9 Learning and First-Order Relational Reasoning

The chapter introduces the basic constructs for the relational representation of formulas in Section 9.1, and subsequently discusses ways in which, and the extent to which, the learning process can be extended to handle these richer representations. We first consider representations for relational reasoning, first by a simple reduction to propositional reasoning in Section 9.2, then via the "grounding trick" in Section 9.3, and finally by an extension of resolution to relational representations in Section 9.5. Unfortunately, we'll observe that we generally can't reason with function symbols in Section 9.4, and then in Section 9.6 that learning with relational representations is intractable, even for very simple domains and representations. But once again, as in Section 8.3, we show in Section 9.7 that we can use implicit learning to circumvent the intractability of finding consistent representations and solve an integrated relational learning and reasoning task.

9.1 Relations, variables, domains, and quantifiers.

So far, we have limited our discussion to a setting in which the world is described using propositional attributes. This formulation is conceptually simple and moreover, it turns out, permitted us to introduce most of the known tractable algorithmic techniques for reasoning. The use of such a simplistic formulation is, unfortunately, not entirely satisfactory for most purposes. Consider for example, our encoding of a planning problem as logical statements in Example 3.3. For *every* step of the plan $t = 1, \ldots, T$ and *every* location the robot could be in, we had separate variables and separate axioms characterizing the effects of the robot's possible actions for that particular location at that particular time. Given a generic set of axioms, we instantiated copies of them for the various time points and sets of locations (as appropriate), so at first this may seem to be merely a small nuisance. This simple approach poses some more fundamental obstacles, however:

1. *Scale:* increasing the number of attributes in this way may lead to an exceptionally large domain, where the computational complexity of our methods for reasoning usually scale polynomially with the number of attributes. Moreover, if the size of the domain is not known in advance, we may not be able to fix an adequate set of attributes up front.

2. *Generalization—or lack thereof:* in the previous part of the book, we suggested that it may be desirable to learn at least some knowledge about our environment from examples. If this knowledge only applies to a particular instantiation at a particular point in time, however, its utility is limited. Indeed, the propositional formulation forbids generalizing from our experiences in one location at one time to rules that apply more broadly. The amount of experience needed to learn a realistic domain is then prohibitively large. (Imagine if every door we encountered posed a novel challenge we had to overcome by trial-and-error.)

First-order logic addresses these challenges by separating the description of the world into two parts, *objects* and *predicates*, where we may write statements using the predicates that range over *all possible* objects. In this way, we obtain a much more compact representation. As usual, this expressive economy raises the danger of rendering the associated computational tasks intractable, and we must make a judicious choice of representation. Nevertheless, in many cases, a first-order representation is essential.

Formally, we will consider representations that instead of simply propositional attributes, now more generally have *atomic formulas* that consist of a *predicate* symbol with all of its *arguments* bound to *terms*; that is the predicate symbol has an associated *arity* a, and there are *constants* or *variables* that may be used as the arguments. It will often be useful to suppose that these arguments, as well as the constants and variables, have associated *types*, where the arguments can only be bound to a constant or variable of the indicated type. Later, we will also consider terms that use *functions*, which also have an associated arity a, and yield another term when its arguments are respectively bound to a terms, which again we may require to be of appropriate types. (If there are no functions, we will say that the logic is *function-free*.) The special case of a predicate of arity 0 (no arguments) is simply a propositional attribute, of the kind we have been considering up until now. Our possible worlds now correspond to truth values for these more general atomic formulas, i.e., truth values of all of the predicates when the arguments are bound to each possible combination of objects. More precisely, an *interpretation* of the logical language consists of a *domain* for the objects, given by a set, and an *interpretation*: a mapping that assigns

- each constant symbol an element of the domain (of the appropriate type)
- each function symbol a function that takes an appropriate tuple of domain elements to another domain element (of the appropriate type)
- each predicate symbol a relation of the appropriate type on the domain.

