Journal of Artificial Intelligence Research 60 (2017) 1165-1213 Submitted 04/17; published 12/17

A Knowledge Level Account of Forgetting

James P. Delgrande JIM@CS.SFU.CA
School of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6 Canada

Abstract

Forgetting is an operation on knowledge bases that has been addressed in different areas of Knowledge Representation and with respect to different formalisms, including classical propositional and first-order logic, modal logics, logic programming, and description logics. Definitions of forgetting have been expressed in terms of manipulation of formulas, sets of postulates, isomorphisms between models, bisimulations, second-order quantification, elementary equivalence, and others. In this paper, forgetting is regarded as an abstract belief change operator, independent of the underlying logic. The central thesis is that forgetting amounts to a reduction in the language, specifically the signature, of a logic. The main definition is simple: the result of forgetting a portion of a signature in a theory is given by the set of logical consequences of this theory over the reduced language. This definition offers several advantages. Foremost, it provides a uniform approach to forgetting, with a definition that is applicable to any logic with a well-defined consequence relation. Hence it generalises a disparate set of logic-specific definitions with a general, high-level definition. Results obtained in this approach are thus applicable to all subsumed formal systems, and many results are obtained much more straightforwardly. This view also leads to insights with respect to specific logics: for example, forgetting in first-order logic is somewhat different from the accepted approach. Moreover, the approach clarifies the relation between forgetting and related operations, including belief contraction.

1. Introduction

In Artificial Intelligence (AI), a common assumption is that an agent is *knowledge-based*, that is, it has a *knowledge base (KB)* comprised of a set of sentences, or formulas in some language, that gives a declarative account of some domain of interest, and that represents the agent's explicit beliefs. Associated with the KB is a set of inference procedures for deriving further sentences, that is, the beliefs that are implicit in an agent's knowledge base. However, a KB is generally not a static, monolithic object; rather a KB may evolve as information is added, becomes out of date, or is discovered to be incorrect. These latter modifications have been studied under the topic of *belief change*, with the dominant approach given by the so-called *AGM approach* (Alchourrón, Gärdenfors, & Makinson, 1985; Gärdenfors, 1988). More recently, a KB change operator called *forget* has received attention, and has been studied in the context of various approaches, including classical propositional logic, first-order logic, logic programming, modal logics, and description logics, among others. Informally, in forgetting, information is in some fashion or another discarded, and an agent becomes ignorant of, or unaware of, that information. A key assumption is that, while the resulting theory will be weaker than before, the resulting KB should entail the same set of sentences that are irrelevant to the forgotten items.

Forgetting has various pragmatic uses.[1] For example, in *query answering*, if one can determine what is relevant with respect to a query, then forgetting the irrelevant part of a knowledge base may yield more efficient query-answering. This notion can be extended to other tasks in which it would be useful to focus on the *relevant* part of a KB, such as in planning, diagnosis, decision-making, and so on. In addition, approaches to nonmonotonic reasoning and belief change are typically phrased in terms of the global consistency of a knowledge base;[2] an account of forgetting may potentially allow a reformulation of such approaches to be with respect to the consistency of a relevant part of the knowledge base. As well, forgetting may provide a formal account and justification of *predicate hiding*, which in turn may find applications in security and in privacy issues. Moreover, forgetting may be applicable in *summarising* a knowledge base by suppressing lesser details, or for *reusing* part of a knowledge base by removing an unneeded part of a larger knowledge base, or in *clarifying* relations between predicates.

There is, however, no generally agreed-upon definition of forgetting. Some work starts with a definition of forgetting originally due to George Boole in propositional logic, and uses this definition as a touchstone to develop an approach in a different setting. Other work, notably in the description logic community, regards forgetting as the dual of uniform interpolation. Still other work begins with some underlying host formalism, provides intuitions for what is desired with respect to a forgetting operator, and then, based on these intuitions, develops a formal definition. These definitions may be based on the manipulation of formulas, sets of postulates, isomorphisms between models, rule transformations, bisimulations between models, second-order quantification, elementary equivalence, and others. Accounts of forgetting are often equated with, or noted to be similar to, other concepts such as uniform interpolation, variable or literal elimination, (ir)relevance, independence, separability, awareness, and others. As a consequence of all this, forgetting has been addressed with respect to various logics, with differing intuitions and definitional frameworks, but with no clear consensus as to what forgetting *means* as a general concept. Consequently, while there has been much interesting and important work in individual areas, it is difficult to determine how different approaches relate. For example, it is not clear in what sense a forgetting operator based on, say, bisimulations in a modal logic, can be said to correspond to a forgetting operator based on program transformations in an answer set program. Phrased another way, if we could somehow factor out the underlying representation formalisms, how can we know when and whether two approaches to forgetting capture the same phenomenon?

This paper is an attempt to address these issues. Forgetting is considered here as an abstract belief change operator, independent of any specific formal system. That is, the goal is to investigate forgetting at the *knowledge level* (Newell, 1981) and thus independent of syntax. Our thesis is that forgetting amounts to a reduction in the language of a logic, in particular to a reduction in the *signature* of a language.

This leads to a very simple abstract framework. We assume that an agent's beliefs are characterised by a deductively-closed theory (called a *belief set*) in the language of some logic over a signature σ. The operation of forgetting a (sub)signature $\rho \subseteq \sigma$ from a knowledge base K is simply the set of consequences of K expressible over $\sigma \setminus \rho$. This definition offers several advantages and

1. See also the work of Lang, Liberatore, and Marquis (2003) and Konev, Walther, and Wolter (2009b), from which several of these items were taken.
2. For example in saying that a bird is assumed to fly if it is consistent that it flies, the notion of consistency is with respect to the contents of the KB.

benefits. While these suggested advantages and benefits are developed throughout the paper, it is worth listing them here:

1. Since forgetting is expressed at a high level of abstraction, the definition is *simple* and *intuitive*. In large part this is due to the fact that the definition specifies what forgetting *is*, without getting into the orthogonal issue of how forgetting may be computed. The result is a definition of a syntax-independent operator where the results of forgetting are independent of how information is expressed in a knowledge base.

2. The approach is *general*, in that it provides a uniform approach to studying forgetting. It proposes a single definition, applicable to *any* logic with a well-defined consequence relation. By "stepping back" and taking a more general view, various issues are clarified, unified, and simplified. Thus, for example, this perspective leads to the suggestion that forgetting as it has been previously addressed in first-order logic is actually composed of (at least) two distinct operations.

3. The general approach is readily shown to have properties deemed desirable for forgetting, where these results are applicable to all subsumed logics. Many of these results are straightforwardly, and sometimes trivially, obtained in the general framework. In the case of propositional logic, the definition is logically equivalent to the one due to George Boole.

4. The perspective provided by this definition in some cases leads to simpler approaches to computing forgetting in specific logics. In particular, the main definition is expressed in terms of the consequence relation of a logic and not, as is often the case, in terms of a logic's model theory; this in some cases suggests a simpler means of computing forgetting.

5. This abstract view leads to insights with respect to individual logics and the relation between approaches to forgetting. So for example, we consider the relation between forgetting in propositional logic and forgetting in the relevance logic $\mathbf{E}_{fde}$; and we show that the approach sheds light on the notion of *literal elimination* (Lang et al., 2003).

6. The fact that this definition of forgetting is expressed in terms of logical inference, rather than in terms of an underlying model theory, has two further consequences. First, it requires less intricate formal machinery, with fewer ontological assumptions than approaches based, say, on isomorphisms or bisimulations between models, while (as we later argue) being sufficiently expressive for a general notion of forgetting.

   Second, this perspective provides a clear delineation between forgetting and the belief change operation of *contraction*: The former is an operation on the *language* of a logic; while the latter concerns beliefs with respect to a domain of application, and so deals unavoidably with notions of *meaning* and *interpretation* of a language.

At the same time, there are some things that the approach does not address:

1. It is not claimed that this is the "one true" definition of forgetting (although we do suggest that this definition is general and intuitive, and is most appropriate for computational concerns). In the next-to-last section, an alternative definition is discussed, based on second-order quantification.

2. Since the approach is expressed at the knowledge level, it is foremost *representational*, providing a characterisation of forgetting via a definition and desired properties, in a logic-independent fashion. Consequently, issues of implementation and inference are not the focus of the paper. Indeed, since forgetting is defined with respect to belief sets, the main computational challenge is to determine, when possible, a finite representation of forgetting; this is just the problem of determining a *uniform interpolant*. (To be clear: Uniform interpolation and forgetting are distinct notions. A uniform interpolant is a syntactic object which may or may not exist, depending on the underlying logic; forgetting for us is syntax independent, and the result of forgetting is well-defined for any logic.)

The rest of the paper is structured as follows. The next section reviews previous work in forgetting and related concepts. The third section gives formal preliminaries regarding languages, logics, etc. The fourth section presents the general approach, giving the main definition and exploring formal results. The following section examines instances of the general approach, specifically with regards to classical propositional logic, first-order logic, the relevance logic $\mathbf{E}_{fde}$, answer set programs, and (briefly) modal and description logics. This is followed by a discussion of the relation of the present framework with that of other definitions and concepts. We conclude with a brief summing up. A précis of the approach was given by Delgrande (2014), while Delgrande and Wang (2015) addressed the application of the approach to disjunctive logic programs covered in Section 5.4.

2. Background

This section reviews previous work in forgetting. A general familiarity with different classes of logics is assumed. For classical logic, the texts by Mendelson (2015) and Enderton (1972) are among the many excellent introductions. For more on model theory, see the works of Chang and Keisler (2012), Doets (1996), or Hodges (1997). Since forgetting has been defined with respect to most, if not all, major classes of logics, each such class of logic is introduced only very briefly; more information can be found in the given references.

Many approaches to forgetting are based on notions of similarity between models. The strongest notion of similarity is that two models may be *isomorphic*, that is, that there is a bijective mapping between them that "preserves structure". The fact that models w_1 and w_2 are isomorphic is denoted $w_1 \cong w_2$. In modal logics, a key notion is that of a *bisimulation*, where a bisimulation between two models is one in which bisimilar possible worlds have a matching assignment to atomic sentences, and where accessible worlds are matched by bisimilarity; see the text by Blackburn et al. (2001) for a formal definition. That w_1 and w_2 are bisimilar is denoted $w_1 \simeq w_2$. Of most interest to us is the notion of *elementary equivalence*. Models w_1 and w_2 are elementary equivalent, $w_1 \equiv w_2$, just if exactly the same formulas are true at w_1 and w_2, i.e. for every formula ϕ in the language, $w_1 \models \phi$ iff $w_2 \models \phi$. We have that, if models w_1 and w_2 are isomorphic, then they are bisimilar (where defined) and elementary equivalent. The converse in general does not hold in either case.

The rest of this section reviews research in forgetting with respect to specific classes of logics. What should be clear from the discussion below is that definitions of forgetting come in a great variety of different forms. Moreover, many of these definitions are local to a specific approach, and don't readily generalise to other approaches. We return to several of these approaches in Section 5, where they are re-examined once the proposed approach has been explored in Section 4.

2.1 Propositional Logic

The best-known definition of forgetting is due to George Boole. Let $\top$ be a designated atom that is always true and define $\bot$ to be $\neg\top$. Then Boole (1854) suggested that, to forget an atom p from a formula ϕ in propositional logic, one would disjoin the result of uniformly substituting $\top$ for p in ϕ with the result of uniformly substituting $\bot$ for p. That is, let $\phi[p/t]$ denote the result of uniformly substituting $t \in \{\top, \bot\}$ for p; then forgetting p in ϕ is defined by

$$\phi[p/\top] \vee \phi[p/\bot]. \tag{1}$$

For example, forgetting q in $(p \supset q) \wedge (q \supset r)$ is equivalent to $p \supset r$ while forgetting p from the same formula is equivalent to $q \supset r$. Semantically, the models of forgetting p in ϕ are given by: replace each model w of ϕ by two models, w and w' where w' is the same as w except that the truth value of p is complemented.

While the definition (1) has many appealing properties, it does not generally extend to other logics. In particular, it leads to incorrect results if applied directly to a modal logic, and it makes no sense applied to many approaches in logic programming or in many description logics. This leads to an important observation ("claim" may be a better word) regarding the Boole conception of forgetting, which is worth emphasizing: *The Boole conception of forgetting is best regarded not as a* definition *of forgetting, but rather as a means of* computing *forgetting in propositional logic.* The following analogy may be helpful: In propositional logic, a key notion is that of the *satisfiability* of a set of formulas; one means of determining (un)satisfiability is via *resolution*. Resolution isn't of course the same as satisfiability, although a set of formulas is unsatisfiable if and only if the empty clause is derivable via resolution. Instead, resolution is a computational means for determining the satisfiability of a set of formulas. The same is suggested here regarding forgetting: forgetting in propositional logic arguably isn't most appropriately defined via the Boole definition (1); instead (1) represents a means of computing forgetting.

Lang, Liberatore, and Marquis (2003) address *independence* in propositional logic, in part by developing two notions of forgetting. The first, semantic notion, *variable forgetting* corresponds to the Boole definition. In the second, more syntactic notion, *literal forgetting*, one may forget a literal l while retaining previous knowledge concerning the complement $\bar{l}$. Subsequently, Lang and Marquis (2010) use variable forgetting for restoring consistency in propositional knowledge bases.

Nayak, Chen, and Lin (2007) explore the possible relation between forgetting and *belief erasure* (Katsuno & Mendelzon, 1992). Specifically they show that in the approach of Dalal (1988), Boole-style forgetting is definable via erasure applied to literals. As well, they show that these results hold if the Dalal measure of distance between interpretations is replaced by a preorder according to Winslett (1988). However these results hold only when the item for forgetting/erasure is a literal; indeed, as they note, it is unclear what forgetting a propositional formula would amount to in their approach.

2.2 First-Order Logic

Quite possibly the best-known paper on forgetting is due to Lin and Reiter (1994), which addresses forgetting in first-order logic. Two types of forgetting are addressed: forgetting a fact (i.e. a ground atomic formula) and forgetting a relation. In each case, the primary definition of forgetting is semantic. In forgetting a fact $P(\vec{t})$, M' is a model characterising the forgetting of $P(\vec{t})$ if M' agrees exactly with some model M of the agent's knowledge, except possibly for the truth assignment to

$P(\vec{t})$. A syntactic characterisation is given for finite theories that has a form analogous to the Boole definition for propositional logic.

In forgetting a relation, a similar semantic definition is given: for predicate symbol P, a relation $\cong_P$ between models[3] is defined so that $M_1 \cong_P M_2$ just if M_1 and M_2 agree on the interpretation of all symbols (viz. are isomorphic) with the possible exception of P. Then the models of forgetting P in a theory T are just those models M' where for some model M of T, $M \cong_P M'$ holds. Thus if T is some theory, then the result of forgetting a predicate symbol P yielding theory T' has models:

$$Mod(T') = \{M' \mid \exists M \in Mod(T), M \cong_P M'\}. \tag{2}$$

An equivalent syntactic definition in second-order logic is also given, where below X is a second-order predicate variable of appropriate arity:

$$forget(\phi, P) = (\exists X)\phi[P/X]. \tag{3}$$

Lin and Reiter note that forgetting in this case is not always first-order definable.

The Lin/Reiter work was motivated by an issue in reasoning about action, specifically where one wants to *progress* an agent's knowledge (Lin & Reiter, 1997) by "forgetting" about earlier actions and states of the world, while not losing knowledge of present and possible future outcomes.[4] In the same vein, Rajaratnam et al. (2014) considers forgetting a ground atomic formula, again in a theory of action. This last approach is compared to the belief change operation of contraction, and is shown to satisfy the basic AGM contraction postulates.

Christoph Wernhard has also studied forgetting and the dual operation of projection in first-order logic in the context of the Herbrand base. To this end, Wernhard (2004) considers forgetting a predicate symbol (more accurately, all ground instances sharing a given predicate symbol), and later (2008) addresses forgetting a literal. Forgetting is expressed in terms of second-order quantifier elimination, and so begins with an expression of the form (3) and re-expresses it in terms of the original logic; see also the work of Wernhard (2009) and of Gabbay, Schmidt, and Szalas (2008).

Forgetting in first-order logic is also studied by Zhang and Zhou (2010). There they identify *strong forgetting* with the Lin and Reiter definition (2). They also define *weak forgetting*, by giving a set of four postulates focussed on a notion of *irrelevance* between a theory and symbols in a signature. A representation result is provided, in terms of elementary equivalence between models, but ignoring the forgotten symbols. Zhou and Zhang (2011) continue this work, in addressing *bounded forgetting*, in which only sentences where the maximum nesting of quantifiers is less than a given constant are considered.

2.3 Answer Set Programming

Answer set programming (ASP) (Gelfond & Lifschitz, 1988; Baral, 2003) is a prominent knowledge representation approach, falling within the logic programming paradigm. The two major approaches deal with *disjunctive* and *normal* logic programs. A *general logic program* over an alphabet of atoms σ is a finite set of rules of the form

$$a_1; \dots; a_m; \sim b_1; \dots; \sim b_n \leftarrow c_1, \dots, c_o, \sim d_1, \dots, \sim d_p, \tag{4}$$

3. Lin and Reiter use $\sim_P$.
4. See also the work of Vassos and Levesque (2013) which, inter alia, distinguishes *strong progression* which is the Lin/Reiter notion, and *weak progression* which is based on elementary equivalence and is always first-order definable.

where $a_i, b_j, c_k, d_l \in \sigma$, where $m, n, o, p \geq 0$, and where $m + n + o + p > 0$. A disjunctive logic program is one where $n = 0$ for all rules, and a normal logic program is a disjunctive program where $m \leq 1$ for all rules. Operators ';' and ',' express disjunction and conjunction respectively, while the unary operator '$\sim$' is (default) negation. A program induces a set of (0 or more) *answer sets*, where an answer set is, roughly, a minimal set of atoms satisfying the rules.

Given the interest in ASP and in developing comprehensive programming environments for ASP (Gebser, Kaminski, Kaufmann, & Schaub, 2012), there has been an attendant interest in addressing forgetting within this paradigm. However, answer set programs are nonmonotonic with respect to the set of answer sets; for example the ASP program $\{a \leftarrow \sim b.\}$ has one answer set $\{a\}$, while $\{a \leftarrow \sim b.,\ b.\}$ has answer set $\{b\}$. This nonmonotonicity substantially complicates any definition of forgetting. It can be noted that the Boole definition for forgetting does not extend to these logic programs, since the disjunction of two disjunctive logic programs isn't a disjunctive program.

Early approaches focussed on using program transformations to compute a notion of forgetting. For example, Wang, Sattar, and Su (2005) transform a normal logic program into a negative program (with no atoms in the body of a rule) and then compute forgetting. In their approach, answer sets are preserved under forgetting. This approach has been further developed and extended to disjunctive logic programs by Eiter and Wang (2006, 2008). The concepts of strong and weak forgetting (Zhang & Foo, 2006) are similarly defined in terms of a set of program transformations of normal logic programs. Last, Eiter et al. (2006) have addressed forgetting in HEX programs, with intended application to ontologies, again by rule transformation. In all these approaches, forgetting may produce different results for two programs that are strongly equivalent (Turner, 2001).[5] Thus, depending on how a logic program is expressed, one may get different results for forgetting the same symbol. This is partly addressed by Wong (2007) (see also Wong, 2009), where a notion of *T-equivalence* is defined via three inference rules, and where it is then shown that Zhang and Foo's strong and weak forgetting preserve T-equivalence.

More recently, attempts have been made to define forgetting under SE models. Wang et al. (2012, 2014) provide an analysis of forgetting in answer set programs via forgetting in the closely-related area of HT-logic, in which they address forgetting with respect to HT stable models (Lifschitz, Pearce, & Valverde, 2001) and FLP stable models (Truszczynski, 2010). As well, Wang et al. (2013) develop a model-based approach to forgetting in general logic programs based on the stable model semantics; they show that it satisfies what they suggest is a suitable set of properties for a nonmonotonic approach. This work does not always guarantee the existence of results of forgetting in some classes of logic programs; see Examples 5 and 6 of Wang et al. (2014) and Example 3 of Wang et al. (2013).

A notable approach to forgetting in answer set programs is presented in the thesis of Wong (2009). The focus in this work is on extending and analysing the approaches to weak and strong forgetting of Zhang and Foo (2006). In so doing, Wong extends these results to disjunctive logic programs, develops inference systems that capture SE equivalence and T-equivalence, and develops syntax-independent versions of these approaches to forgetting.

