further extension which extends RDF and OWL semantics to "fuzzy" or imprecise knowledge. Deliverable 2.5.7 presented a number of different surface syntaxes for encoding knowledge bases interpreted under OWL semantics.

In this deliverable we consider the possibility that the formalisms underlying RDF and OWL family are not necessarily the most appropriate or convenient framework for knowledge representation in all applications, and that simple syntactic translation (as described in D2.5.7) is not always possible. We address how knowledge encoded with "incompatible" semantics can be integrated within a single system. In particular, we present

- a hybrid Prolog-OWL architecture which provides the power of logic programming without eliminating the open-world semantics of OWL,
- a system for combining rules with semantic web knowledge bases based on an embedding into first-order autoepistemic logic,
- an analysis of the relationship between RDF(S) and standard logic formalisms, including F-Logic, first-order logic, and description logics, and
- an architecture for the integration of multiple ontologies which preserves the independent semantics of each integrated component.

Contents

Chapter 1

Introduction

Resource Description Format (RDF) [Bec04] and the Web Ontology Language (OWL) [BvHH+04] are languages defined by the World Wide Web Constortium (W3C) as part of its Semantic Web activity. Widespread use of RDF and OWL has allowed for interoperability between independently-developed tools and applications. Further, the standardized formal semantics of these languages provides a foundation for the design of new services such as query interfaces [PFT+04a, PFT+05] and lay the groundwork for further extension with additional features for increased expressivity [PFT+04b, GHP+06, SSP06]. Even without extension, these languages have proven expressive enough to serve as the "implementation language" for a number of different modeling languages presented to users [PFT+07].

RDF and OWL are not the most expressive languages in existence, however, and tools, applications, and developer expertise grounded in different knowledge representation formalisms can be expected to persist in the marketplace indefinitely. For this reason, wide adoption of RDF and OWL in novel applications is dependent upon the ability to integrate these languages with other semantic formalisms. We explore a number of such integrations.

In Chapter 2, we integrate OWL with one of the most successful and most widely used forms of knowledge representation, namely Prolog, and present a hybrid approach which layers Prolog on top of OWL in such a way that the open-world semantics of OWL becomes directly accessible within the Prolog system.

Chapter 3 addresses integration between Semantic Web knowledge bases and rule systems. In the context of the Semantic Web, several approaches to the combination of ontologies, given in terms of theories of classical first-order logic, and rule bases have been proposed. They either cast rules into classical logic or limit the interaction between rules and ontologies. Autoepistemic logic (AEL) is an attractive formalism which allows to overcome these limitations, by serving as a uniform host language to embed ontologies and nonmonotonic logic programs into it. For the latter, so far only the propositional setting has been considered. We present several embeddings of normal and disjunctive

non-ground logic programs under the stable-model semantics into first-order AEL, and compare them in combination with classical theories, with respect to stable expansions and autoepistemic consequences. Our results reveal differences and correspondences of the embeddings and provide a useful guidance in the choice of a particular embedding for knowledge combination.

In Chapter 4 we address the relationship between the RDF(S) semantics and the semantics of standard knowledge representation formalisms such as logic programming and description logics. We consider embeddings of RDF and RDFS in logic. Using these embeddings, combined with existing results about various fragments of logic, we establish several novel complexity results. The embeddings we consider show how techniques from deductive databases and description logics can be used for reasoning with RDF(S). Finally, we consider querying RDF graphs and establish the data complexity of conjunctive querying for the various RDF entailment regimes.

Finally, in Chapter 5 we address integration between OWL ontologies with incompatible independent interpretations. Several proposals have been put forward to support distributed agent cooperation in the Semantic Web, by allowing concepts and roles in one ontology be reused in another ontology. In general, these proposals reduce the autonomy of each ontology by defining the semantics of the ontology to depend on the semantics of the other ontologies.

We propose a new framework for managing autonomy in a set of cooperating ontologies (or ontology space). In this framework, each language entity (concept/role/individual) in an ontology may have its meaning assigned either locally with respect to the semantics of its own ontology, to preserve the autonomy of the ontology, or globally with respect to the semantics of any neighbouring ontology in which it is defined, thus enabling semantic cooperation between multiple ontologies.

In this way, each ontology has a "subjective semantics" based on local interpretation and a "foreign semantics" based on semantic binding to neighbouring ontologies. We study the properties of these two semantics and describe the conditions under which entailment and satisfiability are preserved. We also introduce two reasoning mechanisms under this framework: "cautious reasoning" and "brave reasoning". Cautious reasoning is done with respect to a local ontology and its neighbours (those ontologies in which its entities are defined); brave reasoning is done with respect to the transitive closure of this relationship. This framework is independent of ontology languages. As a case study, for Description Logic $\mathcal{ALCN}$ we present two tableau-based algorithms for performing each form of reasonings and prove their correctness.

Chapter 2

Any-World Access to OWL from Prolog

2.1 Introduction

The Web Ontology Language OWL has been recommended by the W3C in 2004 for the representation of ontologies, and its usage is spreading rapidly ever since. One of the design issues for OWL has been that it is decidable and based on the open world assumption, and these two properties – which are both inherited from description logics – have served it well in the last two years.

However, with these design decisions come also some drawbacks as they limit expressiveness of OWL in ways which make working with it cumbersome at times. Even more, due to decidability of the language some things cannot be expressed at all in OWL. Efforts are therefore under way to extend OWL with more expressive features, and there is a growing body of work with proposals and studies how to do this best.

The corresponding research can roughly be classified into two different approaches. The first approach deals with extensions of OWL while adhering as much as possible to the conceptual frame of mind spanned by description logic research. The second approach is based on establishing hybrid systems which combine OWL with other established knowledge representation formalisms in such a way that either approach is encompassed in full, possibly using two different reasoning engines, but allowing for information flow between the subsystems. The work which we present in this chapter is of the hybrid kind.

The particular integration which we report on, is based on the following rationales.

- OWL has not been designed to be a stand-alone programming language. OWL ontologies should rather be viewed as declarative knowledge bases, which require programming in some other language for accessing the knowledge and further processing it. It is a natural choice to use a logic-based declarative programming language for this purpose.

- One of the most requested-for extensions of OWL is the ability to formulate rules, in some established rules language.
- It becomes more and more apparent that closed-world features are required alongside the open-world character of OWL.

Our hybrid system addresses the formulated needs by interlacing OWL with one of the most prominent and historic approaches to logic-based knowledge representation, namely with Prolog. Our system layers Prolog on top of OWL by allowing the querying of OWL ontologies via a standard OWL reasoner. A tight integration is achieved by interpreting the answers given by the OWL reasoner in an open-world fashion, and by processing this answer within Prolog in the same open-world fashion. This is achieved by means of the so-called any-world semantics due to Loyer and Straccia [LS05].

Technically speaking, the integration is achieved via a hybrid semantics for a language which incorporates calls to an OWL reasoner into standard logic programming. This hybrid semantics is based on the any-world semantics. Algorithmization and an implementation of the approach is provided by means of a transformation of logic programs under the any-world semantics into standard Prolog, in this case realised using SWI-Prolog.

Besides the aforementioned rationales for our approach, we thus arrive at a system with the following additional features.

- Modularity: The user can develop her programs based on Prolog programming and need not deal with the evaluation of OWL-based reasoning and knowledge. It is possible to offer restricted or controlled access to third party knowledge-bases without problems.
- Maturity: We incorporate the KAON2 reasoner and thus offer the performance of a state-of-the-art DL-reasoner to the logic programming world. The logic programming environment can be handled with little more than basic Prolog knowledge.
- Conformity with standards: Available OWL knowledge bases can be used directly. As we do not need one big formal system comprising both approaches, these can be used with no or only little maintenance to do.
- Bridge between ontology language paradigms: One of the most prominent alternatives to OWL for ontology representation is F-Logic [KLW95b, AL04], which can be used both as an ontology language and as a programming language. As F-Logic in its basic form is basically Prolog extended with further syntactic features, our approach can be used directly for realising a hybrid OWL/F-Logic system.

The structure of the chapter is as follows. In Section 2.2, we review the basic facts we need about the any-world semantics and about OWL in order to make this chapter relatively self-contained. In Section 2.3, we prove a theorem which gives the formal rationale for our algorithmisation. In Section 2.4 we discuss the implemented system which we

provide. In Section 2.5 we give an extended example which shows the possibilities of our approach. In Section 2.6 we discuss related work, and we conclude in Section 2.7.

2.2 Preliminaries

2.2.1 The Any-World semantics

We review the any-world semantics due to Loyer and Straccia [LS05] in some details as it is crucial for understanding our work.

Bilattices

The any-world semantics is based on a truth-space which is a so-called bilattice [Gin88]. This is a potent mathematical structure which particularly provides two partial orders, which permit to represent (logical) truth and the knowledge contained in these truth-values separately.

Formally, a *lattice* $\langle L, \leq \rangle$ is a non-empty set L with a partial order $\leq$, where each subset of L containing two elements has a supremum and infimum regarding $\leq$ (also known as *meet* and *join*). It is a *complete lattice* iff every subset has supremum and infimum regarding $\leq$. We write $x < y$ for $x \leq y$ and $x \neq y$ where $x, y \in L$.

A *bilattice* $\langle B, \leq_t, \leq_k \rangle$ is a non-empty set B with two partial orders, the *truth-order* $\leq_t$ and the *knowledge-order* $\leq_k$, both of which give B the structure of a complete lattice. Due to completeness, the greatest and least element regarding either of the orders always exists and is unique [Gin88]. The greatest element regarding $\leq_t$ is denoted `true`, the least element `false`. Regarding $\leq_k$, the greatest element is $\top$, the least $\bot$. Meet and join under $\leq_t$ which are denoted $\wedge$ and $\vee$, correspond to the well-known two-valued conjunction and disjunction regarding the values `true` and `false`. Under $\leq_k$ meet and join are denoted $\otimes$ and $\oplus$, where $x \otimes y$ extracts the maximum knowledge that is expressed both in x and y whereas $x \oplus y$ unites the knowledge of x and y. Our approach is particularly based on the smallest non-trivial bilattice known as $\mathcal{FOUR}$ [Bel77] which is depicted in Figure 2.2.1. Indeed, although bilattice-based semantics is generally formulated for arbitrary bilattices, $\mathcal{FOUR}$ is currently the only such lattice of practical relevance, and will entirely suffice for our purposes.

An operator $\bullet$ on a lattice is called *monotone* when $x_1 \leq y_1$ and $x_2 \leq y_2$ implies $x_1 \bullet x_2 \leq y_1 \bullet y_2$. We suppose for all bilattices here considered that all of the operators $\wedge, \vee, \otimes, \oplus$ are monotone w.r.t. both the knowledge- and the truth-order; this is called the *infinitary interlacing condition*. We furthermore assume that all bilattices are *infinitary distributive* i.e. that all distributive laws connecting the aforementioned lattice operators hold. Finally, we assume that all lattices have a *negation*, which is an operator denoted $\neg$ that inverses the truth order, does not inflict the knowledge order and satisfies $\neg\neg x =$

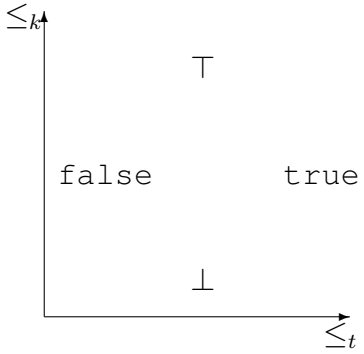


Figure 2.1: The bilattice $\mathcal{FOUR}$

x. These assumptions are standard and generally known to be unproblematic in a logic programming context.

Logic programs

We extend logic programs from the common case and include not only connectives for disjunction, conjunction and negation but for all the operators of a bilattice: $\wedge, \vee, \otimes, \oplus$ and $\neg$. So the knowledge order and its operators are not only a tool of analysis and semantics as used for example in [Fit93] but can be used explicitly to determine how the program treats information from the perspective of knowledge. A logic program is based on a set $\mathcal{P}$ of predicates, $\mathcal{V}$ of variables, $\mathcal{C}$ of constants and $\mathcal{F}$ of functions. A *term* is either an element of $\mathcal{V}$ or $\mathcal{C}$ or of the form $f(t_1, \ldots, t_n)$ where $f \in \mathcal{F}$ and all $t_1, \ldots, t_n$ are terms. The *ground terms* forming the *Herbrand universe* are all the terms that can be built from elements of $\mathcal{C}$ and $\mathcal{F}$. An *atom* is of the form $p(t_1, \ldots, t_m)$ where $p \in \mathcal{P}$ and all $t_1, \ldots, t_m$ are terms. The *ground atoms* forming the *Herbrand base* are all the atoms that can be built from the Herbrand universe. A *literal* is of the form A or $\neg A$ where A is an atom. Furthermore we allow the elements of the bilattice as literals. A *formula* is either any literal, or of the form $\varphi_1 \bullet \varphi_2$ where φ_1 and φ_2 are formulas and $\bullet$ is one of the four lattice operators $\wedge, \vee, \otimes, \oplus$, or one of the expressions $\forall\varphi$ respectively $\exists\varphi$ where φ is a formula. A *rule* is of the form $p(x_1, \ldots, x_m) \leftarrow \varphi(x_1, \ldots, x_m)$ where $p \in \mathcal{P}$, $x_1, \ldots, x_m \in \mathcal{V}$ and φ is a formula. We call p the *head* and φ the *body* of the rule. We suppose that the free variables in φ are among $\{x_1, \ldots, x_m\}$ and are universally quantified. A *logic program* P is a finite set of rules. Not allowing terms in the heads of rules is not a restriction, e.g. the rules (taken from [LS05]):

$$p(s(x)) \leftarrow p(x)$$
$$p(0) \leftarrow \texttt{true}$$

can be rewritten (using a predicate eq defining equality) as:

$$p(y) \leftarrow \exists x(eq(y, s(x)) \wedge p(x))$$
$$p(x) \leftarrow eq(x, 0)$$

With $ground(P)$ we denote all ground instances of members of P over the Herbrand universe.

Interpretations of logic programs

Let B be a bilattice. An *interpretation* of a logic program on B is a mapping I from ground atoms to members of B. It is extended to formulas as follows: $I(b) = b$ where $b \in B$; $I(\varphi_1 \bullet \varphi_2) = I(\varphi_1) \bullet I(\varphi_2)$ where φ_1, φ_2 are formulas and $\bullet$ is one of the operators $\wedge, \vee, \otimes, \oplus$; $I(\neg\varphi) = \neg I(\varphi)$; $I(\exists x\varphi(x)) = \bigvee\{I(\varphi(t))|t \text{ is a ground term}\}$ and finally $I(\forall x\varphi(x)) = \bigwedge\{I(\varphi(t))|t \text{ is a ground term}\}$. The partial orders of the bilattice are point-wise extended to interpretations: $I_1 \leq_t I_2$ iff $I_1(A) \leq_t I_2(A)$ for all ground atoms A. The extension for $\leq_k$ is analogous. Given two interpretations I_1, I_2 we define $(I_1 \bullet I_2)(\varphi) = I_1(\varphi) \bullet I_2(\varphi)$ where $\bullet$ is a lattice operator and φ a formula. Thus the space of all possible interpretations on a bilattice constitutes an infinitary interlaced and distributive bilattice as well. An interpretation I is a *model* of a logic program P iff $I(A) = I(\varphi)$ for all rules $A \leftarrow \varphi$ in P.

Semantics

The semantics is defined via the fixed point of a monotone operator similarly to the well known Kripke-Kleene [Fit85, Fit90] or well-founded [vRS91] semantics. In fact the any-world semantics used here is a generalization of the well-founded semantics. The central idea of the any-world semantics is to overcome the limitations of both the open and the closed world as default assumption. Instead an arbitrary interpretation H called the *hypothesis* is used as default assumption, i.e. the value $H(A)$ is the default value for the atom A. From this point of view, the open world assumption corresponds to the hypothesis $H(A) = \bot$ for all atoms A, we call this hypothesis $H_\bot$. The closed world assumption can be modelled by $H(A) = \texttt{false}$ for all atoms A, this hypothesis is denoted $H_\texttt{f}$. Now the information of the program is combined with knowledge extracted from the hypothesis used. To gather information from the program we use the well known *immediate consequence operator* $\Phi_P(I)(A) = I(\varphi)$ where $A \leftarrow \varphi$ is a rule in P. Now we want to augment the interpretation I with the information from a hypothesis H. This is done similarly to the use of the unfounded set in the well-founded semantics. From a knowledge point of view, the unfounded set is the amount of information contributed to the semantics by the closed world assumption. This concept now is generalized to arbitrary hypotheses H. We usually cannot use all the information of H. Instead we want to extract the maximum knowledge of H, expressed as an interpretation J, so that the assumed knowledge J is entailed by the program w.r.t. the augmented interpretation $I \oplus J$, i.e. we want to make sure that $J(A) \leq_k \Phi_P(I \oplus J)(A)$. This idea is modelled using the so called *safe interpretations*. An interpretation J is *safe* w.r.t. a logic program P, an interpretation I and a hypothesis H if $J \leq_k H$ and $J \leq_k \Phi_P(I \oplus J)$. The *support* provided by H to P and I is the greatest (on the knowledge order) safe interpretation w.r.t. P, I and H. It is

denoted $s_P^H(I)$. Note that this particularily entails that the support is always smaller than the hypothesis.

In order to simplify the treatment of logic programs using fixed-point semantics, we introduce the transformed program P^*. Given a logic program P and a hypothesis H the program P^* contains the following rules:

- $A \leftarrow \varphi_1 \vee \cdots \vee \varphi_n$ if $A \leftarrow \varphi_1, \ldots, A \leftarrow \varphi_n$ are all rules in $ground(P)$ with the head A.
- $A \leftarrow H(A)$ if A is not the head of any rule in P.

The second part enforces that for any atom that is not assigned a truth-value by a rule in the program, it is given its value according to the default assumption, i.e. the hypothesis.

Now we define the operator $\tilde{\Pi}_P^H(I) = \Phi_P(I) \oplus s_P^H(I)$ which works on P^*. The fixed points of $\tilde{\Pi}_P^H$ are called the *H-founded models* of P. In [LS05] it is shown that the support operator $s_P^H(I)$ is monotone in I and H w.r.t. the knowledge order. Furthermore also Φ_P is monotone w.r.t. the knowledge order [Fit93]. By the infinitary interlacing condition, motonicity of $\tilde{\Pi}_P^H$ is guaranteed. So by the well known Knaster-Tarski theorem [Tar55], there is always a (unique) least H-founded model for any logic program, which can be obtained as the least upper bound of the transfinite sequence $(\tilde{\Pi}_P^H \uparrow \alpha)_\alpha$, where α ranges over ordinals, $\tilde{\Pi}_P^H \uparrow 0$ is the least interpretation, $\tilde{\Pi}_P^H \uparrow \alpha + 1 = \tilde{\Pi}_P^H(\tilde{\Pi}_P^H \uparrow \alpha)$ for all α, and $\tilde{\Pi}_P^H \uparrow \alpha = \sup\{\tilde{\Pi}_P^H \uparrow \beta : \beta < \alpha\}$ for limit ordinals α.

The key feature of the any-world semantics is the flexibility of the default assumption. Particularly using a hypothesis that maps ground atoms to the set $\{\texttt{false}, \bot\}$ it is possible to mix closed- and open-world based information, whereon our hybrid semantics relies. It also includes several well known semantics. Using $H_\texttt{f}$, the H-founded model is the well-founded model [LS05]. Let H_{KK} be the interpretation that maps all atoms that are the head of a rule in a given program P to $\bot$, all the other atoms to `false`. Using this hypothesis, the H-founded model of P is its Kripke-Kleene-model [LS05]. This reflects that the Kripke-Kleene semantics uses only the immediate-consequence operator Φ_P and consequently the support part in $\tilde{\Pi}_P^H$ is reduced to $\bot$ by the assignment of $\bot$ to all rule heads. (Recall that the support is always smaller than the hypothesis on the knowledge order). However the Kripke-Kleene semantics is based on the closed world assumption. This is manifested in the hypothesis mapping the other atoms to `false`. The fact that the hypothesis affects only those atoms will be used later for the hybrid semantics of our system.

2.2.2 Description logics

The description logics part of our hybrid system uses the KAON2 OWL DL reasoner [Mot06].[1] Our approach, however, is independent of the specific reasoner used, and can

[1] See also `http://kaon2.semanticweb.org`

indeed be used with any reasoning system based on the open world assumption. OWL DL is based on the description logic $\mathcal{SHOIN}$(D) [HST99], but for the purpose of our exhibition we will not need to give many details about OWL DL. It shall suffice to recall that OWL DL allows to specify axioms describing the subsumption relation between complex concepts C and D, written $C \sqsubseteq D$. The (complex) concepts themselves are composed by means of primitive (or atomic) concepts, logical and other connectives, individuals which correspond to logical constants, and roles which describe relationships between individuals. It is also possible to specify that some individual a belongs to a class C, written $C(a)$, or to explicitly state that two individuals a and b are connected by a role R, written $R(a, b)$. The special concepts $\top$ and $\bot$, respectively, are defined as containing all individuals respectively no individual. OWL DL is given an open-world semantics e.g. by mapping it into first-order logic with equality.

Given a set of OWL DL axioms, called an *ontology*, it is possible to derive logical consequences from it by means of well-established algorithms. The most basic inference tasks are

- checking whether an ontology is satisfiable (i.e. logically consistent),
- checking whether a concept C subsumes a concept D, i.e. whether $C \sqsubseteq D$ is a logical consequence,
- checking whether a concept C is satisfiable, i.e. whether there is a model of the knowledge base in which the extension of C is non-empty, and
- checking whether an individual a is contained in a concept C, i.e. whether $C(a)$ is a logical consequence.

2.3 A program transformation for algorithmising the any-world semantics

We provide an extension for Prolog which implements an any-world logic based on $\mathcal{FOUR}$ and hypotheses that map into $\{\texttt{false}, \bot\}$. The implementation is based on Theorem 2 below, which acts as a bridge between the any-world semantics and Prolog.

Before we provide the theorem, let us define the specific type of hypotheses which we need for our purposes. Recall that the hypothesis $H_{\texttt{f}}$ corresponds to the closed world assumption, while $H_{\bot}$ can be interpreted as an open world semantics. Consequently, the hypotheses of interest are a mix between these two.

Definition 1 *Given a logic program P we define the set of hypotheses* KKS *to be the set of all interpretations that map an atom A to $\bot$ when A is the head of a rule in P, and to either $\bot$ or* `false` *otherwise.*

Note that H_{KK} is in KKS for all programs P.

Theorem 2 *Given a logic program P and a hypothesis $H_{\mathrm{KKS}} \in \mathrm{KKS}$, there exists a program transformation $T^{H_{\mathrm{KKS}}}$ such that the H-founded model of P under H_{KKS} is the same as the H-founded model of $T^{H_{\mathrm{KKS}}}(P)$ under H_{KK}.*

The proof is based on the possibility to add the default assumption chosen as rules of the form $A \leftarrow H(A)$ to the program, such that the resulting program does not have any atoms that are not head of a rule. When evaluated under H_{KK}, accordingly the default assumption `false` is not used for any atom. The assumption $\bot$ for atoms that are heads of a rule, i.e. all atoms, is overridden by the rule $A \leftarrow$ `false` should it exist, as `false` $\oplus \bot =$ `false`. We first prove the following:

Lemma 3 *Let P be a logic program and H_{KKS} a hypothesis from* KKS*. Then $s_P^{H_{KKS}}(I) = H_{KKS}$ for any interpretation I.*

Proof 1 *For the following proof we write $s_P^{H_{KKS}}(A)$ for $s_P^{H_{KKS}}(I)(A)$ as the choice of interpretation is without effect. For an atom A that is not the head of any rule there are two possibilities: (1) If $H_{KKS}(A) = \bot$, then the fact that the support is always smaller than the hypothesis on the knowledge-order requires $s_P^{H_{KKS}}(A) = \bot$. (2) If $H_{KKS}(A) =$ `false`, then the rule $A \leftarrow$ `false` is in P^*. As the support is a safe interpretation we require $s_P^{H_{KKS}}(A) \leq_k \Phi_P(I \oplus s_P^{H_{KKS}})(A)$. This now becomes $s_P^{H_{KKS}}(A) \leq_k I($`false`$) \oplus s_P^{H_{KKS}}($`false`$) =$ `false`. As the support is the largest safe interpretation on the knowledge-order we have $s_P^{H_{KKS}}(A) =$ `false`. Consider now an atom A that is head of a rule in P. Using again that the support is a safe interpretation and thus smaller (in the knowledge order) than the hypothesis, we have that $s_P^{H_{KKS}}(A) = \bot = H_{KKS}$.* □

Now we are ready to prove the theorem:

Proof 2 (of Theorem 2) *We use the notation of the theorem. Furthermore we let $P' = T^{H_{KKS}}(P)$. We now show that the operators $\tilde{\Pi}_P^{H_{KKS}}$ and $\tilde{\Pi}_{P'}^{H_{KK}}$ have the same result on every step of their iteration. As the operator $\tilde{\Pi}$ is defined on P^*, for the evaluation of $\tilde{\Pi}_P^{H_{KKS}}$ P^* is constructed from P under the hypothesis H_{KKS}. In this process the same rules are added as when applying $T^{H_{KKS}}$ to P by definition of T. So P' and P^* constructed under the hypothesis H_{KKS} are identical. For the evaluation of $\tilde{\Pi}_{P'}^{H_{KK}}$ the program P'^* is constructed under the hypothesis H_{KK}. Since in P' all atoms are head of rule, the hypothesis H_{KK} has no influence on P'^*. So we have $P^* = P'^*$, thus $\tilde{\Pi}_P^{H_{KKS}}$ and $\tilde{\Pi}_{P'}^{H_{KK}}$ work on the same program.*

Now consider an arbitrary iteration step α:

$$(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(\varphi) \oplus s_{P'}^{H_{KK}}(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(A).$$

We have $H_{KK}(A) = \bot$ for all atoms A in P' as all atoms are the head of a rule after the transformation. By Lemma 3 the support is equal to the hypothesis (note that H_{KK} is in

KKS*) and so the remaining formula is*

$$(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(\varphi). \tag{2.1}$$

Consider now the operator

$$(\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha)(\varphi) \oplus s_P^{H_{KKS}}(\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha)(A).$$

Again by Lemma 3 we have that $H^{H_{KKS}}(A) = s_P^{H_{KKS}}(\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha)(A)$*. For atoms that are head of a rule in P we obtain*

$$(\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha)(\varphi). \tag{2.2}$$

As P *and* P' *contain the same rules, the operators in (2.1) and (2.2) yield the same results. For atoms that are not the head of a rule we know that either* $H_{KKS}(A) = \bot$ *or* $H_{KKS}(A) = \texttt{false}$*. The following argument is analogous for both cases, we consider the latter. If* $H_{KKS}(A) = \texttt{false}$*, then there is a rule* $A \leftarrow \texttt{false}$ *in* P^* *and accordingly also in* P'*. So we have* $(\tilde{\Pi}_{P}^{H_{KKS}} \uparrow \alpha)(\varphi) = s_P^{H_{KKS}}(\tilde{\Pi}_{P}^{H^{KKS}} \uparrow \alpha)(A) = \texttt{false}$ *as well as* $(\tilde{\Pi}_{P'}^{H^{KK}} \uparrow \alpha)(\varphi) = \texttt{false}$*. So both operators give the same result.* □

So we have the possibility to deal with hypotheses mixing open- and closed-world assumption while we only need to compute the Kripke-Kleene semantics.

2.4 Implementation

In order to arrive at its least fixed point, i.e. at the Kripke-Kleene semantics, Φ_P may need as many as Church-Kleene ω_1 steps. Indeed the Kripke-Kleene semantics is Π_1^1-complete [Fit02] and thus not even semi-decidable. This means that a sound and complete implementation of the Kripke-Kleene semantics cannot be provided for theoretical reasons.

However, the Kripke-Kleene semantics was originally conceived as a declarative semantics which captures the essence of the Prolog procedural semantics, and indeed they are strongly related, as shown e.g. in [Kun87]. For practical purposes, it thus suffices to view Prolog as an approximate implementation of the Kripke-Kleene semantics.

We therefore provide a library that permits using the logic $\mathcal{FOUR}$ with all corresponding lattice operations $\oplus, \otimes, \wedge, \vee$ and $\neg$ in Prolog. The user can write programs in a Prolog-like syntax, which is then compiled to SWI-Prolog[2] such that each predicate is augmented with an additional parameter, which carries the truth-value. A predicate $\mathsf{p}(\mathsf{t}_1, \ldots, \mathsf{t}_n, \mathsf{TV})$ is then deducible in Prolog if $\mathsf{p}(\mathsf{t}_1, \ldots, \mathsf{t}_n)$ has the truth-value TV.

Within this framework, we offer special atoms, so called *DL-atoms*, that are not evaluated according to the logic programming semantics but by querying the DL-reasoner

[2] `http://www.swi-prolog.org`

KAON2. They have the form $dl_q(\mathrm{p}_q)$ where q is a query and p_q the respective vector of parameters. The queries we offer are *subsumes, unsatisfiable* and *disjoint* regarding concepts and *has_role* regarding roles.

Usually queries to a DL-reasoner have two possible answers: the queried information is either demonstrable or not. However, if the answer is negative, then two cases are possible: Either the negation of the query is demonstrable, or the negation of the query is also not demonstrable. In the first case, the refutation of the query is much stronger than in the second.

In order to give an example, consider the knowledge base specified by the following axioms.

$$\begin{aligned} unicorn &\sqsubseteq appears\_in\_novels \\ horned\_animal &\sqsubseteq animal \end{aligned}$$

When queried whether $unicorn \sqsubseteq horned\_animal$ holds, the reasoner responds with `No`, which is entirely appropriate as the knowledge base does not allow to derive any knowledge about the relationship between $unicorn$ and $horned\_animal$, i.e. the relationship $unicorn \sqsubseteq horned\_animal$ can neither be confirmed nor refuted.

Consider now the situation that the knowledge base contains the following additional axioms, where the second describes the assertion that the concepts $unicorn$ and $phantasy\_animal$ are extensionally disjoint.

$$\begin{aligned} unicorn &\sqsubseteq phantasy\_animal \\ animal \sqcap phantasy\_animal &\sqsubseteq \bot \end{aligned}$$

When now queried whether $unicorn \sqsubseteq horned\_animal$ holds, the reasoner again responds with `No`, which is entirely appropriate as the knowledge base implies that $unicorn$ and $horned\_animal$ are in fact extensionally disjoint. The situation compared to the first situation, however, is very different: The first knowledge base did not specify anything about the relation between $unicorn$ and $horned\_animal$, while the second knowledge base strongly refutes the subsumption relation.