Example 9.1: Our domain with the box-moving robot is most naturally represented when the objects under consideration include the robot, the box, the three locations, and the points in time, where we would usually view these as three or four distinct types (e.g., perhaps viewing the robot and box as the same type of object, and otherwise viewing the rest as distinct). Then we could have a ternary predicate AT, where we take $AT(o, l, t)$ as representing that the object (box or robot) o is at location

l at time t, where these might be specific constants, or we might view them as variables in some larger formula. If the robot and box are considered to be the same type of object, we might want unary predicates $ROBOT$ and BOX to distinguish them, e.g., $ROBOT(o)$ is true if o is the robot and false otherwise. Some natural functions to consider in such domains would be the *successor* function on integers, the unary function s such that $s(t)$ gives $t + 1$, and unary functions on locations that give adjacent locations such as $left$ and $right$, where $left(l)$ and $right(l)$ indicate locations to the immediate left and right of l, respectively. Then an action that moves the robot left generically might require an object o, location l, and time t to be fixed such that $ROBOT(o)$ and $AT(o, l, t)$, and then asserts $\neg AT(o, l, s(t))$ and $AT(o, left(l), s(t))$—so the robot is no longer at l, but rather at the location to its left at the next point in time, $t + 1$. These generic formulas can be instantiated for any location and point in time. Note that $left(l)$ is now a term with the location type, as are $left(left(l))$ and $right(left(l))$ (which is presumably identical to l); similarly, $s(t)$, $s(s(t))$, etc. are also terms with the type of a point in time.

The main new feature of first-order logic as compared to propositional logic is the quantifier: supposing φ is a first-order formula and x is a variable, $\forall x\ \varphi$ is also a first-order formula. All occurrences of the variable x in φ are then said to be *bound* in $\forall x\ \varphi$. Strictly speaking, there may be occurrences of x that were already bound, if they appeared in some subformula $\forall x\ \psi$ of φ; we will assume without loss of generality that distinct variable names are used for all of the quantifiers in a formula, which will simplify matters. Indeed, if y is a variable name that does not occur in φ, we can *substitute* y for all occurrences of the variable x in φ and obtain an equivalent formula. If all of the variables in φ are bound, we say φ is a *sentence*. A sentence is true in an interpretation iff the formulas we obtain by substituting all possible combinations of objects of the domain (of the appropriate types) for the variables yields a formula that evaluates to true; when a knowledge base (set of sentences) are simultaneously true for an interpretation, we say that the interpretation is a *model* for the knowledge base. Analogous to before, we say that the sentence φ is *valid for a knowledge base* KB of sentences, denoted $KB \models \varphi$, if φ is true for all models of KB. Noting that all interpretations are vacuously models of the empty knowledge base, $\models \varphi$ denotes that φ is true in all interpretations, and we may simply say that φ is *valid*.

Example 9.2: Continuing the previous example, suppose there is a predicate $MOVE_{LEFT}$ such that $MOVE_{LEFT}(o, t)$ indicates that o moves left at time t. Then our previous formalization of this action can be captured by the first-order sentence

$$\forall o \forall l \forall t\ [ROBOT(o) \wedge AT(o, l, t) \wedge MOVE_{LEFT}(o, t)] \\ \Rightarrow [\neg AT(o, l, s(t)) \wedge AT(o, left(l), s(t))]$$

We can formulate this sentence without the use of functions as follows: suppose instead of the functions, we have binary predicates $LEFT$ and S on locations and time points, respectively, such that $LEFT(l, l')$ is true iff l' is to the left of l and

$S(t,t')$ is true iff t' is $t+1$. Then the above sentence is equivalent to

$$\forall o \forall l \forall l' \forall t \forall t' [ROBOT(o) \wedge AT(o,l,t) \wedge LEFT(l,l') \wedge S(t,t') \wedge MOVE_{LEFT}(o,t)] \Rightarrow [\neg AT(o,l,t') \wedge AT(o,l',t')]$$

In addition to the universal quantifier, in first-order logic one also often encounters the *existential quantifier*: similar to the treatment of connectives such as implication, the usual convention is that we define $\exists x\ \varphi$ to be shorthand for the formula $\neg[\forall x\ \neg\varphi]$. Naturally, this formula is true in an interpretation if there is some substitution of an appropriate object for x in φ that makes φ true, so it is equivalent to the existential quantifier as usually understood. Unfortunately, even though the existential quantifier adds nothing to the ultimate expressive power of the representations, the computational tasks associated with sentences that use both types of quantifiers are much more challenging.

9.2 Naïve propositionalization.

Now that we have introduced the basic language of first-order logic and how it is interpreted, we will start to consider the associated computational tasks. The first and perhaps simplest approach is a straightforward reduction to a propositional representation, which we considered back in Chapter 5.