The work of Gabbay, Pearce, and Valverde (2011) is also relevant to forgetting under SE models. This work is primarily concerned with various logical properties, such as an interpolation property for equilibrium logic and ASP. Gabbay et al. touch briefly on uniform interpolation with respect to disjunctive logic programs; however, a uniform interpolant is generated directly from a program's

5. The notion of *strong equivalence* corresponds to logical equivalence in classical logic, and the notion of *SE model* corresponds to the notion of model in classical logic.

set of answer sets. In common with this work, the approaches of Either and Wang (2006, 2008), Wang et al. (2014), and Wang et al. (2013) determine the result of forgetting through the generation of answer sets/equilibria. Consequently, in these approaches the structure of the original program is lost. Moreover, the size of the result of forgetting may be exponentially larger in the size of the input program.

Delgrande and Wang (2015) present an approach to forgetting in disjunctive logic programs. This approach can be seen as a specific instance of the general approach presented herein. Forgetting an atom from program P is characterised by the set of those SE consequences of P that do not mention the atom to be forgotten. An equivalent, syntactic, characterization of forgetting is provided, where the result of forgetting an atom p is given by those rules in the program that do not mention p, together with rules obtained by a single inference step from those rules that do mention p. The approach is syntax-independent, and the result of forgetting an atom in two strongly equivalent programs is again two strongly equivalent programs. Interestingly, it also proves to be the case that the system F_S, developed by Wong (2009), is formally equivalent to this approach (and thus precedes it), although the underlying definitions differ.[6]

The above approaches and others are extensively surveyed and compared by Gonçalves, Knorr, and Leite (2016a). The paper reviews and compares various properties that have been proposed in the literature for forgetting in answer set programming; as well, different approaches are compared and classified with respect to these properties. In subsequent work (Goncalves, Knorr, & Leite, 2016b), the property of *strong persistence* (which essentially corresponds with uniform interpolation) in answer set programming is analysed, and where it is shown that this property cannot always be guaranteed.

Forgetting an action in the closely-related area of action languages (Gelfond & Lifschitz, 1998) has been addressed by Erdem and Ferraris (2007). Given that an action description has a natural modelling in terms of a transition diagram (viz. a directed graph showing action transitions between states), forgetting an action has an intuitive modelling in terms of deleting edges named by the action to be forgotten. As well, a syntactic characterisation in terms of transforming an action description is also given.

2.4 Modal Logic

In a modal logic (Chellas, 1980; Hughes & Cresswell, 1996; Blackburn et al., 2001), a proposition is true or false at a *possible world*. In the most common formulation of a possible worlds semantics, a model is given by a triple $M = \langle W, R, P \rangle$ where W is a set (of possible worlds), P is an assignment of truth values to atomic sentences at possible worlds, and R is a binary relation between possible worlds, specifying which worlds are *accessible* from which other worlds. In such a model, at a possible world one can distinguish a sentence being true from being *necessarily* true, or true in all accessible possible worlds. In brief, $M, w \models \phi$ asserts that ϕ is true at world w in model M while $M, w \models \Box\phi$ asserts that ϕ is true at every world w' such that Rw, w'. An important use of modal logic is in modelling an agent's beliefs. In such an *epistemic logic*, the modality K is commonly used in place of $\Box$, and the accessibility relation usually specifies an equivalence relation.

Initial work on forgetting in modal logic focussed on computing uniform interpolants. To this end, Ghilardi (1995) investigates uniform interpolation in the base logic K, while Ghilardi and Za-

6. I thank a reviewer for pointing this out to me, as well as alerting me to Wong's (2009) thesis.

wadowski (1995) show that the logic S4 does not have the uniform interpolation property. Herzig and Menguin (2008) extend modal resolution to address variable forgetting in K.

Van Ditmarsch et al. (2009) present an approach to what they call *introspective forgetting*, in which the notion of becoming ignorant is modelled as an event in a dynamic epistemic logic. After forgetting an atom p, all of $\neg Kp$, $\neg K\neg p$ and $K(\neg Kp \wedge \neg K\neg p)$ hold, and so the agent doesn't know that p is true nor false, but does know about its ignorance. According to van Ditmarsch et al. (2009), there is no hard notion of what it means to forget something, and indeed a notion of forgetting-as-becoming-unaware is also explored. They mention inter alia that the three definitions for variable forgetting given by Lang, Liberatore, and Marquis (2003) fail in their framework; this then serves as an example where a definition for forgetting doesn't map over to another formalism.

Baral and Zhang (2005) study knowledge update, where they treat forgetting of knowledge as a special form of belief *erasure*. In forgetting a formula ϕ, the agent would not know the truth values of ϕ, and so both $\neg K\phi$ and $\neg K\neg\phi$ would hold. Zhang and Zhou (2009) also address forgetting in the epistemic logic S5. A set of four postulates is proposed to characterise forgetting, and a characterisation result is given in terms of bisimulations. They also point out that in the work of both van Ditmarsch et al. (2009) and Baral and Zhang (2005), if one restricts the respective approach to classical propositional logic, the resulting notions of forgetting differ from the Boole definition of propositional variable forgetting.

2.5 Description Logic

A description logic (DL) (Baader, Calvanese, McGuiness, Nardi, & Patel-Schneider, 2007) can be regarded as a fragment of first-order logic, expressed in a convenient syntax and typically with good complexity properties, for describing and making assertions about concepts and their instances. A DL knowledge base consists of a *TBox* and an *ABox*. The *TBox* describes *terminology* or vocabulary of the domain, in terms of a structured description. The *ABox* is a set of *assertions* about named individuals in terms of the vocabulary of the TBox. The TBox contains a set of *concepts*, which denote sets of individuals, and a set of *roles*, which denote binary relations between individuals. Given various constructors, one can specify structured concepts (and, in some cases, roles). Assertions in the TBox are given in terms of *concept subsumption* and *equivalence*. For example,

$$\begin{aligned} \mathit{Mother} &\equiv \mathit{Woman} \sqcap \exists hasChild.Person \\ \mathit{MotherWoD} &\equiv \mathit{Mother} \sqcap \forall hasChild.Male \end{aligned}$$

expresses that a mother is a woman with at least one child (who is a person); while a mother without daughters is a mother all of whose children are male. There are a great many possible constructors; by selecting different constructors one obtains different description logics. Perhaps the best known of the more expressive DLs is called $\mathcal{ALC}$; more recently less expressive DLs, notably $\mathcal{EL}$ and *DL-Lite*, have received increasing attention. Description logics have received a great deal of attention in recent years, in part due to the large number of individual logics, but also due to the success in applying them in various domains and applications. This interest has also led to substantial work on areas closely related to forgetting, in particular uniform interpolation, and also in conservative extensions, semantic difference, and the like.

With respect to $\mathcal{ALC}$, a uniform interpolant with respect to a TBox and given signature can't necessarily be expressed in $\mathcal{ALC}$, nor in first-order logic (Ghilardi, Lutz, & Wolter, 2006). The issue of approximating and computing a uniform interpolant, where such exists, has been investigated by

Wang et al. (2009) and Wang et al. (2010b). Lutz and Wolter (2011) address model-theoretic characterizations of uniform interpolants in terms of bisimulations, along with results on computing interpolants. Koopmann and Schmidt (2015) investigate an approach to computing uniform interpolation in $\mathcal{ALC}$ ABoxes, showing that again the results may not always be representable in the language of the ABox; see also the work of Ludwig and Konev (2014).

Regarding less expressive description logics, Wang et al. (2010a) consider forgetting concepts in *DL-Lite*; they provide a model-based definition, noting that the results of forgetting are not always expressible in *DL-Lite*. They subsequently give three algorithms for computing forgetting. Kontchakov et al. (2010) address forgetting in *DL-Lite* in the context of ontology comparison and module extraction; they show that a uniform interpolant always exists in the description logics DL-Lite$^{\mathcal{N}}_{bool}$ and DL-Lite$^{\mathcal{N}}_{horn}$. With regards to $\mathcal{EL}$, Konev, Walther, and Wolter (2009a) develop a framework for forgetting concepts and roles in an extension of $\mathcal{EL}$. Three different notions of forgetting are investigated; each definition is based on a restricted notion of elementary equivalence, restricted to a particular query type. Nikitina and Rudolph (2014) develop an approach for computing a uniform interpolant for a general $\mathcal{EL}$ TBox, if it exists.

3. Formal Preliminaries: Logics and Related Concepts

The general notion of a *logic* is intuitively clear, whether classical propositional or first-order logic, a modal logic, a description logic, or some other approach. Logic as a subject is often described as the formal analysis of the principles of correct reasoning. Interestingly, there is little work on the concept of a logic as an abstract formal entity. A volume edited by Barwise and Feferman (1985) studies logic in the abstract, but with a strong first-order-logic flavour. As well, Israel (1993) provides an overview of the role of logic in AI that includes an informative account of what goes into making up a logic.

In the first subsection below, relevant concepts having to do with the *language* of a logic are introduced, while the second subsection covers notions having to do with deductive consequence. Since our approach to forgetting is based on deductive consequence, this is all that is required regarding background material for the approach. However, in order to compare the present approach with those approaches to forgetting that make use of semantic concepts, there are two further subsections. The third subsection covers semantics, which deals with *models* and which is used to define the notion of an abstract *logic*, the subject of the last subsection.

3.1 Languages

A language in general is simply a set of finite strings, or in this case, *formulas* or *sentences*.[7] Typically in the case of a logic, one assumes first that the set of formulas is spelled out in terms of an inductive definition on a given *vocabulary*. Formulas are intended to assert something regarding an underlying domain of application; that is, formulas are ultimately intended to represent or denote *propositions*. To this end, the vocabulary of a language is made up of two disjoint, countable sets, the *logical* and *non-logical* symbols. The meaning of the logical symbols is fixed, like the keywords in a programming language, and is made up of logical connectives (e.g. perhaps $\wedge$, $\neg$, etc.), and perhaps punctuation and variables. When we come to discuss individual logics, the symbols $\neg$, $\wedge$,

7. "Formula" and "sentence" will be used interchangeably, except in first-order logic where, as usual, a sentence is a formula with no free variables.

$\vee$, $\supset$, and $\equiv$ will have their usual interpretation; however, in general we don't assume the presence of any particular connective in a language. The meaning of the non-logical symbols, or *signature*, is not fixed, but depends on a domain of application, as described in the subsection on semantics. Thus in a propositional logic, a set of propositional atoms makes up the non-logical symbols, while in first-order logic, the non-logical symbols include constant, predicate, and function symbols.

However, in developing the approach, this level of detail is not required. We will take a signature to be some set, informally the set of non-logical symbols, but formally just some set. A *language* $\mathcal{L}$ is a function from a signature to another set, informally the set of *formulas*, subject to a restriction given below. For notation, individual signatures are denoted by σ (σ', etc.) or ρ (ρ', etc.).[8] $\mathcal{L}$ will stand for some language, and so $\mathcal{L}(\sigma)$ will be the instance of $\mathcal{L}$ with signature σ. For readability, $\mathcal{L}(\sigma)$ will be written as $\mathcal{L}_\sigma$. We also talk of $\mathcal{L}_\sigma$ (and other entities) as being *over* signature σ. Formulas are denoted by lower-case Greek letters ϕ, ψ, ... The upper-case Greek letter Γ is used to denote a set of formulas. These symbols may be subscripted or primed, as may other subsequently-introduced terms.

With this notation, our restriction on languages is:

$$\phi \in \mathcal{L}_\sigma \text{ and } \phi \in \mathcal{L}_{\sigma'} \quad \text{iff} \quad \phi \in \mathcal{L}_{\sigma\cap\sigma'} \tag{5}$$

This has two important consequences. First, we obtain for a language $\mathcal{L}$, that if $\sigma \subseteq \sigma'$ then $\mathcal{L}_\sigma \subseteq \mathcal{L}_{\sigma'}$. Second, we can define the signature of a formula ϕ, $\Sigma(\phi)$, as being the minimal signature σ such that $\phi \in \mathcal{L}_\sigma$. That is, $\Sigma(\phi) = \sigma$ iff $\phi \in \mathcal{L}_\sigma$ and for any $\sigma' \subset \sigma$, $\phi \notin \mathcal{L}_{\sigma'}$. The fact that this notion is well-defined follows from (5). This notation extends in the natural way to sets of sentences.

Example 1 *Let $\mathcal{L}$ be the language of propositional logic. For an instance of $\mathcal{L}$, the signature is a sets of atoms, and thus for such a set σ, $\mathcal{L}_\sigma$ is the instance of the language over that signature.*

For $\sigma = \{p, q, r\}$, we have that $p \vee q \in \mathcal{L}_\sigma$ and $p \vee s \notin \mathcal{L}_\sigma$. Further, $\Sigma(p \wedge (q \vee \neg q)) = \{p, q\}$, and for $\Gamma = \{p,\ q \vee \neg q\}$, we have $\Sigma(\Gamma) = \{p, q\}$.

3.2 Deductive Consequence

Deductive consequence deals with what formulas follow deductively from a given set of formulas. Let Γ be a set of formulas and ϕ a formula in some underlying language $\mathcal{L}$ with understood signature; then $\Gamma \vdash \phi$ indicates that ϕ is a (deductive) consequence of Γ. A *theory* is just a set of formulas, while a *belief set* is a deductively-closed set of formulas. That is, Γ is a belief set just if:

$$\phi \in \Gamma \quad \text{iff} \quad \Gamma \vdash \phi.$$

A consequence relation is not arbitrary, but rather must satisfy some constraints. The following are usually taken as a minimal set of such conditions, and a relation satisfying these conditions is generally known as a Tarskian consequence relation:

Reflexivity: If $\phi \in \Gamma$, then $\Gamma \vdash \phi$.

Cut: If $\Gamma \vdash \psi$ for every $\psi \in \Gamma'$, and $\Gamma \cup \Gamma' \vdash \phi$, then $\Gamma \vdash \phi$.

8. Mnemonically, σ and variants will be used to denote the signature of a language, set of models, or some other formal entity, while ρ will denote that part of a signature to be forgotten. Formally there is no difference between σ and ρ: both are simply sets.

Monotony: If $\Gamma \vdash \phi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash \phi$.

For sets of formulas Γ and Γ', the notation $\Gamma \vdash \Gamma'$ asserts that $\Gamma \vdash \psi$ for every $\psi \in \Gamma'$. $\Gamma \leftrightarrow \Gamma'$ abbreviates $\Gamma \vdash \Gamma'$ and $\Gamma' \vdash \Gamma$. For a set with a singleton element we may simplify notation, writing $\Gamma \leftrightarrow \{\phi\}$ as $\Gamma \leftrightarrow \phi$. For language $\mathcal{L}_\sigma$ with signature σ where $\Gamma \subseteq \mathcal{L}_\sigma$, define

$$\mathcal{C}n_\sigma(\Gamma) = \{\phi \mid \Gamma \vdash \phi \text{ where } \phi \in \mathcal{L}_\sigma\}$$

Example 2 *Assume that the underlying logic is classical propositional logic, and let the language have signature $\sigma = \{p, q, r\}$. Let $\phi = p \wedge (q \vee \neg q)$.*

Then $\quad p \vee r \in \mathcal{C}n_\sigma(\{\phi\}), \quad p \vee s \notin \mathcal{C}n_\sigma(\{\phi\}), \quad p \vee s \in \mathcal{C}n_{(\sigma \cup \{s\})}(\{\phi\})$.

Note that $p \vee s \notin \mathcal{C}n_\sigma(\{\phi\})$ because $s \notin \sigma$.

The rest of this subsection covers terms that will be useful in discussing forgetting. Let $\sigma' \subset \sigma$, $\Gamma' \subseteq \mathcal{L}_{\sigma'}$, and $\Gamma \subseteq \mathcal{L}_\sigma$. Then Γ is a *conservative extension* of Γ' if

$$\mathcal{C}n_{\sigma'}(\Gamma') = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'}.$$

That is, Γ is a conservative extension of Γ' if every theorem of Γ' is a theorem of Γ, and moreover any theorem of Γ in the language of Γ' is a theorem of Γ'. In other words, for languages $\mathcal{L}_{\sigma'} \subset \mathcal{L}_\sigma$ and theories Γ' and Γ of $\mathcal{L}_{\sigma'}$ and $\mathcal{L}_\sigma$ respectively, Γ is a conservative extension of Γ' iff for every $\phi \in \mathcal{L}_{\sigma'}$, $\Gamma' \vdash \phi$ iff $\Gamma \vdash \phi$. Moreover, if there is a formula $\chi \in \mathcal{L}_{\sigma'}$ where $\chi \leftrightarrow \Gamma'$, then χ is a *uniform interpolant* of Γ with respect to σ'.

A logic for which a uniform interpolant always exists for finite theories has the *uniform interpolation property*. It is well known that not every logic has this property. For example, classical and intuitionistic propositional logic, and the modal logics K and S5 do, while classical first-order logic and the modal logic S4 do not.

3.3 Semantics

The relation of deductive consequence satisfies various constraints, and different logics will have different consequence relations. The *semantic theory*, or *model theory*, of a logic can be seen as providing an independent, or prior, justification of principles of deductive reasoning. In the semantics, the fundamental notion is that of *logical entailment*; informally, a formula ϕ is logically entailed by a set of formulas Γ just if, in any state of affairs in which Γ holds, ϕ must also hold.

The notion of a "state of affairs" is expressed formally in the definition of a model. A *model* provides an interpretation, or an account of the meaning, of a signature. So for example, in propositional logic a signature σ is comprised of a set of atoms, and a model can be specified by a complete, consistent set of literals drawn from σ. As another example, in first-order logic a model consists of, first, a set of individuals D. Then, a constant symbol in the signature denotes an element of D; each n-place predicate symbol in the signature denotes an n-ary relation over D; while denotations of function symbols are similarly defined.

Generally in reasoning within a logic, a class of models is taken as being over some fixed signature. However given that our interests lie with forgetting, when it comes to examining model-theoretic aspects of the approach, we will need to refer to classes of models having different signatures. To this end we have the following terminology. Let σ and σ' be signatures where $\sigma' \subseteq \sigma$.

Then σ' is a *reduct* of σ, and σ is an *expansion* of σ'.[9] These notions extend to models: If $\sigma' \subseteq \sigma$ are two signatures, w', w are two models with signature σ', σ respectively, and if furthermore w is the same as w' over σ' (and so w' and w agree on the interpretation of symbols in σ'), then w' is the σ'*-reduct* of w, and w is a σ*-expansion* of w'. For model w and signature σ, $w_{|\sigma}$ denotes the reduct of w with respect to σ, whereas for $\Sigma(w) \subseteq \sigma$, $w_{\uparrow\sigma}$ is the set of expansions of w with respect to σ. In general, a model w with signature σ may have many expansions to models with signature $\sigma' \supset \sigma$; however the reduct of w to a given subsignature is unique.

Again, as in the definition of a language, we do not require this level of detail. For our needs, a class of models is given by a function $\mathcal{M}$ from a set, informally a signature, to another set, informally the models with that signature. Similar to languages, $\mathcal{M}(\sigma)$, the class of models over a signature σ, will be written $\mathcal{M}_\sigma$. Individual models are denoted w (w_1, w', etc.). The function $\mathcal{M}$ is subject to the following restriction:

$$\text{If } w \in \mathcal{M}_\sigma \text{ and } \sigma \neq \sigma' \quad \text{then} \quad w \notin \mathcal{M}_{\sigma'}. \tag{6}$$

Thus each model w is associated with a unique signature, denoted $\Sigma(w)$.

The *reduct* of a model with respect to a signature, $\mathit{Reduct}(w, \sigma)$, is a two-place function in which $\mathit{Reduct}(w, \sigma) \in \Sigma(\sigma)$; it is defined only if $\sigma \subseteq \Sigma(w)$. The reduct function is henceforth written using the aforementioned notation, so that $\mathit{Reduct}(w, \sigma)$ is expressed $w_{|\sigma}$. Expansion is defined as a converse of the reduct: if $\Sigma(w) = \sigma$ and $w' = w_{|\sigma'}$ then $w \in w'_{\uparrow\sigma}$. For any model it is assumed that an expansion to a larger signature exists, i.e. for $\Sigma(w) = \sigma$, and $\sigma \subseteq \sigma'$, there is w' where $w' \in \mathcal{M}_{\sigma'}$ and $w = w'_{|\sigma}$.