Our framework provides the means to distinguish between these situations by means of a different choice of truth values. In the first situation, the resulting truth value must be $\bot$, while in the second it must be `false`. Technically, we realise this in such a way that each query to KAON2 results in two calls to the reasoner allowing to retrieve more detailed information. For the atom $dl_{subsumes}(C, D)$, the first query to the reasoner asks for $C \sqsubseteq D$. Given a positive answer, we know that this is demonstrable, thus the DL-atom is evaluated as `true`. When the answer is negative, there are, however, two cases possible: $C \sqsubseteq D$ might be satisfiable, but not formally implied by the knowledge base. In this case the DL-atom should have the value $\bot$ i.e. unknown. On the other hand it is possible that the information in the knowledge-base makes $C \sqsubseteq D$ impossible. Then the DL-atom should be assigned `false`. This is done by the second query, which asks

whether $KB \cup \{C \sqsubseteq D\}$ is satisfiable, where KB is the knowledge-base. We summarize the query in the following table.

result of the query: $C \sqsubseteq D$	result of the query: Is $KB \cup \{C \sqsubseteq D\}$ satisfiable?	value of $dl_{subsumes}(C, D)$
yes	–	`true`
no	yes	$\bot$
no	no	`false`

Note that the queries are executed consecutively, i.e. the second query is only performed if the first returned false.

A useful perspective on this is the following: $C \sqsubseteq D$ results in `true` if it holds in *all* models of the knowledge base. It results in `false` if it holds in *none* of the models of the knowledge base. And it results in $\bot$ if it holds in some, but not all, models of the knowledge base.

The question of the satisfiability of a concept is reducible to subsumption: a concept C is satisfiable iff $C \sqsubseteq \bot$ does not hold, i.e. if there is *some* model in which the extension of C is non-empty. This situation is best understood by considering *unsatisfiabilitiy* of a concept instead of satisfiability, as this allows us to use exactly the argumentation used above: A concept is unsatisfiable if it is extensionally empty in *all* models of the knowledge base. Similarly to the case of subsumption, we arrive at the execution detailed in the following table.

result of the query: $C \sqsubseteq \bot$	result of the query: Is $KB \cup \{C \sqsubseteq \bot\}$ satisfiable?	value of $dl_{unsatisfiable}(C)$
yes	–	`true`
no	yes	$\bot$
no	no	`false`

Querying for extensional disjointness of concepts is treated similarly, by reducing it to subsumption: two concepts C and D are disjoint iff $C \sqsubseteq \neg D$.

The query whether $C(a)$ holds can be resolved as in the following table. Note that $C(a)$ holds if it is true in *all* models.

result of the query: $C(a)$	result of the query: $\neg C(a)$	value of $dl_{member}(C, a)$
yes	–	`true`
no	yes	`false`
no	no	$\bot$

The query $dl_{has\_role}(I_1, R, I_2)$ provides information whether two individuals I_1 and I_2 are connected via a role R. When $\langle I_1, I_2 \rangle \in R$ then the DL-atom is `true`. To evaluate

the other truth-values, we have to restrict ourselves to the known individuals, as it is not possible in OWL to ask for negated roles [ELST04c]. When querying whether two individuals are connected via a role, it is a sensible assumption that this might be possible, i.e. that $\langle I_1, X\rangle \in R$ or $\langle X, I_2\rangle \in R$. So we assign the value `false` to queries when there exists either an $X \neq I_2$ with $\langle I_1, X\rangle \in R$ or an $Y \neq I_2$ with $\langle Y, I_2\rangle \in R$ but $\langle I_1, I_2\rangle \notin R$. All other pairs of individuals get the value $\bot$.

To integrate these DL-atoms flawlessly with the semantics of our logic programming environment which is based on fixed points, we need to guarantee that the values of the DL-atoms are monotone w.r.t. the knowledge order. For now, we assume that the knowledge-base is static, i.e. it cannot change during the program evaluation. Then the evaluation of the DL-atoms always yields the same result, and thus, trivially, is monotone.

The implemented system, called PrOWLog, is available for download from `http://logic.aifb.uni-karlsruhe.de/wiki/PrOWLog`.

2.5 An Example

We exemplify our approach by extending an example given in [LS05], formalising a judge's decision process, as given by the following rules.

$$\begin{aligned} is\_suspect &\leftarrow has\_motive \vee has\_witness \\ is\_cleared &\leftarrow \neg contradict\_alibi \wedge has\_alibi \\ charge &\leftarrow is\_suspect \oplus \neg is\_cleared \end{aligned}$$

The judge collects information suggesting that a person is suspect as well as information that indicates that the person is cleared. To support suspicion he collects information about the existence of a motive or a witness (first line). To enforce innocence the judge considers an alibi, but only if this is not contradicted by the defendant's testimony (second line). Finally he combines this information (third line). Assume now that the only information the judge has about some person is $has\_witness \leftarrow \texttt{false}$. Only relying on this, the suspect shouldn't be charged. Based on the closed-world assumption we get $has\_motive = \texttt{false}$ and thus $is\_suspect = \texttt{false}$. As $has\_alibi = \texttt{false}$, we obtain $is\_cleared = \texttt{false}$. So when evaluating $charge$ the information is contradictory and $charge$ gets the value $\top$.

Using the open-world assumption, giving all atoms the default value $\bot$, we get $is\_suspect = \bot$ because $has\_motive = \bot$ and $\texttt{false} \vee \bot = \bot$. Since we know nothing about has_alibi and $contradict\_alibi$, the default assumption is used again and we get $is\_cleared = \bot$ and finally $charge = \bot$. So neither of the two established assumptions work in a satisfactory way. Consider now the mixed hypothesis H_m defined as follows: $H_m(has\_witness) = \texttt{false}$, $H_m(has\_motive) = \texttt{false}$, $H_m(has\_alibi) = \bot$, $H_m(contradict\_alibi) = \bot$.

Then, like under the closed-world assumption, $is\_suspect$ is false. The contradiction we encountered, however, does not exist any more as $is\_cleared = \bot$ which reflects that the information is not sufficient to make a decision. Consequently $charge = \texttt{false}$. This illustrates that the first line of the program is devised according to the closed-world assumption. The second line however is based on a different idea: For $is\_cleared$ to become $\texttt{false}$, $has\_alibi = \texttt{false}$ is already sufficient. So the meaning of $has\_alibi = \texttt{false}$ is that it has been proven that nobody can provide an alibi for the defendant. Then we need also the possibility to model the fact, that just no alibi is known, which corresponds to $has\_alibi = \bot$, and which should be the default case. So the second line is conceived with an open-world setting in mind. H-founded models enable the use of such programs despite the different approaches involved.

To complete the example, it could be assumed that the judge draws his knowledge from an OWL DL knowledge base, by means of the following rules which query a knowledge base about a person Ted who is under investigation.

$$\begin{aligned} has\_motive &\leftarrow dl_{member}(dl\_has\_motive, Ted) \\ has\_witness &\leftarrow dl_{member}(dl\_has\_witness, Ted) \\ has\_alibi &\leftarrow dl_{member}(dl\_has\_alibi, Ted) \\ contradict\_alibi &\leftarrow dl_{member}(dl\_contradict\_alibi, Ted) \end{aligned}$$

By means of our hybrid semantics, the system will respond with the desired answer.

2.6 Related work

Our approach using DL-atoms to link rule-based and ontology-based reasoning is inspired by the approach of Eiter et. al. presented in [ELST04a], where *extended logic programs* and the *answer set semantics* [GL91a] are modified to incorporate DL-atoms to query external reasoners. This approach permits the flow of information in both directions from the DL-enhanced program to the reasoner and vice versa. Extended logic programs make use of two negation operators, distinguishing explicitly negation as failure and classical negation. The any-world-semantics permits to manage this naturally, giving the negation operator of the lattice different meanings respective to the default assumption of the negated expression. In [EIST06] Eiter et. al. generalize their approach to so called HEX-Programs, where the DL-atoms are replaced by atoms permitting to access a variety of different external sources, not only DL-reasoners. To accomplish this, the rule syntax and the answer set semantics are extended. The evaluation of these programs is made possible by a splitting algorithm based on the dependency structure in the program. Also based on DL-atoms, our approach was developed from another perspective. Given the elegant yet expressive any-world semantics and the ease of use of the hypotheses in KKS, we provide a logic programming environment with access to description logics, while remaining close to Prolog programming. We emphasize the use of the particular kind of information

that can be drawn from DL-reasoners as an open-world based system with an intuitive semantics.

Motik and Rosati present in [MR06] an approach for a system combining rules and DL into one formalism. Based on MKNF [Lif91] they join a decidable FOL fragment with logic programming rules. The modality operators in their so called *hybrid MKNF knowledge bases* allow to formulate rules to enforce closed world reasoning while maintaining the open world assumption for the DL-part. Their system also subsumes Rosati's approach in [Ros06a]. There and more detailed in [Ros05] he discusses the relation of open and closed semantics in these hybrid systems.

2.7 Conclusions and Further Work

We have presented the hybrid reasoning system PrOWLog, which allows to combine OWL DL with Prolog in such a way that the open-world semantics of OWL DL can be captured within the Prolog system. To the best of our knowledge, this is the first work which integrates a logic programming language and OWL in such a way.

We perceive basically two lines of further research to follow up on our results. On the one hand, studies remain to be done which show that the approach is useful in practice. We believe that in particular an integration with F-Logic reasoners is worth investigating, as F-Logic and OWL are two complementary ontology paradigms, which are both used in practice. On the other hand, it remains to be investigated whether the integration of Prolog and OWL can be strengthened by weakening the layering, i.e. by allowing some flow of information back to the OWL knowledge base, perhaps in a way similar to [EIST06].

Chapter 3

Embedding Non-Ground Logic Programs into Autoepistemic Logic for Knowledge-Base Combination

3.1 Introduction

In the context of the ongoing discussion around combinations of rules and ontologies for the Semantic Web, there have been several proposals for integrating classical knowledge bases (ontologies) and rule bases (logic programs). Generally speaking, all these approaches try to define a reasonable semantics for a combined knowledge base consisting of a classical component and a rules component.

Two trends are currently observable. On the one hand, approaches such as SWRL [HPSBT05] extend the ontology with Horn formulas in a classical framework. This approach is straightforward, but prohibits nonmonotonic rules. On the other hand, existing approaches which do allow nonmonotonic rules either (a) distinguish between "classical" and "rules" predicates and limit the domain of interpretation (e.g., [Ros06b]) or (b) restrict the interaction to ground entailment (e.g., [ELST04b]). The main distinction between these approaches is the type of interaction between the classical knowledge base on the one hand and the rule base on the other (cf. [BEPT06] for an examination of this issue).

As for combination, a classical theory and a logic program should be viewed as complementary descriptions of the same domain. Therefore, a syntactic separation between predicates defined in these two components should not be enforced. Furthermore, it is desirable to neither restrict the interaction between the classical and the rules components nor impose any syntactic or semantic restrictions on the individual components. That is, the classical component may be an arbitrary theory Φ of some first-order language with equality, and the rules component may be an arbitrary non-ground normal or disjunctive

logic program P, interpreted using, e.g., the common stable-model semantics [GL88].

The goal is a combined theory, $\iota(\Phi, P)$, in a uniform logical formalism. Naturally, this theory should amount to Φ if P is empty, and to P if Φ is empty. Therefore, such a combination must provide faithful embeddings $\sigma(\Phi)$ and $\tau(P)$ of Φ and P, respectively, in this formalism, given by $\sigma(\Phi) = \iota(\Phi, \emptyset)$ and $\tau(P) = \iota(\emptyset, P)$, respectively. In turn, knowledge combination may be carried out on top of such embeddings $\sigma(\cdot)$ and $\tau(\cdot)$, where in the simplest case one may choose $\iota(\Phi, P) = \sigma(\Phi) \cup \tau(P)$.

This raises the questions (a) which uniform formalism is suitable and (b) which embeddings are suitable and, furthermore, how do embeddings relate to each other and how do they behave under knowledge combination?

Autoepistemic logic (AEL) [Moo85], which extends classical logic with a modal belief operator, is an attractive candidate for a uniform formalism. In fact, embedding a classical theory in AEL is trivial, and several embeddings of logic programs in AEL have been described [GL88, MT93, LS93, Che93, Prz91]. However, all these embeddings have been developed for the propositional case only, whereas we need to deal with non-ground theories and programs. This requires us to consider *first-order autoepistemic logic* (FO-AEL) [Kon91, KR02, LL00], and non-ground versions of these embeddings. Our main contributions are as follows.

We define several embeddings of non-ground logic programs into FO-AEL, taking into account subtle issues of quantification in FO-AEL. We show that these embeddings are faithful in the sense that the stable models of the program and the sets of objective ground atoms in the stable expansions of the embeddings are in a one-to-one correspondence. However, the embeddings behave differently on formulas beyond ground atoms, and when combined with classical theories, even when considering propositional formulas.

Motivated by these differences, we compare the embeddings along two dimensions:

1. We determine correspondences between the stable expansions of different possible embeddings, with respect to various classes of formulas, and present inclusion relations between the sets of autoepistemic consequences of the embeddings.

2. We determine correspondences between stable expansions for combinations with theories from different fragments of classical logic which are important in ontology representation.

Compared to other well-known nonmonotonic formalisms like Reiter's default logic, FO-AEL offers a uniform language in which (nonmonotonic) rules themselves can be expressed at the object level. This conforms with the idea of treating an ontology and a logic program together as a unified theory.

Arguably, none of the embeddings can a priori be considered to be superior to the others. Our results give useful insight into the properties of the different embeddings,

both on its own right and for knowledge combination. They provide a helpful guidance for the selection of an embedding for a particular scenario.

The chapter is further structured as follows. We review the definitions of first-order logic and logic programs in Section 3.2. We proceed to describe first-order autoepistemic logic (FO-AEL) and present a novel characterization of stable expansions for certain kinds of theories in Section 3.3. The embeddings of normal and disjunctive logic programs and our results about the faithfulness of the embeddings are described in Section 3.4. We investigate the relationships between these embeddings themselves, and under combination with first-order theories, in Sections 3.5 and 3.6. We conclude with related and future work in Section 3.7.

3.2 Preliminaries

First-Order Logic A first-order (FO) language $\mathcal{L}$ consists of all formulas over a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, where $\mathcal{F}$ and $\mathcal{P}$ are countable sets of *function* and *predicate symbols*, respectively. Function symbols with arity 0 are called *constants*. $\mathcal{V}$ is a countably infinite set of *variable symbols*. Terms and atomic formulas (atoms) are constructed as usual for first-order logic with equality. Ground terms are also called *names*; $\mathcal{N}_\Sigma$ is the set of names of a given signature Σ. Complex formulas are constructed as usual using the symbols $\neg$, $\wedge$, $\vee$, $\supset$, $\exists$, $\forall$,(, and). A sentence is a formula with no free variables. The universal closure of a formula ϕ is denoted by $\forall\phi$. $\mathcal{L}_g$ is the restriction of $\mathcal{L}$ to ground formulas; $\mathcal{L}_{ga}$ is the restriction of $\mathcal{L}_g$ to atomic formulas. An *FO theory* $\Phi \subseteq \mathcal{L}$ is a set of sentences.

An *interpretation* of a language $\mathcal{L}$ is a tuple $w = \langle U, \cdot^I \rangle$, where U is a nonempty set, called the *domain*, and $\cdot^I$ is a mapping which assigns a function $f^I : U^n \to U$ to every n-ary function symbol $f \in \mathcal{F}$ and a relation $p^I \subseteq U^n$ to every n-ary predicate symbol $p \in \mathcal{P}$. A *variable assignment* B for w is a mapping which assigns an element $x^B \in U$ to every variable $x \in \mathcal{V}$. A variable assignment B' is an *x-variant* of B if $y^B = y^{B'}$ for every variable $y \in \mathcal{V}$ such that $y \neq x$. The interpretation of a term t, denoted $t^{w,B}$, is defined as usual; if t is ground, we write t^w instead of $t^{w,B}$.

An individual k with at least one name $t \in \mathcal{N}$ such that $t^w = k$ is called a *named* individual, and *unnamed* otherwise. In case names are interpreted distinctly, the *unique-names assumption* applies. If, additionally, every individual is named, the *standard-names assumption* applies.

A *variable substitution* β is a set $\{x_1/t_1, ..., x_k/t_k\}$, where $x_1, ..., x_k$ are distinct variables and $t_1, ..., t_k$ are names. β is *total* if it contains some x/n for every variable $x \in \mathcal{V}$. Given variable assignment B and substitution β, if $\beta = \{x/t \mid x \in \mathcal{V}, t^w = x^B, \text{for some name } t\}$, then β is *associated with* B. The *application* of a variable substitution β to some term, formula, or theory, denoted by appending β to it, is defined as syntactical replacement, as usual. Clearly, if the unique-names assumption applies, each variable assignment has a unique associated substitution; if the standard-names assumption applies, each associated

substitution is total.

Example 1 *Consider a language $\mathcal{L}$ with constants $\mathcal{F} = \{a, b, c\}$, and an interpretation $w = \langle U, \cdot^I \rangle$ with $U = \{k, l, m\}$ such that $a^w = k$, $b^w = l$, and $c^w = l$, and the variable assignment B: $x^B = k$, $y^B = l$, and $z^B = m$. B has two associated variable substitutions, $\beta_1 = \{x/a, y/b\}$ and $\beta_2 = \{x/a, y/c\}$, which are not total.*

Logic Programs A *disjunctive logic program* P consists of rules of the form

$$h_1 \mid \ldots \mid h_l \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n, \tag{3.1}$$

where $h_1, \ldots, h_l, b_1, \ldots, b_m, c_1, \ldots, c_n$ are (equality-free) atoms. $H(r) = \{h_1, \ldots, h_l\}$ is the set of *head atoms* of r, $B^+(r) = \{b_1, \ldots, b_m\}$ is the set of *positive body atoms* of r, and $B^-(r) = \{c_1, \ldots, c_n\}$ is the set of *negative body atoms* of r. If $l = 1$, then r is *normal*. If $B^-(r) = \emptyset$, then r is *positive*. If every variable in r occurs in $B^+(r)$, then r is *safe*. If every rule $r \in P$ is normal (resp., positive, safe), then P is normal (resp., positive, safe).

By a *first-order signature*, Σ_P, we understand a superset of the function and predicate symbols which occur in P. Let $\mathcal{L}_P$ denote the first-order language based on Σ_P. We assume that Σ_P contains at least one 0-ary function symbol or only 0-ary predicate symbols. The *Herbrand base*, B_H, of $\mathcal{L}_P$ is the set of ground atomic formulas of $\mathcal{L}_P$. Subsets of B_H are called *Herbrand interpretations*.

The *grounding* of a logic program P, denoted $gr(P)$, is the union of all possible ground instantiations of P, obtained by replacing each variable in a rule r with a name in $\mathcal{N}_{\Sigma_P}$, for each rule $r \in P$.

Let P be a positive program. A Herbrand interpretation M of P is a *model* of P if, for every rule $r \in gr(P)$, $B^+(r) \subseteq M$ implies $H(r) \cap M \neq \emptyset$. A Herbrand model M is *minimal* iff for every model M' such that $M' \subseteq M$, $M' = M$.

Following [GL91b], the *reduct* of a logic program P with respect to an interpretation M, denoted P^M, is obtained from $gr(P)$ by deleting (i) each rule r with $B^-(r) \cap M \neq \emptyset$, and (ii) $\mathit{not}\ c$ from the body of every remaining rule r with $c \in B^-(r)$. If M is a minimal Herbrand model of P^M, then M is a *stable model* of P.

3.3 First-order Autoepistemic Logic

We adopt the definition of first-order autoepistemic logic (FO-AEL) under the any- and all-name semantics following [Kon91], using a novel characterization with *associated variable substitutions*. The benefit of these semantics is that they allow quantification over arbitrary domains and generalize classical first-order logic with equality, thereby allowing a trivial embedding of first-order theories (with equality). Other approaches

[KR02, LL00] require interpretations to follow the unique or standard names assumptions and therefore do not allow such direct embeddings.

An FO-AEL language $\mathcal{L}_{\mathsf{L}}$ is defined relative to a first-order language $\mathcal{L}$:

i any atomic formula in $\mathcal{L}$ is a formula in $\mathcal{L}_{\mathsf{L}}$;

ii if ϕ is a formula in $\mathcal{L}_{\mathsf{L}}$, then $\mathsf{L}\phi$, called a *modal atom*,[1] is a formula in $\mathcal{L}_{\mathsf{L}}$; and

iii complex formulas are constructed as in standard first-order logic.

A formula without modal atoms is an *objective* formula. As usual, a formula with no free variable occurrences is a *sentence*. *Standard autoepistemic logic* is FO-AEL without variables.

An *autoepistemic interpretation* is a pair $\langle w, \Gamma\rangle$ where $w = \langle U, \cdot^I\rangle$ is a first-order interpretation and is $\Gamma \subseteq \mathcal{L}_{\mathsf{L}}$ a set of sentences, called *belief set*. $\langle w, \Gamma\rangle$ *satisfies* an objective atomic formula $p(t_1, \ldots, t_n)$ *relative* to a variable assignment B, denoted $(w, B) \models_\Gamma p(t_1, \ldots, t_n)$, if $(t_1^{w,B}, \ldots, t_n^{w,B}) \in p^I$. Furthermore, $(w, B) \models_\Gamma t_1 = t_2$ iff $t_1^{w,B} = t_2^{w,B}$.

Satisfaction of a formula $\mathsf{L}\phi$ in an interpretation $\langle w, \Gamma\rangle$ with respect to a variable assignment B under the *any-name semantics* (resp., *all-name semantics*) is defined as follows:

$(w, B) \models_\Gamma \mathsf{L}\phi$ iff, for some (resp., all) variable substitution(s) β, associated with B, $\phi\beta$ is closed and $\phi\beta \in \Gamma$.

Satisfiability of complex formulas is defined as usual, with $\phi, \psi \in \mathcal{L}_{\mathsf{L}}$:

- $(w, B) \models_\Gamma \neg\phi$ iff $(w, B) \not\models_\Gamma \phi$,
- $(w, B) \models_\Gamma \phi \wedge \psi$ iff $(w, B) \models_\Gamma \phi$ and $(w, B) \models_\Gamma \psi$,
- $(w, B) \models_\Gamma \phi \vee \psi$ iff $(w, B) \models_\Gamma \phi$ or $(w, B) \models_\Gamma \psi$,
- $(w, B) \models_\Gamma \phi \supset \psi$ iff $(w, B) \models_\Gamma \neg\phi$ or $(w, B) \models_\Gamma \psi$,
- $(w, B) \models_\Gamma \forall x.\phi$ iff for every x-variant B' of B, $(w, B') \models_\Gamma \phi$, and
- $(w, B) \models_\Gamma \exists x.\phi$ iff for some x-variant B' of B, $(w, B') \models_\Gamma \phi$.

Notice that in case the unique names (or the standard names) assumption applies, the any- and all-name semantics coincide.

$\langle w, \Gamma\rangle$ is a *model* of ϕ under the any-(resp. all-)name semantics, denoted $w \models_\Gamma \phi$, if $(w, B) \models_\Gamma \phi$ under the any-(resp. all-)name semantics for every variable assignment B of w. This extends to sets of formulas in the usual way. A set of formulas $A \subseteq \mathcal{L}_{\mathsf{L}}$ *entails* a sentence $\phi \in \mathcal{L}_{\mathsf{L}}$ under the any-(resp. all-)name semantics with respect to a belief set

[1] $\mathsf{L}\phi$ is usually read as "ϕ is known" or "ϕ is believed."

Γ, denoted $A \models_\Gamma \phi$, if for every interpretation w such that $w \models_\Gamma A$ under the any-(resp. all-)name semantics, $w \models_\Gamma \phi$ under the any-(resp. all-)name semantics.

When considering only interpretations for which the standard names assumption applies (so that the any- and all-name semantics coincide), we talk about entailment $A \models_\Gamma \phi$ under the standard names assumption.

Example 2 *Consider a language with constant symbols a, b and unary predicate symbol p, and an interpretation $\langle w, \Gamma\rangle$ with $w = \langle\{k\}, \cdot^I\rangle$ and $\Gamma = \{p(a)\}$. Under the any-name semantics, $w \models_\Gamma \exists x.\mathsf{L}p(x)$; under the all-name semantics, $w \not\models_\Gamma \exists x.\mathsf{L}p(x)$, because $b^w = a^w = k$, but $p(b) \notin \Gamma$.*

We deem this behavior of the all-name semantics counterintuitive; so, following [Kon91], we use the any-name semantics in what follows, unless stated otherwise.

Example 3 *Consider the formula $\phi = \forall x(p(x) \supset \mathsf{L}p(x))$ and some interpretation $\langle w, \Gamma\rangle$. Then: $w \models_\Gamma \phi$ iff for every variable assignment B, $(w, B) \models_\Gamma p(x) \supset \mathsf{L}p(x)$ iff $(w, B) \not\models_\Gamma p(x)$ or $(w, B) \models_\Gamma \mathsf{L}p(x)$. Now, $(w, B) \models_\Gamma \mathsf{L}p(x)$, with $x^B = k$, iff for some $t \in \mathcal{N}_\Sigma$, $t^w = k$, and $p(t) \in \Gamma$. Thus, ϕ is false (unsatisfied) in any interpretation where p^I contains unnamed individuals.*

A *stable expansion* is the set of beliefs of an ideally introspective agent (i.e. an agent with perfect reasoning capabilities, and with knowledge about its own beliefs), given some base set. A belief set $T \subseteq \mathcal{L}_\mathsf{L}$ is a *stable expansion* of a base set $A \subseteq \mathcal{L}_\mathsf{L}$ iff $T = \{\phi \mid A \models_T \phi\}$. With $\mathcal{L}_g$ and $\mathcal{L}_{ga}$ we denote the restriction of $\mathcal{L}$ to ground formulas and to ground atomic formulas, respectively. T_o, T_{og}, and T_{oga} denote the restrictions of T to objective, objective ground, and objective ground atomic formulas, respectively, i.e. $T_o = T \cap \mathcal{L}$, $T_{og} = T \cap \mathcal{L}_g$, and $T_{oga} = T \cap \mathcal{L}_{ga}$.

A formula ϕ is an *autoepistemic consequence* of A if ϕ is included in every stable expansion of A. $Cons(A)$ denotes the set of all autoepistemic consequences of A. $Cons_o(A)$ denotes the restriction of $Cons(A)$ to objective formulas, i.e. $Cons_o(A) = Cons(A) \cap \mathcal{L}$.

Every stable expansion T of A is a stable set, which means that it fulfills the following properties: (a) T is closed under first-order entailment, (b) if $\phi \in T$ then $\mathsf{L}\phi \in T$, and (c) if $\phi \notin T$ then $\neg\mathsf{L}\phi \in T$. Furthermore, if T is consistent, the converse statements of (b) and (c) hold.

Konolige [Kon91] shows that a stable expansion T of a base set A is determined by its objective subset T_o, called the *kernel* of T. He further obtains the following result.

Proposition 1 ([Kon91]) *Let $A \subseteq \mathcal{L}_\mathsf{L}$ be a base set which does not have nested modal operators, and let $\Gamma_o \subseteq \mathcal{L}$ be a set of objective formulas. Then, $\Gamma_o = \{\phi \in \mathcal{L} \mid A \models_{\Gamma_o} \phi\}$ iff $\Gamma_o = T \cap \mathcal{L}$ for some stable expansion T of A.*

We extend this result as follows:

Proposition 2 *Given a base set $A \subseteq \mathcal{L}_{\mathsf{L}}$ with only objective atomic formulas in the context of modal atoms, and a set of objective formulas $\Gamma_o \subseteq \mathcal{L}$, with $\Gamma_{oga} = \Gamma_o \cap \mathcal{L}_{ga}$, then $\Gamma_o = \{\phi \in \mathcal{L} \mid A \models_{\Gamma_{oga}} \phi\}$ iff $\Gamma_o = T \cap \mathcal{L}$ for some stable expansion T of A.*

Proof 3 *Since A only contains atomic formulas in the context of the modal operator we obtain $A \models_{\Gamma_o} \phi$ iff $A \models_{\Gamma_{oga}} \phi$, because, by the definition of satisfaction of modal formulas, non-ground and non-atomic formulas in Γ_o do not affect satisfaction of formulas in A. $\{\phi \in \mathcal{L} \mid A \models_{\Gamma_{oga}} \phi\} = \{\phi \in \mathcal{L} \mid A \models_{\Gamma_o} \phi\}$ follows.*

Since A does not contain any nested modal operators we combine this result with Proposition 1 to obtain $\Gamma_o = \{\phi \in \mathcal{L} \mid A \models_{\Gamma_o} \phi\} = \{\phi \in \mathcal{L} \mid A \models_{\Gamma_{oga}} \phi\}$ iff $\Gamma_o = T \cap \mathcal{L}$ is the kernel of a stable expansion T of A.

3.4 Embedding Non-Ground Logic Programs

We define an embedding as a function which takes a logic program P as its argument and returns a set of sentences in the FO-AEL language obtained from Σ_P.

Since the unique names assumption does not hold in FO-AEL in general, it is necessary to axiomatize default uniqueness of names (as introduced by [Kon91]). By UNA_Σ we denote the set of axioms

$$\neg\mathsf{L}(t_1 = t_2) \supset t_1 \neq t_2, \quad \text{for all distinct } t_1, t_2 \in \mathcal{N}_\Sigma.$$

The reason we axiomatize default uniqueness ($\neg\mathsf{L}(t_1 = t_2) \supset t_1 \neq t_2$), instead of rigid uniqueness ($t_1 \neq t_2$), is that we want to allow a first-order theory which is added to the embedding to override this inequality, rather than introduce an inconsistency. We believe that, with default uniqueness, the combination behaves in a more intuitive way.

Recall that a logic program P consists of a set of rules of the form

$$h_1 \mid \ldots \mid h_l \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n, \tag{3.2}$$

where $h_1, \ldots, h_l, b_1, \ldots, b_m, c_1, \ldots, c_n$ are (equality-free) atoms. If $l = 1$, then r is *normal*. If $B^-(r) = \emptyset$, then r is *positive*. If every variable in r occurs in $B^+(r)$, then r is *safe*. If every rule $r \in P$ is normal (resp., positive, safe), then P is normal (resp., positive, safe).