Suppose we have fixed a finite domain of objects O. Let $\varphi|_{x\leftarrow o}$ denote the result of substituting the object $o \in O$ for the occurrences of the variable x in φ. Now, we observe that a sentence $\forall x\ \varphi$ is equivalent to the formula $\bigwedge_{o\in O} \varphi|_{x\leftarrow o}$. Similarly, recalling De Morgan's law, $\exists x\ \varphi$ is equivalent to $\bigvee_{o\in O} \varphi|_{x\leftarrow o}$. So, if we are given a finite domain of objects O, we can rewrite a sentence of first-order logic to eliminate the quantifiers, thus obtaining an essentially propositional formula. This transformation is known as *propositionalization*.

Apart from the limitation that the domain must be finite and known in advance, propositionalization also has the unfortunate effect of potentially increasing the formula size substantially. This increase in size is controlled by the *quantifier rank* of the formula:

Definition 9.1 The *quantifier rank* of a first-order formula is inductively defined (for the De Morgan basis) as follows. The quantifier rank of an atomic formula is 0. The quantifier rank of $\neg\varphi$ is the same as the quantifier rank of φ, and the quantifier ranks of $\varphi \wedge \psi$ and $\varphi \vee \psi$ are equal to the larger of the respective quantifier ranks of φ and ψ. Finally, the quantifier rank of $\forall x\ \varphi$ is one greater than the quantifier rank of φ.

Proposition 9.2 *Let $|\varphi|$ denote the size of φ in symbols, and let $|O|$ denote the cardinality of the finite set of objects O, which we suppose may be each represented by a single symbol. Suppose φ has quantifier rank k. Then the propositionalization of φ has size at most $O(|O|^k \cdot |\varphi|)$.*

Proof We proceed by induction on the structure of φ. For an atomic formula φ, $|\varphi| = 1$ and the propositionalization of φ also consists of a single symbol. Consider $\varphi \wedge \psi$ of quantifier rank k. By our inductive hypothesis, since both φ and ψ have quantifier rank at most k, they have propositionalizations of size at most $C \cdot |O|^k|\varphi|$ and $C \cdot |O|^k|\psi|$, where C is the big-O constant. Therefore, observe, the propositionalization of $\varphi \wedge \psi$ indeed has size at most $C \cdot |O|^k|\varphi \wedge \psi|$; $\varphi \vee \psi$ is similar. Finally, suppose that $\forall x\ \varphi$ has quantifier rank k. Then since φ has quantifier rank at most $k-1$, it has a propositionalization of size at most $C \cdot |O|^{k-1}|\varphi|$. The propositionalization consists of the conjunction of $|O|$ copies of φ, $\varphi|_{x \leftarrow o}$ for each $o \in O$. This has size $|O| \cdot (C \cdot |O|^{k-1}|\varphi| + 1) \leq C \cdot |O|^k|\forall x\ \varphi|$. □

Hence, if the quantifier rank is constant, we obtain a propositional representation that is of size polynomial in the number of objects in the domain.

Example 9.3: In general, if we have a knowledge base of universally-quantified clauses of bounded width, where the relations also have small arity, then each clause has quantifier rank at most the width times the arity. Since bounded-width resolution can be automated (Exercise 5.9), this gives a first subset of first-order logic that is automatable. In turn, the negation of existentially-quantified conjunctive formulas of bounded width produce universally quantified bounded width formulas, so this subset enables answering existentially quantified conjunctive queries against a knowledge base of universally quantified clauses, when all of these have small width and use relations of small arity.

9.3 The grounding trick and proper+.

Basic propositionalization is only effective when the domain of objects under consideration is relatively small. But, in many cases the domain is not small, or we may not have any idea of what is in it; indeed, in some cases, the domain may even be infinite. It turns out that nevertheless, as long as the formulas in question mention relatively few constants explicitly, we can find a small propositionalization that is sufficient for reasoning.

Let KB be a set of function-free first-order formulas and let q likewise be a function-free first-order formula; suppose the conjunction of KB and $\neg q$ has quantifier rank k. In the following, $\hat{O} \subseteq O$ will be a set of objects consisting of

- all constants appearing in KB and q and
- any k additional objects from O (or, simply, all of O if there are not so many additional objects).