3.4 Logics

To recap to this point: A *language* $\mathcal{L}$ is a function from a set (informally, a signature) to another set (informally the set of formulas with that signature) that satisfies the assumption (5); as described this allows us to identify the signature of a formula. Similarly the class of all models is specified by a (model) function $\mathcal{M}$ from a set (of signature elements) to a class (of models with that signature). This function has a restriction (6) that $\mathcal{M}$ partitions the class of models. The reduct associates a model with a certain signature with another model with a given subsignature. Given these considerations, we can define the notion of an *(abstract) logic*.[10]

Definition 1 *A* logic *is a triple* $L = (\mathcal{L}, \mathcal{M}, \models)$ *where*

1. *$\mathcal{L}$ is a language function,*
2. *$\mathcal{M}$ is a model function,*
3. $\models \; \subseteq \; \{(w, \phi) \mid \exists\sigma \textit{ s.t. } w \in \mathcal{M}_\sigma, \phi \in \mathcal{L}_\sigma\}$ *is the* satisfaction relation *between models and formulas,*

and subject to the following restrictions:

9. These terms are standard in model theory. *Reduct* should not be confused with the distinct notion of *reduct* in answer set programming. Similarly, *expansion* has no relation to the belief change operator of *expansion* in the AGM approach.
10. This definition is adapted from one proposed by Delgrande, Peppas, and Woltran (2016); some notions regarding signatures are adapted from the works of Barwise and Feferman (1985), and García-Matos and Väänänen (2005).

1. $w \models \phi$ *is* $\begin{cases} \textit{true} & \textit{if } \Sigma(\phi) \subseteq \Sigma(w) \textit{ and } (w, \phi) \ \in \models \\ \textit{false} & \textit{if } \Sigma(\phi) \subseteq \Sigma(w) \textit{ and } (w, \phi) \ \notin \models \\ \textit{undefined} & \textit{otherwise} \end{cases}$

2. Reduct Property*:*

   If $\Sigma(\phi) = \sigma'$*, then:* $\quad w \models \phi$ *iff* $w_{|\sigma'} \models \phi$

3. Renaming Property*:*

   $w \models \phi$ *is preserved under uniform renaming of elements of the signature.*

*An (*instance*) of a* logic L over signature σ *is given by* $L_\sigma = (\mathcal{L}_\sigma, \mathcal{M}_\sigma, \models)$.

The first restriction states that the satisfaction relation is defined for an instance $w \models \phi$ only if the signature of the formula ϕ is a subsignature of the model w. The reduct property states that whether $w \models \phi$ holds or not depends only on $\Sigma(\phi)$. The renaming property says that symbols of the signature have no intrinsic meaning, but rather associate elements of the language with parts of models.

Other familiar notions are defined the same as for classical logic. Assume we have an instance of logic $L = (\mathcal{L}, \mathcal{M}, \models)$ over signature σ. If for formula ϕ there is a model $w \in \mathcal{M}_\sigma$ such that $w \models \phi$, we say that ϕ is *satisfiable* and that it is *satisfied by* w or *true at* w; moreover, we say that w is a *model of* ϕ. A set of sentences $\Gamma \subseteq \mathcal{L}_\sigma$ is satisfiable if there is a model $w \in \mathcal{M}_\sigma$ such that $w \models \Gamma$, that is to say, $w \models \phi$ for every $\phi \in \Gamma$. The set of models of ϕ over signature σ, where necessarily $\Sigma(\phi) \subseteq \sigma$, is denoted by $Mod_\sigma(\phi)$; this notation extends to sets of formulas. A formula ϕ is *logically entailed* by a set of formulas Γ, written $\Gamma \models \phi$, just if $Mod_\sigma(\Gamma) \subseteq Mod_\sigma(\phi)$ where σ is any signature in which $\Sigma(\phi), \Sigma(\Gamma) \subseteq \sigma$. The notation for reduct and expansion extends to sets of models in the obvious way. Thus, for $\sigma' \subseteq \sigma$, $\Sigma(\phi) \subseteq \sigma$, and $\Sigma(\phi') \subseteq \sigma'$,

$$\begin{aligned} Mod_\sigma(\phi)_{|\sigma'} &= \{w_{|\sigma'} \mid w \in Mod_\sigma(\phi)\} \quad \text{and} \\ Mod_{\sigma'}(\phi')_{\uparrow\sigma} &= \{w_{\uparrow\sigma} \mid w \in Mod_{\sigma'}(\phi')\} \end{aligned}$$

We will assume in general for a given logic that logical entailment is exactly captured by an associated consequence relation; that is we have the theorem:

$$\Gamma \models \phi \ \text{ iff } \ \Gamma \vdash \phi. \tag{7}$$

Delgrande, Peppas, and Woltran (2016) show that any logic, according to Definition 1, defines a Tarskian consequence relation via (7).

Here are some examples illustrating the above concepts.

Example 3 *In propositional logic, let* $\sigma_1 = \{p, q, r, s, t\}$ *and* $\sigma_2 = \{q, r\}$*. Models can be identified with complete, consistent sets of literals.*

For $w_1 = \{p, \neg q, r, s, t\}$ *and* $w_2 = \{\neg q, r\}$ *we have that* $w_2 = w_{1|\sigma_2}$ *and* $w_1 \in w_{2\uparrow\sigma_1}$*.*

Let $\Gamma = \{p \vee q,\ \neg p \vee r,\ s \wedge t,\ t \supset r\}$*. Then*

$$\begin{aligned} Mod_{\sigma_1}(\Gamma) &= \{\{p, q, r, s, t\}, \{p, \neg q, r, s, t\}, \{\neg p, q, r, s, t\}\} \\ Mod_{\sigma_1}(\Gamma)_{|\sigma_2} &= \{\{q, r\}, \{\neg q, r\}\} \end{aligned}$$

Example 4 *In first-order logic, a model is a pair $\langle D, \mathcal{I} \rangle$ where D is a domain of individuals and $\mathcal{I}$ is an* interpretation mapping.

Assume we have signatures:

$$\sigma_1 = \{chris, leslie, Student\}, \quad \sigma_2 = \{chris, Student\}$$

where $chris$ and $leslie$ are constant symbols and $Student$ is a 1-place predicate symbol. Let w_1 be the model with signature σ_1 with

$$D = \{o_1, o_2, o_3\},\ \mathcal{I}(chris) = o_1,\ \mathcal{I}(leslie) = o_2,\ \mathcal{I}(Student) = D$$

and let w_2 be the model with signature σ_2 with

$$D = \{o_1, o_2, o_3\},\ \mathcal{I}(chris) = o_1, \qquad\qquad \mathcal{I}(Student) = D.$$

Then we have that $w_2 = w_{1|\sigma_2}$.

Note that for $w_2 = w_{1|\sigma_2}$, models w_1 and w_2 must have the same domain and agree on the interpretation of shared symbols in their signatures. Thus if w_2' were just like w_2 except that $D = \{o_1, o_2\}$ or $\mathcal{I}(chris) = o_2$ or $\mathcal{I}(Student) = \{o_1, o_2\}$, then $w_2' \neq w_{1|\sigma_2}$.

Last, we consider notions relating sets of models to the existence of sets of formulas; this will be used in the next section, in relating the approach to work in model-based forgetting. First we introduce the following definition, specifying the set of formulas entailed by a set of models. For the instance of $L = (\mathcal{L}, \mathcal{M}, \models)$ over signature σ, and for $X \subseteq \mathcal{M}_\sigma$, define:

$$t(X) \ = \ \{\phi \mid \forall w \in X, w \models \phi\}$$

Since w in the expression above has signature σ, and $w \models \phi$ is only defined for $\Sigma(\phi) \subseteq \sigma$, we have that $\Sigma(t(X)) \subseteq \sigma$ is implied for any $X \subseteq \mathcal{M}_\sigma$.

For a logic $L = (\mathcal{L}, \mathcal{M}, \models)$ over signature σ, $X \subseteq \mathcal{M}_\sigma$ is *elementary* if there is a set of formulas $\Gamma \subseteq \mathcal{L}_\sigma$ such that $Mod_\sigma(\Gamma) = X$. Thus X is elementary just if $X = Mod_\sigma(t(X))$. This leads to the following definition:

Definition 2 *A logic $L = (\mathcal{L}, \mathcal{M}, \models)$ is* hereditarily elementary, *if for any signature σ and any $X \subseteq \mathcal{M}_\sigma$, if X is elementary then for any $\sigma' \subseteq \sigma$, the reduct $X_{|\sigma'}$ is elementary.*

Phrased slightly differently, if Γ is a set of formulas in a hereditarily elementary logic with signature σ, then for signature $\sigma' \subseteq \sigma$ there is a set of formulas Γ' where $Mod_{\sigma'}(\Gamma') = Mod_\sigma(\Gamma)_{|\sigma'}$.

Classical propositional logic over a finite signature is (trivially) hereditarily elementary, since any set of models over a finite signature defines a set of formulas. For example, for a set of models X, each model in X can be represented by a conjunction of literals true at that model, and the set of models represented by the disjunction of such conjunctions. Propositional Horn logic over a finite signature is a non-trivial example of a hereditarily elementary logic. In Horn logic, not every set of models is elementary; a set of models in Horn logic is elementary iff it is closed under intersection of positive literals in an interpretation. For example if a model is represented by the atoms true at that model, then the set of models $\{\{p\}, \{q\}\}$ is not elementary, while the set $\{\{p\}, \{q\}, \emptyset\}$ is. However, if a set X is elementary then $X_{|\sigma'}$ is also elementary. A further example of a hereditary elementary logic is provided by general logic programs (Lifschitz et al., 2001).

Modal logics and logic programs under the answer set semantics provide examples of logics that are not hereditarily elementary. For example, in a modal logic, say K, let the signature be $\sigma = \{p, q\}$ and consider the set of formulas $\Gamma = \{\Diamond p,\ \Diamond\neg p,\ \Box q\}$. Every model of Γ contains at least two possible worlds, one at which p is true and one at which $\neg p$ is true. Informally one might expect $Mod_\sigma(\Gamma)_{|\{q\}}$ to exactly characterise the models of $\Box q$ over signature $\{q\}$. However, all models in $Mod_\sigma(\Gamma)_{|\{q\}}$ have at least two worlds, while $Mod_{\{q\}}(\{\Box q\})$ also contains single-world models. Consequently we have that $Mod_\sigma(\Gamma)$ is trivially elementary, while $Mod_\sigma(\Gamma)_{|\{q\}}$ is not elementary.

Second, for answer set programs, consider the normal program $P = \{p \leftarrow \sim q.,\ q \leftarrow \sim p.,\ \bot \leftarrow p, q.\}$ with signature $\sigma = \{p, q\}$. $Mod_\sigma(P) = \{(p, p), (q, q)\}$ is elementary by definition. However $Mod_\sigma(P)_{|\{p\}} = \{(p, p), (\emptyset, \emptyset)\}$ is not elementary, since there is no normal logic program (nor, for that matter, disjunctive program) with precisely the given set of models.[11]

This concludes the background definitions required for the approach, introduced next. Again, it can be emphasized that the concepts introduced in Section 3.3 and 3.4 concerning the semantics are not required for the approach itself, but are included to facilitate comparison with other approaches to forgetting.

4. The Approach

As indicated at the outset, our thesis is that forgetting is most profitably regarded as an operation that decreases the language of an agent, in that the signature of the agent's language is reduced. As well, since forgetting is intended as an abstract, knowledge level operation, forgetting should apply to the *content* of an agent's beliefs and be independent of how beliefs are represented. If we equate an agent's (implicit) beliefs with the set of sentences derivable from the agent's knowledge base, this leads to the following definition.

Definition 3 *Assume we are given a logic L with language $\mathcal{L}$ and a signature σ. Let $\Gamma \subseteq \mathcal{L}_\sigma$. The result of* forgetting ρ in Γ, *denoted $\mathcal{F}_{L,\sigma}(\Gamma, \rho)$, is defined as:*

$$\mathcal{F}_{L,\sigma}(\Gamma, \rho) \ \doteq\ \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho}$$

Since the underlying logic will always be clear from the context, as will the signature, for simplicity the subscripts L and σ are dropped, and henceforth we just write $\mathcal{F}(\Gamma, \rho)$.

There are several things to note about this definition.

- It is very simple.
- It specifies a unique, unambiguous result. Given a consequence relation, a set of formulas, and a signature, forgetting that signature with respect to the set of formulas is uniquely determined.
- It is applicable to *any* system with a signature and (Tarskian) consequence relation. Thus, it is applicable to, among others, classical (propositional and first-order) logic, relevance logics,

11. A pair such as (p, p) is an *SE model* (Turner, 2001). Discussing the topic of classes of SE models here would take us too far afield; for more see the work of Eiter, Tompits, and Woltran (2005). I thank an anonymous referee for pointing out this example, in a slightly different context.

answer set programs, modal logics, and description logics. Notably, it applies to logics in which a uniform interpolant may not exist, such as classical first-order logic, the modal logic S4, and various relevance logics and description logics.

- The definition is phrased in terms of logical consequence; there is no reference to semantics or an underlying model theory. Hence the development in Sections 3.3 and 3.4 on the semantics of abstract logics is needed only for comparison to related work.

- The definition is at the *knowledge level*, and consequently is independent of how an agent's knowledge is represented. As we show in the sequel, this allows properties of the operator to be easily (in some cases trivially) shown, and it highlights similarities and differences in forgetting in different logics.

On the other hand, since the definition is intended to characterise the result of forgetting, the resulting set of beliefs, $\mathcal{F}(\Gamma,\rho)$, will in general be infinite. Consequently a key question is to determine those instances of $\mathcal{F}$ that may be finitely characterized; and a key computational issue is to determine this finite representation, or uniform interpolant, when possible.

Clearly $\mathcal{F}(\Gamma,\rho)$ results in a reduced language (except in the trivial case where $\sigma\cap\rho=\emptyset$.) It can also be observed that if there is a formula ϕ such that $\mathcal{F}(\Gamma,\rho)\leftrightarrow\phi$, then ϕ is a uniform interpolant with respect to Γ and $\mathcal{L}_{\sigma\setminus\rho}$. Of course, one may wish to re-express the result of forgetting in the original language of Γ; indeed, many existing approaches to forgetting assume that the underlying language is unchanged. To this end, and for comparative purposes, a variant of Definition 3 is given as follows, where the signature of the underlying language is σ:[12]

$$\mathcal{F}_O(\Gamma,\rho) \doteq \mathcal{C}n_\sigma(\mathcal{F}(\Gamma,\rho)) \tag{8}$$

The result is a theory in the original language, but where the resulting theory carries no contingent information regarding elements of ρ.

The following results are elementary, some essentially being observations, but they show that the definition of forgetting has the "right" properties.

Theorem 1 *Let $L=(\mathcal{L},\mathcal{M},\models)$ be a logic and σ a signature, and let $\Gamma,\Gamma'\subseteq\mathcal{L}_\sigma$.*

1. $\Gamma\vdash\mathcal{F}(\Gamma,\rho)$
2. *If* $\Gamma\vdash\Gamma'$, *then* $\mathcal{F}(\Gamma,\rho)\vdash\mathcal{F}(\Gamma',\rho)$
3. $\mathcal{F}(\Gamma,\rho)=\mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma,\rho))$
4. $\mathcal{F}(\Gamma,\rho)=\mathcal{F}(\mathcal{F}(\Gamma,\rho'),\rho)$ *where* $\rho'\subseteq\rho$.
5. $\mathcal{F}(\Gamma,\rho_1\cup\rho_2)=\mathcal{F}(\Gamma,\rho_1)\cap\mathcal{F}(\Gamma,\rho_2)$
6. $\mathcal{F}(\Gamma,\rho_1\cup\rho_2)=\mathcal{F}(\mathcal{F}(\Gamma,\rho_1),\rho_2))$
7. $\mathcal{F}_O(\Gamma,\rho)$ *is a* conservative extension *of* $\mathcal{F}(\Gamma,\rho)$.

12. Mnemonically this can be thought of as *F*orgetting in the *O*riginal language.

Part 1 states that forgetting results in no consequences not in the original theory. Part 2 implies that the result of forgetting is independent of syntax, since $\Gamma \leftrightarrow \Gamma'$ gives $\mathcal{F}(\Gamma, \rho) \leftrightarrow \mathcal{F}(\Gamma', \rho)$. From Part 3 we obtain that forgetting results in a deductively-closed theory in the reduced signature. The next three parts show that forgetting is decomposable with respect to a signature, either by doing forgetting on successive subsets of a signature (which also demonstrates idempotency with respect to a signature), or else by independently forgetting parts of the signature. Forgetting parts of a signature can be done independently (Part 5) or iteratively (Part 6). This in turn implies that forgetting is a commutative and associative operation with respect to signatures. Finally, if the result of forgetting is re-expressed in the original language, the original theory is a conservative extension of the result of forgetting, and so if forgetting is finitely expressible, the result is a uniform interpolant.

Four conditions are proposed by Zhang and Zhou (2009) for characterising an approach to forgetting in the modal logic S5. An analogous[13] result follows here with respect to forgetting re-expressed in the original signature. To begin, we have the definition:

Definition 4 *Signature ρ is* irrelevant *to Γ, $IR(\Gamma, \rho)$, iff there is Γ' such that $\Gamma \leftrightarrow \Gamma'$ and $\Sigma(\Gamma') \cap \rho = \emptyset$.*

That is, ρ is irrelevant to Γ just if Γ is equivalent to a set of sentences that do not mention any element of ρ. For example, in propositional logic, p is irrelevant to $(p \vee \neg p) \wedge q$ but q is not. The following theorem shows that the conditions identified by Zhang and Zhou also characterise forgetting in arbitrary logics.

Theorem 2 *Let Γ, $\Gamma' \subseteq \mathcal{L}_\sigma$ be sets of sentences in a logic with signature σ and let $\rho \subseteq \sigma$.*
Then $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$ iff all of the following hold

1. $\Gamma \vdash \Gamma'$
2. *If $IR(\phi, \rho)$ and $\Gamma \vdash \phi$, then $\Gamma' \vdash \phi$*
3. *If $IR(\phi, \rho)$ and $\Gamma \nvdash \phi$, then $\Gamma' \nvdash \phi$*
4. $IR(\Gamma', \rho)$

For Conditions 2 and 3 of the theorem, we have that, if a formula ϕ is independent of a signature ρ, then forgetting ρ has no effect on whether that formula is a consequence of the original knowledge base or not. The last condition is analogous to the success postulate in AGM contraction: the result of forgetting ρ yields a theory expressible without ρ. It can be noted that Condition 3 is not independent, since it is implied by Condition 1.

Next, we consider the relation between forgetting in a logic L and a logic weaker than L. It proves to be the case that if one goes from a logic to a "contained" logic, then forgetting in the former subsumes forgetting in the latter. Since our definition of forgetting is expressed in terms of inference, for the next result (only) a logic is characterised by its associated language and consequence relation.

Definition 5 *Let L be a logic with language $\mathcal{L}$ and consequence relation $Cn(\cdot)$, and let L' analogously be a logic defined over $\mathcal{L}'$ and $Cn'(\cdot)$.*
Then L' is a sublogic *of L if for any signatures σ, σ' where $\sigma' \subseteq \sigma$,*

13. Zhang and Zhou express their postulates in terms of logical entailment whereas here they are expressed using deductive consequence.

1. *$\mathcal{L}'_{\sigma'} \subseteq \mathcal{L}_\sigma$, and*

2. *for $\Gamma' \subseteq \mathcal{L}'_{\sigma'}$, $\Gamma \subseteq \mathcal{L}_\sigma$, we have: $\Gamma' \subseteq \Gamma$ implies $\mathcal{C}n'_{\sigma'}(\Gamma') \subseteq \mathcal{C}n_\sigma(\Gamma)$.*

Theorem 3 *Let L be a logic with language $\mathcal{L}$ and consequence relation $\mathcal{C}n(\cdot)$, and let L' be a logic with language $\mathcal{L}'$ and consequence relation $\mathcal{C}n'(\cdot)$.*

If L' is a sublogic of L, then for $\sigma' \subseteq \sigma$, $\Gamma' \subseteq \mathcal{L}'_{\sigma'}$, $\Gamma \subseteq \mathcal{L}_\sigma$:

if $\Gamma' \subseteq \Gamma$, then $\mathcal{F}(\Gamma', \rho) \subseteq \mathcal{F}(\Gamma, \rho)$.

The proof of this result is straightforward. However, it is interesting in a number of respects. First, it shows that, for example, the result of forgetting in intuitionistic logic is subsumed by forgetting in classical propositional logic, and that forgetting in Horn theories is similarly subsumed by forgetting in classical propositional logic. Second, not all approaches to forgetting satisfy this theorem. For example, Zhang and Zhou (2009) note that the approach of van Ditmarsch et al. (2009), which addresses forgetting in the modal logic S5, when restricted to propositional logic does not always result in propositional forgetting.