We proceed to describe three embeddings of normal and three embeddings of disjunctive logic programs into FO-AEL.

3.4.1 Embedding Normal Logic Programs

The first embedding we consider is an extension of the one which originally led Gelfond and Lifschitz to the definition of the stable model semantics [Gel87, GL88]. The

second and third embedding are extensions of the embeddings described by Marek and Truszczynski in [MT93]. The third was independently described by Lifschitz and Schwartz in [LS93] and by Chen in [Che93]. The original motivation for the second and third embedding was the possibility to directly embed programs with strong negation and disjunction. Furthermore, Marek and Truszczynski arrived at the second and third embedding through embeddings of logic programs in *reflexive autoepistemic logic* [Sch92], which is equivalent to McDermott's nonmonotonic modal $\mathrm{Sw5}$ [McD82], and the subsequent embedding of reflexive autoepistemic logic into standard autoepistemic logic. Lifschitz and Schwartz arrived at the third embedding through an embedding of logic programs in Lifschitz's bimodal nonmonotonic logic of *Minimal Belief and Negation-as-Failure* (MBNF) [Lif94] and the subsequent embedding of MBNF in standard autoepistemic logic. Finally, Chen also arrived at the third embedding through an embedding of logic programs in MBNF, but he subsequently embedded MBNF in Levesque's *logic of only knowing* [Lev90], a subset of which corresponds with standard autoepistemic logic (with respect to stable expansions).

Definition 4 *Let r be a rule of the form* (3.2) *with $l = 1$. Then:*

$$\tau_{HP}(r) = (\forall)\ \textstyle\bigwedge_i b_i \wedge \bigwedge_j \neg \mathsf{L} c_j \supset h;$$
$$\tau_{EB}(r) = (\forall)\ \textstyle\bigwedge_i (b_i \wedge \mathsf{L} b_i) \wedge \bigwedge_j \neg \mathsf{L} c_j \supset h;$$
$$\tau_{EH}(r) = (\forall)\ \textstyle\bigwedge_i (b_i \wedge \mathsf{L} b_i) \wedge \bigwedge_j \neg \mathsf{L} c_j \supset h \wedge \mathsf{L} h.$$

For a normal logic program P, we define:

$$\tau_x(P) = \{\tau_x(r) \mid r \in P\} \cup UNA_{\Sigma_P},\ \ x \in \{HP, EB, EH\}.$$

In the above embeddings, "HP" stands for "Horn for Positive rules" (positive rules are translated to objective Horn clauses); "EB" stands for "Epistemic rule Bodies" (the body of a rule can only become true if it is *known* to be true); and "EH" stands for "Epistemic rule Heads" (if the body of a rule is true, the head is *known* to be true). For all three embeddings, we assume $\Sigma_{\tau_x(P)} = \Sigma_P$ (here and henceforth we use "x" as a meta-variable to range over HP, EB, and EH). Furthermore, by τ_x^- we denote the embedding τ_x *without* the UNA axioms: given a normal logic program P, $\tau_x^-(P) = \tau_x(P) - UNA_{\Sigma_P}$. In the examples of embeddings in the remainder of the chapter we do not write the UNA axioms explicitly.

A notable distinction between the embedding τ_{HP} on the one hand and the embeddings τ_{EB} and τ_{EH} on the other is that the contrapositive of the rules in P is included in stable expansions of τ_{HP}, but not in stable expansions of τ_{EB} and τ_{EH}:

Example 4 *Consider $P = \{p \leftarrow q, not\ r\}$. The stable expansion of $\tau_{HP}(P) = \{q \wedge \neg \mathsf{L} r \supset p\}$ includes $\neg p \supset \neg q \vee \mathsf{L} r$; the expansion of $\tau_{EB}(P) = \{q \wedge \mathsf{L} q \wedge \neg \mathsf{L} r \supset p\}$ includes $\neg p \supset \neg \mathsf{L} q \vee \neg q \vee \mathsf{L} r$, but not $\neg p \supset \neg q \vee \mathsf{L} r$.*

For the case of standard autoepistemic logic and ground logic programs, the following correspondence holds:

Proposition 3 ([GL88, MT93]) *A Herbrand interpretation M is a stable model of a ground normal logic program P iff there is a consistent stable expansion T of $\tau_x^-(P)$ in standard autoepistemic logic such that $M = T \cap \mathcal{L}_{ga}$.*

Now consider the case of non-ground programs. The following example illustrates the embeddings:

Example 5 *Consider $P = \{q(a);\ p(x);\ r(x) \leftarrow not\ s(x),\ p(x)\}$, having a single stable model $M = \{q(a), p(a), r(a)\}$. Likewise, each of the embeddings $\tau_{HP}(P)$, $\tau_{EB}(P)$, and $\tau_{EH}(P)$ has a single consistent stable expansion:*

$$\begin{aligned} T^{HP} &= \{q(a), p(a), \mathsf{L}p(a), \neg\mathsf{L}s(a), r(a), \\ &\qquad \forall x(p(x)), \neg\mathsf{L}\forall x(\mathsf{L}p(x)), \ldots\}, \\ T^{EB} &= \{q(a), p(a), \mathsf{L}p(a), \neg\mathsf{L}s(a), r(a), \neg\mathsf{L}\forall x(\mathsf{L}p(x)), \ldots\}, \\ T^{EH} &= \{q(a), p(a), \mathsf{L}p(a), \neg\mathsf{L}s(a), r(a), \forall x(\mathsf{L}p(x)) \ldots\}. \end{aligned}$$

The stable expansions in Example 5 agree on objective ground atoms, but not on arbitrary formulas. We now extend Proposition 3 to the non-ground case.

Lemma 5 *Given a normal logic program P, a stable expansion T of $\tau_x(P)$, and an objective ground atom α, we have that $\tau_x(P) \models_{T_{oga}} \alpha$ under the any- or all-name semantics iff $\tau_x(P) \models_{T_{oga}} \alpha$ under the standard names assumption. Moreover, the same result holds for τ_{HP}^- under the all-name semantics.*

Proof 4 *($\Rightarrow$) By definition, $\tau_x(P) \models_{T_{oga}} \alpha$ under the any- or all-name semantics iff for every interpretation w it holds that whenever $w \models_{T_{oga}} \tau_x(P)$, $w \models_{T_{oga}} \alpha$. If for every interpretation w, whenever $w \models_{T_{oga}} \tau_x(P)$, $w \models_{T_{oga}} \alpha$, then it must hold that for every interpretation w' for which the standard names assumption applies, whenever $w \models_{T_{oga}} \tau_x(P)$, $w \models_{T_{oga}} \alpha$, and thus (again, by definition) $\tau_x(P) \models_{T_{oga}} \alpha$ under the standard names assumption. Therefore, if $\tau_x(P) \models_{T_{oga}} \alpha$ under the any- or all-name semantics, then $\tau_x(P) \models_{T_{oga}} \alpha$ under the standard names assumption.*

($\Leftarrow$) Assume, on the contrary, that $\tau_x(P) \models_{T_{oga}} \alpha$ under the standard names assumption, but $\tau_x(P) \not\models_{T_{oga}} \alpha$ under the any- or all-name semantics. This means that there is some interpretation $w = \langle U, I\rangle$ such that $w \models_{T_{oga}} \tau_x(P)$, but $w \not\models_{T_{oga}} \alpha$.

By the fact that there is no equality in $\tau_x(P)$, there is no occurrence of the equality symbol in T_{oga}. Furthermore, by the UNA axioms, every name in Σ_P must be assigned to a distinct individual in U. We construct the interpretation $w' = \langle U', I'\rangle$ as follows: $U' = \mathcal{N}$, $t^{I'} = t$, for $t \in \mathcal{N}$, and $\langle t_1, \ldots, t_n\rangle \in p^{I'}$ if $\langle k_1, \ldots, k_n\rangle \in p^I$ and $k_i = t_i^I$ for $1 \leq i \leq n$, with p an n-ary predicate symbol. Clearly, the standard names assumption holds for w', and w and w' agree on objective ground atoms: $w \models \alpha$ iff $w' \models \alpha$ for any $\alpha \in \mathcal{L}_{ga}$. We now proceed to show that $w' \models_{T_{oga}} \tau_x(P)$.

Clearly, $\langle w', T_{oga}\rangle$ satisfies the UNA axioms due to the fact that the standard names assumption holds for w' and the fact that T_{oga} does not contain equality atoms. We first

consider the embedding τ_{EH}. Consider a formula $\forall\, b_1 \wedge \mathsf{L}b_1 \wedge \ldots \wedge b_m \wedge \mathsf{L}b_m \wedge \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n \supset h \wedge \mathsf{L}h \in \tau_{EH}(P)$. Since $w \models_{T_{oga}} \tau_{EH}(P)$, $(w, B) \models_{T_{oga}} b_1 \wedge \mathsf{L}b_1 \wedge \ldots \wedge b_m \wedge \mathsf{L}b_m \wedge \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n \supset h \wedge \mathsf{L}h$ for every variable assignment B of w.

Now, consider a variable assignment B' of w' and a corresponding variable assignment B of w which assigns variables only to named individuals: $x^B = k$ iff $x^{B'} = t$ and $t^I = k$. Consider a variable substitution β which is associated with B; since all names are interpreted as distinct individuals (by the UNA axioms), β is the only variable substitution associated with B; therefore, the any- and all-name semantics coincide. Furthermore, by construction of B, β is also associated with B'

If $(w, B) \models_{T_{oga}} h \wedge \mathsf{L}h$, then $(w', B') \models_{T_{oga}} h \wedge \mathsf{L}h$ and if $(w, B) \not\models_{T_{oga}} b_1 \wedge \mathsf{L}b_1 \wedge \ldots \wedge b_m \wedge \mathsf{L}b_m$, then $(w', B') \not\models_{T_{oga}} b_1 \wedge \mathsf{L}b_1 \wedge \ldots \wedge b_m \wedge \mathsf{L}b_m$, by construction of w' (assuring correspondence of unsatisfiability of objective atoms) and since β is the only variable substitution associated with B and the only one associated with B' (assuring correspondence of unsatisfiability of modal atoms). Now, if $(w, B) \not\models_{T_{oga}} \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n$, then it must be the case that $b_1\beta \in T_{oga}, \ldots,$ or $b_n\beta \in T_{oga}$. Therefore, it must hold that $(w', B') \not\models_{T_{oga}} \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n$. So, $(w', B') \models_{T_{oga}} b_1 \wedge \mathsf{L}b_1 \wedge \ldots \wedge b_m \wedge \mathsf{L}b_m \wedge \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n \supset h$.

Thus, we obtain $w' \models_{T_{oga}} \tau_{EH}(P)$. Since w and w' agree on objective ground atoms, $w' \not\models_{T_{oga}} \alpha$, and thus $\tau_{EH}(P) \not\models_{T_{oga}} \alpha$ under the standard names assumption, clearly contradicting the assumption that $\tau_{EH}(P) \models_{T_{oga}} \alpha$ under the standard names assumption. Therefore, $\tau_{EH}(P) \models_{T_{oga}} \alpha$ under the any- and full-name semantics.

The argument for the embeddings τ_{EB} and τ_{HP} is analogous: simply leave out the positive occurrences of modal atoms in the antecedents, respectively consequents and antecedents, in the argument above.

To see why the lemma holds for the embedding τ^-_{HP} under the all-name semantics, leave out the arguments about the UNA axioms and consider a simple adaptation of the argument above: if $(w, B) \not\models_{T_{oga}} \neg\mathsf{L}b_{m+1} \wedge \ldots \wedge \neg\mathsf{L}b_n$, then for all associated variable substitutions β, $b_1\beta \in T, \ldots,$ or $b_n\beta \in T$. One of these variable substitutions is the one associated with B'.

Lemma 6 *Given a normal logic program P and a set $\mathcal{A} \subseteq \mathcal{L}_{ga}$ of objective ground atoms, there exists a stable expansion T of $\tau_x(P)$ under the any- or all-name semantics with $T_{oga} = \mathcal{A}$ iff there exists a stable expansion T' of $\tau_x(gr(P))$ with $T'_{oga} = \mathcal{A}$. Moreover, the same result holds for τ^-_{HP} under the all-name semantics.*

Proof 5 *We first prove the lemma for the special case that the standard names assumption applies. We then use Lemma 5 to extend this result to cases where the standard names assumption does not apply.*

Consider a belief set $\Gamma \subseteq \mathcal{L}_{\mathsf{L}}$ and an interpretation w such that the standard names assumption applies to w. We claim that (*) $w \models_\Gamma \tau_x(gr(P))$ iff $w \models_\Gamma \tau_x(P)$. By the standard names assumption, we have that $w \models_\Gamma \tau_x(P)$ iff for every $\phi \in \tau_x(P)$, $w \models_\Gamma$

ϕ iff for every variable assignment B, $(w, B) \models_\Gamma \phi$ iff for the variable substitution β associated with B, $w \models_\Gamma \phi\beta$. By the standard names assumption, we know that the associated variable substitution β is unique and total. Clearly, by the definition of $gr()$, $\tau_x(gr(P))$ contains all (and only) the formulas of the form $\phi\beta$ with $\phi \in \tau_x(P)$ and β a variable substitution associated with some variable assignment B for w; the claim (*) follows.

($\Rightarrow$) Let T be a stable expansion of $\tau_x(P)$. By Lemma 5 and the above we have:

$$\{\phi \in \mathcal{L}_{ga} \,|\, \tau_x(P) \models_{T_{oga}} \phi\} = \{\phi \in \mathcal{L}_{ga} \,|\, \tau_x(gr(P)) \models_{T_{oga}} \phi\}.$$

Therefore (by Proposition 2),

$$T'_o = \{\phi \in \mathcal{L} \mid \tau_x(gr(P)) \models_{T_{oga}} \phi\}$$

is the kernel of a stable expansion T' of $\tau_x(gr(P))$ and $T' \cap \mathcal{L}_{ga} = T_{ga}$.

($\Leftarrow$) Let T be a stable expansion of $\tau_x(gr(P))$. By Lemma 5 and the above we have:

$$\{\phi \in \mathcal{L}_{ga} \,|\, \tau_x(gr(P)) \models_{T_{oga}} \phi\} = \{\phi \in \mathcal{L}_{ga} \,|\, \tau_x(P) \models_{T_{oga}} \phi\}.$$

Therefore (by Proposition 2),

$$T'_o = \{\phi \in \mathcal{L} \mid \tau_x(P) \models_{T_{oga}} \phi\}$$

is the kernel of a stable expansion T' of $\tau_x(P)$ and $T' \cap \mathcal{L}_{ga} = T_{ga}$.

Theorem 7 *A Herbrand interpretation M of a normal logic program P is a stable model of P iff there is a consistent stable expansion T of $\tau_x(P)$ under the any- or all-name semantics such that $M = T \cap \mathcal{L}_{ga}$. Moreover, the same result holds for τ^-_{HP} under the all-name semantics.*

Proof 6 *(Sketch) By Lemma 6 we can reduce embeddability of non-ground logic programs to embeddability of ground logic programs. The embeddings of $gr(P)$ in first-order autoepistemic logic are then trivially reduced to the respective embeddings of $gr(P)$ in standard autoepistemic logic (Proposition 3).*

Note that this result does not extend to τ^-_{HP} under the any-name semantics. Consider $P = \{p(n_1); r(n_2); q \leftarrow not\ p(x)\}$ such that Σ_P has only two names, n_1 and n_2. P has one stable model, $M = \{p(n_1), r(n_2), q\}$. $\tau^-_{HP}(P) = \{p(n_1); r(n_2); \forall x(\neg \mathsf{L}p(x) \supset q)\}$ has one stable expansion, $T = \{p(n_1),\ r(n_2),\ \mathsf{L}p(n_1),\ \mathsf{L}r(n_2),\ \neg\mathsf{L}p(n_2), \ldots\}$. T does *not* include q. To see why this is the case, consider an interpretation w with only one individual k. $\mathsf{L}p(x)$ is trivially true under the any-name semantics, because there is some name for k such that $p(t) \in T$ (viz. $t = n_1$). In the all-name semantics, this situation does not occur, because for $\mathsf{L}p(x)$ to be true, $p(t)$ must be included in T for *every* name ($t = n_1$ and $t = n_2$) for k. One can similarly verify that the result does not apply to the embeddings τ^-_{EB} and τ^-_{EH} under the all-name semantics, by the positive modal atoms in the antecedents.

3.4.2 Embedding Disjunctive Logic Programs

The embeddings τ_{HP} and τ_{EB} cannot be straightforwardly extended to the case of disjunctive logic programs, even for the propositional case. Consider the program $P = \{a \mid b \leftarrow\}$. P has two stable models: $M_1 = \{a\}$ and $M_2 = \{b\}$. However, a straightforward extension of τ_{HP}, $\tau^{\vee}_{HP}(P) = \{a \vee b\}$, has one stable expansion $T = \{a \vee b, \mathsf{L}(a \vee b), \neg\mathsf{L}a, \neg\mathsf{L}b, \ldots\}$. In contrast, τ_{EH} can be straightforwardly extended because of the modal atoms in the consequent of the implication: $\tau^{\vee}_{EH}(P) = \{(a \wedge \mathsf{L}a) \vee (b \wedge \mathsf{L}b)\}$ has two stable expansions $T_1 = \{a \vee b, a, \mathsf{L}a, \neg\mathsf{L}b, \ldots\}$ and $T_2 = \{a \vee b, b, \mathsf{L}b, \neg\mathsf{L}a, \ldots\}$.

The so-called *positive introspection axioms* (PIAs) [Prz91] remedy this situation for $\tau^{\vee}_{HP}$ and $\tau^{\vee}_{EB}$. Let PIA_{Σ} be the set of axioms

$$\alpha \supset \mathsf{L}\alpha, \quad \text{for every objective ground atom } \alpha \text{ of } \Sigma.$$

The PIA $\alpha \supset \mathsf{L}\alpha$ ensures that every consistent stable expansion contains either α or $\neg\alpha$.

It would have been possible to define the PIAs in a different way: $(\forall)\ \phi \supset \mathsf{L}\phi$ for any objective atomic formula ϕ. This would, however, effectively close the domain of the predicates in Σ_P (see Example 3). We deem this aspect undesirable in combinations with FO theories.

Definition 8 *Let r be a rule of form* (3.2)*. Then:*

$$\begin{aligned}
\tau^{\vee}_{HP}(r) &= (\forall)\ \textstyle\bigwedge_i b_i \wedge \bigwedge_j \neg\mathsf{L}c_j \supset \bigvee_k h_k;\\
\tau^{\vee}_{EB}(r) &= (\forall)\ \textstyle\bigwedge_i (b_i \wedge \mathsf{L}b_i) \wedge \bigwedge_j \neg\mathsf{L}c_j \supset \bigvee_k h_k;\\
\tau^{\vee}_{EH}(r) &= (\forall)\ \textstyle\bigwedge_i (b_i \wedge \mathsf{L}b_i) \wedge \bigwedge_j \neg\mathsf{L}c_j \supset \bigvee_k (h_k \wedge \mathsf{L}h_k).
\end{aligned}$$

For a disjunctive logic program P, we define:

$$\begin{aligned}
\tau^{\vee}_{HP}(P) &= \{\tau^{\vee}_{HP}(r) \mid r \in P\} \cup PIA_{\Sigma_P} \cup UNA_{\Sigma_P};\\
\tau^{\vee}_{EB}(P) &= \{\tau^{\vee}_{EB}(r) \mid r \in P\} \cup PIA_{\Sigma_P} \cup UNA_{\Sigma_P};\\
\tau^{\vee}_{EH}(P) &= \{\tau^{\vee}_{EH}(r) \mid r \in P\} \cup UNA_{\Sigma_P}.
\end{aligned}$$

As before, by $\tau^{\vee -}_{x}$ we denote the embedding $\tau^{\vee}_{x}$ *without* the UNA axioms. We do not write the UNA and PIA axioms explicitly in the examples below.

For the case of standard autoepistemic logic and ground disjunctive logic programs, the correspondence between the stable expansions of the embeddings $\tau^{\vee}_{HP}(P)$ and $\tau^{\vee}_{EH}(P)$ and the stable models of P is known:

Proposition 4 ([Prz91, MT93]) *A Herbrand interpretation M of a ground disjunctive logic program P is a stable model of P iff there is a consistent stable expansion T of $\tau^{\vee -}_{HP}(P)$ (resp., $\tau^{\vee -}_{EH}(P)$) in standard autoepistemic logic such that $M = T \cap \mathcal{L}_{ga}$.*

We generalize this result to the case of FO-AEL and non-ground programs, and additionally for $\tau^{\vee}_{EB}$, similar to the case of normal programs.

Lemma 9 *Given a disjunctive logic program P, a stable expansion T of $\tau_x^\vee(P)$, and an objective ground atom α, we have that $\tau_x^\vee(P) \models_{T_{oga}} \alpha$ under the any- or all-name semantics iff $\tau_x^\vee(P) \models_{T_{oga}} \alpha$ under the standard names assumption. Moreover, the same result holds for ${\tau_{HP}^\vee}^-$ under the all-name semantics.*

Proof 7 *($\Rightarrow$) Trivial, since every interpretation for which the standard names assumption applies is an interpretation; see also the $\Rightarrow$ direction in the proof of Lemma 5.*

($\Leftarrow$) The argument is as straightforward adaptation of the arguments in the $\Leftarrow$ direction in the proof of Lemma 5: simply replace the consequence $h \wedge \mathsf{L}h$ with the disjunction $(h_1 \wedge \mathsf{L}h_1) \vee \cdots \vee (h_l \wedge \mathsf{L}h_l)$. Furthermore, it is also easy to see, by the fact that w and w' agree on ground atomic formulas, that if the PIA axioms are satisfied in $\langle w, T_{oga}\rangle$, then they are satisfied in $\langle w', T_{oga}\rangle$.

Lemma 10 *Given a disjunctive logic program P and a set $\mathcal{A} \subseteq \mathcal{L}_{ga}$ of objective ground atoms, there exists a stable expansion T of $\tau_x^\vee(P)$ under the any- or all-name semantics with $T_{oga} = \mathcal{A}$ iff there exists a stable expansion T' of $\tau_x^\vee(gr(P))$ with $T'_{oga} = \mathcal{A}$. Moreover, the same result holds for ${\tau_{HP}^\vee}^-$ under the all-name semantics.*

Proof 8 *The proof of the lemma is obtained from the proof of Lemma 6 by replacing occurrences of τ_x with $\tau_x^\vee$ and replacing references to Lemma 5 with references to Lemma 9.*

Theorem 11 *A Herbrand interpretation M of a disjunctive logic program P is a stable model of P iff there is a consistent stable expansion T of $\tau_x^\vee(P)$ under the any- or all-name semantics such that $M = T \cap \mathcal{L}_{ga}$. Moreover, the same result holds for ${\tau_{HP}^\vee}^-$ under the all-name semantics.*

Proof 9 *Reduction of embeddability of ground logic programs follows from Lemma 10. Embeddability of $gr(P)$ using $\tau_{HP}^\vee$ and $\tau_{EH}^\vee$ follows from Proposition 4.*

Embeddability of $gr(P)$ using $\tau_{EB}^\vee$ then follows from the embeddability using $\tau_{HP}^\vee$ and the PIA axioms: if the objective part of the antecedent of any formula in $\tau_{EB}^\vee(gr(P))$ is satisfied in a model of $\tau_{EB}^\vee(gr(P))$, then the positive modal part of the antecedent is necessarily included in the stable expansion.

A notable distinction between the embeddings $\tau_{HP}^\vee$ and $\tau_{EB}^\vee$ on the one hand and $\tau_{EH}^\vee$ on the other is the presence and absence of the PIA axioms, respectively:

Example 6 *Consider $P = \{p \mid q \leftarrow\}$, $\tau_{HP}^\vee(P) = \{p \vee q\} \cup PIA_{\Sigma_P}$, and $\tau_{EH}^\vee(P) = \{(p \wedge \mathsf{L}p) \vee (q \wedge \mathsf{L}q)\}$. The stable expansions of $\tau_{HP}^\vee(P)$ are $T_1^{HP} = \{p, \neg q, \mathsf{L}p, \neg\mathsf{L}q, \ldots\}$ and $T_2^{HP} = \{q, \neg p, \mathsf{L}p, \neg\mathsf{L}p, \ldots\}$; the expansions of $\tau_{EH}^\vee(P)$ are $T_1^{EH} = \{p, \mathsf{L}p, \neg\mathsf{L}q, \ldots\}$ and $T_2^{EH} = \{q, \mathsf{L}p, \neg\mathsf{L}p, \ldots\}$. The expansions T_1^{EH} and T_2^{EH} include neither $\neg q$ nor $\neg p$.*

Note that the embedding τ_{HP} cannot be straightforwardly extended to logic programs with strong ("classical") negation [GL91b], even for the propositional case. Take, for

example, the logic program $P = \{p \leftarrow\sim p\}$, with $\sim$ denoting strong negation. P has one stable model $M = \emptyset$. The straightforward extension of the embedding τ_{HP} yields $\{\neg p \supset p\}$ which has one stable expansion which includes p. It was shown in [MT93] that for the propositional case, the embeddings τ_{EB} and τ_{EH} can be easily extended to the case of logic programs with strong negation: consider a rule of the form (3.1) such that $h_1, \ldots, h_l, b_1, \ldots, b_m, c_1, \ldots, c_n$ are either atoms or strongly negated atoms, and an extension of the embeddings τ_{EB}, τ_{EH} such that strong negation $\sim$ is translated to classical negation $\neg$, then Proposition 3 straightforwardly extends to these extended versions of τ_{EB}, τ_{EH} [MT93]. These results can be straightforwardly extended to the non-ground case. Embedding of logic programs with strong negation using τ_{HP} could be done by rewriting P to a logic program without strong negation (see [GL91b] for such a rewriting).

3.5 Relationships between the Embeddings

In this section we explore correspondences between the embeddings presented in the previous section. Recall that we consider the any-name semantics, because of its more intuitive behavior (cf. Example 3).

In the following we compare (i) the stable expansions of such combinations and (ii) the sets of autoepistemic consequences of the individual embeddings. To this end, we introduce the following notation:

Let A_1 and A_2 be FO-AEL theories. We write $A_1 \equiv A_2$ iff A_1 and A_2 have the same stable expansions. Moreover, for $\gamma \in \{g, ga\}$, we write $A_1 \equiv_{o\gamma} A_2$ iff

$$\{T \cap \mathcal{L}_\gamma \mid T \text{ is a stable expansion of } A_1\} = \{T' \cap \mathcal{L}_\gamma \mid T' \text{ is a stable expansion of } A_2\}.$$

Note that, by definition, $A_1 \equiv A_2$ implies $A_1 \equiv_{og} A_2$ and $A_1 \equiv_{og} A_2$ implies $A_1 \equiv_{oga} A_2$.

In our analysis we furthermore use the following classes of programs:

- $\mathcal{P}rg$, $\mathcal{S}afe$, and $\mathcal{G}rnd$ denote the classes of arbitrary, safe, and ground logic programs, respectively.

Observe the following inclusions:

$$\mathcal{G}rnd \longrightarrow \mathcal{S}afe \longrightarrow \mathcal{P}rg$$

Our results on relationships between sets of consequences of the embeddings are summarized in Theorem 12 on page 35.

3.5.1 Relationships between Stable Expansions of Embeddings

In this section, we present our results on the relations between stable expansions of embeddings $\tau_x(P)$, where P is a logic program.

From Theorems 7 and 11 we know that all six embeddings agree on ground atomic formulas in the stable expansions of the embedding of a normal logic program P:

Corollary 1 *If P is a normal logic program, then $\tau_{HP}(P) \equiv_{oga} \tau_{EB}(P) \equiv_{oga} \tau_{EH}(P) \equiv_{oga} \tau^{\vee}_{HP}(P) \equiv_{oga} \tau^{\vee}_{EB}(P) \equiv_{oga} \tau^{\vee}_{EH}(P)$. If P is a disjunctive logic program, then $\tau^{\vee}_{HP}(P) \equiv_{oga} \tau^{\vee}_{EB}(P) \equiv_{oga} \tau^{\vee}_{EH}(P)$.*

This result cannot be immediately extended to the case of objective ground formulas. Consider the logic program $P = \{a \leftarrow b\}$; $\tau_{HP}(P) = \{b \supset a\}$ and $\tau_{EB}(P) = \{b \wedge \mathsf{L}b \supset a\}$. The single stable expansion of $\tau_{HP}(P)$ includes $\neg b \vee a$, but the expansion of $\tau_{EB}(P)$ does not include $\neg b \vee a$. This is because $\neg b \supset \neg \mathsf{L}b$ is included in every stable expansion, but the converse, $\neg \mathsf{L}b \supset \neg b$, is not. The situation changes for the embeddings $\tau^{\vee}_{HP}$ and $\tau^{\vee}_{EB}$, because of the PIA axioms.

Proposition 5 *Given a normal (resp., disjunctive) logic program P, then $\tau_{EB}(P) \equiv_{og} \tau_{EH}(P)$ (resp., $\tau^{\vee}_{HP}(P) \equiv_{og} \tau^{\vee}_{EB}(P)$).*

Proof 10 *For inconsistent expansions, the Proposition trivially holds.*

Let T_{oga} be the set of ground objective atoms in a stable expansion of $\tau_{EB}(P)$. By Corollary 1, T_{oga} is also the set of objective ground atoms of a stable expansion of $\tau_{EH}(P)$ (cq. $\tau^{\vee}_{HP}(P)$, $\tau^{\vee}_{EB}(P)$).

Define T^x_{og} as $T^x_{og} = \{\phi$ is objective and ground $\mid \tau_x(P) \models_{T_{oga}} \phi\}$. By Proposition 2, T^x_{og}, with $x \in \{EB, EH, HP^{\vee}, EB^{\vee}\}$, is the set of ground formulas in a stable expansion of $\tau_{EB}(P)$, $\tau_{EH}(P)$, $\tau^{\vee}_{HP}(P)$, or $\tau^{\vee}_{EB}(P)$.

As for the first part of the proposition concerning normal embeddings $\tau_{EB}(P)$ and $\tau_{EH}(P)$, we will show that, given T_{oga}, $T^{EB}_{og} = T^{EH}_{og}$. We claim that all objective ground formulas in either of the expansions are first-order consequences of the ground atomic formulas in the expansions:

$$T^x_{og} = \{\phi \in \mathcal{L}_g \mid T_{oga} \models \phi\}. \tag{3.3}$$

Since $T^{EB}_{oga} = T^{EH}_{oga}$ (by Corollary 1), $T^{EB}_{og} = T^{EH}_{og}$ follows by this claim.