Definition 9.3 The *propositionalization of a sentence φ with respect to a set of objects $\hat{O}$* that contains the constants in φ, $\mathsf{ground}_{\hat{O}}(\varphi)$ is defined as follows:

- An atomic formula is a sentence only if all of the terms in it are constants, in which case $\mathsf{ground}_{\hat{O}}(\varphi) = \varphi$.
- $\mathsf{ground}_{\hat{O}}(\neg\varphi) = \neg\mathsf{ground}_{\hat{O}}(\varphi)$, $\mathsf{ground}_{\hat{O}}(\varphi \wedge \psi) = \mathsf{ground}_{\hat{O}}(\varphi) \wedge \mathsf{ground}_{\hat{O}}(\psi)$, and similarly, $\mathsf{ground}_{\hat{O}}(\varphi \vee \psi) = \mathsf{ground}_{\hat{O}}(\varphi) \vee \mathsf{ground}_{\hat{O}}(\psi)$
- $\mathsf{ground}_{\hat{O}}(\forall x\ \varphi) = \bigwedge_{o \in \hat{O}} \mathsf{ground}_{\hat{O}}(\varphi|_{x \leftarrow o})$

Importantly, it turns out that including additional objects beyond the quantifier rank does not affect the satisfiability of $\mathsf{ground}_{\hat{O}}(\varphi)$:

Lemma 9.4 *Suppose φ has quantifier rank k. Let $\hat{O}$ consist of the constants of φ plus k additional objects from O, and let $O' \subseteq O$ be a set of objects that contains all of the constants of φ and $\ell \geq k$ additional objects from O. Then $\mathsf{ground}_{\hat{O}}(\varphi)$ is satisfiable iff $\mathsf{ground}_{O'}(\varphi)$ is satisfiable.*

Proof Without loss of generality, we suppose $\hat{O} \subset O'$. First, suppose $\mathsf{ground}_{\hat{O}}(\varphi)$ is satisfiable, and let w be a satisfying possible world. We will define a satisfying world w' for $\mathsf{ground}_{O'}(\varphi)$ as follows. w' will agree with w on all of the atomic formulas appearing in $\mathsf{ground}_{\hat{O}}(\varphi)$. Let $S = O' - \hat{O}$, and consider a subformula ψ appearing in $\mathsf{ground}_{O'}(\varphi)$ that mentions r objects from S, $o'_1, \ldots, o'_r$. Since φ has quantifier rank k, ψ mentions at most k objects that are not among the constants, and hence ψ does not mention at least r of the additional objects from $\hat{O}$, $\hat{o}_1, \ldots, \hat{o}_r$. Now, let w' assign each atomic formula α in ψ the same truth value as w assigned $\alpha|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r}$. Now, by induction on the structure of φ:

- for atomic formulas, by construction $w \models \mathsf{ground}_{\hat{O}}(\alpha|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r})$ iff $w' \models \mathsf{ground}_{O'}(\alpha)$.
- $w \models \mathsf{ground}_{\hat{O}}(\neg\varphi|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r})$ iff $w' \models \mathsf{ground}_{O'}(\neg\varphi)$, $w \models \mathsf{ground}_{\hat{O}}(\varphi \wedge \psi|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r})$ iff $w' \models \mathsf{ground}_{O'}(\varphi \wedge \psi)$ and similarly, $w \models \mathsf{ground}_{\hat{O}}(\varphi \vee \psi|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r})$ iff $w' \models \mathsf{ground}_{O'}(\varphi \vee \psi)$
- $\mathsf{ground}_{O'}(\forall x\ \varphi) = \bigwedge_{o \in O'} \mathsf{ground}_{O'}(\varphi|_{x \leftarrow o})$ where $\varphi|_{x \leftarrow o}$ has quantifier rank $k - 1$. For $o \in \hat{O}$, the subformula $\varphi|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r, x \leftarrow o}$ appears in $\mathsf{ground}_{\hat{O}}(\forall x\ \varphi)$ where by the induction hypothesis, $w' \models \varphi|_{x \leftarrow o}$ iff $w \models \varphi|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r, x \leftarrow o}$. Otherwise, o is some o'_i; then the induction hypothesis gives us that

  $$w' \models \varphi|_{x \leftarrow o'_i} \text{ iff } w \models (\varphi|_{x \leftarrow o'_i})|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r} = (\varphi|_{x \leftarrow \hat{o}_i})|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r}$$

  where, since the substitutions range over precisely $\hat{O}$ now, $w' \models \mathsf{ground}_{O'}(\forall x\ \varphi)$ iff $w \models \mathsf{ground}_{\hat{O}}((\forall x\ \varphi)|_{o'_1 \leftarrow \hat{o}_1, \ldots, o'_r \leftarrow \hat{o}_r})$. □

We now make a simple but important observation, that propositional logic has a property known as *compactness*.