4.1 Model-Theoretic Considerations

The definition of forgetting is given entirely in syntactic, or proof-theoretic, terms. We conclude this section by examining Definition 3 with respect to the models of a theory. Several characterisations are given. First, if Γ has signature σ, the models of $\mathcal{F}(\Gamma, \rho)$ can be determined by "ignoring" the interpretation of the elements of ρ, that is, by taking the reduct of models of Γ with respect to $\sigma \setminus \rho$. If forgetting is to be considered with respect to the original signature σ, then one can take the reduct of models of Γ with respect to $\sigma \setminus \rho$ as before, but then expand each model in this set by σ. In a different direction, a characterisation of forgetting in terms of models is given in terms of elementary equivalence. In each of these cases, we require that a logic be hereditarily elementary, and so for a set of formulas Γ, the reduct of $Mod_\sigma(\Gamma)$ over some signature is guaranteed to be expressible by another set of formulas.

Theorem 4 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a hereditarily elementary logic and σ a signature. Then for any $\Gamma \subseteq \mathcal{L}_\sigma$ and $\sigma' \subseteq \sigma$:*

1. *$Mod_{\sigma'}(\mathcal{F}(\Gamma, \sigma \setminus \sigma')) = Mod_\sigma(\Gamma)_{|\sigma'}$*

2. *$Mod_{\sigma}(\mathcal{F}(\Gamma, \sigma \setminus \sigma')) = (Mod_\sigma(\Gamma)_{|\sigma'})_{\uparrow\sigma}$*

Thus, in forgetting elements of a signature $\rho = \sigma \setminus \sigma'$, the models of the result of forgetting can be determined by either ignoring ρ in a model (Part 1), or else for every model w of Γ, adding models that are the same as w except that they differ in the interpretation of ρ (Part 2). Since the logic is hereditarily elementary by assumption, if a set of models is elementary, then so is any reduct of that set. Thus, in this case Theorem 4 provides a semantic alternative to Definition 3.

Another model-theoretic characterisation of forgetting is given next. Recall that two models w and w' are elementary equivalent, denoted $w \equiv w'$, just if they agree on the satisfaction of all formulas, that is, $w \equiv w'$ iff for every formula ϕ, $w \models \phi$ iff $w' \models \phi$. We can generalise this to w and w' being *elementary equivalent except on signature* ρ, denoted $w \equiv_\rho w'$, by:

Definition 6 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a logic, σ a signature, and let $w, w' \in \mathcal{M}_\sigma$.*
Then $w \equiv_\rho w'$ iff: for every $\phi \in \mathcal{L}_{\sigma \setminus \rho}$: $w \models \phi$ iff $w' \models \phi$.

We obtain the following result, characterising forgetting in terms of elementary equivalence.

Theorem 5 *Let $L = (\mathcal{L}, \mathcal{M} \models)$ be a hereditarily elementary logic, and let σ be a signature. Then for $\Gamma \subseteq \mathcal{L}_\sigma$:*
$Mod_\sigma(\mathcal{F}(\Gamma, \rho)) \ = \ \{w \in \mathcal{M}_\sigma \mid \exists w' \in Mod_\sigma(\Gamma) \text{ such that } w \equiv_\rho w'\}$

5. Applying the Approach

In this section we examine the general approach with respect to specific logics, in particular, classical propositional logic, first-order logic, a relevance logic, disjunctive logic programs, and, very briefly, modal logics and description logics.

Since we will consider means of computing forgetting in different approaches, the following notation will be convenient: Let Γ be a set of sentences in some language, and let ϵ be an element of the signature; define:

$$\Gamma_{\downarrow\epsilon} \ = \ \{\phi \in \Gamma \mid \epsilon \notin \Sigma(\phi)\}.$$

That is, $\Gamma_{\downarrow\epsilon}$ is simply those formulas of Γ that don't mention ϵ. This notion is obviously dependent on syntax; for example $\{a, \neg b\}_{\downarrow b} = \{a\}$ while $\{a \wedge \neg b\}_{\downarrow b} = \{\}$. This won't be an issue, since Γ will be in some normal form when $\downarrow$ is used.

Since in this section we will most often deal with forgetting a single atom, we will freely use $\mathcal{F}(\Gamma, p)$ in place of $\mathcal{F}(\Gamma, \{p\})$, and similarly for related concepts. Forgetting an arbitrary signature is of course expressible via iteratively forgetting an atom using Theorem 1.6.

5.1 Propositional Logic

Let $\mathcal{L}$ be the language of propositional logic, and let σ be a signature consisting of a set of propositional atoms. It will sometimes be easier to work with a set of formulas in conjunctive normal form (CNF) expressed in clause form, rather than arbitrary propositional formulas.[14] For a set of propositional formulas Γ, Γ_{CNF} will be the corresponding set expressed in CNF in clause form, as a set of sets of literals. Since there is no confusion, in the finite case we will freely switch between set-based notation and the corresponding logical formulas.

In the next definition, $Res(S, p)$ is the set of propositional clauses obtained from a set of clauses S by carrying out all possible resolutions with respect to the atom p:

Definition 7 *Let S be a set of propositional clauses and p an atom. Define*

$$Res(S, p) = \{\phi \mid \exists \phi_1, \phi_2 \in S \text{ such that } p \in \phi_1, \neg p \in \phi_2, \text{ and } \phi = (\phi_1 \setminus \{p\}) \cup (\phi_2 \setminus \{\neg p\})\ \}$$

We obtain:

Theorem 6 *Let $\Gamma \subseteq \mathcal{L}$ be a set of formulas and p an atom.*

14. Recall that a formula in clause form is given by a set of sets of literals, where literals in a set are taken as being disjoined and the clauses in a set are taken as conjoined.

1. $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma[p/\top] \vee \Gamma[p/\bot]$ *for finite* Γ.

2. $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma_{CNF\downarrow p} \cup Res(\Gamma_{CNF}, p)$.

The two parts of the theorem provide two means for computing forgetting. Both have the obvious benefit that they capture propositional forgetting in a formula (when the KB is finite) and in a syntax-independent fashion. The formula on the right hand side in the first part is the standard (Boole) definition of forgetting in propositional logic. It has the advantage of being easily computable, and results in no worse than a doubling of the size of a knowledge base. It has several disadvantages: It only works for finite sets of formulas; it results in a disjunction as the top-level connective in a knowledge base (which in general may be awkward to work with); and it does not readily extend to other logics, such as a modal logic, Horn clauses, logic programs, nor, as we will suggest, first-order logic.

The second part of the theorem similarly provides a recipe for computing forgetting: convert a knowledge base to clause form; carry out all possible resolutions with respect to the atom to be forgotten; and add these resolvents to those clauses in the knowledge base that don't mention that atom. There is no need to require a finite knowledge base, and the resulting KB is in a more useful form.[15] Forgetting an atom may result in a quadratic blowup of the knowledge base, although this is with respect to a knowledge base already in clause form. Otherwise, the properties of Theorem 1 and 2 are obviously inherited. As well, these results also hold in the further restriction to propositional Horn clauses (Delgrande & Wassermann, 2013).

Example 5 *Let* $\Gamma = \{ p \vee q \vee r,\ \neg p \vee s,\ \neg p \vee t,\ r \vee u \}$.
Then

$$\begin{aligned}
\Gamma[p/\top] \vee \Gamma[p/\bot] &= \{ ((\top \vee q \vee r) \wedge (\neg\top \vee s) \wedge (\neg\top \vee t) \wedge (r \vee u)) \vee \\
&\qquad ((\bot \vee q \vee r) \wedge (\neg\bot \vee s) \wedge (\neg\bot \vee t) \wedge (r \vee u)) \ \} \\
\Gamma_{\downarrow p} \cup Res(\Gamma, p) &= \{ r \vee u,\ q \vee r \vee s,\ q \vee r \vee t \}.
\end{aligned}$$

These two results are readily shown to be equivalent.

The semantic characterisation of forgetting in propositional theories is provided by a corollary to Theorem 4. In the following, a model with signature σ is identified with a complete, consistent set of literals from σ.

Corollary 1

1. $Mod_{\sigma \setminus \{p\}}(\mathcal{F}(\Gamma, p)) =$
   $\{m \in \mathcal{M}_{\sigma \setminus \{p\}} \mid \exists m' \in Mod_{\sigma}(\Gamma)$ *where* $m = m' \setminus \{p, \neg p\}\}$

2. $Mod_{\sigma}(\mathcal{F}(\Gamma, p)) = Mod_{\sigma}(\Gamma) \ \cup$
   $\{m \in \mathcal{M}_{\sigma} \mid \exists m' \in Mod_{\sigma}(\Gamma)$ *where* $(m \setminus m') \cup (m' \setminus m) = \{p, \neg p\}\}$

15. It might be objected that a KB has to be first converted to CNF. While this is true, it can also be observed that most often a KB will consist of a large number of relatively small formulas, and so will already be "close" to CNF.

For the first part, the models of $\mathcal{F}(\Gamma, p)$ are just the models of Γ where p is excluded from the signature. For the second part, if forgetting is expressed in terms of the original signature, then the models of $\mathcal{F}(\Gamma, p)$ are the models of Γ together with models of Γ where the truth value attached to p is replaced by its complement.

As described in Section 2, Lang, Liberatore, and Marquis (2003) address (among other concepts) *variable forgetting*, which amounts to the usual version of propositional forgetting, and *literal forgetting*, in which only positive (or negative) occurrences of an atom are removed.[16] For a finite set of formulas Γ, the result of forgetting the literal l is given by:

$$ForgetLit(\Gamma, l) \;=\; \Gamma[l/\top] \vee (\bar{l} \wedge \Gamma). \tag{9}$$

Semantically, the models of Γ after forgetting literal l consist of the models of Γ along with those models of Γ that satisfy l, but where the value assigned to l is changed to $\bar{l}$.

Example 6

$$\begin{aligned} ForgetLit((\neg a \vee b) \wedge (a \vee c), \neg a) \;&\equiv\; ((\top \vee b) \wedge (\bot \vee c)) \vee (a \wedge ((\neg a \vee b) \wedge (a \vee c)) \\ &\equiv\; (a \vee c) \wedge (b \vee c). \end{aligned}$$

The following result relates this approach with the present. For Γ in CNF, let $\Gamma_{\Downarrow l}$ be the set of clauses in Γ that don't mention the literal l; for example, $\{p \vee q, \neg p \vee r, s \vee t\}_{\Downarrow p} = \{\neg p \vee r, s \vee t\}$. The following can be compared to Theorem 6.2.

Theorem 7 *Let $\Gamma \subseteq \mathcal{L}$ and let l be a literal.*
$ForgetLit(\Gamma, l) \;\leftrightarrow\; (\Gamma_{CNF})_{\Downarrow l} \;\cup\; Res(\Gamma_{CNF}, p)$.

Corollary 2 $\mathcal{F}(\Gamma, p) \;\leftrightarrow\; ForgetLit(ForgetLit(\Gamma, \neg p), p)$

Literal forgetting has a syntactic flavour, in that a literal and its complement are distinguished. We return to this notion in Section 6, in discussing forgetting in the context of belief change, where we suggest that literal forgetting may (also) be seen as an appropriate base case in an approach to belief update.

5.2 First-Order Logic

We will assume that a signature in first-order logic is made up of three sorts of nonlogical symbols: constants, predicate symbols and function symbols. For the purposes of discussion, we will focus on predicate symbols and constants. Regarding predicate symbols, we have the following result, which is a corollary of Theorem 5:

Corollary 3 *Let Γ be a set of sentences of first-order logic with signature σ, and let P be a predicate symbol. Then:*
$Mod_\sigma(\mathcal{F}(\Gamma, P)) = \{w \in \mathcal{M}_\sigma \mid \exists w' \text{ such that } w' \models \Gamma \text{ and } w \equiv_P w'\}$.

This can be contrasted with Lin and Reiter's (1994) definition of forgetting a predicate symbol:

16. More accurately, those instances that would be positive (or negative) if the formula were expressed in CNF.

Definition 8 *Let Γ be a set of sentences in first-order logic over signature σ, and let P be a predicate symbol. A theory Γ' is a result of* forgetting P in Γ *iff for any model w: $w \models \Gamma'$ iff there is a model w' of Γ such that $w \cong_P w'$.*

In semantic terms, Lin and Reiter's definition refers to isomorphisms between models, whereas the present definition (relativised to models in first-order logic) is in terms of elementary equivalence. Consequently the Lin/Reiter definition is stronger than that presented here. As noted in Section 2, Zhang and Zhou (2010) explore properties of these approaches, which they call *weak* and *strong forgetting*: their definition for strong forgetting is that of Definition 8, while for weak forgetting their definition is in terms of four conditions that are shown to be the same as Definition 3 in the context of first-order logic.[17] We defer a fuller comparison between the Lin/Reiter style of forgetting and the present approach to Section 6, where other definitions of forgetting (in particular, in terms of second-order quantification) are also considered.

Forgetting a constant symbol turns out to be much as one would expect. In the following, to ease notation, a finite set of formulas appearing in a formula will stand for the conjunction of its elements. We obtain the following result:

Theorem 8 *Let $\mathcal{L}_\sigma$ be the language of first-order logic over signature σ, and let $\Gamma \subseteq \mathcal{L}_\sigma$ be a set of sentences. Let c be a constant, and let $\Gamma_c = \{\phi \in \Gamma \mid \phi \text{ mentions } c\}$. If Γ_c is finite, then for variable x not appearing in Γ we have:*

$$\mathcal{F}(\Gamma, c) \;\leftrightarrow\; (\Gamma \setminus \Gamma_c) \;\cup\; \{\exists x\,(\Gamma_c[c/x])\}\,.$$

That is, the result of forgetting a constant c from a set of formulas Γ is the set of those formulas in Γ that don't mention c, together with an existential generalization over the conjunction of formulas mentioning c, replacing c by a new variable.

Example 7 *Let $\phi = Student(john) \wedge Student(sue)$. Then*

$$\mathcal{F}(\phi, john) \;\leftrightarrow\; \exists x(Student(x) \wedge Student(sue))$$

which is equivalent to $Student(sue)$. For $\phi' = \phi \cup \{john \neq sue\}$ we have

$$\mathcal{F}(\phi', john) \;\leftrightarrow\; \exists x(Student(x) \wedge Student(sue) \wedge x \neq sue)$$

and so there is some student other than Sue.

Lin and Reiter (1994), along with others such as Vassos and Levesque (2013) and Rajaratnam et al. (2014), discuss "forgetting about a fact", which is to say, a ground atomic formula. Again, this is covered in detail in Section 6, but it is worth noting here that Definition 3 is inappropriate in this case. Consider defining the forgetting of a ground atomic formula g from Γ by taking the set of consequences Γ' of Γ that don't mention g. That is:

Unsuitable definition: $\quad \mathcal{F}'(\Gamma, g) \;=\; \mathcal{C}n_\sigma(\Gamma) \cap (\mathcal{L}_\sigma \setminus \{\phi \in \mathcal{L}_\sigma \mid \phi \text{ mentions } g\})$.

This definition, first, doesn't produce a deductively closed theory and, second, doesn't produce the correct result, as the following example illustrates.

17. These are the conditions given in Theorem 2, and presented originally by Zhang and Zhou (2009).

Example 8 *Let $g = Student(john)$ and assume that $\Gamma \vdash Student(john)$.*
Then for $\phi = \exists x(x = john \wedge Student(x))$ we have

$$\phi \in \mathcal{F}'(\Gamma, Student(john)).$$

That is, ϕ is a logical consequence of Γ that does not mention $Student(john)$. Moreover, clearly $\mathcal{F}'(\Gamma, Student(john)) \vdash Student(john)$.

So not only does the information $Student(john)$ remain implicitly in $\mathcal{F}'$, but $\mathcal{F}'$ is also not a (deductively-closed) theory.

The diagnosis is that $Student(john)$, or any well-formed language component beyond members of a signature, is not an appropriate object for forgetting, at least, not by the present approach. Instead, we suggest that often the appropriate belief change operation in this case is *contraction*; see Section 6 for details.

5.3 Relevance Logic: $\mathbf{E}_{fde}$

We next consider the relevance logic of *first degree entailment* $\mathbf{E}_{fde}$ (Anderson & Belnap Jr., 1975). $\mathbf{E}_{fde}$ is weaker than classical propositional logic, in that the so-called "paradoxes of implication" such as $(p \wedge \neg p) \rightarrow \phi$ and $\phi \rightarrow (p \vee \neg p)$ are missing. There are several reasons for looking at this logic: it shows how forgetting may be applied in a logic weaker than classical propositional logic, in particular in a logic with an impoverished notion of negation; and it provides an example where the standard Boole definition of forgetting is unsuitable. As well, $\mathbf{E}_{fde}$ has found application in AI with regards to limited reasoning.

Semantically, in $\mathbf{E}_{fde}$ a propositional formula can be true or false, as usual, but also neither true nor false, or both true and false. A formula of $\mathbf{E}_{fde}$ is of the form $\phi \rightarrow \psi$ where ϕ, ψ are propositional formulas formed using the standard connectives $\wedge$, $\vee$, $\neg$ (and not, notably, $\rightarrow$). Intuitively $\phi \rightarrow \psi$ has the meaning that the truth of ϕ entails the truth of ψ. $\mathbf{E}_{fde}$ has the following proof theory:

Axioms:

$\phi \wedge \psi \rightarrow \phi$	$\phi \wedge \psi \rightarrow \psi,$
$\phi \rightarrow \phi \vee \psi$	$\psi \rightarrow \phi \vee \psi$
$\phi \rightarrow \neg\neg\phi$	$\neg\neg\phi \rightarrow \phi$
$\phi \wedge (\psi \vee \mu) \rightarrow (\phi \wedge \psi) \vee \mu$	

Rules of Inference:

From $\phi \rightarrow \psi$ and $\psi \rightarrow \mu$ infer $\phi \rightarrow \mu$
From $\phi \rightarrow \psi$ and $\phi \rightarrow \mu$ infer $\phi \rightarrow \psi \wedge \mu$
From $\phi \rightarrow \mu$ and $\psi \rightarrow \mu$ infer $\phi \vee \psi \rightarrow \mu$
From $\phi \rightarrow \psi$ infer $\neg\psi \rightarrow \neg\phi$

If we write $\phi \leftrightarrow \psi$ for $\phi \rightarrow \psi$ and $\psi \rightarrow \phi$, then it can be shown that $\phi \leftrightarrow \phi'$ where ϕ' is ϕ in CNF. Moreover, for ϕ, ψ in CNF, determining whether $\phi \rightarrow \psi$ holds can be determined in quadratic time: The entailment holds just when for each clause (disjunction) c in ψ there is a clause in ϕ whose literals are among those in c.

With these facts, it is straightforward to define forgetting in $\mathbf{E}_{fde}$. For Γ a set of propositional formulas formed using the connectives $\wedge$, $\vee$, $\neg$, and for signature $\sigma \supseteq \Sigma(\Gamma)$, define $\mathcal{C}n_\sigma(\Gamma)$ by:

$$\mathcal{C}n_\sigma(\Gamma) = \{\phi \mid \Gamma \rightarrow \phi\}.$$

Then Definition 3 applies. We obtain the following:

Theorem 9 *Let* Γ *be a set of propositional formulas formed using* $\wedge$, $\vee$, $\neg$. *Then, in* $\mathbf{E}_{fde}$,

$$\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma_{CNF\downarrow p}$$

Thus the result of forgetting an atom p is just the set of clauses in Γ_{CNF} that don't mention p. Compare this with Theorem 6.2 for propositional logic, which has an additional term. So forgetting here is trivial, although of course converting to CNF may result in an exponential blowup.

5.4 Answer Set Programming

We turn next to the answer set programming (ASP) (Gelfond & Lifschitz, 1988; Baral, 2003) approach in logic programming. As described in Section 2, there has been substantial effort in addressing forgetting in ASP. Here we show that the general approach leads to a semantic definition of forgetting in the class of disjunctive logic programs.

Recall that a *general logic program* over a signature σ is a finite set of rules of the form

$$a_1; \ldots; a_m; \sim b_1; \ldots; \sim b_n \leftarrow c_1, \ldots, c_o, \sim d_1, \ldots, \sim d_p,$$

where $a_i, b_j, c_k, d_l \in \sigma$, where $m, n, o, p \geq 0$, and where $m + n + o + p > 0$. Recall also that a disjunctive logic program has $n = 0$ for all rules. While ASP is nonmonotonic with respect to the set of answer sets of a program, it has been put on a monotonic footing via a notion called *strong equivalence*[18] (Lifschitz et al., 2001; Turner, 2001) that provides an account of logical equivalence for programs. Subsequently, Wong (2008) has provided an inferential system for rules that preserves strong equivalence; and his notion of *SE consequence* is shown to be sound and complete with respect to the semantic notion of *SE models* (Turner, 2001). Wong's inference system is given as follows, where lower case letters are atoms, upper case letters are sets of atoms, and for a set of atoms $C = \{c_1, \ldots, c_n\}$, $\sim C$ stands for $\{\sim c_1, \ldots, \sim c_n\}$.