Every entailed objective ground formula can be written as a set of ground clauses of the form $c = l_1 \vee \ldots \vee l_k$, where each literal l_i is either an atom p_i or a negated atom $\neg p_i$. Clearly, $T_{oga} \models c$ if and only if there is an l_i in c such that $l_i \in T_{oga}$. In order to prove our claim, we observe first that obviously $T^x_{og} \supseteq \{\phi \in \mathcal{L}_g \mid T_{oga} \models \phi\}$. For the converse direction, we proceed indirectly: assume that $\tau_x(P) \models_{T_{oga}} c$ (and thus $c \in T_{og}$) and there is no l_i in c such that $l_i \in T_{oga}$; therefore, for each l_i in c, $\tau_x(P) \models_{T_{oga}} \neg \mathsf{L}l_i$. Consider any

interpretation w such that $w \models_{T_{oga}} \tau_x(P)$; then, $w \models_{T_{oga}} c$. Let w' be the interpretation obtained from w by flipping the truth value of each atom p_i contained in a literal l_i in c such that w satisfies l_i; clearly, $w' \not\models_{T_{oga}} c$. We now show that $w' \models_{T_{oga}} \tau_x(P)$.

If (the non-ground version of) p_i occurs in the antecedent of an axiom originating from a rule in P, then also $\mathsf{L}p_i$ is there, but is false by assumption, so switching p_i does not change satisfaction of the axiom. If (the non-ground version of) p_i occurs in the head of an axiom originating from a rule, then (at least one of the modal atoms) in the body must be false in $\langle w, T_{oga}\rangle$; otherwise p_i would have been included in T_{oga}. Therefore, $w' \models_{T_{oga}} \tau_x(P)$ and thus $\tau_x(P) \not\models_{T_{oga}} c$, which is a contradiction. This establishes (3.3).

As for ground equivalence of the disjunctive embeddings $\tau^{\vee}_{HP}(P)$ and $\tau^{\vee}_{EB}(P)$, we exploit the properties of the PIA axioms: for any objective ground atom α, either α or $\neg\alpha$ is included in the stable expansion T^x ($x \in \{HP^{\vee}, EB^{\vee}\}$); thus, $T^x_{og} = \{\phi \in \mathcal{L}_g \mid T_{oga} \models \phi\}$ (by induction over the formulas). $T^{HP^{\vee}}_{og} = T^{EB^{\vee}}_{og}$ follows immediately.

For non-ground formulas we obtain the following result:

Proposition 6 *Given a safe normal logic program P, $\tau_{EB}(P) \equiv \tau_{EH}(P)$.*

Proof 11 *Let T_{oga} be the set of objectives ground atoms of a stable expansion T of either $\tau_{EB}(P)$ or $\tau_{EH}(P)$; by Corollary 1 we know that the stable expansions of $\tau_{EB}(P)$ and $\tau_{EH}(P)$ correspond with respect to objective ground atomic formulas.*

We show below (1) that if P is a safe program we can, given any interpretation w, construct an interpretation w' with no unnamed individuals such that $w \models_{T_{oga}} \tau_x(P)$ iff $w' \models_{T_{oga}} \tau_x(P)$ ($x \in \{EB, EH\}$). It follows that, given a formula ϕ, $\tau_x(P) \models_{T_{oga}} \phi$ iff for every interpretation w with no unnamed individuals it holds that whenever $w \models_{T_{oga}} \tau_x(P)$, $w \models_{T_{oga}} \phi$.

Clearly, given an interpretation w, if $w \models_{T_{oga}} \tau_{EH}(P)$, then $w \models_{T_{oga}} \tau_{EB}(P)$. Therefore, for any formula ϕ must hold that whenever $\tau_{EB}(P) \models_{T_{oga}} \phi$, it must be the case that $\tau_{EH}(P) \models_{T_{oga}} \phi$. To show the converse direction, we proceed by contradiction. Suppose that $\tau_{EH}(P) \models_{T_{oga}} \phi$, but $\tau_{EB}(P) \not\models_{T_{oga}} \phi$. Consider an interpretation with no unnamed individuals w such that $w \models_{T_{oga}} \tau_{EB}(P)$, but $w \not\models_{T_{oga}} \phi$. By (1), there is such an interpretation. We proceed to show that $w \models_{T_{oga}} \tau_{EH}(P)$, contradicting the assumption that $\tau_{EH}(P) \models_{T_{oga}} \phi$.

Suppose, on the contrary, that there is some formula $B \supset h \wedge \mathsf{L}h \in \tau_{EH}(P)$ such that $w \not\models_{T_{oga}} B \supset h \wedge \mathsf{L}h$. Since $B \supset h \in \tau_{EB}(P)$, it must be the case that $w \models_{T_{oga}} B$ and $w \not\models_{T_{oga}} \mathsf{L}h$, and thus $h \notin T_{oga}$. However, since $w \models_{T_{oga}} B$, it must be the case that for $B = b_1 \wedge \mathsf{L}b_1 \wedge \cdots \wedge b_m \wedge \mathsf{L}b_m \wedge \neg\mathsf{L}c_1 \wedge \cdots \wedge \neg\mathsf{L}c_n$, $b_1, \ldots, b_m \in T_{oga}$ and $c_1, \ldots, c_n \notin T_{oga}$. Now, since T is a stable expansion and $B \supset h \in \tau_{EB}(P)$, it must be the case that $h \in T_{oga}$. Therefore, $w \models_{T_{oga}} B \supset h \wedge \mathsf{L}h$. By contradiction we obtain that $w \models_{T_{oga}} \tau_{EH}(P)$.

(1) Given an interpretation $w = \langle U, \cdot^I\rangle$, we construct $w' = \langle U', \cdot^{I'}\rangle$ as follows: $U' = \{k \mid$

$k \in U$ and k is named}; for every predicate symbol p : $p^{I'} = \{t \mid t \in p^I$ and t does not contain an unnamed individual}; and, for every n-ary function symbol f and every n-ary tuple t of U', if $f^I(t) = k$ and k is named, then $f^{I'} = k$.

Consider $\tau_{EB}(P)$. Now, $\tau_{EB}(P)$ contains two kinds of formulas: UNA axioms and axioms of the form $(\forall) \bigwedge(b_i \wedge \mathsf{L}b_i) \wedge \bigwedge \neg\mathsf{L}c_j \supset h$. UNA axioms are obviously true in $\langle w', T_{oga}\rangle$, since they do not involve variables. Consider the open formula $\bigwedge(b_i \wedge \mathsf{L}b_i) \wedge \bigwedge \neg\mathsf{L}c_j \supset h$ and the variable assignment B for w. Since P is a safe program, every variable in h or in some c_j occurs in some b_i. Therefore, if B assigns any variable in the formula to an unnamed individual, the formula is trivially true in $\langle w, T_{oga}\rangle$ with respect to B. If B assigns all variables in the formula to named individuals, then the formula must be true in $\langle w, T_{oga}\rangle$ with respect to B iff the formula is true in $\langle w', T_{oga}\rangle$ with respect to B, by construction of w'. This argument is straightforwardly extended to formulas of $\tau_{EB}(P)$. The case of $\tau_{EH}(P)$ is analogous. Thus, we have established that $w \models_{T_{oga}} \tau_x(P)$ iff $w' \models_{T_{oga}} \tau_x(P)$.

Note that this result cannot be extended to the embeddings τ_{HP} and $\tau^{\vee}_{HP}$. Consider the logic program $P = \{q(x) \leftarrow p(x)\}$. $\tau_{HP}(P) = \{\forall x(q(x) \supset p(x))\}$ has one stable expansion T such that $\{\forall x(p(x) \supset q(x))\} \in T$. $\tau_{EB}(P) = \{\forall x(p(x) \wedge \mathsf{L}p(x) \supset q(x))\}$ has one stable expansion such that $\{\forall x(p(x) \supset q(x))\} \notin T$. This difference is caused by the fact that $\mathsf{L}p(x)$ will be false in case an unnamed individual is assigned to x by some variable assignment B for some model w. Similar for $\tau^{\vee}_{HP}$; the PIAs do not help, since they are only concerned with ground atoms and thus do not apply to unnamed individuals.

To see that the embeddings τ_{EB} and τ_{EH} differ when considering arbitrary logic programs, consider $P = \{p(x); q(x) \leftarrow p(x)\}$. The embedding $\tau_{EH}(P) = \{\forall x(p(x) \wedge \mathsf{L}p(x)), \forall x(p(x) \wedge \mathsf{L}p(x) \supset q(x) \wedge \mathsf{L}q(x))\}$ has one consistent stable expansion which includes $\forall x(q(x))$, while $\tau_{EB}(P) = \{\forall x(p(x)), \forall x(p(x) \wedge \mathsf{L}p(x) \supset q(x))\}$ has one consistent stable expansion which does not include $\forall x(q(x))$, because $\forall x(\mathsf{L}p(x))$ is not necessarily true when $\forall x(p(x))$ is true; in other words, the converse Barcan formula $(\mathsf{L}\forall x(\phi(x)) \supset \forall x(\mathsf{L}\phi(x)))$ is not universally valid, which is a property of FO-AEL under the any-name semantics [Kon91].

Proposition 7 *Given a ground disjunctive logic program P, $\tau^{\vee}_{HP}(P) \equiv \tau^{\vee}_{EB}(P)$.*

Proof 12 *By the PIA axioms $b_i \supset \mathsf{L}b_i$ we can eliminate the modal atoms of the form $\mathsf{L}b_i$ from the antecedents of the formulas in $\tau^{\vee}_{EB}(P)$ originating from rules in P. The remaining theory is logically equivalent to $\tau^{\vee}_{HP}(P)$ and thus the stable expansions correspond.*

Note that this result cannot be extended to the embedding $\tau^{\vee}_{EH}$; this embedding does not include the PIA axioms, and thus the argument used in the proof of the proposition does not apply.

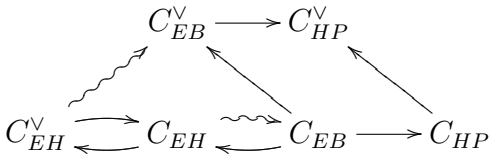


Figure 3.1: Relationships between sets of consequences; $C_x^{(\vee)}$ stands for $Cons_o(\tau_x^{(\vee)}(P))$, $\rightarrow$ stands for $\subseteq$, and $\rightsquigarrow$ stands for $\subseteq$ in case P is safe.

3.5.2 Relationships between Consequences of Embeddings

In order to investigate the relationships between the embeddings with respect to autoepistemic consequences, we first compare the embeddings with respect to their autoepistemic models. Recall that an autoepistemic model $\langle w, T\rangle$ consists of a first-order interpretation w and a belief set $T \subseteq \mathcal{L}_{\mathsf{L}}$. We now present the results on the relations between the FO-AEL models of the embeddings.

Proposition 8 *Given a normal logic program P and a model $\langle w, T\rangle$, if $w \models_T \tau_{EH}(P)$ then $w \models_T \tau_{EB}(P)$ and if $w \models_T \tau_{HP}(P)$ then $w \models_T \tau_{EB}(P)$.*

Proof 13 *(Sketch) Every axiom in $\tau_{EB}(P)$ is subsumed by an axiom in $\tau_{EH}(P)$; the first implication follows immediately.*

Consider a model $\langle w, T\rangle$, a variable assignment B such that $(w, B) \models_T \tau_{HP}(P)$, and an axiom in $\tau_{HP}(P)$ originating from a rule in P such that $(w, B) \models_T \bigwedge b_i \wedge \bigwedge \neg \mathsf{L} c_j$. Obviously, $(w, B) \models_T h$. Consider the corresponding axiom in $\tau_{EB}(P)$. Now, if $(w, B) \not\models_T \bigwedge(b_i \wedge \mathsf{L} b_i) \wedge \bigwedge \neg \mathsf{L} c_j$, then the axiom is trivially satisfied. Also, if $(w, B) \models_T \bigwedge(b_i \wedge \mathsf{L} b_i) \wedge \bigwedge \neg \mathsf{L} c_j$, the axiom is satisfied, since $(w, B) \models_T h$.

Proposition 9 *Given a disjunctive logic program P, a first-order interpretation w and a belief set T, if $w \models_T \tau^{\vee}_{HP}(P)$ then $w \models_T \tau^{\vee}_{EB}(P)$.*

Proof 14 *Follows from the proof of Proposition 8.*

Proposition 10 *Given a safe disjunctive logic program P, a first-order interpretation w, and a belief set T, if $w \models_T \tau^{\vee}_{EB}(P)$ then $w \models_T \tau^{\vee}_{EH}(P)$.*

Proof 15 *As in the proof of Proposition 6, we can restrict our attention to the case of named individuals. Now, because of the PIA axioms, we have that, for any interpretation $\langle w, T\rangle$ such that $w \models_T \tau^{\vee}_{EB}(P)$, variable assignment B, associated variable substitution β, and objective atomic formula α, whenever $(w, B) \models_T \alpha$, $w \models_T \mathsf{L}\alpha\beta$. Therefore, $w \models_T \tau^{\vee}_{EH}(P)$.*

We now consider the relative behavior of the embeddings with respect to autoepistemic consequences.

Theorem 12 *Let P be a (safe) normal (resp., disjunctive) logic program, and let $\tau_x^{(\vee)}$ and $\tau_y^{(\vee)}$ be embedding functions, for $x, y \in \{HP, EB, EH\}$. Then, the relations $Cons_o(\tau_x^{(\vee)}(P)) \subseteq Cons_o(\tau_y^{(\vee)}(P))$ hold as depicted in Figure 3.1 (with the respective provisos).*

Proof 16 *Recall that a formula ϕ is an autoepistemic consequence of a base set A if it is included in all stable expansions. Recall also that the stable expansions of all embeddings correspond with respect to objective ground atoms, by Corollary 1. From this and the fact that, by Proposition 2, every stable expansion T of an embedding $\tau_x^{(\vee)}(P)$ is determined by T_{oga} and $\tau_x^{(\vee)}(P)$, we can conclude that, if every model $\langle w, T_{oga}\rangle$ of an embedding $\tau_x^{(\vee)}(P)$ is also a model of another embedding $\tau_y^{(\vee)}(P)$, then $Cons_o(\tau_y^{(\vee)}(P)) \subseteq Cons_o(\tau_x^{(\vee)}(P))$. In the remainder, $C_x^{(\vee)}$ is short for $Cons_o(\tau_x^{(\vee)}(P))$.*

- *$C_{EH} \subseteq C_{EH}^{\vee}$ and $C_{EH}^{\vee} \subseteq C_{EH}$ follows from the fact that, for P is normal, $\tau_{EH}(P)$ and $\tau_{EH}^{\vee}(P)$ coincide.*
- *$C_{EH} \subseteq C_{EB}$ for P is safe follows from the fact that the stable expansions of τ_{EH} and τ_{EB} coincide, by Proposition 6.*
- *$C_{EB} \subseteq C_{EH}$ and $C_{EB} \subseteq C_{HP}$ follow from Proposition 8.*
- *$C_{EB} \subseteq C_{EB}^{\vee}$ and $C_{HP} \subseteq C_{HP}^{\vee}$ follow from the definition of the embeddings.*
- *$C_{EB}^{\vee} \subseteq C_{HP}^{\vee}$ follows from Proposition 9.*
- *Finally, $C_{EH}^{\vee} \subseteq C_{EB}^{\vee}$ for safe programs follows from Proposition 10.*

Notice that by the Theorems 7 and 11, all embeddings obviously agree on objective ground atomic consequences. Notice also that, if the stable expansions of two embeddings or combinations correspond with respect to a certain class of formulas, then the embeddings or combinations will also agree on autoepistemic consequences with respect to this class of formulas.

3.6 Combinations with First-Order Theories

In this section we explore correspondences between the embeddings presented in combinations with FO theories. In our simple setting, we define the combination of a program P and an FO theory Φ as

$$\iota_x^{(\vee)}(\Phi, P) = \Phi \cup \tau_x^{(\vee)}(P) \subseteq \mathcal{L}_{\mathsf{L}},^{2}$$

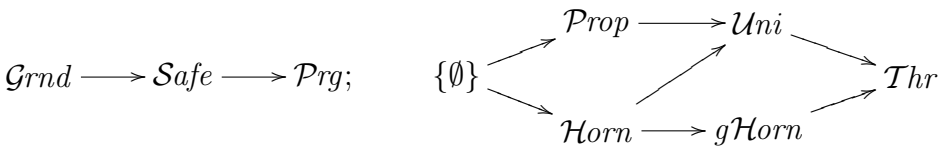
where $\Sigma_{\mathcal{L}_{\mathsf{L}}}$ is the union of the signatures Σ_Φ and Σ_P. Recall that we consider the any-name semantics, because of its more intuitive behavior (cf. Example 3).

[2]One could imagine other, more involved, embeddings of the classical theory. Such embeddings are a topic for future investigations.

In our analysis we furthermore use the following classes of programs and theories:

- $\mathcal{P}rg$, $\mathcal{S}afe$, and $\mathcal{G}rnd$ denote the classes of arbitrary, safe, and ground logic programs, respectively; and
- $\mathcal{T}hr$, $\mathcal{U}ni$, $g\mathcal{H}orn$, $\mathcal{H}orn$, $\mathcal{P}rop$, and $\{\emptyset\}$ denote the classes of arbitrary ($\mathcal{T}hr = 2^{\mathcal{L}}$), universal, generalized Horn,[3] Horn, propositional, and empty FO theories ($\mathcal{U}ni, g\mathcal{H}orn, \mathcal{H}orn, \mathcal{P}rop, \{\emptyset\} \subseteq 2^{\mathcal{L}}$).

Observe the following inclusions:



Our results on the relationships between stable expansions are summarized in Theorem 13 on page 39.

3.6.1 Relationships between Stable Expansions of Combinations

In this section, we present our results on the relations between stable expansions of combinations $\iota_x(\Phi, P)$, where P is a logic program and Φ is a first-order theory from a particular class of theories.

Proposition 11 *Given a safe normal logic program $P \in \mathcal{S}afe$ and a first-order theory $\Phi \subseteq \mathcal{L}$, $\iota_{EB}(\Phi, P) \equiv \iota_{EH}(\Phi, P)$.*

Proof 17 *We show that, given a first-order interpretation w and a stable expansion T of $\iota_x(\Phi, P)$, with $x \in \{EB, EH\}$, $\langle w, T\rangle$ is a model of $\iota_{EB}(\Phi, P)$ iff $\langle w, T\rangle$ is a model of $\iota_{EH}(\Phi, P)$.*

($\Leftarrow$) The axioms in $\iota_{EB}(\Phi, P)$ logically follow from (are entailed by) the axioms in $\iota_{EH}(\Phi, P)$, i.e. every model of $\iota_{EH}(\Phi, P)$ is a model of $\iota_{EB}(\Phi, P)$; recall that the only difference between $\iota_{EB}(\Phi, P)$ and $\iota_{EH}(\Phi, P)$ is the fact that rules in P are embedded as formulas of the form $B \supset h$ in $\tau_{EB}(P)$, whereas they are embedded as formulas of the form $B \supset h \wedge \mathsf{L}h$.

($\Rightarrow$) Recall that axioms in $\iota_{EH}(\Phi, P)$ which originate from rules in P have the following form:

$$(\forall) \bigwedge(b_i \wedge \mathsf{L}b_i) \wedge \bigwedge \neg \mathsf{L}c_j \supset (h \wedge \mathsf{L}h). \tag{3.4}$$

[3] Generalized Horn formulas are Horn formulas which additionally allow existentially quantified variables in the consequent of the material implication.

• *Let T be a stable expansion of $\iota_{EH}(\Phi, P)$. Consider a variable assignment B of w, and an associated variable substitution β. Consider an axiom of the form (3.4) such that $(w, B) \models_T \bigwedge(b_i \wedge \mathsf{L}b_i) \wedge \bigwedge \neg \mathsf{L}c_j$. Because the positive modal atoms in $\bigwedge(b_i \wedge \mathsf{L}b_i) \wedge \bigwedge \neg \mathsf{L}c_j$ are true in $\langle w, T\rangle$ with respect to B, and all variable occurrences in these modal atoms are free, all variables in (3.4) are assigned to named individuals by B. Therefore, $b_i\beta$, $c_j\beta$, and $h\beta$ are closed. It follows that $b_i\beta \in T, c_j\beta \notin T$, and thus, since (3.4) is in $\iota_{EH}(\Phi, P)$, $h\beta \in T$.*

• *The case of T being a stable expansion of $\iota_{EB}(\Phi, P)$ is similar.*

To see the difference between $\iota_{EB}(\Phi, P)$ and $\iota_{EH}(\Phi, P)$ for arbitrary logic programs, consider a language with two unary predicate symbols p, q and one constant symbol a. Consider the FO theory $\Phi = \{\neg p(a), \exists x(p(x))\}$ and the logic program $P = \{q(x) \leftarrow\}$. Then, $\iota_{EB}(\Phi, P)$ has one consistent stable expansion, while $\iota_{EH}(\Phi, P)$ has only one stable expansion which is the inconsistent expansion, because $\tau_{EH}(P) = \{\forall x(q(x) \wedge \mathsf{L}q(x))\}$, which means that every individual must be named and must be in q^w for every model $\langle w, T\rangle$ of $\iota_{EH}(\Phi, P)$. However, Φ requires that there is an individual $k \in p^w$ such that $k \neq a^w$, and because there is at most one named individual, represented by a, there must be an unnamed individual in every model w of Φ. Summarizing, $\tau_{EH}(P)$ requires every individual in every model to be named and Φ requires at least one unnamed individual in every model; this is a contradiction.

Proposition 12 *Given a normal logic program P and a first-order theory $\Phi \in \mathcal{U}ni$, $\iota_{EB}(\Phi, P) \equiv_{og} \iota_{EH}(\Phi, P)$.*

Proof 18 *If Φ is universal, we can disregard models which have unnamed individuals for determining ground entailment. The equivalence $\iota_{EB}(\Phi, P) \equiv_{og} \iota_{EH}(\Phi, P)$ now follows directly from the proof Proposition 11, with the difference that we can restrict ourselves to the case that B assigns variables in h only to named individuals, not because of the modal atoms in the antecedent (the rules are not necessarily safe), but because we are only interested in ground entailment.*

Proposition 13 *Given a normal logic program P and a first-order theory $\Phi \in \mathcal{H}orn$, then $\iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$.*

Proof 19 *If Φ is an equality-free Horn theory, there is no syntactical difference between the formulas in Φ and the formulas in τ_{HP} which originate from positive rules. It is well known that unnamed individuals do not play a role in inference of ground atomic formulas from Horn theories. The lack of UNA axioms for the names in Σ_Φ does not affect the truth of positive modal atoms in the antecedent of the formulas in τ_{EB} (by the any-name semantics). The correspondence follows straightforwardly from Corollary 1.*

To see why the correspondence holds for Horn formulas with equality, consider a formula $\forall(t_1 = t_2) \in \Phi$, with t_1 and t_2 terms. Obviously, all substitution instances of the formula are first-order entailed by $\iota_x(\Phi, P)$; and, since the formula is universal, we do

not need to take unnamed individuals into account. Imagine that $a = b$, with a, b ground terms, is first-order entailed by $\iota_x(\Phi, P)$, then if $p(a)$ is entailed, obviously $p(b)$ is also entailed. Thus, if $p(a)$ and $a = b$ are included in a stable expansion, then so is $p(b)$. $\iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$ for Φ is Horn with equality follows.

It turns out that $\iota_{HP}(\Phi, P)$ and $\iota_{EB}(\Phi, P)$ already differ with respect to objective ground atoms in case $\Phi \in \mathcal{P}rop$. Consider the theory $\Phi = \{p \vee q\}$ and the logic program $P = \{r \leftarrow p; r \leftarrow q\}$. $\iota_{HP}(\Phi, P)$ has one stable expansion which includes r; $\iota_{EB}(\Phi, P)$ also has one stable expansion, but it does not include r.

Proposition 14 *Given a ground normal logic program P and a first-order theory $\Phi \in g\mathcal{H}orn$, then $\iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$.*

Proof 20 *Let Φ' be a Skolemization of Φ. Φ' is Horn and thus, by Proposition 13, $\iota_{HP}(\Phi', P) \equiv_{oga} \iota_{EB}(\Phi', P)$. It remains to be shown that if T is a stable expansion of $\iota_x(\Phi, P) \subseteq \mathcal{L}_{\mathsf{L}}$, then there exists a stable expansion T' of $\iota_x(\Phi', P) \subseteq \mathcal{L}'_{\mathsf{L}}$ such that $T \cap \mathcal{L}_{ga} = T' \cap \mathcal{L}_{ga}$.*

We now transform $\iota_x(\Phi, P)$ to a first-order theory J with respect to T_{oga} in the following way: we treat each combination of the modal symbol L and a predicate symbol p as a new predicate $\mathsf{L}p$; we add the atom $\mathsf{L}\alpha$ to J if $\alpha \in T_{oga}$, and $\neg\mathsf{L}\alpha$ otherwise, for each $\alpha \in \mathcal{L}_{ga}$. Intuitively, J "fixes" the value of $\mathsf{L}\alpha$ according to T. We now have that for any objective ground atom $\alpha \in \mathcal{L}_{ga}$, $\iota_x(\Phi, P) \models_T \alpha$ iff $J \models \alpha$.

We obtain J' from $\iota_x(\Phi', P)$ in the same way, and have that $\iota_x(\Phi', P) \models_T \alpha$ iff $J' \models \alpha$.

J' is a Skolemized version of J, and thus equi-satisfiable. This remains true if we add a negated objective ground atom $\neg\alpha$, with $\alpha \in \mathcal{L}_{ga}$, to J or J'. We can conclude that

$$\iota_x(\Phi, P) \models_{T_{oga}} \alpha \text{ iff } \iota_x(\Phi', P) \models_{T_{oga}} \alpha.$$

Furthermore, consistency is preserved, and thus we can extend T_{oga} to a set of ground atoms T'_{oga} over $\mathcal{L}'_{\mathsf{L}}$ by including ground atoms with Skolem terms such that $T'_{oga} = T' \cap \mathcal{L}'_{ga}$ with T' the stable expansion of $\iota_x(\Phi', P)$.

Proposition 15 *Given a ground disjunctive logic program $P \in \mathcal{G}rnd$ and a first-order theory $\Phi \subseteq \mathcal{L}$, $\iota^{\vee}_{HP}(\Phi, P) \equiv \iota^{\vee}_{EB}(\Phi, P)$.*

Proof 21 *Follows from the proof of Proposition 7.*

Proposition 16 *Given a disjunctive logic program P and a first-order theory $\Phi \in \mathcal{P}rop$, then $\iota^{\vee}_{HP}(\Phi, P) \equiv_{og} \iota^{\vee}_{EB}(\Phi, P)$.*

Proof 22 *Follows from the proof of Proposition 5 and the fact that there is a PIA for each propositional symbol which occurs in any formula in $\tau^{\vee}_{HP}(P)$ or $\tau^{\vee}_{EB}(P)$.*

$\Phi \backslash P$	$\mathcal{P}rg$	$\mathcal{S}afe$	$\mathcal{G}rnd$
$\mathcal{T}hr$	$\iota_{EH} \equiv \iota^{\vee}_{EH}$	$\iota_{EB} \equiv \iota_{EH}$	$\iota^{\vee}_{HP} \equiv \iota^{\vee}_{EB}$
$\mathcal{U}ni$	$\iota_{EB} \equiv_{og} \iota_{EH}$		
$g\mathcal{H}orn$			$\iota_{HP} \equiv_{oga} \iota_{EB}$
$\mathcal{H}orn$	$\iota_{HP} \equiv_{oga} \iota_{EB}$		
$\mathcal{P}rop$	$\iota^{\vee}_{HP} \equiv_{og} \iota^{\vee}_{EB}$		
$\{\emptyset\}$	$\iota_{HP} \equiv_{oga} \iota_{EB} \equiv_{oga}$ $\iota_{EH} \equiv_{oga} \iota^{\vee}_{HP} \equiv_{oga}$ $\iota^{\vee}_{EB} \equiv_{oga} \iota^{\vee}_{EH}$		

Table 3.1: Correspondence between expansions of combinations; $\iota^{(\vee)}_{x}$ is short for $\iota^{(\vee)}_{x}(\Phi, P)$.

To show that the correspondence between $\iota^{\vee}_{HP}(\Phi, P)$ and $\iota^{\vee}_{EB}(\Phi, P)$ does not hold for arbitrary $\Phi \in \mathcal{L}$, consider the FO theory $\Phi = \{\forall x(p(x)), (\forall x(q(x)) \supset r)\}$ and the logic program $P = \{q(x) \leftarrow p(x)\}$. $\iota^{\vee}_{HP}(\Phi, P)$ has one stable expansion which includes r; $\iota^{\vee}_{EB}(\Phi, P)$ also has one stable expansion, but it does not include r.

The following Theorem summarizes our results on the relationships between expansions.

Theorem 13 *Let P be a normal (resp., disjunctive) logic program and Φ be a first-order theory. Then, the relations depicted in Table 3.1 (with the respective provisos) hold, providing P and Φ belong to the classes listed there.*

Proof 23 $\Phi \in \mathcal{T}hr, P \in \mathcal{P}rg : \iota_{EH}(\Phi, P) \equiv \iota^{\vee}_{EH}(\Phi, P)$*, by the fact that the embeddings τ_{EH} and $\tau^{\vee}_{EH}$ are equivalent for normal programs.*

$\Phi \in \mathcal{U}ni, P \in \mathcal{P}rg : \iota_{EB}(\Phi, P) \equiv_{og} \iota_{EH}(\Phi, P)$*, by Proposition 12.*

$\Phi \in \mathcal{H}orn, P \in \mathcal{P}rg : \iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$*, by Proposition 13.*

$\Phi \in \mathcal{P}rop, P \in \mathcal{P}rg : \iota^{\vee}_{HP}(\Phi, P) \equiv_{og} \iota^{\vee}_{EB}(\Phi, P)$*, by Proposition 16.*

$\Phi = \emptyset, P \in \mathcal{P}rg : \iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P) \equiv_{oga} \iota_{EH}(\Phi, P) \equiv_{oga} \iota^{\vee}_{HP}(\Phi, P) \equiv_{oga} \iota^{\vee}_{EB}(\Phi, P) \equiv_{oga} \iota^{\vee}_{EH}(\Phi, P)$*, by Corollary 1.*

$\Phi \in \mathcal{T}hr, P \in \mathcal{S}afe : \iota_{EB}(\Phi, P) \equiv \iota_{EH}(\Phi, P)$*, by Proposition 11.*

$\Phi \in \mathcal{T}hr, P \in \mathcal{G}rnd : \iota^{\vee}_{HP}(\Phi, P) \equiv \iota^{\vee}_{EB}(\Phi, P)$*, by Proposition 15.*

$\Phi \in g\mathcal{H}orn, P \in \mathcal{G}rnd : \iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$*, by Proposition 14.*

In case the standard names assumption applies, we additionally obtain the following result.