Theorem 9.5 (Compactness) *Suppose every finite subset of a countably infinite set of formulas KB has a satisfying possible world. Then there exists a possible world that satisfies all of KB.*

Proof Since the formulas are finite and countably infinite, in particular the propositional attributes are countably infinite, and hence may be indexed by the integers,

$\{X_i\}_{i=1}^{\infty}$. We will define a satisfying possible world by induction on the index of these attributes: specifically, we will find $x_1, \ldots, x_n$ such that all finite subsets of KB have a satisfying possible world that is consistent with $X_1 = x_1, \ldots, X_n = x_n$. For the base case, every subset has a satisfying assignment that is consistent with the empty partial assignment, so we suppose that we have found a satisfactory set of values, $x_1, \ldots, x_n$.

If, for every finite subset S of KB, there is some satisfying possible world such that $X_1 = x_1, \ldots, X_n = x_n$ and $X_{n+1} = 1$, then we take $x_{n+1} = 1$, and this is sufficient by hypothesis. If not, we claim that it suffices to take $x_{n+1} = 0$: indeed, otherwise, there is a finite subset S^* of KB for which every satisfying possible world such that $X_1 = x_1, \ldots, X_n = x_n$ has $X_{n+1} = 0$. For any other finite subset S' of KB, we now consider the subset $S^* \cup S'$. Since, by induction hypothesis, $S^* \cup S'$ had a satisfying possible world that was consistent with $X_1 = x_1, \ldots, X_n = x_n$, and since any satisfying possible world on $S^* \cup S'$ is also a satisfying possible world for S^*, this satisfying possible world has $X_{n+1} = 0$. But likewise, this is also a satisfying possible world for S', and hence S' indeed has a satisfying possible world that is consistent with $X_1 = x_1, \ldots, X_n = x_n$ and $X_{n+1} = 0$, as needed. This completes the induction step. □

Corollary 9.6 *A countably infinite set of clauses KB is unsatisfiable iff there is a finite subset of KB with a (treelike) resolution refutation.*

Proof By compactness, if KB is unsatisfiable, it must have a finite unsatisfiable subset. By the completeness of treelike resolution (recall, Corollary 5.13), there is a refutation of this set of clauses. Conversely, if finite subset of KB has a refutation, the soundness of resolution ensures that it does not have a satisfying possible world, where any possible world satisfying KB would, in particular, necessarily satisfy this subset. □

Compactness is usually stated in the contrapositive of this form, that is, if all finite subsets of KB are *consistent*, then KB itself is also consistent. We now have the pieces to show that the finite grounding suffices:

Theorem 9.7 *Suppose KB contains only function-free universally quantified clauses of quantifier rank k, and consider any interpretation with a countable set of objects O. Let $\hat{O}$ consist of the constants of KB plus k additional objects from O. Then there is a treelike resolution refutation of $\mathsf{ground}_{\hat{O}}(\neg q \wedge \bigwedge_{\varphi \in KB} \varphi)$ iff $KB \models q$ on the domain of objects O.*

Proof Let $\mathsf{GROUND}(KB \cup \{\neg q\})$ denote the set of possible groundings of formulas from KB and $\neg q$ on O. By definition, $KB \models q$ iff $\mathsf{GROUND}(KB \cup \{\neg q\})$ is unsatisfiable for all interpretations. By compactness (Theorem 9.5), if $\mathsf{GROUND}(KB \cup \{\neg q\})$ is unsatisfiable, there is a finite unsatisfiable subset of $\mathsf{GROUND}(KB \cup \{\neg q\})$. The conjunction of this subset is a subformula of $\mathsf{ground}_{\hat{O} \cup O'}(\neg q \wedge \bigwedge_{\varphi \in KB} \varphi)$ for some finite $O' \subset O$. By Lemma 9.4, $\mathsf{ground}_{\hat{O}}(\neg q \wedge \bigwedge_{\varphi \in KB} \varphi)$ is therefore also unsatisfiable and hence, by Corollary 5.13, has a treelike resolution refutation. Conversely, if there is a refutation of $\mathsf{ground}_{\hat{O}}(\neg q \wedge \bigwedge_{\varphi \in KB} \varphi)$, this is also a refutation of $\mathsf{GROUND}(KB \cup \{\neg q\})$, which therefore cannot have a satisfying interpretation; hence, $KB \models q$. □