Inference Rules for SE Consequence:

Taut $x \leftarrow x$

Contra $\leftarrow x, \sim x$

Nonmin From $A \leftarrow B, \sim C$ infer $A; X \leftarrow B, Y, \sim C, \sim Z$

WGPPE From $A_1 \leftarrow B_1, x, \sim C_1$ and $A_2; x \leftarrow B_2, \sim C_2$
infer $A_1; A_2 \leftarrow B_1, B_2, \sim C_1, \sim C_2$

S-HYP

$$\begin{array}{lrcl} \text{From} & A_1 & \leftarrow & B_1, \sim x_1, \sim C_1, \\ & & & \ldots, \\ & A_n & \leftarrow & B_n, \sim x_n, \sim C_n, \\ & A & \leftarrow & x_1, \ldots, x_n \sim C \\ \text{infer} & A_1; \ldots; A_n & \leftarrow & B_1, \ldots, B_n, \sim C_1, \ldots, \sim C_n, \sim A, \sim C \end{array}$$

18. Two programs P and Q are strongly equivalent just if, for any program R, $P \cup R$ and $Q \cup R$ have the same set of answer sets.

Given this inference system, Definition 3 is applicable, and so we can provide a syntax-independent account of forgetting in disjunctive logic programs. Moreover, Theorem 6 suggests an appropriate strategy for computing *forget* in a logic program: We first define a notion $ResLP$ corresponding to Res for forgetting in propositional logic. In propositional logic, Res was used to compute all resolvents on an atom p. Here the same thing is done, except it is more intricate, since one has to consider instances of **WGPPE** and **S-HYP** in place of propositional resolution.

Definition 9 *Let P be a disjunctive logic program and $a \in \sigma$. Define*

$$
\begin{aligned}
&ResLP(P,a) = \\
&\quad \{r \mid \exists r_1, r_2 \in P \textit{ such that} \\
&\qquad r_1 = A_1 \leftarrow B_1, a, {\sim}C_1, \\
&\qquad r_2 = A_2; a \leftarrow B_2, {\sim}C_2, \\
&\qquad r = A_1; A_2 \leftarrow B_1, B_2, {\sim}C_1, {\sim}C_2 \ \} \\
&\quad \cup \\
&\quad \{r \mid \exists r_1, \ldots, r_n, r' \in P \textit{ such that } a = a_1, \\
&\qquad r_i = A_i \leftarrow B_i, {\sim}a_i, {\sim}C_i, \quad 1 \leq i \leq n, \\
&\qquad r' = A \leftarrow a_1, \ldots a_n, {\sim}C, \quad \textit{and} \\
&\qquad r = A_1; \ldots; A_n \leftarrow B_1, \ldots, B_n, {\sim}C_1, \ldots, {\sim}C_n, {\sim}A, {\sim}C \ \}
\end{aligned}
$$

We obtain the following:

Theorem 10 *Let P be a disjunctive logic program and $a \in \sigma$. Then:*
$\mathcal{F}(P,a) \ \leftrightarrow \ P_{\downarrow a} \ \cup \ ResLP(P,a)$.

The relation $\leftrightarrow$ is equivalence in the underlying inference relation, which is to say capturing strong equivalence. Since we inherit the results of Theorems 1 and 2, we get that the results of forgetting are independent of syntax, even though the expression on the right hand side of Theorem 10 is a set of rules obtained by transforming and selecting rules in P. Again, it can be observed that forgetting an atom results in at worst a quadratic blowup in the size of the program, while forgetting a set of atoms may result in an exponential blowup.

Example 9 *Let $P = \{p \leftarrow {\sim}q, \ r \leftarrow p\}$. Forgetting p yields $\{r \leftarrow {\sim}q\}$ (where $r \leftarrow {\sim}q$ is obtained by an application of* **WGPPE***), while forgetting q and r yield programs $\{r \leftarrow p\}$ and $\{p \leftarrow {\sim}q\}$ respectively.*

As outlined in Section 2, Wong (2007, 2009) presents an extensive analysis of forgetting in ASP based on strong and weak forgetting (Zhang & Foo, 2006). There are, for our purposes, two parts to this analysis. First, Wong presents what can be viewed as a "reverse engineering" of the current approach with respect to strong and weak forgetting. That is, strong and weak forgetting don't preserve strong equivalence, in that forgetting applied to two strongly-equivalent programs may yield programs that are not strongly equivalent. Wong asks what the underlying inference relation is, that is preserved by these approaches to forgetting. To this end, he develops a notion called *T-equivalence*. A syntactic and model-theoretic definition of T-equivalence is provided and the approaches of Zhang and Foo (2006) are shown to preserve T-equivalence. Thus the general definition of forgetting given here (Definition 3) doesn't apply to Zhang and Foo's (2006) approach with respect to strong equivalence, but it does with respect to T-equivalence.

Wong (2009) goes on to investigate these notions of forgetting in the context of SE semantics. A set of properties is first proposed by which the appropriateness of a definition can be judged. Then an abstract specification is given for forgetting an atom a from a program P: a logic program is first extended in conformity with a set of conditions to yield a program $C(P, a)$; and the resulting program is then appropriately pruned to produce the forgetting result. The strong version of this specification is named F_S. It is shown that F_S co-incides with $\mathcal{F}(P, a)$ in Theorem 10; see also Theorem 4.30 of Wong's (2009) thesis. Hence Theorem 10 can be seen as giving an alternative formulation of Wong's result for F_S.

5.5 Other Logics

Clearly the overall approach is applicable to other classes of logics. For example, Zhang and Zhou (2009) examine forgetting in the modal logic S5. Their definition is in terms of the four conditions mentioned in Theorem 2; as well a semantic characterisation is given in terms of bisimulations. As a consequence of Theorem 2, their approach to forgetting corresponds with the current approach in the context of S5. Arguably then, Definition 3 provides an independent justification of their approach. Moreover, there is no obstacle in principle to providing a syntactic definition (analogous, say, to Theorem 10) for forgetting in S5, for example by exploiting the inferential system RS5 of Enjalbert, Farinas, and del Cerro (1989). Analogously, the procedure by Herzig and Mengin (2008) for computing uniform interpolants in the modal logic K can be regarded as a syntactic analogue of the present approach in the context of this modal logic.

Further, the central definitions here are applicable to description logics in general, and in particular to those description logics (of which there seems to be many) which lack the uniform interpolation property. It can be noted however that work in description logics focuses on notions of (or closely related to) uniform interpolation, while the current approach focuses on a more general conception of forgetting. The present approach then may prove effective as a tool for analysis for forgetting in such logics.

5.6 Computational Considerations

It was suggested at the outset that the perspective provided by the general approach of Section 4 may in some cases lead to simpler formulations to computing forgetting. Our main definition (Definition 3) is expressed in terms of the consequence relation of a logic and not, as is often the case, in terms of a logic's model theory. In the preceding subsections, a common technique is used for computing forgetting in propositional logic (including propositional Horn logic), disjunctive logic programs, and, trivially, first-degree entailment: to determine the result of forgetting an atom p in a set of formulas Γ, Γ is "compiled" into a reduced normal form Γ' such that Γ' has the appropriate reduced signature and ϕ is in the result of forgetting just when $\Gamma' \vdash \phi$. In each case this is done by applying a cut-like rule to formulas in Γ that mention p to produce a formula that does not mention p. The resulting set of formulas, together with those formulas in Γ that don't mention p, constitute the result of forgetting Γ'. The proof that this works in each case is done by showing that a proof of $\Gamma \vdash \phi$ where ϕ doesn't mention p can be transformed to a proof that $\Gamma' \vdash \phi$.

This technique relies on two features of the underlying logic: the existence of appropriate "cut-like" rules (such as propositional resolution or **WGPPE**), together with the existence of an appropriate normal form for any set of formulas. In the case of propositional logic, this normal form would be clause form; in the case of disjunctive logic programs, the rules can be considered as already being in a normal form. Two points should be noted: First there is nothing formal about this suggested technique for computing forgetting; instead, it appears to be a (sometimes) useful means of computing forgetting when a logic has the right properties (and perhaps an indication of potential difficulties when it does not). Second, this is not a new approach. As previously mentioned, Herzig and Menguin (2008) employ modal resolution in this fashion to compute forgetting in the modal logic K.

6. Discussion

As noted earlier, the term *forgetting* has been used in the literature in different senses, with different meanings, and in different contexts. In the previous sections, a general approach was presented that characterises an arguably-appropriate, generally-applicable, notion of forgetting. In this section, other notions of forgetting are compared to the present one. Two approaches are examined in detail. The first, involving second-order quantification, provides a possible alternative definition to forgetting that is also broadly applicable, and that is no less strong than that developed here. The second approach is narrower, involving forgetting a fact (or ground atomic formula). In this case, it is suggested that the phenomenon under consideration may more appropriately be regarded as an instance of *belief contraction*.

6.1 Forgetting via Second-Order Quantification

A strength of the approach to forgetting developed here is its generality, in that it is applicable to any system that qualifies as a logic. This subsection briefly considers an alternative approach, based on second-order quantification, that has been used for forgetting in classical logic. As before, the goal is to reduce the signature of an agent's language, preserving where possible the agent's beliefs over the reduced signature. The intuitions behind second-order quantification are appealing: Assume that the goal is to forget the symbol ρ of sort s from a signature σ. Intuitively, ρ is intended to denote some specific object of sort s; for example in first-order logic, ρ could be a predicate symbol denoting some relation in the domain. Forgetting ρ in a formula then could be argued to yield the formula where, in place of ρ is a symbol intended to denote some indefinite object of sort s; that is, ρ is replaced by an existentially-quantified variable of the appropriate sort. For example, the result of forgetting predicate symbol $Student$ from

$$(Student(john) \vee Student(sue)) \wedge Teacher(joan)$$

would be

$$\exists X.(X(john) \vee X(sue)) \wedge Teacher(joan)$$

where X is a unary predicate variable. In general, forgetting symbol ρ from a formula ϕ can be defined by

$$forget(\phi, \rho) \ \doteq \ \exists X.\phi[\rho/X] \tag{10}$$

where X is a variable of the same sort and arity as ρ.

Except for straightforward cases such as forgetting a constant in first-order logic, this definition results in a formula not in the original language. Consequently, the major computational challenge is to re-express (10) in the original language, thereby eliminating the second-order quantifier. In some cases this is not a problem. For example, it is easily seen that in propositional logic, the above definition reduces to the standard Boole definition (and so also to the present definition localised to classical propositional logic). However, there are situations, notably in first-order logic, where the results of forgetting may not be first-order expressible; see for example the approaches of Lin and Reiter (1997) and Vassos and Levesque (2013). On the other hand, there are many situations in which the results of forgetting are first-order expressible; this topic has received significant attention – see for example the work of Gabbay and Ohlbach (1992), Gabbay, Schmidt, and Szalas (2008), and Wernhard (2004, 2008).

As noted earlier, forgetting via second-order quantification in many cases is strictly stronger than forgetting as given in Definition 3, since if the result of (10) is first-order expressible, then it also follows via Definition 3. So second-order quantification offers the possibility of an alternative, general, approach to forgetting. While a fully-general account of such forgetting would require as prerequisite a general specification of second-order quantification in arbitrary logics, nonetheless some points of comparison can be made between the present approach and forgetting via (10) in classical logic.

Both definitions have the advantage of being simple. The definition developed here is expressed within the formal system in question and so, as a consequence, it is easy to prove basic properties (e.g. Theorems 1 and 2). However, the result of forgetting is a belief set, and so the main computational challenge is to determine a uniform interpolant (where such exists). The definition in terms of second-order quantification (10) is also not immediately usable, in that one has to re-express the result (where possible) in the original language. However, in both cases one may still carry out query answering, to determine whether a given formula is in the result of forgetting. For the present approach, it is obvious that determining whether $\mathcal{F}(\Gamma, \rho) \vdash \phi$ holds is no harder than determining whether $\Gamma \vdash \phi$ holds. With second-order quantification, a query of the form $\{\exists X(\wedge\Gamma)\} \vdash \phi$ (where X is a predicate variable and ϕ a formula of first-order logic) can be replaced by one of the form $\{(\wedge\Gamma[X/P])\} \vdash \phi$ by substituting a fresh predicate symbol P for the variable X. (See for example the work of Craig (1957), and also of Wernhard (2016) for use of this principle in a theorem prover.)

A major conceptual difference between the two approaches is that for our approach to forgetting, there is no reference to semantics or model theory, but instead the definition is in terms of inference. To the extent that forgetting is a syntactic notion that deals with a reduction in an agent's signature, this is appropriate. Forgetting according to (10) on the other hand, appears to be more "semantic", in that the main definitions are typically expressed in terms of isomorphisms (or other model-theoretic constructs such as bisimulations) between models. For Definition 3, in contrast, one can re-express the main definition in model-theoretic terms (using the notion of elementary equivalence), but the central definition remains based on a notion of inference.

The above points don't conclusively establish one definition of forgetting over another, nor are they intended to. While a specific definition is advocated here, the choice of one definition over another will to some extent be a matter of taste and opinion. However, at least with regards to first-order logic, a possible resolution is to suggest that the full power of first-order logic (as given by Tarskian semantics) isn't required for reasoning, or at least for commonsense reasoning and for reasoning in AI. This point is developed by Levesque (1984) and Genesereth and Kao (2015), who have proposed variants of first-order logic with named, countable universes, and adopting a substitutional interpretation of quantifiers (Leblanc, 1984). In these approaches the notions of isomorphism between models and elementary equivalence coincide.[19] These authors also argue that their logics are suitable vehicles for general reasoning in AI. Thus, in adopting one of these approaches to first-order reasoning, the two approaches to forgetting compared here would coincide.

6.2 Forgetting and Belief Change

A variant of forgetting that has appeared in the literature is *forgetting a fact* or forgetting a ground atomic formula. This has been considered by, for example, Lin and Reiter (1994), Vassos and Levesque (2013), and Rajaratnam et al. (2014). In general, a "fact" can be an element of a signature, as in propositional logic, or a complex language element, as in first-order logic. In Section 5.2, it was noted that the present approach doesn't extend to such a conception of "forgetting" when a "fact" is not an element of a signature. In this subsection, we examine the notion of "forgetting a fact". The idea of removing a fact from a knowledge base is (of course) a coherent notion. However, we suggest that the operation of removing a fact is conceptually and formally quite different from forgetting, as construed here as decreasing a signature, as well as from related notions such as uniform interpolation or second-order quantification. Instead, we suggest that such work may be best regarded as a specific problem in *belief contraction*. From this we go on and further examine the relation between forgetting and belief change.

Recall that adapting our definition of forgetting, Definition 3, to accommodate ground atomic formulas produces unacceptable results. The result of such "forgetting" isn't a belief set, and the formula to be forgotten remains derivable. Thus, if a knowledge base contained the formula $Student(john)$, in the purported forgetting the formula $\exists x(x = john \land Student(x))$ would be in the result, and so $Student(john)$ would still be entailed.

We claim that this isn't a problem with the definition, but rather that $Student(john)$, or any compound language element for that matter, is not an appropriate object for forgetting. The reason for this is that $Student(john)$ is an assertion about a domain of application, and the removal of this fact entails the removal of a *proposition*. That is, in removing a fact, the propositional content of the agent's knowledge base decreases, while the agent's ability to express beliefs about the domain is unchanged. In contrast, in forgetting, the agent's signature decreases, and the agent loses the capability of expressing knowledge about a domain. So while the propositional content of the agent's beliefs also decreases, forgetting is defined solely with respect to elements of a language, with no necessary reference to a domain of application, and consequently independent of semantic notions.

It is instructive to consider the implications of attempting to extend the current approach to satisfactorily allow forgetting a fact. Consider again the example wherein attempting to forget the formula $Student(john)$ using Definition 3 resulted in $\exists x(x = john \land Student(x))$ still being in the agent's KB, and so $Student(john)$ was still entailed. The problem, of course, is that one doesn't want $Student(john)$ to be entailed. An obvious solution is to consider just those belief sets that don't entail $Student(john)$. However, one doesn't want just any such belief set, and an obvious desiderata is that such belief sets should contain as much as possible of the original belief set. Now, it is conceivable that there may be more than one such maximal belief set that doesn't entail $Student(John)$; thus some extra-logical means would be needed to select a preferred belief

19. Thus for example in Levesque's (1984) approach, for model w, the only model that is elementary equivalent to w is w itself.

set (or preferred belief sets, in which case one could take their intersection). Then, this procedure would presumably yield an acceptable belief set, in which $Student(john)$ is not entailed. However, the construction sketched above is, of course, the original construction for belief *contraction* in the AGM approach (Alchourrón et al., 1985; Gärdenfors, 1988).

These considerations indicate that, in removing a fact, the appropriate belief change operation is *contraction*,[20] or perhaps, in a dynamic domain, Katsuno and Mendelzon's (1992) *belief erasure*. Indeed this appears to be sometimes acknowledged, implicitly or explicitly, in the literature. For example, according to Lin and Reiter (1994), the result of "forgetting" $Student(john)$ from $\Gamma = \{\forall x\ Student(x)\}$ is $\{\forall x\ (x \neq john) \supset Student(x)\}$; this may be viewed as a contraction in which the *belief* that John is a student is no longer held. Hence, the Lin/Reiter construction may be regarded as a base case for defining contraction in first-order logic. In the same vein, Rajaratnam et al. (2014) consider "forgetting" a ground atomic formula, with intended application to progression in a theory of action. Their approach is compared to the belief change operation of contraction, and in fact is shown to satisfy the basic AGM contraction postulates. So, to the extent that the AGM contraction postulates characterise the class of contraction functions, it would seem that this work is best considered as dealing with contraction.[21]

6.2.1 FORGETTING AS BELIEF CHANGE

While forgetting is not usually regarded as a belief change operator in the sense of the AGM framework, given the preceding discussion it is worth briefly considering it in this light. The AGM approach studies how a rational agent may alter its beliefs in the presence of new information concerning some domain. The two primary belief change operations are belief *contraction*, in which an agent may reduce its stock of beliefs, and *revision*, in which new information is consistently incorporated into the agent's belief corpus. For incorporating information about a changing world, these operations have counterparts called *erasure* and *update* (Katsuno & Mendelzon, 1992) respectively. AGM belief change[22] is with respect to some domain of application, and so is fundamentally involved with the *interpretation* of the agent's language.[23] Thus for example, if you believed that a friend had a bicycle, but later learned that this was false, you might well also withdraw the belief that the friend had a bicycle tire repair kit. On the other hand, our contention is that forgetting solely involves the signature of the agent's language. Hence, in forgetting an atom p, say, the result will be a decrease in an agent's language, and so a decrease in the agent's ability to express knowledge about the domain.

Nonetheless, several papers have proposed using a forgetting operator (in the sense of reducing a language's signature) for defining an update operator via a definition analogous to the Levi identity; this includes work by Lang, Liberatore, and Marquis (2003), Nayak, Chen and Lin (2006), Zhang and Zhou (2009), and van Ditmarsch et al. (2009). In a narrowly-defined context this may work,

20. This is not to say of course that forgetting a fact is exactly the same as contracting by as fact, but rather that such "forgetting" can be reduced to contraction. Hence, "forgetting" a ground atomic formula ϕ from a belief set K can be expressed as $(K - \phi) - \neg\phi$ (where one of the contraction operations has no effect) or by a package contraction $K - \{\phi, \neg\phi\}$ (Hansson, 1999).

21. It should be clear that these points are not a criticism of the substantial contribution of these papers; rather the suggestion is that the contribution is in a related but different category.

22. For simplicity we will also include the Katsuno-Mendelzon approach to update under this term.

23. This is not to say that there aren't domain-independent approaches to belief change; for example Dalal (1988) presents an approach to revision based on a notion of distance between interpretations based in turn on differing truth values assigned to atoms.

for example, in using forgetting to implement symmetric erasure (Katsuno & Mendelzon, 1992) as a means to obtain an update operator for literals (Nayak et al., 2006). However, in a more general context it is problematic. The following remarks concern using a forgetting operator to implement update; they apply equally well to using forgetting to implement revision.