Proposition 17 *Given a normal logic program P and an FO theory Φ, if the standard names assumption applies, then $\iota_{EB}(\Phi, P) \equiv \iota_{EH}(\Phi, P)$.*

Proof 24 *Follows from the proof of Proposition 6.*

Note that the differences between the embeddings, shown in the preceding subsections, do not depend on the use of negation in the program. Generally speaking, the differences originate from the positive use of the modal operator in the antecedent and the consequent, and the use of the PIA axioms. We illustrate the use of the results with an example.

Example 7 *Consider the logic program $P = \{q(a);\, p(x);\, r(x) \leftarrow not\ s(x),\, p(x)\}$ from Example 5. P is neither safe nor ground; to determine correspondence between embeddings, we need to use the first column of Table 3.1. Since P is normal, all equations in this column are applicable. We have that $\tau_{EB}(P) \equiv_{og} \tau_{EH}(P)$ and $\tau^{\vee}_{HP}(P) \equiv_{og} \tau^{\vee}_{EB}(P)$. Let Φ be a Horn theory, then $\iota_{HP}(\Phi, P) \equiv_{oga} \iota_{EH}(\Phi, P) \equiv_{oga} \iota_{EB}(\Phi, P)$ and $\iota_{EH}(\Phi, P) \equiv \iota^{\vee}_{EH}(\Phi, P)$.*

Additionally, since autoepistemic consequence is defined through the intersection of all stable expansions, we can conclude that $\tau_{EB}(P)$ and $\tau_{EH}(P)$, and also $\tau^{\vee}_{HP}(P)$ and $\tau^{\vee}_{EB}(P)$, agree on objective ground autoepistemic consequence and that $\iota_{HP}(\Phi, P)$, $\iota_{EH}(\Phi, P)$, and $\iota_{EB}(\Phi, P)$ agree on objective ground atomic autoepistemic consequence.

Most of the relations given in Figure 3.1 do not extend to combinations with FO theories. Consider, e.g., $P = \{r \leftarrow not\ p, not\ q\}$ and $\Phi = \{p \vee q\}$. Then, $\tau_{HP}(P) = \{\neg \mathsf{L}p \wedge \neg \mathsf{L}q \supset r\}$ and $\tau^{\vee}_{HP}(P) = \{\neg \mathsf{L}p \wedge \neg \mathsf{L}q \supset r\} \cup PIA_{\Sigma_P}$ both have a single stable expansion; the stable expansions of $\tau_{HP}(P)$ and $\tau_{HP}(P)$ both contain $\neg \mathsf{L}p$, $\neg \mathsf{L}q$ and r. The combination $\tau_{HP}(P) \cup \Phi$ has one stable expansion which includes $\neg \mathsf{L}p$, $\neg \mathsf{L}q$, and r; $\tau^{\vee}_{HP}(P) \cup \Phi$ has two stable expansions $\{p, \mathsf{L}p, \neg \mathsf{L}q, \ldots\}$ and $\{q, \mathsf{L}q, \neg \mathsf{L}p, \ldots\}$, neither of which includes r. Thus, r is an autoepistemic consequence of $\iota_{HP}(\Phi, P)$, but not of $\iota^{\vee}_{HP}(\Phi, P)$. Therefore, $Cons_o(\iota_{HP}(\Phi, P)) \not\subseteq Cons_o(\iota^{\vee}_{HP}(\Phi, P))$.

Using the results obtained in this section, we can make a number of observations about the embeddings:

(1) The stable expansions of embeddings with and embeddings without the PIAs generally tend to differ. However, we can note that the former are generally stronger in terms of the number of objective autoepistemic consequences (cf. Figure 3.1 and Example 6).

(2) The embeddings τ_{HP} and $\tau^{\vee}_{HP}$ are generally the strongest in terms of consequences (see Figure 3.1), when comparing to other and embeddings without and with PIAs, respectively. They allow to derive the contrapositive of rules (cf. Example 4) and the bodies of rules are applicable to unnamed individuals, whereas the antecedents of the axioms in

the other embeddings are only applicable to named individuals, because of the positive modal atoms in the bodies.

(3) For unsafe programs, the embeddings τ_{EH} and $\tau_{EH}^{\vee}$ are generally not comparable with the others; embeddings of unsafe rules may result in axioms of form $\forall x \mathsf{L} p(x)$ (cf. Example 5), which require all individuals to be named.

(4) In case the programs are safe, or one assumes that all individuals are named, τ_{EB} and τ_{EH} coincide.

Special care needs to be taken if one selects an embedding which includes the PIA axioms ($\tau_{HP}^{\vee}$ and $\tau_{EB}^{\vee}$), since they ensure that either α or $\neg\alpha$ (or both for the inconsistent expansion) is included in every stable expansion, for every ground atom of Σ_P. Note that the PIA axioms have no effect when considering individuals which do not correspond to ground terms in Σ_P.

We conclude this section with an example which demonstrates possibly unexpected or undesirable effects of the UNA axioms in their interaction with an FO theory.

Example 8 *Consider $P = \{p(a); p(b)\}$ and $\Phi = \{a \neq b \supset r\}$. Then, r is included in any stable expansion of $\Phi \cup \tau_x(P)$, for any τ_x, in view of the UNA axioms.*

Special care needs to be taken if one selects an embedding which includes the PIA axioms ($\tau_{HP}^{\vee}$ and $\tau_{EB}^{\vee}$), since they ensure that either α or $\neg\alpha$ (or both for the inconsistent expansion) is included in every stable expansion, for every ground atom of Σ_P. Note that the PIA axioms have no effect when considering individuals which do not correspond to ground terms in Σ_P.

3.7 Related and Future Work

In this chapter, we have studied the combination of logic programs and ontologies (FO theories) using embeddings in a unifying formalism (FO-AEL). One could imagine, in contrast, extensions of semantics for logic programs or ontologies to incorporate (parts of) the other formalism. One such extension of logic programming semantics is that of open domains [GP93, VBDDS97, HVNV]. Such extended semantics can be used to accommodate incomplete knowledge, an important aspect of ontology languages. Nonmonotonic extensions of description logics (an FO-based formalism suitable for ontologies) have been presented in the literature [BH95, DNR02, BLW06]. Such approaches might be extended to accommodate logic programs.

We have investigated basic correspondences between different embeddings of non-ground programs in FO-AEL, and simple combinations with FO theories. Choosing different embeddings for logic programs, but also possibly different embeddings for first-order theories, will give rise to different properties of such combinations [BEPT06]. In future work, we will investigate these properties, as well as the relationship with exist-

ing approaches to combine logic programs and classical theories [HPSBT05, ELST04b, Ros06b].

So far, we have only considered equality-free logic programs. We expect that equality in rule bodies will not pose a problem, since still only the trivial equalities are derivable. Allowing equality in rule heads is a topic for further research.

We expect that the proposed combinations of rules and ontologies based on FO-AEL will give rise to the definition of novel decidable fragments and for sound (but possibly incomplete) algorithms for specific reasoning tasks for such combinations. Additionally, we will consider other nonmonotonic logics (e.g., default logic and circumscription) as formalisms for combining logic programs and classical knowledge bases.

Chapter 4

Logical Foundations of (e)RDF(S): Complexity and Reasoning

4.1 Introduction

The Resource Description Framework RDF [KC04], together with its vocabulary description language RDFS [BG04], constitutes the basic language for the semantic Web. More expressive semantic Web languages such as the description logic-based OWL DL [PSHH04a] and future semantic Web rule languages[1] are supposed to extend it. However, certain properties of the RDF semantics posed layering problems in the definition of OWL DL [HPSvH03]; it was decided to extend only a part of RDF. This has led to a situation in which the user communities of RDF and OWL DL are increasingly diverging, leading to a fragmentation of the semantic Web. There is a possibility that a similar situation will occur if a possible future rules language for the semantic Web does not adequately account for RDF(S).

Research has been done to uncover some of the formal properties of RDF (e.g. [Hor05b, GHM04, BFT05]). However, so far little research has been done into the formal relationships between RDF and the logical language paradigms of description logics and logic programming.[2] Therefore, we deem it worthwhile to investigate these relationships, in order to facilitate the RDF-compatibility of a future logic-based rules language for the semantic Web, and to see whether the RDF and description logic worlds can be brought closer together. Additional benefits of such an investigation include the possible use of techniques from logic programming and description logics for reasoning with RDF(S).

The RDF semantics specification [Hay04] defines three increasingly expressive kinds of entailment, namely *simple*, *RDF* and *RDFS entailment*. Furthermore, it describes *extensional RDFS* (eRDFS) entailment as a possible extension of RDFS entailment which is more in line with description logic-like languages. We refer to these kinds of entailment

[1] `http://www.w3.org/2005/rules/`

[2] A notable exception is [BFT05].

as *entailment regimes*.

To investigate the relationship between RDF and logic we embed the various RDF entailment regimes in F-Logic [KLW95a], which is a syntactic extension of first-order logic with object oriented modeling constructs. It turns out that the attribute value construct in F-Logic is exactly equivalent to the triple construct in RDF, and the typing (class membership) construct in F-Logic is very close in spirit to the one in RDF. Additionally, F-Logic, like RDFS, allows to use the same identifier as a class, instance, or property identifier, while still having a standard first-order logic-based semantics.

These embeddings can be used for RDF reasoning using Datalog engines. Furthermore, they lead to several novel complexity results about RDF; see Table 4.3 on page 57 for an overview of the complexity results for the various entailment regimes.

We then show the embedding of a large subset of *extensional* RDFS in FOL, and we show that it can be embedded in the tractable description logic (contextual) $DL\text{-}Lite_{\mathcal{R}}$ [CDGL$^+$06].

Finally, we define a notion of conjunctive queries over RDF graphs, and establish the data complexity of query answering for the respective entailment regimes.

The structure of the remainder of the chapter is as follows. In Section 4.2 we review F-Logic, $DL\text{-}Lite_{\mathcal{R}}$, and RDF. In Section 4.3 we define embeddings of RDF in F-Logic and FOL, and demonstrate the relationship with $DL\text{-}Lite_{\mathcal{R}}$. In Section 4.4 we use these embeddings and correspondences to obtain several novel complexity results for RDF. In Section 4.5 we define conjunctive query answering in RDF and exhibit its complexity. We discuss some implications of the results in this chapter, and compare it with related work, in Section 4.6. We conclude the chapter and outline future work in Section 4.7.

4.2 Preliminaries

4.2.1 F-Logic

F-Logic[3] extends first-order logic with constructs for object-oriented modeling (we use the object typing and attribute value construct), while staying in a first-order semantic framework.

The signature of an F-language $\mathcal{L}$ is of the form $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F}$ and $\mathcal{P}$ disjoint sets of function and predicate symbols, each with an associated arity $n \geq 0$. Let $\mathcal{V}$ be a set of variable symbols disjoint from $\mathcal{F}$ and $\mathcal{P}$. Terms and atomic formulas are defined in the usual way; $\bot$ is an atomic formula. A molecule in F-Logic is one of the following: (i) an *is-a* molecule of the form $C : D$, which states that an individual C is of the type D,

[3]Note that F-Logic is also often used as an extension of nonmonotonic logic programming; however, we follow the original definition which is strictly first-order.

or (ii) a *data molecule* of the form $C[D \twoheadrightarrow E]$ which states that an individual C has an attribute D with the value E, where C, D and E terms,. A molecule is *ground* if it does not contain variable symbols.

Formulas of an F-language $\mathcal{L}$ are either atomic formulas, molecules, or compound formulas, which are constructed in the usual way from atomic formulas, molecules, the logical connectives $\neg$, $\wedge$, $\vee$ and $\supset$, the quantifiers $\exists$ and $\forall$, and the auxiliary symbols ')' and '('. An F-Logic theory $\Phi \subseteq \mathcal{L}$ is a set of formulas.

F-Logic Horn formulas are of the form $(\forall)B_1 \wedge \ldots \wedge B_n \supset H$, with $B_1, \ldots, B_n, H$ atomic formulas or molecules. F-Logic Datalog formulas are F-Logic Horn formulas with no function symbols of arity higher than 0 such that every variable in H occurs in some equality-free $B_1, \ldots, B_n$. F-Logic Horn and Datalog theories are sets of F-Logic Horn and Datalog formulas, respectively.

An *F-structure* is a tuple $\mathbf{I} = \langle U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$, where U is a non-empty, countable set (the domain) and $\in_U$ is a binary relation over U. An n-ary function symbol $f \in \mathcal{F}$ is interpreted as a function over the domain U: $\mathbf{I}_F(f) : U^n \rightarrow U$. An n-ary predicate symbol $p \in \mathcal{P}$ is interpreted as a relation over the domain U: $\mathbf{I}_P(p) \subseteq U^n$. $\mathbf{I}_{\twoheadrightarrow}$ associates a binary relation over U with each $k \in U$: $\mathbf{I}_{\twoheadrightarrow}(k) \subseteq U \times U$. Variable assignments are defined as usual.

Given an F-structure $\mathbf{I}$ of an F-language $\mathcal{L}$, a variable assignment B, and a term t of $\mathcal{L}$, $t^{\mathbf{I},B}$ is defined as: $x^{\mathbf{I},B} = x^B$ for $x \in \mathcal{V}$ and $t^{\mathbf{I},B} = \mathbf{I}_F(f)(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B})$ for t of the form $f(t_1, \ldots, t_n)$, with $f \in \mathcal{F}$ an n-ary function symbol and $t_1, \ldots, t_n$ terms.

Satisfaction of atomic formulas and molecules ϕ in $\mathbf{I}$, given the variable assignment B, denoted $(\mathbf{I}, B) \models_{\mathsf{f}} \phi$, is defined as: $(\mathbf{I}, B) \not\models_{\mathsf{f}} \bot$, $(\mathbf{I}, B) \models_{\mathsf{f}} p(t_1, \ldots, t_n)$ iff $(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B}) \in \mathbf{I}_P(p)$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 : t_2$ iff $t_1^{\mathbf{I},B} \in_U t_2^{\mathbf{I},B}$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1[t_2 \twoheadrightarrow t_3]$ iff $\langle t_1^{\mathbf{I},B}, t_3^{\mathbf{I},B} \rangle \in \mathbf{I}_{\twoheadrightarrow}(t_2^{\mathbf{I},B})$, and $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 = t_2$ iff $t_1^{\mathbf{I},B} = t_2^{\mathbf{I},B}$. This extends to arbitrary formulas in the usual way.

The notion of a model is defined in the usual way. A theory $\Phi \subseteq \mathcal{L}$ *F-entails* a formula $\phi \in \mathcal{L}$, denoted $\Phi \models_{\mathsf{f}} \phi$, if for all F-structures $\mathbf{I}$ such that $\mathbf{I} \models_{\mathsf{f}} \Phi$, $\mathbf{I} \models_{\mathsf{f}} \phi$.

4.2.2 FOL and DL-$Lite_{\mathcal{R}}$

Classical first-order logic (FOL) is F-Logic without molecules. *Contextual first-order logic* [CKW93] (contextual FOL) is classical FOL where $\mathcal{F}$ and $\mathcal{P}$ are not required to be disjoint, function and predicate symbols do not have associated arities, and for every structure $\mathbf{I} = \langle U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$ it is the case that $\mathbf{I}_F$ assigns a function $\mathbf{I}_F(f) : U^i \rightarrow U$ to every $f \in \mathcal{F}$ and $\mathbf{I}_P$ assigns a relation $\mathbf{I}_P(p) \subseteq U^i$ to every $p \in \mathcal{P}$, for every nonnegative integer i. We denote satisfaction and entailment in classical and contextual FOL using the symbols $\models$ and $\models_c$, respectively.

F-Logic can be straightforwardly embedded in FOL, as shown in the following proposition.

Proposition 18 *Let Φ and ϕ be an F-Logic theory and formula which do not contain the binary and ternary predicate symbols isa and $data$, respectively, and let Φ' and ϕ' be the FOL theory and formula obtained from Φ and ϕ by replacing every is-a molecule $a : b$ with the atomic formula $isa(a, b)$ and every data molecule $a[b \twoheadrightarrow c]$ with the atomic formula $data(a, b, c)$. Then,*
– *Φ has a model iff Φ' has a model and*
– *$\Phi \models_{\mathsf{f}} \phi$ iff $\Phi' \models \phi'$.*

$DL\text{-}Lite_{\mathcal{R}}$ [CDGL+06] is a description logic (DL) with certain desirable computational properties; most reasoning tasks are polynomial, and query answering has LogSpace data complexity.

A classical (resp., contextual) $DL\text{-}Lite_{\mathcal{R}}$ language consists of pairwise disjoint (resp., possibly non-disjoint) sets of concept ($\mathcal{C}$), role ($\mathcal{R}$), and individual ($\mathcal{F}$) identifiers. Concepts and roles in $DL\text{-}Lite_{\mathcal{R}}$ are defined as follows:

$$\begin{array}{rcl} C_l & \longrightarrow & A \mid \exists R \\ C_r & \longrightarrow & A \mid \exists R \mid \neg A \mid \neg \exists R \\ R, R' & \longrightarrow & P \mid P^- \end{array}$$

with $A \in \mathcal{C}$, $P \in \mathcal{R}$, C_l (resp., C_r) a left- (resp., right-)hand side concept, and $R, R' \in \mathcal{R}$ roles.

A $DL\text{-}Lite_{\mathcal{R}}$ knowledge base $\mathcal{K}$ consists of a TBox $\mathcal{T}$ which is a set of inclusion axioms of the forms $C_l \sqsubseteq C_r$ and $R \sqsubseteq R'$, and an ABox $\mathcal{A}$ which is a set of concept and role membership assertions of the forms $A(a)$ and $P(a_1, a_2)$, with $a, a_1, a_2 \in \mathcal{F}$.

We define the semantics of $DL\text{-}Lite_{\mathcal{R}}$ through a translation to classical (resp., contextual) FOL, using the mapping function π, which is defined in Table 4.1; π extends naturally to sets of axioms and assertions.

$\pi(A, X)$	$=$	$A(X)$	$\pi(C_l \sqsubseteq C_r)$	$=$	$\forall x(\pi(C_l, x) \supset \pi(C_r, x))$
$\pi(P, X, Y)$	$=$	$P(X, Y)$	$\pi(R_1 \sqsubseteq R_2)$	$=$	$\forall x, y(\pi(R_1, x, y) \supset \pi(R_2, x, y))$
$\pi(P^-, X, Y)$	$=$	$P(Y, X)$			
$\pi(\exists R, X)$	$=$	$\exists y(R(X, y))$	$\pi(A(a))$	$=$	$A(a)$
$\pi(\neg A, X)$	$=$	$\neg\pi(A, X)$	$\pi(P(a_1, a_2))$	$=$	$P(a_1, a_2)$
$\pi(\neg\exists R, X)$	$=$	$\neg\exists y(R(X, y))$			
y is a new a variable					

Table 4.1: Mapping $DL\text{-}Lite_{\mathcal{R}}$ to FOL

Given a classical (resp., contextual) $DL\text{-}Lite_{\mathcal{R}}$ knowledge base $\mathcal{K}$, the classical (resp., contextual) *FOL equivalent* of $\mathcal{K}$ is the theory $\Phi = \pi(\mathcal{K}) = \pi(\mathcal{T}) \cup \pi(\mathcal{A})$.

4.2.3 RDF

RDF is a data language, where the central notion is the *graph*, which is a set of *triples* of the form $\langle s, p, o\rangle$; s is the *subject*, p is the *predicate*, and o is the *object* of the triple.

A *vocabulary* $V = \langle \mathcal{F}, \mathcal{PL}, \mathcal{TL}, \mathcal{B}\rangle$ consists of a set $\mathcal{F}$ of URI references, a set $\mathcal{PL}$ of plain literals (i.e. Unicode character strings with an optional language tag), a set $\mathcal{TL}$ of typed literals (i.e. pairs (s, u) of a Unicode string s and a URI u denoting a datatype), and a set $\mathcal{B}$ of blank nodes (i.e. existentially quantified variables); see [KC04, Sections 6.4, 6.5, 6.6] for more details about the specific form of these symbols. Terms are URI references, plain or typed literals, or blank nodes. A *graph* S of a vocabulary V is a set of *triples* $\langle s, p, o\rangle$, with $s, p, o \in \mathcal{F} \cup \mathcal{PL} \cup \mathcal{TL} \cup \mathcal{B}$.[4] With $bl(\langle s, p, o\rangle)$ (resp., $bl(S)$) we denote the set of blank nodes in $\langle s, p, o\rangle$ (resp., S). A triple $\langle s, p, o\rangle$ (resp., graph S) is *ground* if $bl(\langle s, p, o\rangle) = \emptyset$ (resp., $bl(S) = \emptyset$).

An interpretation of a vocabulary V is a tuple $I = \langle U_R, U_P, U_L, I_{\mathcal{F}}, I_L, I_{ext}\rangle$, where U_R is a countable non-empty set, called the domain, U_P is a countable set of properties, $U_L \subseteq U_R$ is a countable set of literal values with $\mathcal{PL} \subseteq U_L$, $I_{\mathcal{F}}$ is an interpretation function for URI references $I_{\mathcal{F}} : \mathcal{F} \to U_R \cup U_P$, I_L is an interpretation function for typed literals $I_L : \mathcal{TL} \to U_R$, and I_{ext} is an *extension function* $I_{ext} : U_P \to 2^{(U_R \times U_R)}$.

Given an interpretation I of a vocabulary V and a subset of the blank nodes $\mathcal{B}' \subseteq \mathcal{B}$, we define a mapping $A : \mathcal{B}' \to U_R$ which is used to interpret blank nodes. For a term t we define $t^{I,A}$ as: (a) if $t \in \mathcal{F}$, then $t^{I,A} = I_{\mathcal{F}}(t)$, (b) if $t \in \mathcal{PL}$, then $t^{I,A} = t$, (c) if $t \in \mathcal{TL}$, then $t^{I,A} = I_L(t)$, and (d) if $t \in \mathcal{B}'$, then $t^{I,A} = A(t)$.

An interpretation I *satisfies* a triple $\langle s, p, o\rangle$ with respect to a mapping $A : \mathcal{B}' \to U_R$, with $bl(\langle s, p, o\rangle) \subseteq \mathcal{B}'$, denoted $(I, A) \models \langle s, p, o\rangle$, if $p^{I,A} \in U_P$ and $\langle s^{I,A}, o^{I,A}\rangle \in I_{ext}(p^{I,A})$. I satisfies a graph S with respect to a mapping $A : bl(S) \to U_R$, denoted $(I, A) \models S$, if $(I, A) \models \langle s, p, o\rangle$ for every $\langle s, p, o\rangle \in S$. An interpretation I is a *model* of a graph S, denoted $I \models S$, if $(I, A) \models S$ for some $A : bl(S) \to U_R$.

Any interpretation is an *s-interpretation* (simple interpretation). An interpretation I is an *rdf-* (resp., *rdfs-*, *erdfs-*)interpretation if it interprets the RDF (resp., RDFS) vocabulary, satisfies the RDF (resp., RDFS) axiomatic triples, and satisfies a number of conditions, as specified in [Hay04].

An RDF graph S s-(resp., rdf-, rdfs-, or erdfs-)entails an RDF graph E if every s-(resp., rdf-, rdfs-, or erdfs-)interpretation which is a model of S is also a model of E. We refer to these kinds of entailment as *entailment regimes*, and use the symbol $\models_x$, with $x \in \{s, rdf, rdfs, erdfs\}$, to denote entailment under the respective regimes.

Intuitively, the difference between the RDFS and eRDFS entailment regimes is that for the latter, whenever an ontological relation (e.g. subclass or property domain) implic-

[4] Note that we allow literals in subject (s), and literals and blank nodes in predicate (p) positions, whereas the RDF specification [KC04] does not. Nonetheless, our results immediately apply to standard RDF graphs as defined in [KC04].

itly holds in an interpretation, the corresponding RDF statement (`subClassOf`, `domain`) must be true, whereas this is not always the case with the RDFS entailment regime. The following example illustrates this difference.

Example 9 *Let S be the following graph: $S = \{\langle parent, \texttt{domain}, Person\rangle, \langle mother, \texttt{subPropertyOf}\ parent\rangle$ which says that the domain of $parent$ is $Person$, and the property $mother$ is a sub-property of $parent$. Using eRDFS entailment we can conclude from S that the domain of $mother$ is also $Person$:*

$$S \models_{erdfs} \langle mother, \texttt{domain}, Person\rangle;$$

it is always the case that the subject of any $mother$ triple has the type $Person$; thus, $mother$ implicitly has the domain $Person$. We cannot draw this conclusion when using RDFS entailment; in RDFS, only explicitly asserted domain constraints can be derived.

4.3 RDF(S) Embedding

In this section we first define an embedding of the various entailment regimes in F-Logic. We then consider an embedding of eRDFS entailment in FOL and DL.

4.3.1 Embedding RDF in F-Logic

We embed a graph as a conjunction of data molecules; URI references and literals are treated as constant symbols, and blank nodes are treated as existentially quantified variables. In the remainder we assume that RDF graphs are finite.

Given a vocabulary $V = \langle \mathcal{F}, \mathcal{PL}, \mathcal{TL}, \mathcal{B}\rangle$, an F-language $\mathcal{L}$ *corresponds to* V if it has a signature of the form $\Sigma = \langle \mathcal{F}' \supseteq \mathcal{F} \cup \mathcal{PL} \cup \mathcal{TL}, \mathcal{P}\rangle$.[5]

Definition 14 *Let S be an RDF graph of a vocabulary V, let $\langle s, p, o\rangle \in S$ be a triple in S, and let $\mathcal{L}$ be an F-language which corresponds to V. Then,*

$$\begin{aligned} (tr(\langle s, p, o\rangle) \in \mathcal{L}) &= s[p \twoheadrightarrow o] \text{ and} \\ (tr(S) \in \mathcal{L}) &= \{\exists\, bl(S)(\bigwedge\{tr(\langle s, p, o\rangle) \mid \langle s, p, o\rangle \in S\})\}. \end{aligned}$$

If ϕ is an F-Logic formula or theory in prenex normal form with only existential quantifiers, then ϕ^{sk} denotes the *Skolemization* of ϕ, i.e. every existentially quantified variable is replaced with a globally unique new constant symbol.

We use a set of formulas $\Psi^x \subseteq \mathcal{L}$, as defined in Table 4.2, to axiomatize the semantics of an entailment regime $x \in \{s, rdf, rdfs, erdfs\}$.

[5] Even though typed literals are pairs in RDF, we treat them simply as constant symbols in our embedding.

Proposition 19 *Let S be an RDF graph of a vocabulary V. Then, $tr(S)^{sk} \cup \Psi^x$, with $x \in \{s, rdf, rdfs\}$, can be equivalently rewritten to a set of F-Logic Datalog formulas.*

Note that Ψ^{erdfs} cannot be equivalently rewritten to a set of Datalog formulas, because of the use of universal quantification in the antecedents of some of the implications in Ψ^{erdfs}.