We thus see that the "grounding trick" of using only a suitable finite set of objects $\hat{O}$ gives a complete proof system for universally quantified clausal knowledge bases. Keep in mind, however, that completeness alone does not guarantee efficiency: CNF satisfiability is still a NP-complete problem. More specifically, in general, these treelike proofs may be large, even if the clauses are small and hence the quantifier rank is low and corresponding ground formula is not too large. In particular, propositional logic is a special case so Corollary 5.24 still shows that the pigeonhole formulas PHP_q^p require (sub)exponentially long resolution refutations, and moreover Exercise 5.10 shows that the treelike refutations must be exponentially long. In addition, even if the proof is small, Theorem 5.26 shows that it is NP-hard to determine that a small refutation exists. What the grounding trick does for us is to at least demonstrate that the main barriers (at least, for function-free universal clauses) are actually only from the hardness of the propositional problem. The unbounded domain of objects itself poses no fundamental barrier. It thus also presents natural subsets for which reasoning is tractable: CNFs for which the grounding has a small width or small space treelike refutation.

We can extend this approach slightly beyond clauses, to a subset of formulas of *first-order logic with equality* known as *proper+*. First-order logic with equality extends first-order logic with atomic formulas that are *equality formulas*, $t_1 = t_2$ for terms t_1 and t_2. It may be natural to think of = as another predicate, except that we fix its interpretation to be that when we substitute objects $o_1, o_2 \in O$ for t_1 and t_2, the formula is true iff o_1 is the same object as o_2. An *equality expression* is a De Morgan formula with equality formulas as its atomic formulas.

Definition 9.8 The set of *equality expressions* is inductively defined as follows:

- A formula of the form $t_1 = t_2$ for terms t_1 and t_2 is an equality expression.
- If φ is an equality expression, so is $\neg\varphi$.
- If φ and ψ are equality expressions, so are $\varphi \wedge \psi$ and $\varphi \vee \psi$.

Now, a *proper+* knowledge base is given by a set of formulas of the form $e \Rightarrow c$ where e is an equality expression and c is a clause; more precisely, it consists of the *universal closure* of such formulas, i.e., the sentences given by universal quantifiers over each of the variables appearing in one of these formulas $e \Rightarrow c$:

Definition 9.9 A *universally quantified proper+ clause* is a formula of the form $\forall x_1 \cdots \forall x_k \; e \Rightarrow c$ where e is an equality expression, c is an OR of a set of *literals*, where the literals consist of atomic formulas or negations of atomic formulas, and all variables occurring in c or e are included in $x_1, \ldots x_k$. A *proper+ knowledge base* is a set of universally quantified proper+ clauses.

The key observation is that when we ground a formula from a function-free proper+ knowledge base, since the interpretation of e is determined by the binding, we can evaluate e, and obtain a clause as before: if e is false in the binding, the clause is

vacuous, and otherwise we simply obtain the binding of c. Precisely now, note that

$$\mathsf{ground}_{\hat{O}}\left(\bigwedge_{\substack{(e \Rightarrow c)(x_1,\ldots,x_k)\\ \in KB}} \forall x_1 \cdots \forall x_k e \Rightarrow c\right) = \bigwedge_{\substack{(e \Rightarrow c)(x_1,\ldots,x_k)\\ \in KB}} \bigwedge_{\substack{o_1,\ldots,o_k \in \hat{O}^k:\\ e|_{x_1 \leftarrow o_1,\ldots,x_k \leftarrow o_k}=1}} c|_{x_1 \leftarrow o_1,\ldots,x_k \leftarrow o_k}.$$

The reader may verify that the proofs hold for proper+ knowledge bases, since the groundings simply give us sets of clauses. We thus obtain

Corollary 9.10 *Let KB be a function-free proper+ knowledge base of quantifier rank k, and consider any interpretation with a countable set of objects O. Let $\hat{O}$ consist of the constants of KB plus k additional objects from O. Then there is a treelike resolution refutation of* $\mathsf{ground}_{\hat{O}}(\neg q \wedge \bigwedge_{\varphi \in KB} \varphi)$ *iff $KB \models q$ on the domain of objects O.*

Since this extension means that we can refute proper+ knowledge bases, where we prove a query q by refuting $KB \cup \{\neg q\}$, we note that the queries we can prove take the form of "proper+ conjunctions:"

Definition 9.11 A *existentially quantified proper+ conjunction* is a formula of the form

$$\exists x_1 \cdots \exists x_k \; e \wedge c$$

where e is an *equality expression* and c is a conjunction, an AND of a set of literals (again, atomic formulas or negations of atomic formulas).

9.4 Resolution with functions is hard.

In the previous section, we saw that grounding gives a method for deciding the validity of function-free first-order logic when there are finitely many constants. This raises the question of whether these methods can be extended to capture functions. Unfortunately, the answer is no, in a very strong sense.