Assume that we have a propositional language over a signature σ. Then for knowledge base Γ one might define an update (or similarly, revision) operation in analogy to the Levi identity by:

$$Update(\Gamma, \phi) \;=\; forget(\Gamma, \Sigma(\phi)) + \phi. \tag{11}$$

Clearly, this *does* yield an update (or revision) operator, but an operator that in general is too drastic to be useful. To see this, consider a knowledge base which asserts (factually) that in British Columbia, hummingbirds arrive in the spring with the flowering of Salmonberry bushes. However, let's say that due to the effects of climate change, hummingbirds now arrive with the flowering of Blackcurrents. According to the above recipe (11), to update by this new information, one would first forget *all* knowledge involving one or both of hummingbirds or Salmonberrys. So in forgetting about hummingbirds one would lose the information about their size, colouring, etc. Or if one forgot about Salmonberrys then all information concerning these shrubs would be lost. Clearly this is much too extreme. If on the other hand the *assertion* that "hummingbirds arrive with the flowering of Salmonberrys" was to be removed, then one would want to lose this information, while keeping intact other knowledge regarding hummingbirds and Salmonberrys. That is, as argued above, this doesn't involve forgetting, but rather an *erasure* operation: the world has changed and the assertion is no longer true.

There is one instance where forgetting and contraction (or equally, erasure) may converge, and that is in dealing with propositional symbols. For example, Nayak, Chen, and Lin (2007) show that forgetting an atom in propositional logic satisfies the AGM postulates. Similarly, Lang, Liberatore, and Marquis (2003) address literal forgetting, a syntactic operation in which a literal, but not its negation, is "forgotten". Of interest is their definition in their Proposition 14:

$$Mod(ForgetLit(\Gamma, l)) \;=\; Mod(\Gamma) \cup \{Force(w, \bar{l}) \mid w \models \Gamma\} \tag{12}$$

$Force(w, l)$ results in a model the same as w, except that $Force(w, l) \models l$. So, via (12), one can "forget" that l is true without necessarily believing that l is false. $Force$ applied to Γ with respect to l would thus seem to have the effect of an update operator, in that via $Force$ one could assert on a model-by-model basis that the world has changed so that l is not true. In fact, Delgrande, Jin, and Pelletier (2008) take just this tack, having separately proposed the use of $Force$ and (12) as the base cases of an approach to compositional update and erasure operators, in which the update and erasure of arbitrary formulas are defined recursively in terms of these base cases. As a result, the results of Lang, Liberatore, and Marquis (2003) regarding literal forgetting may also be seen as covering a base case of belief change operators.

So, to conclude, forgetting can be considered as a belief change operator. However, it is not an operator like revision or contraction (or update and erasure). These operators fundamentally refer to a domain of application, and to the propositional content of an agent's beliefs. Moreover, given an agent's set of beliefs, the standard AGM or Katsuno-Mendelzon approaches don't specify a unique operator but a class of operators: postulates are given on the one hand for characterising the range of potential operators, while on the other hand specifications are provided for constructing the operators in this class. Forgetting is not like this in that, given a logic and a set of beliefs in a KB, forgetting elements of the signature of the KB is uniquely determined.

Forgetting is however similar in some respects to the operation of *expansion* in the AGM framework. The expansion of a belief set K by a formula ϕ is given by the deductive closure of K together with ϕ:

$$K + \phi \doteq \mathcal{C}n(K \cup \{\phi\}) \tag{13}$$

Similar to other belief change operators, expansion can be defined via a set of postulates, as is done by Gärdenfors (1988). However, since these postulates determine a unique outcome, it is simpler and more perspicuous to define expansion via (13), with the postulates then being derived properties. In the same way that forgetting seems similar to contraction, so too can expansion be seen as similar to revision. It has been noted above that forgetting is too drastic to be a useful contraction function. Similarly, expansion is too drastic to be a useful revision function; in particular, if $K \vdash \neg\phi$ then the expansion $K + \phi$ is inconsistent. Thus, while forgetting isn't a dual of expansion, the two operators are broadly analogous.

6.3 Relation to Other Concepts

As Section 2 makes clear, there is no universally-agreed on use of the term *forget* in the literature, and different approaches have used the term *forgetting* to address conceptually different phenomena. In this paper, forgetting refers to modification in an agent's knowledge resulting from a reduction in the signature of the language. Hence, a given logic and signature uniquely define a forget operator. This definition is independent of a domain of application, to the extent that forgetting doesn't involve losing information about the domain *per se* but rather involves losing the ability to express, or represent, information about a domain.

Other approaches have described forgetting as "becoming ignorant" or have based intuitions on the forgetting of knowledge or on the forgetting of the truth value of a proposition. Such approaches then deal with the relation between the assertions in an agent's knowledge base and the domain of application. Hence, as argued above, such approaches are best regarded as addressing some form of *contraction* in an agent's knowledge.

Forgetting has often been identified with uniform interpolation, particularly in the description logic community, and indeed there is a close relation. A uniform interpolant is a syntactic object, which may or may not exist for a given logic. Forgetting, on the other hand, is well-defined for any logic with a well-defined consequence relation. So even in a logic that lacks the uniform interpolation property, the question of whether a formula ϕ is entailed by the result of some forgetting is still well-defined. In the case that a uniform interpolant exists, it then would correspond to a (finite) representation of the result of forgetting.

7. Conclusion

This paper has presented an account of forgetting as an abstract belief change operator, independent of any particular underlying logic. The central thesis is that forgetting amounts to a reduction in the vocabulary, specifically the signature, of the language of a logic. The main definition is simple: the result of forgetting some elements of a signature in a knowledge base is simply the set of sentences derivable from the knowledge base that don't mention the forgotten elements of the signature. This definition is applicable to any logic with a well-defined consequence relation and, since the result is a logically closed set of formulas, the result of forgetting is independent of syntax. A key point is that, since forgetting is defined independently of a model theory, it has nothing to say about the interpretation of symbols, and so is independent of a domain of application.

The approach offers several advantages. Since it is expressed at a high level of abstraction, it is simple and intuitive, while at the same time its generality provides a uniform approach for studying forgetting in a logic-independent fashion. Moreover, the approach is readily shown to have various desirable properties, and these properties hold in all subsumed logics. Similarly, any result shown to hold in the general approach applies in any subsumed logic, and doesn't need to be re-proved for each logic or class of logics. We have also suggested that in some cases the main definition may lead to simpler approaches to computing forgetting in a specific logic (for example in propositional logic and in disjunctive logic programs); and it leads to insights with respect to individual logics and to the relation between approaches (for example, between propositional and Horn logic, and between variable and literal forgetting). Arguably also the relations between closely-related notions, such as uniform interpolation, literal elimination, and belief contraction, are clarified.

The definition excludes some approaches that have been called "forgetting" elsewhere, most notably the forgetting of a ground atomic formula. While we claim that the present approach captures a broad and interesting conception of forgetting, with various advantages and benefits, this not to say that there aren't alternative definitions; to this end an alternative approach based on second-order quantification was briefly compared to the present.

Much other work on forgetting, even when the goal is to forget an element of a signature, is based on the model theory of a specific logic. Such work is often phrased in terms of, for example, isomorphisms or bisimulations between models. While the present approach can also be phrased in terms of models (using elementary equivalence), it may be less expressive than those based on isomorphism or bisimulation between models. Nonetheless, arguably this potential difference in expressivity is not a drawback, for the following reasons. First, if one accepts the premiss that forgetting involves a decrease of the signature of an agent's language, then there is no need to appeal to the underlying model theory; in fact, a principle of parsimony suggests that if there is no need to refer to the model theory, then one ought not to refer to it. Second, from one point of view, the overall goal of Artificial Intelligence is to come up with intelligent reasoning agents, where reasoning is based on notions of deductive consequence. The present approach similarly is based on deductive consequence, which would then seem to be the appropriate level of granularity for such an operator. Last, in the specific case of first-order logic, it has been suggested that classical first-order logic under Tarskian semantics is overly expressive with respect to the needs of AI, and that a more curtailed semantics essentially loses nothing with respect to expressivity, while offering formal and perhaps computational advantages (Genesereth & Kao, 2015; Levesque, 1984). In this latter case, there is no difference with regards to expressivity.

Acknowledgements

I thank a reviewer for very helpful and insightful suggestions regarding the formalization in Section 3. I thank a second reviewer for helpful comments, particularly with regards to work in answer set programmming. Financial support from the Natural Sciences and Engineering Research Council of Canada is also gratefully acknowledged.

Appendix A. Proof of Theorems

A.1 Proof of Theorem 1

1. From Definition 3 we have that if $\mathcal{F}(\Gamma, \rho) \vdash \phi$ then $\Gamma \vdash \phi$, from which the result follows.

2. This also follows from Definition 3, along with the fact that $\vdash$ is a Tarskian consequence relation: If $\Gamma \vdash \Gamma'$, then $\mathcal{C}n_\sigma(\Gamma) \supseteq \mathcal{C}n_\sigma(\Gamma')$, and so $\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho} \supseteq \mathcal{C}n_\sigma(\Gamma') \cap \mathcal{L}_{\sigma\setminus\rho}$. Hence $\mathcal{F}(\Gamma, \rho) \supseteq \mathcal{F}(\Gamma', \rho)$ and so $\mathcal{F}(\Gamma, \rho) \vdash \mathcal{F}(\Gamma', \rho)$.

3. This is a result of the fact that we have an underlying Tarskian consequence relation.

   Given that $\mathcal{F}(\Gamma, \rho)$ has signature $\sigma \setminus \rho$, we get $\mathcal{F}(\Gamma, \rho) \subseteq \mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma, \rho))$ from Reflexivity.

   To show $\mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma, \rho)) \subseteq \mathcal{F}(\Gamma, \rho)$, let $\phi \in \mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma, \rho))$. Then $\mathcal{F}(\Gamma, \rho) \vdash \phi$ from the definition of $\mathcal{C}n(\cdot)$. From the definition of $\mathcal{F}$, this means that $\mathcal{C}n_\sigma(\Gamma) \vdash \phi$ and $\phi \in \mathcal{L}_{\sigma\setminus\rho}$. Then $\phi \in \mathcal{C}n_\sigma(\Gamma)$ by the definition of $\vdash$ and the fact that $\mathcal{C}n_\sigma(\Gamma) = \mathcal{C}n_\sigma(\mathcal{C}n_\sigma(\Gamma))$.

   Since we have $\phi \in \mathcal{C}n_\sigma(\Gamma)$ and $\phi \in \mathcal{L}_{\sigma\setminus\rho}$, we obtain $\phi \in \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho}$ and so $\phi \in \mathcal{F}(\Gamma, \rho)$, which was to be shown.

4. Assume $\rho' \subseteq \rho$.

$$\begin{array}{rcll} \mathcal{F}(\Gamma, \rho) & = & \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho} & \text{def. } \mathcal{F} \\ & = & \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho} \cap \mathcal{L}_{\sigma\setminus\rho'} & \text{set theory, since } \mathcal{L}_{\sigma\setminus\rho} \subseteq \mathcal{L}_{\sigma\setminus\rho'} \\ & = & (\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho'}) \cap \mathcal{L}_{\sigma\setminus\rho} & \text{reordering} \\ & = & \mathcal{F}(\Gamma, \rho') \cap \mathcal{L}_{\sigma\setminus\rho} & \text{def. } \mathcal{F} \\ & = & \mathcal{F}(\mathcal{F}(\Gamma, \rho'), \rho) & \text{def. } \mathcal{F} \end{array}$$

5.

$$\begin{array}{rcll} \mathcal{F}(\Gamma, \rho_1 \cup \rho_2) & = & \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus(\rho_1\cup\rho_2)} & \text{def. } \mathcal{F} \\ & = & \mathcal{C}n_\sigma(\Gamma) \cap (\mathcal{L}_{\sigma\setminus\rho_1} \cap \mathcal{L}_{\sigma\setminus\rho_2}) & \text{def. } \mathcal{L}_\sigma\text{, set theory} \\ & = & (\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho_1}) \cap (\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho_2}) & \text{set theory} \\ & = & \mathcal{F}(\Gamma, \rho_1) \cap \mathcal{F}(\Gamma, \rho_2) & \text{def. } \mathcal{F} \end{array}$$

6.

$$\begin{array}{rcll} \mathcal{F}(\Gamma, \rho_1 \cup \rho_2) & = & \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus(\rho_1\cup\rho_2)} & \text{def. } \mathcal{F} \\ & = & \mathcal{C}n_\sigma(\Gamma) \cap (\mathcal{L}_{\sigma\setminus\rho_1} \cap \mathcal{L}_{\sigma\setminus\rho_2}) & \text{def. } \mathcal{L}_\sigma\text{, set theory} \\ & = & (\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho_1}) \cap \mathcal{L}_{\sigma\setminus\rho_2} & \text{associativity} \\ & = & \mathcal{F}(\Gamma, \rho_1) \cap \mathcal{L}_{\sigma\setminus\rho_2} & \text{def. } \mathcal{F} \\ & = & \mathcal{F}(\mathcal{F}(\Gamma, \rho_1), \rho_2)) & \text{def. } \mathcal{F} \end{array}$$

7. According to the definition of a conservative extension, we need to show that $\mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma, \rho)) = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho}$.

   But from Part 3 we have that $\mathcal{C}n_{\sigma\setminus\rho}(\mathcal{F}(\Gamma, \rho)) = \mathcal{F}(\Gamma, \rho)$ and so, substituting in, we need to show that $\mathcal{F}(\Gamma, \rho) = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\rho}$. But this is just Definition 3.

   □

A.2 Proof of Theorem 2

For left to right, the definition of forget clearly satisfies Conditions 1-4. Briefly:

Assume that $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$.

For Condition 1, we have that $\mathcal{F}(\Gamma, \rho) \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$, and by assumption that $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$. This implies that $\Gamma' \leftrightarrow \mathcal{F}(\Gamma, \rho)$. Theorem 1.1 is $\Gamma \vdash \mathcal{F}(\Gamma, \rho)$ and this together with $\Gamma' \leftrightarrow \mathcal{F}(\Gamma, \rho)$ implies that $\Gamma \vdash \Gamma'$.

For Conditions 2 and 3, assume that $IR(\phi, \rho)$ and let ϕ' be such that $\phi' \in \mathcal{L}_{\sigma \setminus \rho}$ and $\phi \leftrightarrow \phi'$. We obtain:

$\Gamma \vdash \phi$ iff $\Gamma \vdash \phi'$.
$\Gamma \vdash \phi'$ iff $\mathcal{F}_O(\Gamma, \rho) \vdash \phi'$ since $\phi' \in \mathcal{L}_{\sigma \setminus \rho}$.
$\mathcal{F}_O(\Gamma, \rho) \vdash \phi'$ iff $\Gamma' \vdash \phi'$ from our assumption that $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$.
$\Gamma' \vdash \phi'$ iff $\Gamma' \vdash \phi$ since $\phi \leftrightarrow \phi'$.

Putting this all together we have that $\Gamma \vdash \phi$ iff $\Gamma' \vdash \phi$, which is just Condition 2 + 3.

For Condition 4, from the assumption that $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$, and since $\mathcal{F}_O(\Gamma, \rho) \leftrightarrow \mathcal{F}(\Gamma, \rho)$, we obtain $\Gamma' \leftrightarrow \mathcal{F}(\Gamma, \rho)$. Since $\mathcal{F}(\Gamma, \rho) \subseteq \mathcal{L}_{\sigma \setminus \rho}$ we obtain $IR(\Gamma', \rho)$.

For right to left, assume that Γ and Γ' satisfy the four given conditions. To show that $\Gamma' \leftrightarrow \mathcal{F}_O(\Gamma, \rho)$, we show the equivalent equality $\mathcal{C}n_\sigma(\Gamma') = \mathcal{F}_O(\Gamma, \rho)$.

$\subseteq$: We need to show that if $\phi \in \mathcal{C}n_\sigma(\Gamma')$, then $\phi \in \mathcal{F}_O(\Gamma, \rho)$.

Assume that $\phi \in \mathcal{C}n_\sigma(\Gamma')$ and so $\Gamma' \vdash \phi$. From Condition 4 we have $IR(\Gamma', \rho)$ from which it follows that $IR(\phi, \rho)$. Hence there is $\phi' \in \mathcal{L}_{\sigma \setminus \rho}$ such that $\phi \leftrightarrow \phi'$.

Since $\Gamma' \vdash \phi$ and $\phi \leftrightarrow \phi'$, so $\Gamma' \vdash \phi'$. This together with $IR(\phi', \rho)$ implies that $\Gamma \vdash \phi'$ via the contrapositive of Condition 3.

Therefore $\phi' \in \mathcal{C}n_\sigma(\Gamma)$ where $\phi' \in \mathcal{L}_{\sigma \setminus \rho}$, and so $\phi' \in \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho}$; that is, $\phi' \in \mathcal{F}(\Gamma, \rho)$. Thus $\phi \in \mathcal{C}n_\sigma(\mathcal{F}(\Gamma, \rho))$. Hence $\phi \in \mathcal{F}_O(\Gamma, \rho)$, which was to be shown.

$\supseteq$: Assume that $\phi \notin \mathcal{C}n_\sigma(\Gamma')$; we are to show that $\phi \notin \mathcal{F}_O(\Gamma, \rho)$.

Since $\phi \notin \mathcal{C}n_\sigma(\Gamma')$, we have $\Gamma' \nvdash \phi$.

There are two cases.

- $\Gamma \nvdash \phi$.
  In this case, by monotonicity, $\mathcal{F}_O(\Gamma, \rho) \nvdash \phi$. This implies $\phi \notin \mathcal{F}_O(\Gamma, \rho)$.
- $\Gamma \vdash \phi$.
  Since by assumption $\Gamma' \nvdash \phi$ but $\Gamma \vdash \phi$, by Condition 2 we obtain that not $IR(\phi, \rho)$.
  This means that $\nexists \phi'$ where $\phi \leftrightarrow \phi'$ and $\phi' \in \mathcal{L}_{\sigma \setminus \rho}$. That is $\nexists \phi'$ where $\phi \leftrightarrow \phi'$ and $\phi' \in \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho}$, or for every ϕ' where $\phi \leftrightarrow \phi'$ we have $\phi' \notin \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho}$.
  Consequently for every such ϕ', $\phi' \notin \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho}$, that is, $\phi' \notin \mathcal{F}(\Gamma, \rho)$, or $\phi \notin \mathcal{C}n_\sigma(\mathcal{F}(\Gamma, \rho))$, which is to say, $\phi \notin \mathcal{F}_O(\Gamma, \rho)$.

Thus in either case, we obtain $\phi \notin \mathcal{F}_O(\Gamma, \rho)$ as desired.

□

A.3 Proof of Theorem 3

Assume that $\mathcal{L}', \mathcal{C}n'(\cdot)$ is a sublogic of $\mathcal{L}, \mathcal{C}n(\cdot)$, and that $\Gamma' \subseteq \mathcal{L}'_{\sigma'}$, $\Gamma \subseteq \mathcal{L}_\sigma$, where $\sigma' \subseteq \sigma$.

From Definition 3 we have that $\mathcal{F}(\Gamma', \rho) = \mathcal{C}n_{\sigma'}(\Gamma') \cap \mathcal{L}_{\sigma' \setminus \rho}$.

We are given that, first, $\mathcal{C}n_{\sigma'}(\Gamma') \subseteq \mathcal{C}n_\sigma(\Gamma)$ and, second, that $\mathcal{L}'_{\sigma'} \subseteq \mathcal{L}_\sigma$.

Since $\mathcal{L}'_{\sigma'} \subseteq \mathcal{L}_\sigma$, so $\mathcal{L}'_{\sigma' \setminus \rho} \subseteq \mathcal{L}_{\sigma \setminus \rho}$.

Consequently, via set theory, $\mathcal{C}n_{\sigma'}(\Gamma') \cap \mathcal{L}'_{\sigma' \setminus \rho} \subseteq \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho}$.

From the definition of forgetting we get $\mathcal{F}(\Gamma', \rho) \subseteq \mathcal{F}(\Gamma, \rho)$.

$\Box$

A.4 Proof of Theorem 4

Lemma 1 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a logic and σ a signature. Let $\Gamma, \Gamma' \subseteq \mathcal{L}_\sigma$. If $Mod_\sigma(\Gamma) \subseteq Mod_\sigma(\Gamma')$, then $Mod_\sigma(\Gamma)_{|\sigma'} \subseteq Mod_\sigma(\Gamma')_{|\sigma'}$*

Proof of Lemma 1:

Assume that $Mod_\sigma(\Gamma) \subseteq Mod_\sigma(\Gamma')$ and let $w' \in Mod_\sigma(\Gamma)_{|\sigma'}$.

Then there is $w \in Mod_\sigma(\Gamma)$ where $w' = w_{|\sigma'}$.

By our given assumption this means that $w \in Mod_\sigma(\Gamma')$.

Hence $w' \in Mod_\sigma(\Gamma')_{|\sigma'}$ by the definition of a reduct.

$\Box$

Lemma 2 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a logic, σ a signature, and let $\Gamma \subseteq \mathcal{L}_\sigma$. Then:*
$\mathcal{C}n_\sigma(\Gamma) = t(Mod_\sigma(\Gamma))$.