$\Psi^s = \emptyset$
$\Psi^{rdf} = \Psi^s \cup \{tr(\langle s,p,o\rangle) \mid \langle s,p,o\rangle$ is an RDF axiomatic triple$\}\cup$ $\{wellxml(t) \mid t \in \mathcal{TL}$ is a well-typed XML literal$\}\cup$ $\{illxml(t) \mid t \in \mathcal{TL}$ is an ill-typed XML literal$\}\cup$ $\{\forall x(\exists y, z(y[x \twoheadrightarrow z]) \supset x[\texttt{type} \twoheadrightarrow \texttt{Property}]),$ $\forall x(wellxml(x) \supset x[\texttt{type} \twoheadrightarrow \texttt{XMLLiteral}]),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{XMLLiteral}] \wedge illxml(x) \supset \bot)\}$
$\Psi^{rdfs} = \Psi^{rdf} \cup \{tr(\langle s,p,o\rangle) \mid \langle s,p,o\rangle$ is an RDFS axiomatic triple$\}\cup$ $\{pl(t) \mid t \in \mathcal{PL}\}\cup$ $\{\forall x,y,z(x[y \twoheadrightarrow z] \supset x[\texttt{type} \twoheadrightarrow \texttt{Resource}] \wedge z[\texttt{type} \twoheadrightarrow \texttt{Resource}]),$ $\forall u,v,x,y(x[\texttt{domain} \twoheadrightarrow y] \wedge u[x \twoheadrightarrow v] \supset u[\texttt{type} \twoheadrightarrow y]),$ $\forall u,v,x,y(x[\texttt{range} \twoheadrightarrow y] \wedge u[x \twoheadrightarrow v] \supset v[\texttt{type} \twoheadrightarrow y]),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{Property}] \supset x[\texttt{subPropertyOf} \twoheadrightarrow x]),$ $\forall x,y,z(x[\texttt{subPropertyOf} \twoheadrightarrow y] \wedge y[\texttt{subPropertyOf} \twoheadrightarrow z] \supset x[\texttt{subPropertyOf} \twoheadrightarrow z]),$ $\forall x,y(x[\texttt{subPropertyOf} \twoheadrightarrow y] \supset \forall z_1,z_2(z_1[x \twoheadrightarrow z_2] \supset z_1[y \twoheadrightarrow z_2])),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \supset x[\texttt{subClassOf} \twoheadrightarrow \texttt{Resource}]),$ $\forall x,y(x[\texttt{subClassOf} \twoheadrightarrow y] \supset \forall z(z[\texttt{type} \twoheadrightarrow x] \supset z[\texttt{type} \twoheadrightarrow y])),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \supset x[\texttt{subClassOf} \twoheadrightarrow x]),$ $\forall x,y,z(x[\texttt{subClassOf} \twoheadrightarrow y] \wedge y[\texttt{subClassOf} \twoheadrightarrow z] \supset x[\texttt{subClassOf} \twoheadrightarrow z]),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{ContainerMembershipProperty}] \supset x[\texttt{subPropertyOf} \twoheadrightarrow \texttt{member}]),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{Datatype}] \supset x[\texttt{subClassOf} \twoheadrightarrow \texttt{Literal}]),$ $\forall x(pl(x) \supset x[\texttt{type} \twoheadrightarrow \texttt{Literal}]),$ $\forall x(x[\texttt{type} \twoheadrightarrow \texttt{Literal}] \wedge illxml(x) \supset \bot)\}$
$\Psi^{erdfs} = \Psi^{rdfs} \cup \{\forall x,y(\forall u,v(u[x \twoheadrightarrow v] \supset u[\texttt{type} \twoheadrightarrow y]) \supset x[\texttt{domain} \twoheadrightarrow y]),$ $\forall x,y(\forall u,v(u[x \twoheadrightarrow v] \supset v[\texttt{type} \twoheadrightarrow y]) \supset x[\texttt{range} \twoheadrightarrow y]),$ $\forall x,y(x[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge y[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge \forall u,v(u[x \twoheadrightarrow v] \supset$ $u[y \twoheadrightarrow v]) \supset x[\texttt{subPropertyOf} \twoheadrightarrow y]),$ $\forall x,y(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge y[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge \forall u(u[\texttt{type} \twoheadrightarrow x] \supset u[\texttt{type} \twoheadrightarrow y]) \supset$ $x[\texttt{subClassOf} \twoheadrightarrow y])\}$

Table 4.2: Axiomatization of the RDF entailment regimes

Theorem 15 *Let S and E be RDF graphs of a vocabulary V and let $x \in \{s, rdf, rdfs, erdfs\}$ be an entailment regime. Then,*

$$S \models_x E \quad \textit{iff} \quad tr(S) \cup \Psi^x \models_{\mathsf{f}} tr(E), \textit{ and}$$
$$S \textit{ is } x\textit{-satisfiable} \quad \textit{iff} \quad tr(S) \cup \Psi^x \textit{ has a model.}$$

Proof 25 *We first show that $S \not\models_x E$ iff $tr(S) \cup \Psi^x \not\models_{\mathsf{f}} tr(E)$. From this follows immediately that $S \models_x E$ iff $tr(S) \cup \Psi^x \models_{\mathsf{f}} tr(E)$.*

($\Rightarrow$) Let $V = \langle \mathcal{F}, \mathcal{PL}, \mathcal{TL}, \mathcal{B} \rangle$ be the vocabulary of S and E and let $\mathcal{L}$ be a corresponding F-language. Assume that $S \not\models_x E$. This means that there is an x-interpretation $I = \langle U'_R, U_P, U_L, I_{\mathcal{F}}, I_L, I_{ext} \rangle$ such that $I \models S$ and $I \not\models E$. We construct a corresponding F-Logic interpretation $\mathbf{I} = \langle U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$ in the following way:

- *(i) $U = U'_R \cup U_P$,*
- *(ii) $\mathbf{I}_F(t) = I_{\mathcal{F}}(t)$ for every URI reference $t \in \mathcal{F}$, $\mathbf{I}_F(t) = t$ for every plain literal $t \in \mathcal{PL}$, $\mathbf{I}_F(t) = I_L(t)$ for every typed literal $t \in \mathcal{TL}$,*
- *(iii) $\mathbf{I}_{\twoheadrightarrow}(k) = I_{ext}(k)$ for every $k \in U_P$, and*
- *(iv) $\mathbf{I}_P(pl) = \mathcal{PL}$, $\mathbf{I}_P(illD) = \{u \mid t\mathcal{TL}$ is an ill-typed literal and $, I_L(t) = u\}$, $\mathbf{I}_P(wellxml) = \{u \mid t\mathcal{TL}$ is a well-typed XML literal and $, I_L(t) = u\}$ and $\mathbf{I}_P(wellD) = \{u \mid t\mathcal{TL}$ is a well-typed literal and $, I_L(t) = u\}$.*

It is easy to verify that $\mathbf{I} \models_{\mathsf{f}} tr(S) \cup \Psi^x$ and $\mathbf{I} \not\models_{\mathsf{f}} tr(E)$. Hence, $tr(S) \cup \Psi^x \not\models_{\mathsf{f}} tr(E)$.

($\Leftarrow$) Assume that $tr(S) \cup \Psi^x \not\models_{\mathsf{f}} tr(E)$. This means that there is a (by classical results) Herbrand F-structure $\mathbf{I} = \langle U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$ such that $\mathbf{I} \models_{\mathsf{f}} tr(S) \cup \Psi^x$ and $\mathbf{I} \not\models_{\mathsf{f}} tr(E)$. Since $\mathbf{I}$ is a Herbrand structure, U includes all constant symbols, and every constant symbol is interpreted as itself. We construct a corresponding interpretation $I = \langle U'_R, U_P, U_L, I_{\mathcal{F}}, I_L, I_{ext} \rangle$ as follows:

- *(i) $U_P = \{p \mid \langle p, \mathbf{I}_F(\texttt{Property}) \rangle \in \mathbf{I}_{\twoheadrightarrow}(\mathbf{I}_F(\texttt{type}))\} \cup \{p \mid \exists s, o. \langle s, o \rangle \in \mathbf{I}_{\twoheadrightarrow}(p)\}$,*
- *(ii) $U_L = \mathcal{PL} \cup \{xml(s) \mid ((s, \texttt{XMLLiteral}) \in \mathcal{TL} \wedge (s, \texttt{XMLLiteral})$ is a well-typed XML literal$)\} \cup \{l \mid \langle l, \mathbf{I}_F(\texttt{Literal}) \rangle \in \mathbf{I}_{\twoheadrightarrow}(\mathbf{I}_F(\texttt{type}))\}$,*
- *(iii) $U'_R = U \cup U_L$,*
- *(iv) $I_{\mathcal{F}}(t) = \mathbf{I}_F(t)$ for every URI reference $t \in \mathcal{F}$,[6] $I_L((s,u)) = xml(s)$ if $(s,u) \in \mathcal{TL}$ is a well-typed XML literal; $I_L((s,u)) = \mathbf{I}_F((s,u))$ for $(s,u) \in \mathcal{TL}$ if $(s,u) \in \mathcal{TL}$ is not a well-typed XML literal,*
- *(v) for any $p \in U$ and any $\langle s, o \rangle \in \mathbf{I}_{\twoheadrightarrow}(p)$: $\langle s', o' \rangle \in I_{ext}(p)$, where s' (resp., o') is: if $\exists (t, \texttt{XMLLiteral}) \in \mathcal{TL}.((t, \texttt{XMLLiteral})$ is a well-typed XML literal $\wedge \mathbf{I}_F((t, \texttt{XMLLiteral})) = s)$ (resp., ... $= o$) then $s' = xml(t)$ (resp., $o' = xml(t)$); otherwise $s' = s$ (resp., $o' = o$).*

It is easy to verify that I is an x-interpretation, $I \models S$, and $I \not\models E$. Hence, $S \not\models E$. The second part of the theorem can be shown analogously.

[6] Note that plain literals are always interpreted as themselves in RDF interpretations.

The following corollary follows immediately from Theorem 15 and the classical results about Skolemization (see e.g. [Fit96]). For the case of s-entailment, the result was implicitly stated in [Hay04, Skolemization lemma].

Corollary 2 *Let S and E be RDF graphs and let $x \in \{s, rdf, rdfs, erdfs\}$ be an entailment regime. Then,*

$$S \models_x E \quad \textit{iff} \quad tr(S)^{sk} \cup \Psi^x \models_{\mathsf{f}} tr(E).$$

Since, by Proposition 19, $tr(S)^{sk}$, $tr(S)^{sk} \cup \Psi^{rdf}$ and $tr(S)^{sk} \cup \Psi^{rdfs}$ are equivalent to sets of Datalog formulas, this result implies that simple, RDF, and RDFS entailment can be computed using existing F-Logic rule reasoners[7] such as FLORA-2, and Ontobroker, as well as any rule reasoners which supports Datalog (see Proposition 18). Notice that, in the corollary, $tr(E)$ can be seen as a boolean conjunctive query (i.e. a yes/no query) in which the existentially quantified variables in $tr(E)$ are the non-distinguished variables.

We now consider an alternative, direct embedding of the extensional RDFS semantics (erdfs-entailment) which eliminates part of the RDFS vocabulary from the embedded graph, yielding a set of Datalog formulas.

We first define the notion of *nonstandard use* of the RDFS vocabulary, which intuitively corresponds to using the vocabulary in locations where it has not been intended, for example in places where it redefines the semantics of RDF constructs such as in the triple $\langle$`type`, `subPropertyOf`, $a\rangle$.

We say that a term t occurs in a *property position* if it occurs as the predicate of a triple, as the subject or object of a `subPropertyOf` triple, as the subject of a `domain` or `range` triple, or as the subject of a triple $\langle t,$ `type`, `Property`$\rangle$ or $\langle t,$ `type`, `ContainerMembershipProperty`$\rangle$. A term t occurs in a *class position* if it occurs as the subject or object of a `subClassOf` triple, as the object of a `domain`, `range`, or `type` triple, as the subject of a triple $\langle t,$ `type`, `Class`$\rangle$ or $\langle t,$ `type`, `Datatype`$\rangle$. Otherwise, we say that t occurs in an *individual position*.

Definition 16 *Let S be an RDF graph. Then, S has* nonstandard use *of the RDFS vocabulary if*
– `type`*,* `subClassOf`*,* `domain`*,* `range` *or* `subPropertyOf` *occurs in the subject or object position of a triple in S or*
– `ContainerMembershipProperty`*,* `Resource`*,* `Class`*,* `Datatype` *or* `Property` *occurs in any position other than the object position of a* `type`*-triple in S.*

We conjecture that large classes of RDF graphs will not have any nonstandard use of the RDFS vocabulary. We now proceed to define a direct embedding of the extensional RDFS entailment regime in F-Logic.

[7] Note that a Datalog formula with $\bot$ in the antecedent corresponds to an integrity constraint, i.e. a query which may not have an answer set.

Definition 17 *Let $\langle s, p, o\rangle$ be an RDF triple. Then,*

$$\begin{aligned}
tr^{erdfs}(\langle s, \texttt{type}, \texttt{Datatype}\rangle) &= \forall x(x:s \supset x:\texttt{Literal}),\\
tr^{erdfs}(\langle s, \texttt{type}, o\rangle) &= s:o,\\
tr^{erdfs}(\langle s, \texttt{subClassOf}, o\rangle) &= \forall x(x:s \supset x:o),\\
tr^{erdfs}(\langle s, \texttt{subPropertyOf}, o\rangle) &= \forall x, y(x[s \twoheadrightarrow y] \supset x[o \twoheadrightarrow y]),\\
tr^{erdfs}(\langle s, \texttt{domain}, o\rangle) &= \forall x, y(x[s \twoheadrightarrow y] \supset x:o),\\
tr^{erdfs}(\langle s, \texttt{range}, o\rangle) &= \forall x, y(x[s \twoheadrightarrow y] \supset y:o), \textit{ and}\\
tr^{erdfs}(\langle s, p, o\rangle) &= s[p \twoheadrightarrow o], \textit{ otherwise.}
\end{aligned}$$

Let S be an RDF graph of a vocabulary $V = \langle \mathcal{F}, \mathcal{PL}, \mathcal{TL}, \mathcal{B}\rangle$. Then,

$$\begin{aligned}
tr^{erdfs}(S) &= \{\exists\, bl(S)(\textstyle\bigwedge\{tr^{erdfs}(\langle s, p, o\rangle) \mid \langle s, p, o\rangle \in S\})\}, \textit{ and}\\
\Psi^{erdfs-V} &= \{tr^{erdfs}(\langle s, p, o\rangle) \mid \langle s, p, o\rangle \textit{ is an RDF(S) axiomatic triple with no non-standard use of the RDF(S) vocabulary}\} \cup \{t:\texttt{XMLLiteral} \mid t \in \mathcal{TL} \textit{ is a well-typed XML literal}\} \cup \{illxml(t) \mid t \in \mathcal{TL} \textit{ is an ill-typed XML literal}\} \cup\\
&\quad \{t:\texttt{Literal} \mid t \in \mathcal{PL}\} \cup \{\forall x(x:\texttt{Literal} \wedge illxml(x) \supset \bot)\})
\end{aligned}$$

The *property* (resp., *class*) *vocabulary* of an RDF graph S consists of all the symbols occurring in property (resp., class) positions in S and the RDF(S) axiomatic triples with no nonstandard use of the RDF(S) vocabulary.

Given two RDF graphs S and E, we write $E \trianglelefteq S$ if the property and class vocabularies of E are subsets of the property and class vocabularies of S (modulo blank node renaming and instantiation, i.e. replacement of blank nodes with URI references or literals) and `Resource`, `ContainerMembershipProperty`, `Class`, `Property` and `Datatype` do not occur in E.

Theorem 18 *Let S, E be RDF graphs with no nonstandard use of the RDFS vocabulary such that* `Resource`*,* `Class`*,* `Property`*,* `ContainerMembershipProperty` *and* `Datatype` *do not occur in E. Then,*

- *whenever $E \trianglelefteq S$,*

$$S \models_{erdfs} E \textit{ iff } tr^{erdfs}(S) \cup \Psi^{erdfs-V} \models_{\mathsf{f}} tr^{erdfs}(E);$$

- *$(tr^{erdfs}(S))^{sk}$ is a conjunction of F-Logic Datalog formulas, and whenever E does not contain the terms* `subClassOf`*,* `domain`*,* `range`*, and* `subPropertyOf`*, $tr^{erdfs}(E)$ is a conjunction of atomic molecules prefixed by an existential quantifier and*

$$S \models_{erdfs} E \textit{ iff } (tr^{erdfs}(S))^{sk} \cup \Psi^{erdfs-V} \models_{\mathsf{f}} tr^{erdfs}(E).$$

Since $(tr^{erdfs}(S))^{sk} \cup \Psi^{erdfs-V}$ is a set of Datalog formulas we have that, if the RDF graphs fulfill certain conditions, query answering techniques from the area of deductive databases can be used for checking extensional RDFS entailment.

4.3.2 Embedding Extensional RDFS in First-Order Logic

We now consider an embedding of extensional RDFS entailment in FOL, based on the direct embedding of extensional RDFS in F-Logic considered above (Definition 17).

An F-Logic theory or formula is *translatable* to contextual FOL if for molecules of the forms $t_1[t_2 \twoheadrightarrow t_3]$ and $t_1 : t_2$ holds that t_2 is a constant symbol (i.e. 0-ary function symbol).

Let Φ (resp., ϕ) be an F-Logic theory (resp., formula) which is translatable to contextual FOL, then $(\Phi)^{FO}$ (resp, $(\phi)^{FO}$) is the contextual FOL theory obtained from Φ (resp., ϕ) by:
– replacing every data molecule $t_1[t_2 \twoheadrightarrow t_3]$ with $t_2(t_1, t_3)$, and
– replacing every is-a molecule $t_1 : t_2$ with $t_2(t_1)$.

The following proposition follows immediately from a result in [BH08].

Proposition 20 *Let Φ, ϕ be an equality-free F-Logic theory and formula which are translatable to contextual FOL. Then,*

$$\Phi \models_{\mathsf{f}} \phi \ \textit{iff}\ (\Phi)^{FO} \models_c (\phi)^{FO}.$$

An RDF graph S is a *non-higher-order* RDF graph if S does not contain blank nodes in class and property positions, and does not contain nonstandard use of the RDFS vocabulary. A non-higher-order RDF graph S is a *classical* RDF graph if the sets of URIs occurring in individual, class and property positions in S and its context (e.g. entailing or entailed graph) are mutually disjoint. Notice that every ground RDF graph which does not contain nonstandard use of the RDFS vocabulary is a non-higher-order RDF graph. One can also verify that every OWL DL graph, as defined in [PSHH04a], is a classical RDF graph, but there are classical RDF graphs which are not OWL DL graphs.

The following theorem identifies subsets of extensional RDFS which have a natural correspondence to contextual and classical FOL. Observe that if S is a non-higher-order RDF graph, then $tr^{erdfs}(S)$ is translatable to contextual FOL.

Theorem 19 *Let S and E be non-higher-order RDF graphs such that $E \trianglelefteq S$. Then,*

$$S \models_{erdfs} E \textit{ iff } (tr^{erdfs}(S))^{FO} \models_c (tr^{erdfs}(E))^{FO}.$$

If, additionally, S, E are classical graphs, then $(tr^{erdfs}(S))^{FO}$ and $(tr^{erdfs}(E))^{FO}$ are theories of classical first-order logic, and

$$S \models_{erdfs} E \ \textit{iff}\ (tr^{erdfs}(S))^{FO} \models (tr^{erdfs}(E))^{FO}.$$

Proof 26 *Follows immediately from Theorem 18, the fact that $(tr^{erdfs}(S))^{FO}$ and $(tr^{erdfs}(E))^{FO}$ do not contain the equality symbol, and Proposition 20.*

Proposition 21 *Let S be a ground non-higher-order graph*[8]*. Then, $(tr^{erdfs}(S))^{FO}$ can be equivalently rewritten to the FOL equivalent Φ of a contextual $DL\text{-}Lite_{\mathcal{R}}$ knowledge base $\mathcal{K}$.*

If S is a classical RDF graph, then $(tr^{erdfs}(S))^{FO}$ can be equivalently rewritten to the FOL equivalent Φ of a classical $DL\text{-}Lite_{\mathcal{R}}$ knowledge base.

Proof 27 ((Proof Sketch)) *Φ is obtained from $(tr^{erdfs}(S))^{FO}$ in the following way:*
- Class membership or property value statements of the forms $A(a), P(a_1, a_2)$ are included as such,

- Subclass and subproperty statements are included as such,

- Domain constraints of the form $\forall x, y(P(x, y) \supset A(x))$ are rewritten to role-typing statements of the form $\forall x(\exists y(P(x, y)) \supset A(x))$, and

- Range constraints of the form $\forall x, y(P(x, y) \supset A(y))$ are rewritten to role-typing statements of the form $\forall x(\exists y(P(y, x)) \supset A(x))$.

Φ and $(tr^{erdfs}(S))^{FO}$ are obviously equivalent, and it is easy to verify that Φ is the FOL equivalent of a (contextual) $DL\text{-}Lite_{\mathcal{R}}$ knowledge base.

4.4 Complexity

In this section we review the complexity of the various forms of entailment in RDF and present several novel results, based on the embeddings presented in the previous section.

The complexity of simple entailment and RDFS entailment is well known, and the complexity of RDF entailment follows immediately. Note that, although the set of axiomatic triples is infinite, only a finite subset, linear in the size of the graphs, needs to be taken into account when checking entailment.

Proposition 22 ([GHM04, Hor05b, BFT05]) *Let S and E be graphs. Then, the problem of checking $S \models_s E$, $S \models_{rdf} E$, or $S \models_{rdfs} E$ is NP-complete in the combined size of the graphs, and polynomial in the size of S. If E is ground, then the respective problems are polynomial in the combined size of the graphs.*

Additionally, the problem of checking $S \models_{erdfs} E$ is NP-hard in the size of the graphs.

Using Corollary 2 and known results about Datalog we construct a novel membership proof; we give a direct proof of hardness of all for entailment regimes through a reduction from 3-satisfiability.

[8]Note that, when considering a variant of $DL\text{-}Lite_{\mathcal{R}}$ which allows existentials in the ABox – also allowed in OWL DL – this restriction could be relaxed to S being a non-higher-order RDF graph with no blank nodes outside of individual positions.

Proof 28 *(Membership) Assume that E is ground. By Proposition 19 and Corollary 2, the problems $S \models_s E$, $S \models_{rdf} E$, and $S \models_{rdfs} E$ can be reduced to ground entailment in Datalog: $tr(S)^{sk} \cup \Psi^x$, with $x \in \{s, rdf, rdfs\}$, is equivalent to a Datalog knowledge base with $tr(S)^{sk}$ the data and Ψ^x the program; $tr(E)$ is equivalent to a conjunction of ground atoms. Ground entailment in Datalog is polynomial in the size of the data [DEGV01], which corresponds to the size of the RDF graphs in our case. Note that Ψ^{rdf} and Ψ^{rdfs} contain infinitely many statements involving $\_1, \ldots, \_2$, because of the axiomatic triples. However, only those statements involving vocabulary which is used in S or E need to be taken into account; this is linear in the size of the graphs. This establishes polynomiality of the respective problems in the size of S and E, for E is ground, and in the size of S in the general case.*

If E is not ground, then the problem $S \models_x E$, with $x \in \{s, rdf, rdfs\}$, can be decided using the following non-deterministic polynomial algorithm:

1. *Guess a ground substitution θ for the variables in $tr(E)$ with symbols occurring in $tr(S)^{sk}$.*
2. *Verify whether $tr(S)^{sk} \cup \Psi^x \models_{\mathsf{f}} tr(E)\theta$.*

It is easy to verify that this algorithm is correct. This establishes membership in NP of the problems $S \models_s E$, $S \models_{rdf} E$, and $S \models_{rdfs} E$.

(Hardness) We show a novel proof of the NP-hardness of simple, rdf, rdfs, and erdfs entailment through a reduction from 3-satisfiability. Let ϕ be of the form $(l_{11} \wedge l_{12} \wedge l_{13}) \vee \ldots \vee (l_{m1} \wedge l_{m2} \wedge l_{m3})$ where l_{ij} is a propositional literal, i.e. a possibly negated proposition $a \in \{a_1, \ldots, a_n\}$. Let S be the smallest RDF graph such that

- *$\langle a_i, value, true\rangle, \langle a_i, value, false\rangle \in S$ for $1 \leq i \leq n$,*
- *for $1 \leq j \leq m$ $\langle c_j, has1st, c_{j1}\rangle, \langle c_j, has2nd, c_{j2}\rangle, \langle c_j, has3rd, c_{j3}\rangle \in S$ and for $1 \leq k \leq n, 1 \leq o \leq 3$,*
  - *$\langle c_{jo}, a_k, true\rangle \in S$ if $l_{jo} = a_k$,*
  - *$\langle c_{jo}, a_k, false\rangle \in S$ if $l_{jo} = \neg a_k$, and*
  - *$\langle c_{jo}, a_k, true\rangle, \langle c_{jo}, a_k, false\rangle \in S$, otherwise,*

and let E be the smallest RDF graph such that

- *$\langle a_i, value, \_ : a_i\rangle \in E$ for $1 \leq i \leq n$,*
- *$\langle \_ : c, has1st, \_ : c_1\rangle, \langle \_ : c, has2nd, \_ : c_2\rangle, \langle \_ : c, has3rd, \_ : c_2\rangle \in E$, and*
- *$\langle \_ : c_1, a_i, \_ : a_i\rangle, \langle \_ : c_2, a_i, \_ : a_i\rangle, \langle \_ : c_3, a_i, \_ : a_i\rangle \in E$ for $1 \leq i \leq n$,*

with $\_ : c, \_ : c_1, \_ : c_2, \_ : c_3, \_ : a_i$ *blank nodes.*

The triples of the form $\langle a_i, value, true\rangle, \langle a_i, value, false\rangle$ *in* S *encode the possible variable assignments for* a_i*. Each of the conjuncts* c_{jo} *in a conjunction* c_j *encodes the possible value assignments which satisfies* l_{jo}*.*

In the graph E*, the triples of the form* $\langle a_i, value, \_ : a_i\rangle$ *are used to "guess" a valuation, and the other triples are used to "check" whether there is a conjunction in the original formula which is satisfied in this valuation.*

Clearly, $S \models_s E$ *iff* $S \models_{rdf} E$ *iff* $S \models_{rdfs} E$ *iff* $S \models_{erdfs} E$ *iff* ϕ *is satisfiable. Since the construction of the graphs* S, E *is polynomial in the size of* ϕ*, this establishes* NP*-hardness.*

From the embedding in F-Logic, together with the complexity of nonrecursive Datalog [DEGV01], we obtain the following novel characterization of the complexity of simple and RDF entailment.

Theorem 20 *Let* S *and* E *be RDF graphs. Then, the problems* $S \models_s E$ *and* $S \models_{rdf} E$ *are in* LogSpace *in the size of* S*, and in the combined size of the graphs if* E *is ground.*

Proof 29 ((*Proof Sketch*)) *Notice that transforming* $tr(S)$ *to* $tr(S)^{sk}$ *does not require any rewriting; it simply requires removing the existential quantification, and interpreting the variables as constant symbols. It is easy to see that the only fact which could potentially be recursively derived from* Ψ^{rdf} *is* $\texttt{type}[\texttt{type} \twoheadrightarrow \texttt{Property}]$*; however,* $\texttt{type}[\texttt{type} \twoheadrightarrow \texttt{Property}] \in \Psi^{rdf}$*. Thus,* $tr(S)^{sk}$ *and* $tr(S)^{sk} \cup \Psi^{rdf}$ *may be treated as nonrecursive Datalog programs.*

The proposition then follows straightforwardly from Corollary 2 and the fact that ground entailment in nonrecursive Datalog is in LogSpace *in the size of the data [AHV95], with the data being the input RDF graphs.*

Using the correspondence of Proposition 21, the results on the complexity of reasoning in $DL\text{-}Lite_{\mathcal{R}}$ [CDGL+06], and the classical results on skolemization [Fit96] we obtain the following result for extensional RDFS entailment.

Theorem 21 *Let* S *and* E *be RDF graphs with no nonstandard use of the RDFS vocabulary such that* $E \trianglelefteq S$*. Then, the problem of deciding* $S \models_{erdfs} E$ *is* NP-complete *in the size of the graphs, and polynomial if* E *is ground.*

Proof 30 *Notice that* $((tr_{erdfs}(S))^{sk})^{FO}$ *is a theory of contextual FOL. If* E *is ground, then, as a straightforward consequence from Theorems 18 and 19,*

$$S \models_{erdfs} E \text{ iff } ((tr^{erdfs}(S))^{sk})^{FO} \models_c (tr^{erdfs}(E))^{FO}.$$

Given a contextual $DL\text{-}Lite_{\mathcal{R}}$ *theory* Φ^c *(resp., formula* ϕ^c*), the corresponding classical* $DL\text{-}Lite_{\mathcal{R}}$ *theory* Φ *(resp., formula* ϕ*) is obtained from* Φ^c *(resp.,* ϕ^c*) by replacing*

every occurrence of A as class symbol with $A_{\mathcal{C}}$, replacing every occurrence of P as role symbol with $P_{\mathcal{R}}$, and replacing occurrence of a as an individual symbol with $a_{\mathcal{F}}$. We can use this transformation to reduce reasoning in contextual $DL\text{-}Lite_{\mathcal{R}}$ to reasoning in $DL\text{-}Lite_{\mathcal{R}}$; it is easy to verify that

$$\Phi^c \models_c \phi^c \textit{ iff } \Phi \models \phi,$$

for any formula ϕ^c.

Since this transformation is linear in the size of the knowledge base, the complexity of satisfiability and entailment for contextual $DL\text{-}Lite_{\mathcal{R}}$ are the same as for $DL\text{-}Lite_{\mathcal{R}}$, namely polynomial in the size of the knowledge base [CDGL$^+$06].

Assume that E is ground. Since $((tr^{erdfs}(S))^{sk})^{FO}$ and $(tr^{erdfs}(E))^{FO}$ can be (linearly) equivalently rewritten to contextual FOL theories which are equivalent to contextual $DL\text{-}Lite_{\mathcal{R}}$ knowledge bases, and entailment in contextual $DL\text{-}Lite_{\mathcal{R}}$ is polynomial, the problem $S \models_{erdfs} E$ can be solved in polynomial time as well.

This result immediately leads to the following nondeterministic polynomial-time algorithm for deciding $S \models_{erdfs} E$, in case E is not ground:

1. Guess a substitution θ of blank nodes in E with terms in $((tr^{erdfs}(S))^{sk})^{FO}$.
2. Check whether $((tr^{erdfs}(S))^{sk})^{FO} \models_c (tr^{erdfs}(E)\theta)^{FO}$, which can be done in polynomial time, by the above result.

Correctness of the overall algorithm follows from the fact that all formulas in $((tr^{erdfs}(S))^{sk})^{FO}$ are Horn, and thus we only need to take terms occurring in $((tr^{erdfs}(S))^{sk})^{FO}$ into account (by the classical results by Herbrand), justifying the use of the substitution θ. This establishes membership in NP.

NP-hardness follows from the proof of Proposition 22 (observe that the RDFS vocabulary is not used in the hardness proof). Therefore, checking extensional RDFS entailment $S \models_{erdfs} E$, where S has no nonstandard use of the RDFS vocabulary, is NP-complete.

Entailment	Restrictions on S	Restrictions on E	Complexity
$\models_s$,$\models_{rdf}$,$\models_{rdfs}$	none	none	NP-complete
$\models_s$,$\models_{rdf}$	none	ground	LogSpace
$\models_{rdfs}$	none	ground	P
$\models_{erdfs}$	none	none	NP-hard
$\models_{erdfs}$	no nonst. RDFS	no nonst. RDFS	NP-complete
$\models_{erdfs}$	no nonst. RDFS	ground, no nonst. RDFS	P

Table 4.3: Complexity of Entailment $S \models_x E$ in RDF, measured in the size of S, E

Table 4.3 summarizes the complexity of reasoning with the entailment regimes of RDF; "No nonst. RDFS" stands for "no nonstandard use of the RDFS vocabulary; S and E

are such that the property and class vocabularies of E are subsets of the property and class vocabularies of S (modulo blank node renaming and instantiation); and `Resource`, `Class`, `Property`, `ContainerMembershipProperty` and `Datatype` do not occur in E". The results in the first and third line of the table were obtained in [GHM04, BFT05, Hor05b], and the fourth line follows immediately. To the best of our knowledge, the other results are novel.

4.5 Querying

In this section we consider conjunctive queries over RDF graphs using the RDF entailment regimes we considered throughout this chapter.

Given a countable set $\mathcal{V}$ of variable symbols, disjoint from the symbols in V, *a generalized RDF triple* is a tuple of the form $\langle s, p, o\rangle$, with s, p and o terms or variable symbols. A *conjunctive query* $q(\vec{x})$ over an RDF graph S is a set of generalized RDF triples $q(\vec{x})$ such that $\vec{x}$ is a vector of variables occurring in q, also called the *distinguished variables* of q; the blank nodes occurring in q, $bl(q)$, are the *non-distinguished variables* of q.

Given an RDF graph S and conjunctive query $q(\vec{x})$, then a tuple of terms $\vec{a}$ is an *answer* of a query under x-entailment if $S \models_x q(\vec{a})$, with $x \in \{s, rdf, rdfs, erdfs\}$ an entailment regime. The complexity of query answering is related to the complexity of the corresponding recognition problem: the recognition problem associated with a query $q(\vec{x})$ is the decision problem of checking whether, given an RDF graph S and a tuple $\vec{a}$ of terms, the entailment $S \models_x q(\vec{a})$ holds. The *data complexity* of query answering under the x entailment regime corresponds to the complexity of the corresponding recognition problem, in the size of S.