Theorem 9.12 (Church's Theorem) *There is no algorithm that decides whether or not sentences in first-order logic are valid.*

In particular, Church's Theorem holds regardless of the amount of time and memory used by the algorithm. The trouble, briefly, is that first-order logic with functions can be used to define the result of an arbitrary computation, including those that never terminate. Turing's proof of the theorem (which motivated his work on modeling computation) proceeded roughly as follows: Given an algorithm, we can in particular construct a sentence that is valid iff the given algorithm terminates; but, Turing also proved that no algorithm can correctly decide whether or not another algorithm terminates. As an algorithm to decide validity of arbitrary first-order sentences could in turn decide whether or not the computation terminates, no such algorithm can exist.

One glimpse of the power of functions in first-order logic is the following:

Theorem 9.13 *Let $\forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \varphi$ be an arbitrary first-order sentence. Then we can define* Skolem functions $s_2(x_1), s_4(x_1, x_3), \ldots, s_{2i}(x_1, x_3, \ldots, x_{2i-1}), \ldots$ *such that $\forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \varphi$ is satisfiable iff the* Skolemized sentence

$$\forall x_1 \forall x_3 \cdots \forall x_{2i-1} \cdots \varphi|_{x_2 \leftarrow s_2(x_1), x_4 \leftarrow s_4(x_1,x_3), \ldots, x_{2i} \leftarrow s_{2i}(x_1,\ldots,x_{2i-1}), \ldots}$$

is satisfiable.

Proof Suppose that for some binding of $x_1 \ldots, x_{2i-1}$, to $o_1, \ldots, o_{2i-1}$ for which a formula $\exists x_{2i} \psi|_{x_1 \leftarrow o_1, \ldots, x_{2i-1} \leftarrow o_{2i-1}}$ is satisfiable. Then there is a binding of x_{2i} to some o_{2i} for which $\psi|_{x_1 \leftarrow o_1, \ldots, x_{2i-1} \leftarrow o_{2i-1}, x_{2i} \leftarrow o_{2i}}$ is also satisfiable. Thus, we can use the interpretation for s_{2i} that maps $o_1, \ldots, o_{2i-1}$ to o_{2i} to satisfy $\psi|_{x_1 \leftarrow o_1, \ldots, x_{2i-1} \leftarrow o_{2i-1}, x_{2i} \leftarrow s_{2i}(o_1, \ldots, o_{2i-1})}$; and conversely, if the Skolemized sentence is satisfiable, given a satisfying interpretation of s_{2i}, the binding of x_{2i} to $s_{2i}(o_1, \ldots, o_{2i-1})$ shows that the original sentence is also satisfiable for the binding of $x_1 \ldots, x_{2i-1}$, to $o_1, \ldots, o_{2i-1}$ □

We remark that given the ability to represent existentially quantified variables, we can follow the construction in Theorem 3.29 to represent any Boolean formula as a 3CNF using a possibly larger set of variables (but no larger than the number of connectives in the original formula), using new constants for truth values 0 and 1, and a new predicate T appearing in the additional clauses $T(1)$ and $\neg T(0)$ (enforcing that T interprets the truth values as intended). Thus, any first-order sentence may be written as a sentence in which the body is a 3CNF, and then this sentence may be Skolemized – removing the new variables – to obtain a universally quantified 3CNF that is satisfiable iff the original formula was:

Corollary 9.14 *There is a polynomial-time algorithm that, on input an arbitrary first-order sentence $\forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \varphi$, constructs a universally quantified 3CNF using $O(|\varphi|)$ additional functions, two additional constants, and one additional predicate that is satisfiable iff the input sentence is.*

Actually, since the family of possible groundings is (only) countably infinite, we obtain the following consequence of compactness (Corollary 9.6):

Corollary 9.15 *An arbitrary first-order sentence $\forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \varphi$ is unsatisfiable iff there is a finite collection of groundings of the universally quantified 3CNF formula constructed in Corollary 9.14 with a treelike resolution refutation.*

So resolution is essentially (refutation) complete for first-order logic following these transformations. Unfortunately, it follows immediately from Church's Theorem that there can be no way of identifying the appropriate groundings in general when functions are present.

9.5 Lifted resolution.

To summarize, when we consider first-order logic with functions, Corollary 9.15 together with Church's Theorem shows that even resolution on universally quantified clauses is too powerful to obtain an algorithm with good theoretical guarantees. We generally cannot identify a set of groundings to use as a starting point for propositional resolution. Although we still cannot obtain good guarantees, it is useful to consider an alternative form of resolution that operates at a "lifted" level, i.e., on universally quantified clauses directly.