Proof of Lemma 2:

Let $\psi \in \mathcal{L}_\sigma$.

We have that: $\psi \in \mathcal{C}n_\sigma(\Gamma)$ iff
$\Gamma \vdash \psi$ iff
$\Gamma \models \psi$ iff
$\psi \in \{\phi \mid Mod_\sigma(\Gamma) \subseteq Mod_\sigma(\phi)\}$ iff
$\psi \in \{\phi \mid \forall w \in Mod_\sigma(\Gamma), w \models \phi\}$ iff
$\psi \in t(Mod_\sigma(\Gamma))$.

Thus for any formula ψ we have $\psi \in \mathcal{C}n_\sigma(\Gamma)$ iff $\psi \in t(Mod_\sigma(\Gamma))$.

Hence $\mathcal{C}n_\sigma(\Gamma) = t(Mod_\sigma(\Gamma))$.

$\Box$

Lemma 3 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a logic and σ a signature. Let $\Gamma \subseteq \mathcal{L}_\sigma$ and $\sigma' \subseteq \sigma$. Then* $t(Mod_\sigma(\Gamma)_{|\sigma'}) = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'}$.

Proof of Lemma 3:

$LHS \subseteq RHS$:

We show for arbitrary ϕ that if $\phi \in LHS$ then $\phi \in RHS$.

Assume that $\phi \in t(Mod_\sigma(\Gamma)_{|\sigma'})$. From the definition of $t(\cdot)$ this implies that $\Sigma(\phi) \subseteq \sigma'$.

Also by the definition of $t(\cdot)$, for every $w' \in Mod_\sigma(\Gamma)_{|\sigma'}$, we have $w' \models \phi$.

Hence for every such w', there is $w \in Mod_\sigma(\Gamma)$ where $w' = w_{|\sigma'}$ and $w \models \phi$.

This means that for every $w \in Mod_\sigma(\Gamma)$, we have $w \models \phi$. (That is, if this were not the case and we had $w \in Mod_\sigma(\Gamma)$ but $w \not\models \phi$, then we would have $w_{|\sigma'} \in Mod_\sigma(\Gamma)_{|\sigma'}$ but $w_{|\sigma'} \not\models \phi$, by the reduct property, contradiction.)

Thus $\phi \in t(Mod_\sigma(\Gamma))$ by the definition of $t(\cdot)$. From Lemma 2 we have $t(Mod_\sigma(\Gamma)) = \mathcal{C}n_\sigma(\Gamma)$, and since $\phi \in t(Mod_\sigma(\Gamma))$ so $\phi \in \mathcal{C}n_\sigma(\Gamma)$.

Moreover, $\Sigma(\phi) \subseteq \sigma'$ and so $\phi \in \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'}$, as was to be shown.

$RHS \subseteq LHS$:

We show for arbitrary ϕ that if $\phi \notin LHS$, then $\phi \notin RHS$.

So assume for formula ϕ that $\phi \notin t(Mod_\sigma(\Gamma)_{|\sigma'})$.

If $\Sigma(\phi) \not\subseteq \sigma'$, then clearly $\phi \notin RHS$, and we're done.

Thus assume that $\Sigma(\phi) \subseteq \sigma'$. Since $\phi \in \mathcal{L}_{\sigma'}$, we need to show that $\phi \notin \mathcal{C}n_\sigma(\Gamma)$.

From the assumption that $\phi \notin t(Mod_\sigma(\Gamma)_{|\sigma'})$ we have by the definition of $t(\cdot)$ that there is $w' \in Mod_\sigma(\Gamma)_{|\sigma'}$ such that $w' \not\models \phi$.

Since $w' \in Mod_\sigma(\Gamma)_{|\sigma'}$ there is $w \in Mod_\sigma(\Gamma)$ where $w' = w_{|\sigma'}$. By the reduct property we also obtain that $w \not\models \phi$.

$w \in Mod_\sigma(\Gamma)$ gives $w \models \Gamma$, and since $w \not\models \phi$ we have that $\Gamma \not\models \phi$. Thus, by the assumed soundness of the logic, $\Gamma \not\vdash \phi$, which is to say, $\phi \notin \mathcal{C}n_\sigma(\Gamma)$, which was to be shown. $\Box$

Lemma 4 *Let $L = (\mathcal{L}, \mathcal{M}, \models)$ be a logic and σ a signature. Let $\Gamma' \subseteq \mathcal{L}_{\sigma'}$ where $\sigma' \subseteq \sigma$. Then $Mod_\sigma(\Gamma') = Mod_{\sigma'}(\Gamma')_{\uparrow\sigma}$*

Proof of Lemma 4:

For left to right, let $w \in Mod_\sigma(\Gamma')$, and let $w' = w_{|\sigma'}$. Then $w' \models \Gamma'$ by the reduct property and since $\Sigma(\Gamma') = \sigma'$. Thus $w' \in Mod_{\sigma'}(\Gamma')$. But this means that $w \in Mod_{\sigma'}(\Gamma')_{\uparrow\sigma}$ again using the reduct property.

For right to left, let $w \in Mod_{\sigma'}(\Gamma')_{\uparrow\sigma}$. This means that w has signature σ and $w \models \Gamma'$, and so $w \in Mod_\sigma(\Gamma')$.

$\Box$

Proof of Theorem 4:
We have that logic $L = (\mathcal{L}, \mathcal{M}, \models)$ is hereditarily elementary. Let σ be a signature.

1. Let $\Gamma \subseteq \mathcal{L}_\sigma$ and $\sigma' \subseteq \sigma$.

   By assumption L is hereditarily elementary; so let $\Gamma' \subseteq \Gamma$ be a set of formulas where $\Sigma(\Gamma') \subseteq \sigma'$ and $Mod_{\sigma'}(\Gamma') = Mod_\sigma(\Gamma)_{|\sigma'}$.

An instance of Lemma 2 is $\mathcal{C}n_{\sigma'}(\Gamma') = t(Mod_{\sigma'}(\Gamma'))$ which, substituting using the previous equality, gives $\mathcal{C}n_{\sigma'}(\Gamma') = t(Mod_\sigma(\Gamma)_{|\sigma'})$. (i)

An instance of Lemma 3 is $t(Mod_\sigma(\Gamma)_{|\sigma'}) = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'}$. (ii)

Then from (i) and (ii) we get that $\mathcal{C}n_{\sigma'}(\Gamma') = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'}$.

Equal sets of formulas have the same models, and so from the last equality we get

$Mod_{\sigma'}(\mathcal{C}n_{\sigma'}(\Gamma')) = Mod_{\sigma'}(\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'})$, or $Mod_{\sigma'}(\Gamma') = Mod_{\sigma'}(\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'})$. (iii)

Γ' was defined so that $Mod_{\sigma'}(\Gamma') = Mod_\sigma(\Gamma)_{|\sigma'}$; from this and (iii) we get $Mod_\sigma(\Gamma)_{|\sigma'} = Mod_{\sigma'}(\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma'})$.

From Definition 3, we obtain $Mod_\sigma(\Gamma)_{|\sigma'} = Mod_{\sigma'}(\mathcal{F}(\Gamma, \sigma \setminus \sigma'))$, which was to be shown.

2. From the previous part we have that $Mod_{\sigma'}(\mathcal{F}(\Gamma, \sigma \setminus \sigma')) = Mod_\sigma(\Gamma)_{|\sigma'}$.

   Taking the expansion by σ on each side gives $Mod_{\sigma'}(\mathcal{F}(\Gamma, \sigma \setminus \sigma'))_{\uparrow\sigma} = (Mod_\sigma(\Gamma)_{|\sigma'})_{\uparrow\sigma}$. (iv)

   An instance of Lemma 4 (substituting $\mathcal{F}(\Gamma, \sigma \setminus \sigma')$ for Γ' in the lemma) is

   $$Mod_\sigma(\mathcal{F}(\Gamma, \sigma \setminus \sigma')) = Mod_{\sigma'}(\mathcal{F}(\Gamma, \sigma \setminus \sigma'))_{\uparrow\sigma}. \quad \text{(v)}$$

   Consequently, from (iv) and (v) we get

   $$Mod_\sigma(\mathcal{F}(\Gamma, \sigma \setminus \sigma')) = (Mod_\sigma(\Gamma)_{|\sigma'})_{\uparrow\sigma}$$

   which was to be shown.

   □

A.5 Proof of Theorem 5

The result follows trivially if Γ is unsatisfiable; so assume that Γ is satisfiable.

$LHS \subseteq RHS$:

Let $w \in Mod_\sigma(\mathcal{F}(\Gamma, \rho))$. To reduce clutter below, let $\sigma' = \sigma \setminus \rho$, so $w \in Mod_\sigma(\mathcal{F}(\Gamma, \sigma \setminus \sigma'))$.

From Theorem 4.2 we get $w \in (Mod_\sigma(\Gamma)_{|\sigma'})_{\uparrow\sigma}$.

Thus there is w° where $w^\circ = w_{|\sigma'}$ and $w^\circ \in Mod_\sigma(\Gamma)_{|\sigma'}$.

From $w^\circ \in Mod_\sigma(\Gamma)_{|\sigma'}$ in turn we get that there is w' where $w^\circ = w'_{|\sigma'}$ and $w' \in Mod_\sigma(\Gamma)$, that is, $w' \models \Gamma$.

From $w^\circ = w_{|\sigma'}$ and $w^\circ = w'_{|\sigma'}$ we get that $w_{|\sigma'} = w'_{|\sigma'}$.

But this means that for every $\phi \in \mathcal{L}_{\sigma'}$, $w \models \phi$ iff $w' \models \phi$, that is $w \equiv_{\sigma \setminus \sigma'} w'$ or $w \equiv_\rho w'$.

Consequently, for any $w \in Mod_\sigma(\mathcal{F}(\Gamma, \rho))$, we have shown that there is a w' (viz. w' above) such that $w \equiv_\rho w'$ and $w' \models \Gamma$.

$RHS \subseteq LHS$:

Let $w \in \mathcal{M}_\sigma$ be such that for some $w' \in \mathcal{M}_\sigma$ we have $w' \models \Gamma$ and $w \equiv_\rho w'$.

That is, for every $\phi \in \mathcal{L}_{\sigma \setminus \rho}$ we have $w \models \phi$ iff $w' \models \phi$.

Hence, for every ϕ where $\Gamma \models \phi$ and $\phi \in \mathcal{L}_{\sigma \setminus \rho}$, since $w' \models \phi$ and $w \equiv_\rho w'$ we also have $w \models \phi$.

But this means that $w \in Mod_\sigma(\mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \rho})$ or $w \in Mod_\sigma(\mathcal{F}(\Gamma, \rho))$.

□

A.6 Proof of Theorem 6

The proof of the first part of the theorem makes reference to the second part, so we begin with the second part.

2. Observe that if Γ is tautologous, then it follows immediately that $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma$ and

   $Res(\Gamma_{CNF}, p) \leftrightarrow \Gamma$, from which our result is obtained.

   So assume that Γ is non-tautologous.

   Let LHS be $\mathcal{F}(\Gamma, p)$ and let RHS be $\Gamma_{CNF\downarrow p} \;\cup\; Res(\Gamma_{CNF}, p)$.

   LHS $\vdash$ RHS:

   Clearly, for every $\phi \in \Gamma_{CNF\downarrow p}$ or $\phi \in Res(\Gamma_{CNF}, p)$ we have that $\Gamma \vdash \phi$ and $\Sigma(\phi) \cap \{p\} = \emptyset$; consequently $\mathcal{F}(\Gamma, p) \vdash \phi$. Hence $LHS \vdash RHS$.

   RHS $\vdash$ LHS:

   We show for arbitrary ϕ that if $LHS \vdash \phi$ then $RHS \vdash \phi$. So assume that $\mathcal{F}(\Gamma, p) \vdash \phi$; we need to show that $RHS \vdash \phi$.

   We have by the definition of $\mathcal{F}$ that $\mathcal{F}(\Gamma, p) \vdash \phi$ iff $\Gamma \vdash \phi$ and $\phi \in \mathcal{L}_{\sigma \setminus \{p\}}$. $\Gamma \vdash \phi$ iff $\Gamma \cup \{\neg\phi\}$ is unsatisfiable, and $\Gamma \cup \{\neg\phi\}$ is unsatisfiable iff there is a resolution refutation of $\Gamma \cup \{\neg\phi\}$.

   Assume without loss of generality that Γ and $\neg\phi$ are in clause form, and let γ_1, $\dots$, γ_n be a refutation proof of $\Gamma \cup \{\neg\phi\}$, where γ_n is $\{\}$ and for each γ_l where $1 \leq l \leq n$:

   (a) γ_l is a clause of $\Gamma \cup \{\neg\phi\}$, or
   (b) γ_l is obtained from γ_j, γ_k where $j \neq k$ and $j, k < l$, by a resolution step.

   For simplicity, assume without loss of generality that no γ_l is a tautology.

   If no γ_l mentions p, then all clauses from Γ used in the proof are in $\Gamma_{\downarrow p}$ and so $RHS \vdash \phi$.

   Otherwise, some γ_l mentions p. In this case the proof is by reverse induction on the maximum index i of a proof step in $\gamma_1, \dots, \gamma_n$ such that

   (a) γ_i doesn't mention p, but
   (b) γ_i is obtained from γ_j, γ_k by a resolution step on p, and
   (c) $\gamma_j \notin \Gamma$ or $\gamma_k \notin \Gamma$.

As the base case, if there is no γ_i as above, then every clause that mentions p is in $Res(\Gamma, p)$ (since it must be that any such γ_i is obtained from γ_j and γ_k where $\gamma_j, \gamma_k \in \Gamma$), and we're done.

So let i be maximum index of a proof step such that (a)–(c) above hold.

Then the corresponding γ_i is obtained from formulas γ_j and γ_k by a resolution step involving p. Assume without loss of generality that $\gamma_j \not\in \Gamma$, and if $\gamma_k \not\in \Gamma$ then $j > k$.

At this point it is useful to introduce the notation, used here only, of

$$Resolve(\psi_1, \psi_2, p) \rightarrow \psi$$

to indicate that ψ is obtained from ψ_1 and ψ_2 via resolution on p. (The last argument to $Resolve$ isn't needed, but helps with respect to readability.)

So to this point we have that $Resolve(\gamma_j, \gamma_k, p) \rightarrow \gamma_i$ and (since $\gamma_j \not\in \Gamma$) for some atom x and clauses $\gamma_{j'}$, $\gamma_{j''}$ that $Resolve(\gamma_{j'}, \gamma_{j''}, x) \rightarrow \gamma_j$.

The proof can be transformed by "swapping" the two resolution steps. There are two cases to consider:

(a) The atom p occurs in exactly one of $\gamma_{j'}$, $\gamma_{j''}$, say, $\gamma_{j'}$.
The proof is transformed by replacing
$Resolve(\gamma_{j'}, \gamma_{j''}, x) \rightarrow \gamma_j$
$Resolve(\gamma_j, \gamma_k, p) \rightarrow \gamma_i$,
by
$Resolve(\gamma_{j'}, \gamma_k, p) \rightarrow \gamma_{k'}$,
$Resolve(\gamma_{k'}, \gamma_{j''}, x) \rightarrow \gamma_i$.

(b) The atom p occurs in both $\gamma_{j'}$ and $\gamma_{j''}$.
In this case replace
$Resolve(\gamma_{j'}, \gamma_{j''}, x) \rightarrow \gamma_j$
$Resolve(\gamma_j, \gamma_k, p) \rightarrow \gamma_i$,
by
$Resolve(\gamma_{j'}, \gamma_k, p) \rightarrow \gamma_{k'}$,
$Resolve(\gamma_{j''}, \gamma_k, p) \rightarrow \gamma_{k''}$,
$Resolve(\gamma_{k'}, \gamma_{k''}, x) \rightarrow \gamma_i$.

In either case, we obtain a modified resolution refutation of $\Gamma \cup \{\neg\phi\}$ where the maximum index satisfying conditions (a)–(c) above is strictly less than in the previous proof sequence.

Since a proof is finite, we conclude by induction that we can obtain a resolution refutation of $\Gamma \cup \{\neg\phi\}$ where premisses are in $\Gamma_{CNF\downarrow p}$ or $Res(\Gamma, p)$.

It then follows that $\Gamma_{CNF\downarrow p} \cup Res(\Gamma, p) \vdash \phi$.

1. Given a finite set of formulas Γ, we show that $\Gamma[p/\top] \vee \Gamma[p/\bot] \leftrightarrow \Gamma_{CNF\downarrow p} \cup Res(\Gamma_{CNF}, p)$. The desired result then follows immediately from the previous part.

   Assume that Γ is finite; and assume without loss of generality that Γ is in CNF and that no clause in Γ is a tautology. Define a three-fold partition of Γ by:

   - Γ^+_p is the set of clauses in Γ with a positive occurrence of p.

- Γ_p^- is the set of clauses in Γ with a negative occurrence of p.
- $\Gamma_{\downarrow p}$ is the set of clauses in Γ that don't mention p.

Recall that (set-based) clause notation may be freely mixed with logical connectives.

We obtain:

$$
\begin{array}{lll}
\Gamma[p/\bot] \vee \Gamma[p/\top] & & \\
\quad \equiv & (\Gamma_p^+ \cup \Gamma_p^- \cup \Gamma_{\downarrow p})[p/\bot] \;\vee\; (\Gamma_p^+ \cup \Gamma_p^- \cup \Gamma_{\downarrow p})[p/\top] & \text{Γ partition} \\
\quad \equiv & (\Gamma_p^+[p/\bot] \cup \Gamma_p^-[p/\bot] \cup \Gamma_{\downarrow p}[p/\bot]) \vee & \\
 & \qquad (\Gamma_p^+[p/\top] \cup \Gamma_p^-[p/\top] \cup \Gamma_{\downarrow p}[p/\top]) & \text{distr. substitution} \\
\quad \equiv & (\Gamma_p^+[p/\bot] \cup \{\top\} \cup \Gamma_{\downarrow p}) \;\vee\; (\{\top\} \cup \Gamma_p^-[p/\top] \cup \Gamma_{\downarrow p}) & \text{simplification} \\
\quad \equiv & (\Gamma_p^+[p/\bot] \cup \Gamma_{\downarrow p}) \;\vee\; (\Gamma_p^-[p/\top] \cup \Gamma_{\downarrow p}) & \text{more simplification} \\
\quad \equiv & \Gamma_{\downarrow p} \cup (\Gamma_p^+[p/\bot] \vee \Gamma_p^-[p/\top]) & \text{distributivity} \\
\quad \equiv & \Gamma_{\downarrow p} \cup \{c_1 \cup c_2 \mid c_1 \in \Gamma_p^+[p/\bot] \text{ and } c_2 \in \Gamma_p^-[p/\top]\} & \text{distributivity} \\
\quad \equiv & \Gamma_{\downarrow p} \cup Res(\Gamma, p) & \text{def. resolution}
\end{array}
$$

□

A.7 Proof of Theorem 7

Assume that Γ is finite, and assume without loss of generality that Γ is in CNF. We use a similar notation to that in Theorem 6.2, defining the following three-way partition of Γ:

- Γ_l^+ is the set of the clauses of Γ with a positive occurrence of l.
- Γ_l^- is the set of clauses of Γ with a complemented occurrence of l.
- $\Gamma_{\downarrow l}$ is set of the clauses of Γ that don't mention the atom corresponding to l.

Again (set-based) clause notation may be freely mixed with logical connectives.