From the preceding results on the complexity of the various entailment regimes we obtain the following characterization of the complexity of query answering.

Theorem 22 *Let S be an RDF graph, let $x \in \{s, rdf, rdfs\}$ be an entailment regime, and let $q(\vec{x})$ be a conjunctive query. Then, the data complexity of query answering under the x entailment regime is*

- *in LogSpace, if $x \in \{s, rdf\}$,*
- *polynomial, if $x \in \{rdfs\}$, and*
- *polynomial in the size of S, and LogSpace in the size of S^{-ont}, which is the subset of S without the RDFS ontology vocabulary, if $x = erdfs$, S and q have no non-standard use of the RDFS vocabulary, and q contains no blank nodes or variables in class or property position.*

Proof 31 *The first and second bullet follow immediately from Propositions 22 and 20. The third bullet follows from the complexity results about $DL\text{-}Lite_{\mathcal{R}}$ [CDGL+06, CGL+05]*

and the proof of Theorem 21.

4.6 Discussion and Related Work

In this section we discuss implications of the results in this chapter, and place it in the context of related work. We distinguish between work done on RDF and on RDF querying.

RDF There have been several investigations [GHM04, BFT05, Hor05b] into the semantics of RDF. The investigation in [GHM04] reconstructs the semantics from a graph database perspective, and the one in [BFT05] reconstructs the semantics from a logical language perspective. The investigation of the RDF semantics in [Hor05b] stays very close to the RDF specification. Additionally, [Hor05b] shows that the entailment rules presented in the original specification [Hay04] are not complete with respect to the semantics. These reconstructions have led to a number of complexity results for RDF entailment. In this chapter, we built upon these results and complemented them with several novel results for simple, RDF, and extensional RDFS entailment.

The investigation in [BFT05] is close in spirit to our investigation, albeit that [BFT05] bases its logical reconstruction on (contextual) first-order logic, rather than F-Logic.

RDFS(FA) [PH07] defines a new (extensional) semantics for RDFS which is in line with the semantics of OWL DL, as well as a number of syntactic restrictions to achieve a layered meta-modeling architecture. It is currently not known what the precise relationship is between RDFS(FA) and the RDFS semantics defined in the standard [Hay04].

Finally, we mention [MPG07], in which the authors identify a syntactic subset of RDFS which allows for efficient reasoning ($O(n \log n)$), while still being expressive enough to capture large classes of ontologies.

RDF Querying SPARQL [PS07] is a query language for RDF, currently under development at W3C. Of all the RDF entailment regimes, SPARQL currently only considers simple entailment. However, the use of other regimes is considered a possible future extension.

The queries we considered in Section 4.5 are conjunctive queries and correspond to what are called "[SPARQL] graph pattern expressions constructed by using only AND" in [PAG06]. Therefore, not surprisingly, data complexity of conjunctive query answering when considering simple entailment corresponds to the data complexity of evaluating such graph pattern expressions; they are both in LogSpace (cf. Theorem 22 and [PAG06, Theorem 4]).

Finally we mention [Pol07], in which a translation from SPARQL queries to Datalog is described. The combination of such a translation with an embedding of the RDF or

RDFS semantics, as described in Theorem 15, could be used for evaluating SPARQL queries using the respective entailment regimes.

4.7 Conclusions and Future Work

We have presented embeddings of the different RDF entailment regimes in F-Logic, and we have shown how deductive database and description logic technology can be used for reasoning with RDF. An implementation of answering conjunctive queries over RDF graphs under the RDF and RDFS entailment regimes, and restricted RDF graphs under the eRDFS entailment regime, based on the Datalog reasoner IRIS[9], can be found at: `http://tools.deri.org/rdfs-reasoner`. It is planned to extend this reasoner with support for more expressive query languages, such as SPARQL, considering the embedding in Datalog presented in [Pol07].

In the course of our investigation we have presented several novel complexity results. To the best of our knowledge, ours is the first comprehensive investigation of the extensional RDFS entailment regime. These results could be used for, for example, rule-based extensions of RDF, or increasing the alignment between RDF and logic-based semantic Web languages (e.g. OWL DL).

Our future work includes the extension of the considered embeddings with more extensive treatment of datatypes, in the form of *D*-entailment [Hay04], and *D**-entailment [Hor05b], as well as more expressive query languages such as SPARQL.

Several proposals have been made for rule extensions of RDF graphs (e.g. [Hor05a, AADW05]), and several rule-based systems which deal with RDF exist (e.g. Jena, CWM). In an earlier version of the present chapter [BH07] we considered extensions of RDF graphs with logical rules and axioms, based on the embeddings we have presented. However, such extensions are not entirely faithful with respect to the model-theoretic semantics of RDF. Therefore, our future work includes an investigation of combinations of logical rules and RDF based on a notion of common models, i.e. an interpretation is a model of a combination if it is a model of both the logical theory and the RDF graph.

Finally, we plan to investigate the precise relationship between eRDFS and OWL DL entailment, taking the subset compatible with $DL\text{-}Lite_{\mathcal{R}}$ (see Proposition 21) as a starting point.

[9] `http://sourceforge.net/projects/iris-reasoner`

Chapter 5

Semantic Cooperation and Knowledge Reuse by Using Autonomous Ontologies

5.1 Introduction

The Semantic Web vision is to develop a distributed environment in which software agents can automatically, conveniently and effectively interpret and apply the data that is available on the Web. To this end, a system of knowledge representation which supports semantic cooperation between distributed agents is required. Such a system must be based on ontologies which define the terms and relationships used in a particular application domain. Each such ontology reflects the objective and shared views of a community of users working in that domain. However, the original use of the word "ontology" in philosophy was to describe a complete, self-contained domain of discourse. This usage does not scale to the open and distributed Web, where there are ontologies for each different application domain and even different ontologies for the same domain. Thus, to support semantic cooperation between agents, it is necessary to manage and reason about multiple ontologies, which we call an ontology space. How to do this effectively is a major research problem for the Semantic Web.

AI researchers have also studied management and reasoning in multiple representations of application domains using contextual reasoning [Giu93, GS94, GG01]. For this reason, it is a natural and interesting issue to combine ontology-based and context-based approaches so that the advantages of both ontology and contextual reasoning can be employed in the same system. This idea led a series of interesting work.

C-OWL [BGvH$^+$03] sets up relations outside the ontologies by a set of "bridge rules" between the concepts (individuals) from different ontologies. Its semantics relies on the *domain relation* in DDL [BS03], which is a directional mapping from the elements of one domain to the other domain. $\mathcal{E}$-Connection [GPS04] puts the relations inside the ontology, by extending OWL with a new kind of "link property". It connects two sets

of strictly disjoint concepts from different domains. P-DL [BCH06] treats every foreign term as an imported relation, and semantically interprets it by an *image domain relation* which is a one-to-one and compositionally consistent mapping between two domains. Semantic Importing [PSZ06] focuses on the overlaps of domains; it allows a subconcept which falls into the conjunction of two domains to be semantically imported and used in the other ontology. Conservative Extension [GK07] restricts multiple ontology modules in the same global interpretation domain and allows them to be interpreted using standard semantics.

It is easy to see that the above approaches somehow weaken the autonomy of an ontology. In order to bridge the gaps between the semantics of different ontologies, a class of approaches based on cross-domain relations (i.e., domain relations in C-OWL, directional binary relation in $\mathcal{E}$-connection, and image domain relation in P-DL) needs the information of the domain element in the other ontology to interpret a bridge rule in C-OWL, a link property in $\mathcal{E}$-connection, or an importing relation in P-DL. For example, suppose that there is well-accepted ontology on the Web called "Vehicle", in which a concept "Car" is defined as: "A car consists of two parts: engine and body". In C-OWL, this concept is represented as $Vehicle : Car$ which indicates that the concept "Car" is defined in the ontology "Vehicle". Both BMW and Toyota may wish to borrow the concept "Car" when they design their own ontologies. These two companies have different interpretations on "engine" and "body". In C-OWL, $Vehicle : Car$ has different local interpretations in BMW and Toyota. However, a user cannot distinguish this difference and confusion may be caused when other people use $Vehicle : Car$. Semantic Importing does not rely on domain relations, but one needs to know exactly the domain elements and how the concepts/roles are interpreted in the other ontology. Conservative extensions require a single global domain, and prevent each module from local interpretation its axioms under its own context.

In this chapter, we propose a new framework for managing autonomy in a distributed ontology space. On the one hand a language entity (concept/role/individual) is interpreted totally under local domain semantics in order to preserve the autonomy of an ontology; on the other hand a (shared) language entity is restricted by a *semantic binding* if necessary in order to enable semantic cooperation among several ontologies. In this way, one ontology is able to express its "subjective" opinion by local interpretation, and to receive its *foreign semantics* by semantic binding. We use the term "foreign semantics" of one ontology to express the "semantic meaning" of an foreign entity from another ontology connected by semantic binding. Accordingly, we also introduce two reasoning mechanisms: *cautious reasoning* and *brave reasoning*. The former relies on an ontology and its foreign semantics from its neighbors' ontologies connected by semantic binding, but it does not trust the foreign semantics of its neighbor from their neighbors. The latter believes an ontology and its foreign semantics, and also its neighbor ontologies and their foreign semantics.

The main contribution of this chapter are the followings:

- We introduce a novel approach to define semantic cooperation between different ontologies. By semantic binding, the semantics of language entities from one ontology is able to be accessed in other ontologies. This is different from the existing domain-relation based approaches.
- We formalize two forms of reasoning mechanisms: cautious reasoning and brave reasoning. The former only does reasoning in one ontology and its neighbors' ontologies associated by its semantic binding; the latter does reasoning in one ontology and its neighbours' ontologies and their neighbours' ontologies and so on.

In the rest, we briefly review ontologies and ontology spaces in Section-5.2, and introduce the notion of an autonomous ontology in Section-5.3. In Section-5.4 we introduce the two reasoning mechanisms and the algorithms. The conclusion is given in Section-5.5.

5.2 Preliminaries: Ontology Space and foreign entity

In general understanding [PSHH04b], an ontology is a set of annotated *terminological axioms* and *facts*. Current discussion is based on normal Description Logic (DL) [BCM$^+$03]. The proposed framework can be restricted or generalized to some DL languages such as OWL, SHOIN(D+), etc.

5.2.1 Ontology and ontology space

Let $\mathbb{C}$ be a finite set of concept names, $\mathbb{R}$ a finite set of role names, and $\mathbb{E}$ a finite sets of individual names. A *language* $\mathbb{L}$ has a vocabulary of the disjoint union of $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{E}$.

Definition 23 (Ontology) *Let $\mathbb{L}$ be the language. An* Ontology O *is a tuple $\langle T, A\rangle$, where T and A are TBox and ABox respectively in Description Logic on $\mathbb{L}$.*

Definition 24 (Ontology interpretation) *An* (abstract) ontology interpretation $\mathcal{I}$ *is a pair $\left\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \right\rangle$, in where $\Delta^{\mathcal{I}}$ is an nonempty domain, and $\cdot^{\mathcal{I}}$ is a mapping that assigns*

1. *to each concept name $c \in \mathbb{C}$ a subset of $\Delta^{\mathcal{I}}$,*
2. *to each role name $R \in \mathbb{R}$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,*
3. *to each individual name $e \in \mathbb{E}$ an element of $\Delta^{\mathcal{I}}$.*

Definition 25 (*Ontology Space*) *Let I be a set of indexes, standing for a set of URIs for ontologies. Let $\mathbb{L}_I = \{\mathbb{L}_i\}_{i\in I}$ be a set of languages. An* Ontology Space $\mathbb{O}_I$ *on $\mathbb{L}_I$ is a family $\{O_i\}_{i\in I}$,* s.t. *every O_i is an ontology on language $\mathbb{L}_i$, where $i \in I$.*

In ontology space $\mathbb{O}_I = \{O_i\}_{i\in I}$, we denote, by $\mathbb{C}_i$ the set of concept names in ontology O_i; we define $\mathbb{R}_i$ and $\mathbb{E}_i$ analagously. Actually *language* $\mathbb{L}_i$ has a vocabulary of the disjoint union of $\mathbb{C}_i$, $\mathbb{R}_i$ and $\mathbb{E}_i$. In the rest of the chapter, we use *language entity* to denote a concept, role, or individual in one ontology.

5.2.2 Foreign Entity

In ontology space, sometimes a language entity is defined in one ontology, but could be used in another ontology. So we partition the language $\mathbb{L}_i$ in two parts: the *local entity* and the *foreign entity* (originated from local language and foreign language in [BGvH+03]). Intuitively, local entities are the roles, concepts, and individuals that one invites in her own ontology; foreign entities are the roles, concepts, and individuals that she borrows from the other ontologies in order to define something in her ontology.

In this chapter, when we are talking about semantics and reasoning, we always tell a language entity in the ontology space by a way showing (1) where it is used, and (2) where it is originally defined. Suppose that $C \in \mathbb{L}_i$ and $i, j \in I$, then formally in ontology space we have a language entity like, $(i : j : C)$, which means a language entity C appears in ontology O_i, but is originally defined in ontology O_j. This kind of denotation is applied to concepts, roles, and individuals in ontology space.

One of the advantages of this denotation is, in syntax two concepts/roles with the same name but used in different ontologies are distinguishable. For example, suppose we use $(BMW : auto : engine)$ for the engines in BMW car ontology, and $(Toyota : auto : engine)$ for those in Toyota car ontology; obvious these two concepts should be different, and it is easy to see from the syntax: $(BMW : auto : engine) \neq (Toyota : auto : engine)$. This denotation is important in this chapter because, the approach proposed in this chapter assigns local semantics to this kind of concepts/roles, and treats them as totally different entities.

5.3 Autonomous Ontology

In one ontology space, each ontology reflects the subjective opinion on a partial structure of the universe. In Semantic Web, in general one party presents her personal knowledge (understanding) by her ontology. Thereafter we argue each ontology should be semantical independent and keeping autonomy.

Formally an *autonomous ontology* is composed by two parts: one is an ontology which is to be interpreted locally in order to keep the autonomy of one party; the other is a set of foreign entities, which is called *semantic binding* in this chapter, in order to accept foreign information or knowledge from the other parties.

Definition 26 (Autonomous Ontology) *Let $\mathbb{O}_I = \{O_i\}_{i \in I}$ be an ontology space, where I is a set of indexes. An* autonomous ontology *is a tuple $AO_i = \langle \mathbb{B}_i, O_i \rangle$, in which $O_i \in \mathbb{O}_I$, and $\mathbb{B}_i$ is the* semantic binding *of AO_i, which is a set of foreign entities.*

An *autonomous ontology space* $\mathbb{AO}_I$ is a set of autonomous ontologies. In an autonomous ontology space $\mathbb{AO}_I = \{AO_i\}_{i \in I}$, if an entity $(j : j : x)$, which is original

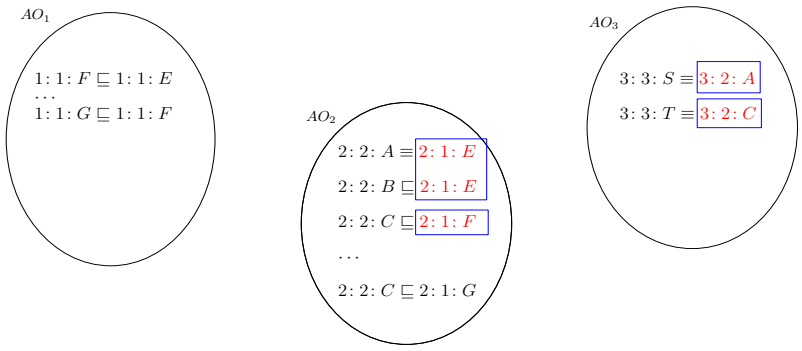


Figure 5.1: Autonomous ontology space.

defined in AO_j, is semantic bounded in AO_i; i.e., $(i:j:x) \in \mathbb{B}_i$, then we say AO_j is the *binding neighbour* of AO_i.

In fact it is easy to treat an autonomous ontology as a normal ontology; for example in OWL we could just introduce a binding annotation like:

```
Annotation ( binding <http://www.auto.org/engine\#> )
```

to express the semantic binding. So in the rest of the chapter sometime we also mean an autonomous ontology by ontology.

Example 10 (Autonomous Ontology) *Consider the autonomous ontology space in Figure-5.1, suppose we have following semantic bindings:* $\mathbb{B}_1 = \emptyset$, $\mathbb{B}_2 = \{(2:1:E),\ (2:1:F)\}$, *and* $\mathbb{B}_3 = \{(3:2:A),\ (3:2:C)\}$. *So* AO_1 *is the binding neighbour of* AO_2, *and* AO_2 *is the binding neighbour of* AO_3. *We note that in autonomous ontology it allows some foreign entity outside of the semantic binding, e.g.,* $(2:1:G)$ *in* AO_2. □

5.3.1 Local interpretation

Definition 27 (Local Interpretation) *For autonomous ontology* $AO_i = \langle \mathbb{B}_i, O_i \rangle$, *a* local interpretation $\mathcal{I}_i$ *is a pair* $\langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \rangle$, *in where* $\Delta^{\mathcal{I}_i}$ *is an nonempty domain, and* $\cdot^{\mathcal{I}_i}$ *is a mapping* , s.t. $\cdot^{\mathcal{I}_i}$ *assigns*

1. *(for local entities)*
   (a) *a subset of* $\Delta^{\mathcal{I}_i}$ *to each local concept name* $(i:i:c)$;
   (b) *a subset of* $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$ *to each local role name* $(i:i:r)$;
   (c) *an element of* $\Delta^{\mathcal{I}_i}$ *to each local individual name* $(i:i:e)$,
2. *and for* $i \neq j$ *(for foreign entities)*

(a) a subset of $\Delta^{\mathcal{I}_i}$ *to each foreign concept name* $(i:j:c)$*;*

(b) a subset of $\Delta^{\mathcal{I}_i}\mathcal{T}imes\Delta^{\mathcal{I}_i}$ *to each foreign role name* $(i:j:r)$*;*

(c) an element of $\Delta^{\mathcal{I}_i}$ *to each foreign individual name* $(i:j:e)$*.*

As we see in the above definition, in local interpretation not only local entities but also foreign entities including those in the semantic binding are interpreted under local domain. Actually, the local interpretation of an autonomous ontology is the interpretation of the ontology. From this aspect the abstract interpretation of ontology in Section-5.2 is also a local interpretation. The difference is, the definition in the last section is for a single ontology, but here we focus on one ontology among an ontology space.

Following common understanding, if an axiom $a \in A$ is true under an interpretation $\mathcal{I}$, we say that interpretation $\mathcal{I}$ *satisfies* the axiom a , and denote this by $\mathcal{I} \models a$.

Definition 28 (Satisfiability of autonomous ontology) *Let* AO_i *be an autonomous ontology and an* $\mathcal{I}$ *its local interpretation, we say that* $\mathcal{I}$ satisfies AO_i*, if for any axiom* $a \in AO_i$*, we have* $\mathcal{I} \models a$*. We call* $\mathcal{I}$ *a* local model *of* AO_i*, and denote this fact by* $\mathcal{I} \models_{\mathcal{L}} AO_i$*.*

Definition 29 (Local Entailment) *Let* AO_i *be an autonomous ontology,* λ *a concept description or an assertion. We say that* λ *is a* local entailment *of* AO_i*, iff for any local model* $\mathcal{I}_i$ *of* AO_i*,* $\mathcal{I}_i \models \lambda$*. This fact is denoted by* $AO_i \models_{\mathcal{L}} \lambda$*.*

Example 11 (Local semantics) *Consider the autonomous ontology space in Figure-5.1. We have* $AO_2 \models_{\mathcal{L}} (2:2:B) \sqsubseteq (2:2:A)$*, because for any local model* $\mathcal{I}$*,*

$$\frac{\begin{array}{l}\mathcal{I} \models (2{:}2{:}B) \sqsubseteq (2{:}1{:}E), \quad and \\ \mathcal{I} \models (2{:}2{:}A) \equiv (2{:}1{:}E)\end{array}}{\mathcal{I} \models (2{:}2{:}B) \sqsubseteq (2{:}2{:}A)}.$$

5.3.2 C-binding Consistency

For autonomous ontology $AO_i \in \mathbb{AO}_I$, let $\mathbb{B}_i = \{\bigcup_{j\in I} \mathbb{B}_{ij}\}$ be the semantic binding [1], in which $\mathbb{B}_{ij} = \{(i:j:C)\}_{i,j\in I}$ contains all of the semantic-bounded foreign entities which are original defined in O_j.

In autonomous ontology $AO_i \in \mathbb{AO}_I$, a *j-concept* (*j-role*) is an class (property) description which is composed by the entities in $\mathbb{B}_{ij}$. For example, a *j*-concept of AO_i could be $\exists(i:j:hasChild).(i:j:Male)$.

Let λ_j be a *j*-concept , obviously in λ_j all of the entities are prefixed by "$i:j$ ". If we change the prefix of every entity in λ_j from "$i:j$ " to "$j:j$ ", and then we get λ'_j. We call λ'_j the *original image* of λ_j in O_j. For example, let λ_j be $(i:j:Person) \sqcap \forall(i:$

[1] It is possible that some foreign entities are not included in $\mathbb{B}_i$. Actually these foreign entities are not semantic-bounded; they are "free-access" entities according to [ZSG06].

symbol	$\mathcal{I}_{21}$	$\mathcal{I}_{22}$
$\mathcal{T}op$	$\{\alpha, \beta\}$	$\{\alpha, \beta\}$
$(2{:}2{:}A)$	$\{\alpha\}$	$\{\alpha, \beta\}$
$(2{:}2{:}B)$	$\{\alpha\}$	$\{\alpha\}$
$(2{:}2{:}C)$	$\{\alpha, \beta\}$	$\{\beta\}$
$(2{:}1{:}E)$	$\{\alpha\}$	$\{\alpha, \beta\}$
$(2{:}1{:}F)$	$\{\alpha, \beta\}$	$\{\beta\}$
$(2{:}1{:}G)$	$\{\alpha, \beta\}$	$\{\alpha, \beta\}$
$\cdots$	$\cdots$	$\cdots$
$(2{:}1{:}F) \sqsubseteq (2{:}1{:}E)$	$Not\ satisfied$	$Satisfied$

Table 5.1: Example of interpretations for the autonomous ontology AO_2.

$j : hasChild).(i : j : Female)$, then its original image λ'_j is $(j : j : Person) \sqcap \forall(j : j : hasChild).(j{:}j{:}Female)$.

We note that the original image λ'_j of a j-concept λ_j may not has a concept name in O_j; it may not be explicitly defined there.

Definition 30 (C(*j*)-binding Consistent Model) *Let $AO_i = \langle \mathbb{B}_i, O_i \rangle$ be an autonomous ontology, and $j \in I$. Let $\mathcal{I}$ be a local model of AO_i, if for any* j*-concept λ_j, we have*

$$\mathcal{I} \models \lambda_j \quad \text{iff} \quad AO_j \models_{\mathcal{L}} \lambda'_j \tag{5.1}$$

then we say $\mathcal{I}$ is the C(j)-binding consistent model *of AO_i. This fact is denoted by $\mathcal{I} \models_{\mathcal{C}(j)} AO_i$*

Example 12 (C(j)-binding consistent model) *Let $\Delta_2 = \{\alpha, \beta\}$ be the domain of AO_2 in Figure-5.1. Considering following two local interpretations of AO_2:*

Obviously both $\mathcal{I}_{21}$ and $\mathcal{I}_{22}$ are local models of AO_2, i.e., $\mathcal{I}_{21} \models_{\mathcal{L}} AO_2$ and $\mathcal{I}_{22} \models_{\mathcal{L}} AO_2$. Let $\lambda = (2 : 1 : F) \sqsubseteq (2 : 1 : E)$, from Table-5.1 we find $\mathcal{I}_{21} \not\models \lambda$. Since $\mathbb{B}_{21} = \{(2{:}1{:}E),\ (2{:}1{:}F)\}$, λ is the 1-concept, and $AO_1 \models_{\mathcal{L}} \lambda$, we have $\mathcal{I}_{21}$ is not a C(1)-binding consistent model. Actually $\mathcal{I}_{22}$ is a C(1)-binding consistent model of AO_2. In this example we also note that foreign entity outside of the semantic binding does not carry any semantical information from its original ontology; e.g., since $(2{:}1{:}G) \notin \mathbb{B}_{21}$, although $AO_1 \models_{\mathcal{L}} (1{:}1{:}G) \sqsubseteq (1{:}1{:}F)$, in AO_2 C(1)-binding consistent model $\mathcal{I}_{22}$ does not need to satisfy $(2{:}1{:}G) \sqsubseteq (2{:}1{:}F)$. □

Definition 31 (C(*j*)-binding Entailment) *Let AO_i be an autonomous ontology, λ a concept description*[2] *or an assertion. We say that λ is the* C(*j*)-binding entailment *of AO_i, iff for any C(j)-binding consistent model $\mathcal{I}$ of AO_i, $\mathcal{I} \models \lambda$. This fact is denoted by $AO_i \models_{\mathcal{C}(\mathrm{j})} \lambda$.*

[2]Here for convenient we treat a subsumption $x \sqsubseteq y$ as a concept description $\neg x \sqcup y$. Same for the rest of the chapter.

In Example-12 we have $AO_2 \models_{\mathcal{C}(I)} (2{:}1{:}F) \sqsubseteq (2{:}1{:}E)$, and $AO_2 \models_{\mathcal{C}(I)} (2{:}2{:}C) \sqsubseteq (2{:}2{:}A)$.

Definition 32 (C-binding Consistency) *Let $\mathcal{I}$ be a local model of AO_i. If for any $j \neq i \in I$, $\mathcal{I}$ is the C(j)-binding consistent model of AO_i, then $\mathcal{I}$ is the* C-binding consistent model *of AO_i. We say AO_i is* C-binding consistent *in $\mathbb{AO}_I$ if there exists a C-binding consistent model.*

We say an autonomous ontology $AO_i = \langle \mathbb{B}_i, O_i \rangle$ is *C-satisfiable*, if it is C-binding consistent and O_i is satisfiable.

Let AO_i be an autonomous ontology and $i \neq j \in I$, we note that not every local model is a C(j)-binding consistent model; it is not necessary for a local model to satisfy a *j*-concept , but it is for a C(j)-binding consistent model. We also note that not every C(j)-binding consistent model is a C-binding consistent model; a C(j)-binding consistent model may not satisfies a k-concept for $k \neq j$. So an autonomous ontology which is satisfiable under the local semantics could be unsatisfiable under the autonomous semantic.

Definition 33 (C-entailment) *Let AO_i be an autonomous ontology, λ a concept description or an assertion. We say that λ is the* C-entailment *of AO_i, iff for any C-binding consistent model $\mathcal{I}_i$ of AO_i, $\mathcal{I}_i \models \lambda$. This fact is denoted by $AO_i \models_{\mathcal{C}} \lambda$.*

Lemma 34 *Let AO_i be an autonomous ontology, λ a concept description or an assertion. We have $AO_i \models_{\mathcal{L}} \lambda \implies AO_i \models_{\mathcal{C}} \lambda$.*

In this chapter, we use $\Pi_{\mathcal{C}}(AO_i) = \{\lambda |\ AO_i \models_{\mathcal{C}} \lambda\}$ to denote the set of C-entailments of AO_i, and call it *Cautious theory* of AO_i. Comparably, we also use $\Pi_{\mathcal{L}}(AO_i) = \{\lambda |\ AO_i \models_{\mathcal{L}} \lambda\}$ to denote the set of local entailment of AO_i under local semantics, and call it *local theory* of AO_i.

Theorem 35 *For a autonomous ontology AO_i, we have $\Pi_{\mathcal{L}}(AO_i) \subseteq \Pi_{\mathcal{C}}(AO_i)$.*

Proposition 23 *Let $AO_i = \langle \mathbb{B}_i, O_i \rangle$ be C-binding consistent, for any $j \neq i \in I$ and any* j*-concept λ of AO_i, we have $AO_i \models_{\mathcal{C}} \lambda$ if $AO_j \models_{\mathcal{L}} \lambda'$.*

Example 13 *In the autonomous ontology space in Figure-5.1, we have $AO_2 \models_{\mathcal{C}} (2:2:C) \sqsubseteq (2:2:A)$. However for AO_3, since $AO_2 \not\models_{\mathcal{L}} (2:2:C) \sqsubseteq (2:2:A)$, $AO_3 \not\models_{\mathcal{C}} (3{:}2{:}C) \sqsubseteq (3{:}2{:}A)$, and then we do not have AO_3 entails $(3{:}3{:}T) \sqsubseteq (3{:}3{:}S)$ under C-entailment.* □

In this chapter when we say *cautious semantics* of an autonomous ontology we mean the C-binding model(s). We say an autonomous ontology space $\mathbb{AO}_I$ is *C-binding consistent* if every autonomous ontology is C-binding consistent.

Definition 36 (C-entailment of Ontology Space) *Suppose autonomous ontology space $\mathbb{AO}_I = \{AO_i\}_{i \in I}$ is C-binding consistent. Let $AO_i \in \mathbb{AO}_I$, and λ a concept description*

or an assertion. We say that λ is the C-entailment of autonomous ontology space $\mathbb{AO}_I$, *denoted by $\mathbb{AO}_I \models_{\mathcal{C}} \lambda$, iff there exists $i \in I,\ s.t.\ AO_i \models_{\mathcal{C}} \lambda$. We also say $i \in I$ is the* provenance *of the entailment λ .*

5.3.3 B-binding Consistency

B-binding stands for "brave binding". One autonomous ontology not only relies on its binding neighbors, it also trusts the neighbors of its binding neighbors. In this way B-binding could build more stronger semantic cooperation among multiple ontologies than C-binding, in the sense that some information in one ontology is transitively reused by, not only its neighbour but also the neighbour's neighbour.

For example in the autonomous ontology space in Figure-5.1, AO_3 does not entail $(3{:}3{:}T) \sqsubseteq (3{:}3{:}S)$ under C-binding semantics because $AO_2 \not\models_{\mathcal{L}} (2{:}2{:}C) \sqsubseteq (2{:}2{:}A)$; but it could entail this subsumption under B-binding semantics because $AO_2 \models_{\mathcal{C}} (2{:}2{:}C) \sqsubseteq (2{:}2{:}A)$. Details will be given later.