In (Lang et al., 2003) it is noted that (9) is equivalent to

$$ForgetLit(\Gamma, l) \;=\; (\bar{l} \wedge \Gamma[l/\bot]) \vee \Gamma[l/\top].$$

We obtain:

$$
\begin{array}{lll}
(\bar{l} \wedge \Gamma[l/\bot]) \vee \Gamma[l/\top] & & \\
\quad \equiv & (\bar{l} \wedge (\Gamma_l^+ \cup \Gamma_l^- \cup \Gamma_{\downarrow l})[l/\bot]) \;\vee\; (\Gamma_l^+ \cup \Gamma_l^- \cup \Gamma_{\downarrow l})[l/\top] & \text{Γ partition} \\
\quad \equiv & (\bar{l} \wedge (\Gamma_l^+[l/\bot] \cup \Gamma_l^-[l/\bot] \cup \Gamma_{\downarrow l}[l/\bot])) \vee & \\
 & \qquad (\Gamma_l^+[l/\top] \cup \Gamma_l^-[l/\top] \cup \Gamma_{\downarrow l}[l/\top]) & \text{distr. substitution} \\
\quad \equiv & (\{\bar{l}\} \cup \Gamma_l^+[l/\bot] \cup \{\top\} \cup \Gamma_{\downarrow l}) \;\vee\; (\{\top\} \cup \Gamma_l^-[l/\top] \cup \Gamma_{\downarrow l}) & \text{simplification} \\
\quad \equiv & (\{\bar{l}\} \cup \Gamma_l^+[l/\bot] \cup \Gamma_{\downarrow l}) \;\vee\; (\Gamma_l^-[l/\top] \cup \Gamma_{\downarrow l}) & \text{simplification} \\
\quad \equiv & \Gamma_{\downarrow l} \cup ((\{\bar{l}\} \cup \Gamma_l^+[l/\bot]) \vee \Gamma_l^-[l/\top]) & \text{distributivity} \\
\quad \equiv & \Gamma_{\downarrow l} \cup \{\{\bar{l}\} \cup c \mid c \in \Gamma_l^-[l/\top]\} \cup & \\
 & \qquad \{c_1 \cup c_2 \mid c_1 \in \Gamma_l^+[l/\bot] \text{ and } c_2 \in \Gamma_l^-[l/\top]\} & \text{distributivity} \\
\quad \equiv & \Gamma_{\downarrow l} \cup \Gamma_l^- \cup Res(\Gamma, l) & \text{def. resolution} \\
\quad \equiv & \Gamma_{\Downarrow l} \cup Res(\Gamma, l) & \text{def. } \Gamma_{\Downarrow l}
\end{array}
$$

□

A.8 Proof of Theorem 8

Let LHS be $\mathcal{F}(\Gamma, c)$ and let RHS be $(\Gamma \setminus \Gamma_c) \cup \{\exists x\, (\Gamma_c[c/x])\}$.

LHS $\vdash$ RHS:

We have for every $\phi \in RHS$ that $\phi \in LHS$, from which our result is obtained. In a bit more detail:

- If $\phi \in \Gamma \setminus \Gamma_c$, then $\phi \in \mathcal{F}(\Gamma, c)$, since $\phi \in \Gamma$ and $\Sigma(\phi) \cap \{c\} = \emptyset$.
- If ϕ is $\exists x(\Gamma_c[c/x])$ then, since $\Gamma_c \in \mathcal{C}n_\sigma(\Gamma)$, so $\phi \in \mathcal{C}n_\sigma(\Gamma)$ via existential generalization. Moreover ϕ doesn't mention c, and so we obtain $\phi \in \mathcal{F}(\Gamma, c)$.

RHS $\vdash$ LHS:

We show that if $LHS \vdash \phi$ then $RHS \vdash \phi$.

Assume that $LHS \vdash \phi$, i.e., $\mathcal{F}(\Gamma, c) \vdash \phi$, or, via the definition of $\mathcal{F}$, $\Gamma \vdash \phi$ and $c \notin \Sigma(\phi)$.

By compactness there is a finite subset $\Gamma' \subseteq \Gamma$ such that $\Gamma' \vdash \phi$.

Γ' can be written as $\Gamma'_c \cup \Gamma'_{\bar{c}}$ where

- formulas in Γ'_c mention c
- formulas in $\Gamma'_{\bar{c}}$ don't mention c.

From $\Gamma'_c \cup \Gamma'_{\bar{c}} \vdash \phi$ and the fact that ϕ doesn't mention c, using existential instantiation we obtain

$$\exists x(\Gamma'_c[c/x]) \cup \Gamma'_{\bar{c}} \vdash \phi \qquad \text{(i)}$$

where x is a new variable not occurring in Γ' or ϕ.

It also follows that $RHS \vdash \exists x(\Gamma'_c[c/x]) \cup \Gamma'_{\bar{c}}$:

- First, $\exists x(\Gamma_c[c/x]) \in RHS$ and $\exists x(\Gamma_c[c/x]) \vdash \exists x(\Gamma'_c[c/x])$; thus $RHS \vdash \exists x(\Gamma'_c[c/x])$.
- Second, clearly $RHS \vdash \Gamma'_{\bar{c}}$

Consequently $RHS \vdash \exists x(\Gamma'_c[c/x]) \cup \Gamma'_{\bar{c}}$. This along with (i) yields $RHS \vdash \phi$, as desired.

$\square$

A.9 Proof of Theorem 9

By definition, $\mathcal{F}(\Gamma, p) = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma\setminus\{p\}}$. The right hand side is equal to

$$\{\phi \mid \Gamma \rightarrow \phi \text{ and } \phi \in \mathcal{L}_{\sigma\setminus\{p\}}\} \qquad \text{(i)}$$

Since for any formula or set of formulas we have $\Gamma \leftrightarrow \Gamma_{CNF}$, we can assume without loss of generality that Γ and ϕ are in CNF in clause form. Consequently, in $\mathbf{E}_{fde}$ (i) is the same as

$$\{\phi \mid \forall c \in \phi, \exists c' \in \Gamma \text{ such that } c' \subseteq c \text{ where } \Sigma(c) \subseteq \sigma \setminus \{p\}\}.$$

This is equivalent to $\{\phi \mid \Gamma_{CNF\downarrow p} \rightarrow \phi\}$ which is just $\mathcal{C}n_\sigma(\Gamma_{CNF\downarrow p})$.

Consequently we obtain that $\mathcal{F}(\Gamma, p) = \mathcal{C}n_\sigma(\Gamma_{CNF\downarrow p})$, i.e. that $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma_{CNF\downarrow p}$.

$\square$

A.10 Proof Sketch of Theorem 10

$\mathcal{F}(P, a)$ is defined to be the set of SE consequences of program P, where every such consequence does not mention atom a. The proof involves showing that if $P \vdash \phi$, where $\vdash$ is SE consequence and ϕ is a rule not mentioning a, then there is a proof of ϕ from $P_{\downarrow a} \cup ResLP(P, a)$ that does not mention a. (The other direction is immediate.)

The proof proceeds by induction, similar to the proof of Theorem 6.2: Assume there is a proof of ϕ from P, represented as a sequence of rules. Since ϕ does not mention a, there is a last rule in the proof, γ, that does not mention a, but is obtained from rules that do mention a; let δ be the last rule in the sequence that does so. Then γ is obtained from δ (and other rules) by instances of **WGPPE** or **S-HYP**. There are several cases, but in each case it can be shown that the proof can be transformed to one where either all rules in the proof mentioning a belong to $ResLP(P, a)$, or else the index of the last rule mentioning a is decreased and the index of no rule mentioning a is increased.

In the second case above, the process must terminate, since a proof is a finite structure. Moreover all premisses will be either rules of P that do not mention a, elements of $ResLP(P, a)$, or tautologies. $\Box$

References

Alchourrón, C., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, *50*(2), 510–530.

Anderson, A., & Belnap Jr., N. (1975). *Entailment: The Logic of Relevance and Necessity, Vol. I*. Princeton University Press, Princeton.

Baader, F., Calvanese, D., McGuiness, D., Nardi, D., & Patel-Schneider, P. (Eds.). (2007). *The Description Logic Handbook* (second edition). Cambridge University Press, Cambridge.

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge.

Baral, C., & Zhang, Y. (2005). Knowledge updates: Semantics and complexity issues. *Artificial Intelligence*, *164*(12), 209 – 243.

Barwise, J., & Feferman, S. (Eds.). (1985). *Model-Theoretic Logics: Background and Aims*. Springer, New York.

Blackburn, P., de Rijke, M., & Venema, Y. (2001). *Modal Logic*, Vol. 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK.

Boole, G. (1854). *An Investigation of the Laws of Thought*. Walton, London. (Reprinted by Dover Publications, 1954).

Chang, C. C., & Keisler, H. J. (2012). *Model Theory* (third edition). Dover Publications.

Chellas, B. (1980). *Modal Logic*. Cambridge University Press, Cambridge.

Craig, W. (1957). Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, *22*(3), 269–285.

Dalal, M. (1988). Investigations into theory of knowledge base revision.. In *Proceedings of the (AAAI) Conference on Artificial Intelligence*, pp. 449–479, St. Paul, Minnesota.

Delgrande, J., & Wang, K. (2015). A syntax-independent approach to forgetting in disjunctive logic programs. In *Proceedings of the (AAAI) Conference on Artificial Intelligence*, pp. 1482–1488.

Delgrande, J. (2014). Toward a knowledge level analysis of forgetting. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, pp. 606–609, Vienna, Austria.

Delgrande, J. P., Jin, Y., & Pelletier, F. J. (2008). Compositional belief update. *Journal of Artificial Intelligence Research*, *32*, 757–791.

Delgrande, J. P., Peppas, P., & Woltran, S. (2016). Generalised belief revision. Tech. rep. DBAI-TA-2016-100, TU Vienna. Submitted for publication.

Delgrande, J., & Wassermann, R. (2013). Horn clause contraction functions. *Journal of Artificial Intelligence Research*, *48*, 475–511.

Doets, K. (1996). *Basic Model Theory*. CSLI Publications.

Eiter, T., Ianni, G., Schindlauer, R., Tompits, H., & Wang, K. (2006). Forgetting in managing rules and ontologies. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006)*, pp. 411–419.

Eiter, T., Tompits, H., & Woltran, S. (2005). On solution correspondences in answer set programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 97–102.

Eiter, T., & Wang, K. (2006). Forgetting and conflict resolving in disjunctive logic programming. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pp. 238–243. AAAI Press.

Eiter, T., & Wang, K. (2008). Forgetting in answer set programming. *Artificial Intelligence*, *172*(14), 1644–1672.

Enderton, H. (1972). *A Mathematical Introduction to Logic*. Academic Press.

Enjalbert, P., & Fariñas del Cerro, L. (1989). Modal resolution in clausal form. *Theoretical Computer Science*, *65*(1), 1–33.

Erdem, E., & Ferraris, P. (2007). Forgetting actions in domain descriptions. In *Proceedings of the (AAAI) Conference on Artificial Intelligence*, Vancouver, BC.

Gabbay, D. M., Schmidt, R. A., & Szałas, A. (2008). *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*, Vol. 12 of *Studies in Logic: Mathematical Logic and Foundations*. College Publications.

Gabbay, D. M., & Ohlbach, H. J. (1992). Quantifier elimination in second–order predicate logic. In Nebel, B., Rich, C., & Swartout, W. (Eds.), *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning*, pp. 425–435. Morgan Kaufmann.

Gabbay, D. M., Pearce, D., & Valverde, A. (2011). Interpolable formulas in equilibrium logic and answer set programming. *J. Artif. Intell. Res. (JAIR)*, *42*, 917–943.

García-Matos, M., & Väänänen, J. (2005). Abstract model theory as a framework for universal logic. In Beziau, J.-Y. (Ed.), *Logica Universalis*, pp. 19–33. Birkhuser Basel.

Gärdenfors, P. (1988). *Knowledge in Flux: Modelling the Dynamics of Epistemic States*. The MIT Press, Cambridge, MA.

Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2012). *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In Kowalski, R., & Bowen, K. (Eds.), *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pp. 1070–1080. The MIT Press.

Gelfond, M., & Lifschitz, V. (1998). Action languages. *Electronic Transactions on AI*, *3*.

Genesereth, M., & Kao, E. (2015). The Herbrand manifesto. In Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., & Roman, D. (Eds.), *Rule Technologies: Foundations, Tools, and Applications*, Vol. 9202 of *Lecture Notes in Computer Science*, pp. 3–12. Springer International Publishing.

Ghilardi, S. (1995). An algebraic theory of normal forms. In *Annals of Pure and Applied Logic*, Vol. 71, pp. 189–245.

Ghilardi, S., Lutz, C., & Wolter, F. (2006). Did I damage my ontology? A case for conservative extensions in description logics. In Doherty, P., Mylopoulos, J., & Welty, C. A. (Eds.), *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning*, Lake District, UK. AAAI Press.

Ghilardi, S., & Zawadowski, M. (1995). Undefinability of propositional quantifiers in the modal system S4. *Studia Logica*, *55*, 259–271.

Goncalves, R., Knorr, M., & Leite, J. (2016a). The ultimate guide to forgetting in answer set programming. In Baral, C., Delgrande, J., & Wolter, F. (Eds.), *Proceedings of the Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, pp. 135–144, Cape Town, South Africa. AAAI Press.

Goncalves, R., Knorr, M., & Leite, J. (2016b). You can't always forget what you want: On the limits of forgetting in answer set programming. In *Proceedings of the European Conference on Artificial Intelligence*, pp. 957–965. IOS Press.

Hansson, S. O. (1999). *A Textbook of Belief Dynamics*. Applied Logic Series. Kluwer Academic Publishers.

Herzig, A., & Mengin, J. (2008). Uniform interpolation by resolution in modal logic. In Hölldobler, S., Lutz, C., & Wansing, H. (Eds.), *Logics in Artificial Intelligence, 11th European Conference (JELIA 2008)*, Vol. 5293 of *Lecture Notes in Computer Science*, pp. 219–231. Springer.

Hodges, W. (1997). *A Shorter Model Theory*. Cambridge University Press, Cambridge, UK.

Hughes, G., & Cresswell, M. (1996). *A New Introduction to Modal Logic*. Routledge., London and New York.

Israel, D. J. (1993). The role(s) of logic in artificial intelligence. In Gabbay, D. M., Hogger, C. J., & Robinson, J. A. (Eds.), *Nonmonotonic Reasoning and Uncertain Reasoning*, Vol. 1 of *Handbook of Logic in Artifical Intelligence and Logic Programming*, pp. 1–29. Oxford.

Katsuno, H., & Mendelzon, A. (1992). On the difference between updating a knowledge base and revising it. In Gärdenfors, P. (Ed.), *Belief Revision*, pp. 183–203, Cambridge. Cambridge University Press.

Konev, B., Walther, D., & Wolter, F. (2009a). Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$.. In *Description Logics'09*.

Konev, B., Walther, D., & Wolter, F. (2009b). Forgetting and uniform interpolation in large-scale description logic terminologies.. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 830–835.

Kontchakov, R., Wolter, F., & Zakharyaschev, M. (2010). Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artificial Intelligence*, *174*(15), 1093 – 1141.

Koopmann, P., & Schmidt, R. A. (2015). Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 175–181. AAAI Press.

Lang, J., Liberatore, P., & Marquis, P. (2003). Propositional independence : Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, *18*, 391–443.

Lang, J., & Marquis, P. (2010). Reasoning under inconsistency: A forgetting-based approach. *Artificial Intelligence*, *174*(1213), 799 – 823.

Leblanc, H. (1984). Alternatives to first-order semantics. In Gabbay, D., & Guenthner, F. (Eds.), *Handbook of Philosophical Logic*. D. Reidel Pub. Co.

Levesque, H. (1984). Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, *23*, 155–212.

Lifschitz, V., Pearce, D., & Valverde, A. (2001). Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, *2*(4), 526–541.

Lin, F., & Reiter, R. (1994). Forget it!. In *AAAI Fall Symposium on Relevance*, New Orleans.

Lin, F., & Reiter, R. (1997). How to progress a database. *Artificial Intelligence*, *92*(1-2), 131–167.

Ludwig, M., & Konev, B. (2014). Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, Vienna, Austria.

Lutz, C., & Wolter, F. (2011). Foundations for uniform interpolation and forgetting in expressive description logics. In Walsh, T. (Ed.), *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 989–995. IJCAI/AAAI.

Mendelson, E. (2015). *Introduction to Mathematical Logic* (6th edition). CRC Press.

Nayak, A., Chen, Y., & Lin, F. (2006). Forgetting and knowledge update. In Sattar, A., & Kang, B. (Eds.), *Proceedings of the Nineteenth Australian Joint Conference of Artificial Intelligence (AI-06)*, Vol. 4304 of *Lecture Notes in Artificial Intelligence*, pp. 131–140. Springer Verlag.

Nayak, A., Chen, Y., & Lin, F. (2007). Forgetting and update – an exploration. In Bonanno, G., Delgrande, J., Lang, J., & Rott, H. (Eds.), *Formal Models of Belief Change in Rational Agents*, No. 07351 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Newell, A. (1981). The knowledge level. *AI Magazine*, *2*(2), 1–20.

Nikitina, N., & Rudolph, S. (2014). (Non-)succinctness of uniform interpolants of general terminologies in the description logic. *Artificial Intelligence*, *215*, 120 – 140.

Rajaratnam, D., Levesque, H. J., Pagnucco, M., & Thielscher, M. (2014). Forgetting in action. In Baral, C., & do Giacomo, G. (Eds.), *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*. AAAI Press.

Truszczynski, M. (2010). Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence*, *174*(16-17), 1285–1306.

Turner, H. (2001). Strong equivalence for logic programs and default theories (made easy). In *LPNMR '01: Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 81–92. Springer-Verlag.

van Ditmarsch, H. P., Herzig, A., Lang, J., & Marquis, P. (2009). Introspective forgetting. *Synthese*, *169*(2), 405–423.

Vassos, S., & Levesque, H. J. (2013). How to progress a database III. *Artificial Intelligence*, *195*, 203–221.

Wang, K., Sattar, A., & Su, K. (2005). A theory of forgetting in logic programming. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pp. 682–688. AAAI Press.

Wang, K., Wang, Z., Topor, R., Pan, J., & Antoniou, G. (2009). Concept and role forgetting in $\mathcal{ALC}$ ontologies. In Bernstein, A., Karger, D., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., & Thirunarayan, K. (Eds.), *Proceedings of ISWC*, Vol. 5823 of *Lecture Notes in Computer Science*, pp. 666–681. Springer.

Wang, Y., Wang, K., & Zhang, M. (2013). Forgetting for answer set programming revisited. In *Proceedings, The 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1162–1168.

Wang, Y., Zhang, Y., Zhou, Y., & Zhang, M. (2012). Forgetting in logic programs under strong equivalence. In *Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Wang, Y., Zhang, Y., Zhou, Y., & Zhang, M. (2014). Knowledge forgetting in answer set programming. *Journal of Artificial Intelligence Research*, *50*, 31–70.

Wang, Z., Wang, K., Topor, R. W., & Pan, J. Z. (2010a). Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence*, *58*(1-2), 117–151.

Wang, Z., Wang, K., Topor, R. W., & Zhang, X. (2010b). Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *ECAI'10*, pp. 47–52.

Wernhard, C. (2004). Semantic knowledge partitioning. In Alferes, J. J., & Leite, J. A. (Eds.), *Logics in Artificial Intelligence, 9th European Conference, JELIA*, Vol. 3229 of *Lecture Notes in Artificial Intelligence*, pp. 552–564. Springer.

Wernhard, C. (2008). Literal projection for first-order logic. In Hölldobler, S., Lutz, C., & Wansing, H. (Eds.), *Logics in Artificial Intelligence: 11th European Conference (JELIA 08)*, Vol. 5293 of *Lecture Notes in Artificial Intelligence*, pp. 389–402. Springer.

Wernhard, C. (2009). *Automated Deduction for Projection Elimination*. No. 324 in Dissertations in Artificial Intelligence. AKA Verlag/IOS Press, Heidelberg, Amsterdam.

Wernhard, C. (2016). The PIE environment for first-order-based proving, interpolating and eliminating. In Fontaine, P., Schulz, S., & Urban, J. (Eds.), *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning*, Vol. 1635 of *CEUR Workshop Proceedings*, pp. 125–138.

Winslett, M. (1988). Reasoning about action using a possible models approach. In *Proceedings of the (AAAI) Conference on Artificial Intelligence*, pp. 89–93, St. Paul, Minnesota.

Wong, K.-S. (2007). A stronger notion of equivalence for logic programs. In Dahl, V., & Niemelä, I. (Eds.), *23rd International Conference on Logic Programming (ICLP 2007)*, Vol. 4670 of *Lecture Notes in Computer Science*, pp. 453–454. Springer.

Wong, K.-S. (2008). Sound and complete inference rules for SE-consequence. *Journal of Artificial Intelligence Research*, *31*(1), 205–216.

Wong, K.-S. (2009). *Forgetting in Logic Programs*. Ph.D. thesis, School of Computer Science and Engineering, University of New South Wales.

Zhang, Y., & Foo, N. (2006). Solving logic program conflict through strong and weak forgetting. *Artificial Intelligence*, *170*, 739–778.

Zhang, Y., & Zhou, Y. (2009). Knowledge forgetting: Properties and applications. *Artificial Intelligence*, *173*(16-17), 1525–1537.

Zhang, Y., & Zhou, Y. (2010). Forgetting revisited. In Lin, F., & Sattler, U. (Eds.), *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, pp. 602–604, Toronto. AAAI Press.

Zhou, Y., & Zhang, Y. (2011). Bounded forgetting. In Burgard, W., & Roth, D. (Eds.), *Proceedings of the (AAAI) Conference on Artificial Intelligence*, San Francisco, CA. AAAI Press.