Definition 37 (B-binding Entailment) *Let $AO_i = \langle \mathbb{B}_i, O_i \rangle$ be an autonomous ontology, and $\mathcal{I}$ a local interpretation of it. We say $\mathcal{I}$ is a* B-binding consistent model *of AO_i, which is denoted by $\mathcal{I} \models_{\mathcal{B}} AO_i$, if*

1 $\mathcal{I}$ is a local model of AO_i, and

2 for any $j \in I$ and any j-*concept λ_j we have $\mathcal{I} \models \lambda_j$ iff*

(a) $AO_j \models_{\mathcal{L}} \lambda'_j$, or

(b) $AO_j \models_{\mathcal{C}} \lambda'_j$, or

(c) $AO_j \models_{\mathcal{B}} \lambda'_j$

We say AO_i is B-binding consistent *in $\mathbb{AO}_I$ if there exists a B-binding consistent model. Let ψ be a formula. For any B-binding model $\mathcal{I}$ of AO_i, if $\mathcal{I} \models \psi$, then we say ψ is the B-entailment of AO_i. This fact is denoted by $AO_i \models_{\mathcal{B}} \psi$*

Above is a recursive definition on the B-binding semantics for an autonomous ontology . Local knowledge in one ontology is used to support a logical result in its *binding neighbour-reached* ontology. We say AO_j is binding neighbour-reached from AO_i, if there exists a sequence $(AO_{x,1},\ AO_{x,2},\ \cdots,\ AO_{x,k})$ such that for $1 \leq y < k$ $AO_{x,(y+1)}$ is a binding neighbour of $AO_{x,y}$ and $AO_{x,1} = AO_i$ and $AO_{x,k} = AO_j$.

From above definition it is easy to get the following lemma.

Lemma 38 *For a autonomous ontology $AO_i \in \mathbb{AO}_I$, every B-binding model is a C-binding model.*

Lemma 39 *Let AO_i be an autonomous ontology, λ a concept description or an assertion. We have $AO_i \models_{\mathcal{C}} \lambda \implies AO_i \models_{\mathcal{B}} \lambda$.*

In this chapter, we use $\Pi_{\mathcal{B}}(AO_i) = \{\lambda |\ AO_i \models_{\mathcal{B}} \lambda\}$ to denote the set of B-entailments of AO_i, and call it *brave theory* of AO_i. From Lemma-38 and Lemma-39, the following theorem is obvious.

Theorem 40 *For a autonomous ontology AO_i, we have $\Pi_{\mathcal{C}}(AO_i) \subseteq \Pi_{\mathcal{B}}(AO_i)$.*

According to Definition-37 the following proposition is obvious.

Proposition 24 *Let $AO_i = \langle \mathbb{B}_i, O_i \rangle$ be B-binding consistent, for any $j \neq i \in I$ and any* j*-concept λ of AO_i, we have $AO_i \models_{\mathcal{B}} \lambda$ if $AO_j \models_{\mathcal{C}} \lambda'$.*

Example 14 *In the autonomous ontology space in Figure-5.1, we have $AO_3 \models_{\mathcal{B}} (3:3:T) \sqsubseteq (3:3:S)$ under the brave semantics.*

In this chapter when we say *brave semantics* of an autonomous ontology we mean the B-binding model(s). We say an autonomous ontology space $\mathbb{AO}_I$ is *B-binding consistent* if every autonomous ontology is B-binding consistent.

Definition 41 (B-entailment of Ontology Space) *Suppose autonomous ontology space $\mathbb{AO}_I = \{AO_i\}_{i \in I}$ is B-binding consistent. Let $AO_i \in \mathbb{AO}_I$, and λ a concept description or an assertion. We say that λ is the* B-entailment of autonomous ontology space $\mathbb{AO}_I$*, denoted by $\mathbb{AO}_I \models_{\mathcal{B}} \lambda$, iff there exists $i \in I,\ s.t.\ AO_i \models_{\mathcal{B}} \lambda$. We also say $i \in I$ is the* provenance *of the entailment λ .*

5.4 Tableaux Algorithms of Reasoning on $\mathcal{ALCN}$

In this section, we present two distributed tableaux algorithms to realize *cautious reasoning* (under cautious semantics) and b*rave reasoning* (under brave semantics) in autonomous ontology space respectively.

Here we consider ontologies represented as $\mathcal{ALCN}$TBoxes (which consists of only class axioms). These algorithms are designed for verifying class satisfiability in an autonomous ontology space $\mathbb{AO}_I$, and can also be used to verify class subsumption[3].

5.4.1 Preliminary of Tableaux Algorithm

Tableaux algorithms are very useful to solve class satisfiability problem. They test the satisfiability of a class λ[4] by trying to construct an interpretation for λ, which is repre-

[3]Since subsumption relation $C \sqsubseteq D$ in AO_i *w.r.t.* $\mathbb{AO}_I$ iff $C \sqcap \neg D$ is unsatisfiable in AO_i *w.r.t.* $\mathbb{AO}_I$.

[4]Here we assume λ is in negation normal form; i.e., negation is only applied to class names.

sented by a *completion tree* $\mathcal{T}$ which is formally defined as following: A *completion tree* is a tuple $\mathcal{T} = \langle x_0, N, E, \mathcal{L} \rangle$, where x_0 is the root of $\mathcal{T}$, N and E are the sets of nodes and edges, respectively, of $\mathcal{T}$, and $\mathcal{L}$is a function that maps a node x in $\mathcal{T}$ to its label $\mathcal{L}(x)$, and an edge $\langle x, y \rangle$ in $\mathcal{T}$ to its $\mathcal{L}(\langle x, y \rangle)$, respectively.

A tableaux algorithm starts from an labeled initial tree (usually simply a root node), and is expanded by repeatedly applying the completion rules. The algorithm terminates either when $\mathcal{T}$ is *complete* (no further completion rules can be applied) or when an obvious contradiction, or *clash*, has been revealed.

Intuitively, our tableaux algorithm expands a completion tree *w.r.t.* the local axiom box, and then project some part of the tree (which is related to other autonomous ontology) for further expansion *w.r.t.* the axiom boxes of the neighbour autonomous ontology , (e.g., sending the original image of a *j*-concept to AO_j and start a new tableaux algorithm to check the satisfiability) and then back-project some semantics results to the local completion tree. We say that a completion tree $\mathcal{T}$ is S(*j*)-bound if there exist some *j*-concept or *j*-role descriptions in the labels of all nodes and edges of $\mathcal{T}$. In this section, we use procedure Tab$(AO_i, \mathcal{T})$ as a well known (local) $\mathcal{ALCN}$tableaux algorithm to expand $\mathcal{T}$ *w.r.t.* a local ontology AO_i. Tab$(AO_i, \mathcal{T})$ has two distinguished features that we need: (1) it takes not only a single node but an arbitrary initial completion tree, (2) the algorithm can cache reasoning states, *i.e.*, backtracking points. In the algorithms below we also use $backtrack$(Tab$(AO_i, \mathcal{T})$) to denote the operation to expand tree $\mathcal{T}$ from the backtracking point and return a completion tree.

Projection is to bring information from one autonomous ontology to the binding neighbour.

Definition 42 (Projection of Completion Tree) *Let $AO_i = \langle \mathbb{B}_i, O_i \rangle$ be an autonomous ontology, and $\mathbb{B}_{ij} \in \mathbb{B}_i$ the set of semantic binding of foreign entities from another autonomous ontology AO_j. Let $\mathcal{T}$ be a clash-free completion tree with root x_0 in AO_i. The* projection *of $\mathcal{T}$* w.r.t. *$\mathbb{B}_{ij}$, denoted as $P(\mathcal{T}, \mathbb{B}_{ij})$, is a completion tree $\mathcal{T}' = \langle x'_0, N', E', \mathcal{L} \rangle$ generated in the following way:*

1. $N' = \{x' \mid \text{if } x \in N\}$,
2. $E' = \{\langle x', y' \rangle \mid \text{if } \langle x, y \rangle \in E\}$,
3. $\mathcal{L}(x') = \{ (j\!:\!j\!:\!C) | \text{ if } (i\!:\!j\!:\!C) \in \mathcal{L}(x) \text{ and } (i\!:\!j\!:\!C) \in \mathbb{B}_{ij}\}$,
4. $\mathcal{L}(\langle x', y' \rangle) = \{ (j\!:\!j\!:\!R) | \text{ if } (i\!:\!j\!:\!R) \in \mathcal{L}(\langle x, y \rangle) \text{ and } (i\!:\!j\!:\!R) \in \mathbb{B}_{ij}\}$,

5.4.2 Cautious Reasoning

Cautious reasoning relies on the knowledge of an ontology as well as the knowledge of its binding neighbour ontologies.

Given an autonomous ontology space $\mathbb{AO}_I = \{AO_i\}_{i \in I}$, the procedure $\mathcal{C}$-Tab($\mathbb{AO}_I$,

k, λ)[5] verifies the satisfiability of an $\mathcal{ALC}$ class description λ in ontology AO_k under the cautious semantics *w.r.t.* $\mathbb{AO}_I$.

Algorithm A-1: $\mathcal{C}$-Tab($\mathbb{AO}_I$, k, λ)

1: Let $\mathcal{T} := Tab(AO_k, \langle x_0, \{x_0\}, \emptyset, \{\mathcal{L}(x_0) = \{\lambda\}\}\rangle)$ // local expansion *w.r.t.* AO_k
2: **repeat**
3: **if** $\mathcal{T}$ has a clash **then**
4: **return** *unsatisfiable*
5: **end if**
6: **for** every binding neighbour autonomous ontology AO_i ($i \in I$) of AO_k **do**
7: **if** there exist $S(i)$-bound maximal sub-trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of $\mathcal{T}$ with roots $x_1, ..., x_n$, respectively **then**
8: $\mathcal{T}'_1 := P(\mathcal{T}_1, \mathbb{B}_{ki}), \ldots, \mathcal{T}'_n := P(\mathcal{T}_n, \mathbb{B}_{ki})$ // sub-trees projection to AO_i
9: $\mathcal{T}'_1 := Tab(AO_i, \mathcal{T}'_1), \ldots, \mathcal{T}'_n := Tab(AO_i, T'_n)$ // local expansion of $\mathcal{T}'_1, \ldots, \mathcal{T}'_n$ *w.r.t.* AO_i
10: **if** any of $\mathcal{T}'_1, \ldots, \mathcal{T}'_n$ has a clash **then**
11: **if** $\mathcal{T}$ is backtrackable **then**
12: $\mathcal{T}$:=backtrack(Tab($AO_i, \mathcal{T}$)) // backtrack and expand
13: **else**
14: **return** *unsatisfiable*
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **return** *satisfiable*
20: **until** false

In this algorithm, $\mathcal{T}$ is initialized with a root x_0 with $\mathcal{L}(x_0) = \{\lambda\}$, and is expanded by local completion rules *w.r.t.* AO_k (line 1 of **A-1**). As $\mathcal{T}$ can have multiple binding neighbour ontologies, each of them should be taken care (line 6 of **A-1**). Note that $\mathcal{T}$ might not be k-bound, the algorithm just project the maximal k-bound sub-trees, and then expand them by local completion rules *w.r.t.* AO_i, and expanded *w.r.t.* $\mathbb{AO}_I$ (lines 7-9 of **A-1**). If any of the projected sub-tree has a clash, $\mathcal{T}$ needs to be backtracked (line 12 of **A-1**), expanded and start the checking all over again.

Theorem 43 *$\mathcal{C}$ -Tab($\mathbb{AO}_I$, k, λ) is a decision procedure to verify the cautious semantics satisfiability of an $\mathcal{ALCN}$-class description λ in ontology AO_k* w.r.t. $\mathbb{AO}_I$.

To prove the theorem, we need to show that: (1) The algorithm always terminates. (2) The algorithm returns unsatisfiable if $\mathbb{AO}_I \not\models_{\mathcal{C}} \lambda$. (3) The algorithm returns satisfiable if $\mathbb{AO}_I \models_{\mathcal{C}} \lambda$. Due to limited space, we skip the detail.

[5]This Algorithm is originated from [PSZ06], in which only positive concepts/roles can be projected to the original ontology, but the approach in this chapter does not have this restriction.

5.4.3 Brave Reasoning

Brave reasoning not only relies on its binding neighbors, but also trusts the neighbors of its binding neighbors.

Given an autonomous ontology space $\mathbb{AO}_I = \{AO_i\}_{i \in I}$, the procedure $\mathcal{B}$ -Tab($\mathbb{AO}_I$, k, λ) verifies the satisfiability of an $\mathcal{ALC}$ class description λ in ontology AO_k *w.r.t.* $\mathbb{AO}_I$ under brave semantics. It calls a recursive procedure $\mathcal{DB}$-Tab($\mathbb{AO}_I$, k, $\mathcal{T}$) to expands a completion tree $\mathcal{T}$ of AO_k *w.r.t.* $\mathbb{AO}_I$ under brave semantics.

Algorithm A-2: $\mathcal{B}$ -Tab($\mathbb{AO}_I$, k, λ)

1: Let $\mathcal{T} := \langle x_0, \{x_0\}, \emptyset, \{\mathcal{L}(x_0) = \{\lambda\}\}\rangle$)
2: $\mathcal{T} :=$ $\mathcal{DB}$-Tab(k, $\mathcal{T}$)
3: **if** $\mathcal{T}$ has a clash **then**
4: **return** *unsatisfiable*
5: **else**
6: **return** *satisfiable*
7: **end if**

Algorithm A-3: $\mathcal{DB}$-Tab($\mathbb{AO}_I$, k, $\mathcal{T}$)

1: let $\mathcal{T}$:=Tab($AO_k, \mathcal{T}$) // local expansion *w.r.t.* AO_k
2: **repeat**
3: **if** $\mathcal{T}$ has a clash **then**
4: **return** $\mathcal{T}$ // *unsatisfiable*
5: **end if**
6: **for** every binding neighbour autonomous ontology AO_i of AO_k **do**
7: **if** there exist $S(i)$-bound maximal sub-trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of $\mathcal{T}$ with roots $x_1, ..., x_n$, respectively **then**
8: $\mathcal{T}_1' := P(\mathcal{T}_1, \mathbb{B}_{ki}), \ldots, \mathcal{T}_n' := P(\mathcal{T}_n, \mathbb{B}_{ki})$ // sub-trees projection from AO_k to AO_i
9: $\mathcal{T}_1' :=$ $\mathcal{DB}$-Tab($\mathbb{AO}_I, i, \mathcal{T}_1'), \ldots, \mathcal{T}_n' :=$ $\mathcal{DB}$-Tab($\mathbb{AO}_I, i, \mathcal{T}_n'$) // recursive calling $\mathcal{DB}$-Tab for brave reasoning on $\mathcal{T}_1', \ldots, \mathcal{T}_n'$ *w.r.t.* AO_i
10: **if** any of $\mathcal{T}_1', \ldots, \mathcal{T}_n'$ has a clash **then**
11: **if** $\mathcal{T}$ is backtrackable **then**
12: $\mathcal{T}$:=backtrack(Tab($O_x i, \mathcal{T}$)) // backtrack and expand
13: **else**
14: **return** $\mathcal{T}$ // *unsatisfiable*
15: **end if**
16: **else**
17: $\mathcal{T}_1 := \beta(\mathbb{B}_{ki}, \mathcal{T}_1, \mathcal{T}_1'), \ldots, \mathcal{T}_n := \beta(\mathbb{B}_{ki}, \mathcal{T}_n, \mathcal{T}_n')$ //back-project
18: **end if**
19: **end if**
20: **end for**
21: **if** $\mathcal{T}$ is not changed, **then**
22: **return** $\mathcal{T}$ // *satisfiable*
23: **end if**
24: **until** false

In this algorithm, initially $\mathcal{T}$ has a root x_0 with $\mathcal{L}(x_0) = \{\lambda\}$ (line 1 of **A-2**), and then it is expanded by local completion rules *w.r.t.* AO_k (line 1 of **A-3**). As $\mathcal{T}$ can have multiple binding neighbour ontologies, each of them should be taken care (line 6 of **A-3**). Note

that $\mathcal{T}$ might not be $S(i)$-bound, maximal $S(i)$-bound sub-trees then should be projected, and expanded by local completion rules *w.r.t.* AO_i, if possible it also need to project to the binding neighbour of AO_i. (lines 7-9 of **A-3**). If any of the projected sub-tree has a clash, $\mathcal{T}$ needs to be backtracked (line 12 of **A-3**), expanded and start the checking all over again; otherwise, we need to back-project the new $S(i)$-bound labels back to $\mathcal{T}$ (line 17 of **A-3**). The algorithm **A-3** would not stop until $\mathcal{T}$ is not changed.

Theorem 44 *$\mathcal{B}$ -Tab($\mathbb{AO}_I$, k, λ) is a decision procedure to verify the brave semantics satisfiability of an $\mathcal{ALCN}$-class description λ in ontology AO_k* w.r.t. $\mathbb{AO}_I$.

To prove the theorems, we need to show that: (1) The algorithms always terminates. (2) The algorithm returns unsatisfiable if $\mathbb{AO}_I \not\models_{\mathcal{B}} \lambda$. (3) The algorithm returns satisfiable if $\mathbb{AO}_I \models_{\mathcal{B}} \lambda$. Due to limited space, we skip the proof.

5.5 Conclusions

In general understanding, ontologies are used for describing the structure of domain knowledge. Techniques for (partial) ontology reuse are important for ontology building, ontology discovery, and practical application of ontologies.

How to realise semantic cooperation among multiple ontologies is an important problem in the field of (partial) ontology reuse. In this chapter we have proposed and analyzed a new framework for managing multiple ontologies that both preserves the autonomy of individual ontologies and also enables the semantic cooperation of different ontologies. We have also proposed two different reasoning mechanisms, called cautious reasoning and brave reasoning, for this framework and studied their properties. As we discussed in Section-5.1, this work is related to DDL [BS03], C-OWL [BGvH$^+$03], $\mathcal{E}$-Connection based approach [GPS04], P-DL [BCH06], Semantic Importing [PSZ06], and conservative extension [GK07].

Ontology modularization [Rec03, SK04, GPSK, PMT05, GPSK06] is another interesting problem in this field. It attempts to partition one ontology and isolate functional modules. How to present formal functional modules in a single ontology by semantic binding (as proposed in this chapter) could be an interesting extension of this approach.

Another interesting task is to explore whether our approach to ontology spaces and semantic binding can be applied to problems of forgetting, importing and extending in ontology spaces.

A natural task for the immediate future is to implement our two reasoning mechanisms and empirically evaluate their performance.

Bibliography

[AADW05] Anastasia Analyti, Grigoris Antoniou, Carlos Viegas Damásio, and Gerd Wagner. Stable model theory for extended RDF ontologies. In *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, pages 21–36. Springer, 2005.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AL04] Jürgen Angele and Georg Lausen. Ontologies in F-logic. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[BCH06] J. Bao, D. Caragea, and V. Honavar. Towards collaborative environments for ontology construction and sharing. In *International Symposium on Collaborative Technologies and Systems (CTS 2006)*, pages 90–108. IEEE Press, 2006.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, 2003.

[Bec04] Dave Beckett. RDF/XML Syntax Specification. URL, February 10 2004. W3C Recommendation, `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`.

[Bel77] Nuel D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logic*, pages 5–37. Reidel, Dordrecht, Netherlands, 1977.

[BEPT06] Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits. On representational issues about combinations of classical theories with nonmonotonic rules. In *Proceedings of the 1st International Conference on Knowledge Science, Engineering and Management (KSEM2006)*. Springer, 2006.

[BFT05] Jos de Bruijn, Enrico Franconi, and Sergio Tessaris. Logical reconstruction of normative RDF. In *Proceedings of the Workshop OWL: Experiences and Directions (OWLED-2005)*, 2005.

[BG04] Dan Brickley and Ramanathan V. Guha. RDF vocabulary description language 1.0: RDF schema. Recommendation 10 February 2004, W3C, 2004.

[BGvH+03] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Proc. Sencond Internatinal Semantic Web Conference*, pages 164–179, 2003.

[BH95] Franz Baader and Bernard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14:149–180, 1995.

[BH07] Jos de Bruijn and Stijn Heymans. RDF and logic: Reasoning and extension. In *Proceedings of the 6th International Workshop on Web Semantics (WebS 2007), in conjunction with the 18th International Conference on Database and Expert Systems Applications (DEXA 2007)*. IEEE Computer Society Press, 2007.

[BH08] Jos de Bruijn and Stijn Heymans. On the relationship between description logic-based and f-logic-based ontologies. *Fundamenta Informaticae*, 2008. Accepted for publication.

[BLW06] Piero Bonatti, Carsten Lutz, and Frank Wolter. Expressive non-monotonic description logics based on circumscription. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 400–410. AAAI Press, 2006.

[BS03] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantic*, 1:153–184, 2003.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein eds. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/, Feb 2004.

[CDGL+06] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 260–270. AAAI Press, 2006.

[CGL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Dl-lite: Tractable description logics for

ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI2005)*, pages 602–607. AAAI Press, 2005.

[Che93] Jianhua Chen. Minimal knowledge + negation as failure = only knowing (sometimes). In *Proceedings of the 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR1993)*, pages 132–150. MIT Press, 1993.

[CKW93] Weidong Chen, Michael Kifer, and David Scott Warren. HILOG: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.

[DEGV01] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR)*, 33(3):374–425, 2001.

[DNR02] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Description logics of minimal knowledge and negation as failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.

[EIST06] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Proceedings*, volume 4011 of *Lecture Notes in Computer Science*, pages 273–287. Springer, Berlin/Heidelberg, 2006.

[ELST04a] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR2004: Principles of Knowledge Representation and Reasoning*, pages 141–151. AAAI Press, Menlo Park, California, 2004.

[ELST04b] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*. AAAI Press, 2004.

[ELST04c] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Well-founded semantics for description logic programs in the semantic web. In Grigoris Antoniou and Harold Boley, editors, *Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings*, volume 3323 of *Lecture Notes in Computer Science*, pages 81–97. Springer, Berlin/Heidelberg, 2004.

[Fit85] Melvin Fitting. A kripke-kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Fit90] Melvin C. Fitting. Bilattices in logic programming. In *20th International Symposium on Multiple-Valued Logic, Charlotte*, pages 238–247. IEEE CS Press, Los Alamitos, 1990.

[Fit93] Melvin Fitting. The family of stable models. *Journal of Logic Programming*, 17(2/3&4):197–225, 1993.

[Fit96] Melvin Fitting. *First Order Logic and Automated Theorem Proving (second edition)*. Springer Verlag, 1996.

[Fit02] Melvin Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[Gel87] Michael Gelfond. On stratified autoepistemic theories. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI1987)*, pages 207–211. Morgan Kaufmann Publishers, 1987.

[GG01] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.

[GHM04] Claudio Gutierrez, Carlos Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In *Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS2004)*, pages 95–106. ACM Press, 2004.

[GHP+06] B. Cuenca Grau, I. Horrocks, B. Parsia, P. Patel-Schneider, U. Sattler, and R. Shearer. Next steps for owl. Kweb deliverable: D2.5.5, 2006.

[Gin88] Matthew L. Ginsberg. Multivalued logics: A uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4(3):265–316, 1988.

[Giu93] F. Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993.

[GK07] Bernardo Cuenca Grau and Oliver Kutz. Modular ontology languages revisited. In *Proc. of the IJCAI-2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition*, 2007.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming (ICLP1988)*, pages 1070–1080. MIT Press, 1988.

[GL91a] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[GL91b] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3–4):365–386, 1991.

[GP93] Michael Gelfond and Halina Przymusinska. Reasoning on open domains. In *Proceedings of the 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR1993)*, pages 397–413, 1993.

[GPS04] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *Proceedings of the 3thd International Semantic Web Conference (ISWC-2004)*, 2004.

[GPSK] B.C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularizing OWL Ontologies. In *Proc. of the KCAP-2005 Workshop on Ontology Management*.

[GPSK06] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006).*, 2006.

[GS94] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.

[Hay04] Patrick Hayes. RDF semantics. Recommendation 10 February 2004, W3C, 2004.

[Hor05a] Herman J. ter Horst. Combining RDF and part of OWL with rules: Semantics, decidability, complexity. In *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*. Springer, 2005.

[Hor05b] Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2–3):79–115, 2005.

[HPSBT05] Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, and Dmitry Tsarkov. OWL rules: A proposal and prototype implementation. *Journal of Web Semantics*, 3(1):23–40, 2005.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 161–180. Springer, Berlin/Heidelberg, 1999.

[HVNV] Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir. Open answer set programming with guarded programs. *ACM Transactions on Computational Logic*. Accepted for Publication.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. Recommendation 10 February 2004, W3C, 2004.

[KLW95a] Michael Kifer, Geord Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.

[KLW95b] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[Kon91] Kurt Konolige. Quantification in autoepistemic logic. *Fundamenta Informaticae*, 15(3–4):275–300, 1991.

[KR02] Michael Kaminski and Guy Rey. Revisiting quantification in autoepistemic logic. *ACM Transactions on Computational Logic*, 3(4):542–561, 2002.

[Kun87] Kenneth Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Lev90] Hector J. Levesque. All i know: a study in autoepistemic logic. *Artificial Intelligence*, 42(2–3):263–309, 1990.

[Lif91] V. Lifschitz. Nonmonotonic databases and epistemic queries. In *Proceedings of IJCAI-91*, pages 381–386, San Mateo, CA., 1991. Morgan Kaufmann.

[Lif94] Vladimir Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70(1–2):53–72, 1994.

[LL00] Hector J. Levesque and Gerhard Lakemeyer. *The Logic of Knowledge Bases*. MIT Press, 2000.

[LS93] Vladimir Lifschitz and Grigori Schwarz. Extended logic programs as autoepistemic theories. In *Proceedings of the 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR1993)*, pages 101–114. MIT Press, 1993.

[LS05] Yann Loyer and Umberto Straccia. Any-world assumptions in logic programming. *Theoretical Computer Science*, 342(2-3):351–381, 2005.

[McD82] Drew McDermott. Nonmonotonic logic ii: Nonmonotonic modal theories. *Journal of the ACM*, 29(1):33–57, 1982.

[Moo85] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[Mot06] Boris Motik. *Reasoning in description logics using resolution and deductive databases*. PhD thesis, Universität Karlsruhe, 2006.

[MPG07] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. Minimal deductive systems for RDF. In *Proceedings of the 4th European Semantic Web Conference (ESWC2007)*, pages 53–67. Springer, 2007.

[MR06] Boris Motik and Riccardo Rosati. Closing semantic web ontologies. Technical report, University of Manchester, UK, 2006.

[MT93] V. Wiktor Marek and Miroslaw Truszczynski. Reflexive autoepistemic logic and logic programming. In *Proceedings of the 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR1993)*, pages 115–131. MIT Press, 1993.

[PAG06] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. In *Proceedings of the 5th International Semantic Web Conference (ISWC2006)*, pages 30–43. Springer, 2006.

[PFT+04a] J. Pan, E. Franconi, S. Tessaris, B. Glimm, W. Siberski, V. Tzouvaras, G. Stamou, I. Horrocks, L. Li, and H. Wache. Report on query language design and standardisation. Kweb deliverable: D2.5.2, 2004.

[PFT+04b] J. Pan, E. Franconi, S. Tessaris, Stamou, V. Tzouvaras, L. Serafini, I. Horrocks, and B. Glimm. Specification of coordination of rule and ontology. Kweb deliverable: D2.5.1, 2004.

[PFT+05] J. Pan, E. Franconi, S. Tessaris, D. Tsarkov, I. Horrocks, G. Stoilos, G. Stamou, H. Wache, D. Turi, S. Bechhofer, and L. Li. Report on inplementataion and optimisation techniques for ontology query systems. Kweb deliverable: D2.5.3, 2005.

[PFT+07] J. Pan, E. Franconi, S. Tessaris, D. Tsarkov, I. Horrocks, G. Stoilos, G. Stamou, H. Wache, D. Turi, S. Bechhofer, and L. Li. Report on inplementataion and optimisation techniques for ontology query systems. Kweb deliverable: D2.5.7, 2007.

[PH07] Jeff Z. Pan and Ian Horrocks. RDFS(FA): Connecting RDF(S) and OWL DL. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):192–206, 2007.

[PMT05] Elena B. Paslaru, Malgorzata Mochol, and Robert Tolksdorf. Case Studies on Ontology Reuse. In *Proc. of I-Know05*, 2005.

[Pol07] Axel Polleres. From SPARQL to rules (and back). In *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, pages 787–796. ACM Press, 2007.

[Prz91] Teodor C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9(3–4):401–424, 1991.

[PS07] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Candidate Recommendation 14 June 2007, W3C, 2007.

[PSHH04a] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. Recommendation 10 February 2004, W3C, 2004.

[PSHH04b] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax, 10 February 2004.

[PSZ06] Jeff Z. Pan, Luciano Serafini, and Yuting Zhao. Semantic import: An approach for partial ontology reuse. In *Proc. of the ISWC2006 Workshop on Modular Ontologies (WoMO).*, 2006.

[Rec03] Alan L Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owls. In *Proc. the International Conference on Knowledge Capture, (K-CAP 2003)*, 2003.

[Ros05] Riccardo Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In François Fages and Sylvain Soliman, editors, *Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005, Dagstuhl Castle, Germany, September 11-16, 2005, Proceedings*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.

[Ros06a] Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 68–78. AAAI Press, 2006.

[Ros06b] Riccardo Rosati. $\mathcal{DL}$+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 68–78. AAAI Press, 2006.

[Sch92] Grigori Schwartz. Reflexive autoepistemic logic. *Fundamenta Informaticae*, 17(1–2):157–173, 1992.

[SK04] Heiner Stuckenschmidt and Michel Klein. Structure-based partitioning of large concept hierarchies. In *Proc. of Int. Semantic Web Conf.*, 2004.

[SSP06] G. Stoilos, G. Stamou, and J. Pan. Fuzzy reasoning extensions. Kweb deliverable: D2.5.6, 2006.

[Tar55] Alfred Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[VBDDS97] Kristof Van Belleghem, Marc Denecker, and Danny De Schreye. A strong correspondence between description logics and open logic programming. In *Proceedings of the 14th International Conference on Logic Programming (ICLP1997)*, pages 346–360. MIT Press, 1997.

[vRS91] Allen van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[ZSG06] Yuting Zhao, Luciano Serafini, and Fausto Giunchiglia. Autonomous ontology: Operations and semantics **`or`** local semantics with semantic binding on foreign entity. In *Proceeding of The 1st Asian Semantic Web Conference (ASWC'06)*, Beijing, China, 2006.