For example, suppose we have formulas $\forall x R(x)$ and $\forall y \neg R(f(y))$. For any binding of $y \leftarrow o$ and any interpretation of f, we can consider the binding of $x \leftarrow f(o)$, and obtain $\bot$ from $R(f(o))$ and $\neg R(f(o))$. But abstractly, we can consider a substitution that simply maps $x \leftarrow f(y)$. Then we can obtain the clauses $\forall y R(f(y))$ and $\forall y \neg R(f(y))$ to which we could apply a syntactic resolution rule on the atomic formula $R(f(y))$ which has a free variable.

More generally, the lifted resolution rule takes universally quantified clauses of the form $\alpha \vee C$ and $\neg\beta \vee D$ for atomic formulas α and β, and a substitution σ such that $\alpha|_\sigma = \beta|_\sigma$. One then infers the universally quantified clause $C|_\sigma \vee D|_\sigma$. Actually, in full generality:

Definition 9.16 *Lifted resolution* is a proof system for which the lines consist of universally quantified clauses. For any universally quantified clauses $\forall x \cdots A \vee C$ and $\forall x \cdots B \vee D$, and any substitution on variables such that for some atomic formula α $A|_\sigma$ simplifies to α and $B|_\sigma$ simplifies to $\neg\alpha$, we may infer the universally quantified clause $\forall x \cdots C|_\sigma \vee D|_\sigma$. We call the substitution σ a *unifier* for A and B.

We need this variant of the rule to refute unfortunate pairs like $\forall x \forall y R(x) \vee R(y)$ and $\forall x \forall y \neg R(x) \vee \neg R(y)$ that cannot be handled one term at a time.

As stated, we can take any grounding as a substitution, so lifted resolution can directly simulate grounded resolution. It is therefore also essentially complete, and inherits the same fundamental barriers as ground resolution. But, it has the advantage of being able to compactly represent a large set of possible groundings—indeed, the set of *all possible* groundings that can be obtained for these clauses, if we use a *"most general unifier."*

Definition 9.17 A unifier σ for clauses A and B is a *most general unifier* if for any other unifier τ there is a third substitution θ such that $A|_\tau$ is the result of applying θ to $A|_\sigma$ and $B|_\tau$ is the result of applying θ to $B|_\sigma$. That is, we may write $\tau = \theta \circ \sigma$, where $\circ$ denotes composition.

Most general unifiers, at least, can be computed quite efficiently, in linear time. We'll develop a much simpler near-linear time algorithm here. These fast algorithms obtain a unifier by constructing a hierarchy of equivalence classes of terms:

Definition 9.18 A *unification relation* is an equivalence relation on terms satisfying the following properties

- *homogeneous*: no equivalence class contains terms of the form $f(\cdots)$ and $g(\cdots)$ for functions $f \neq g$
- *acyclic*: no equivalence class contains both a term and a proper subterm of that term
- *unification axiom*: for any function f and subterms $s_1, \ldots, s_k$ and $t_1, \ldots, t_k$, if $f(s_1, \ldots, s_k)$ is in the same class as $f(t_1, \ldots, t_k)$ then for each $i = 1, \ldots, k$, s_i is in the same equivalence class as t_i.

The *unification closure* of two terms s and t is the least unification relation for which s and t are equivalent.

Here, we will regard constants as functions with arity 0.

Lemma 9.19 *If there exists a variable substitution σ such that $s|_\sigma = t|_\sigma$, then there exists a unification closure for s and t.*

Proof First observe that since σ only substitutes for variables, the sets of terms that are equivalent after substitution by σ is indeed a unification relation. Now, consider the relation obtained by first taking s and t as equivalent, and then considering the closure under the unification axiom. Observe that this is indeed a unification relation. Since any unification relation must satisfy the unification axiom, this is therefore the least unification relation, and hence a unification closure. □

A unification closure implicitly specifies a most general unifier.

Definition 9.20 For a unification closure $\equiv$, let $\sigma_\equiv$ be a substitution for variables defined as follows. For each variable term s:

- If s is only equivalent to other variables in $\equiv$, let $\sigma_\equiv$ substitute some canonical member of the equivalence class for s.
- Otherwise, let $s|_{\sigma_\equiv}$ be a canonical constant member of the equivalence class for s.

Theorem 9.21 *Terms s and t are unifiable iff $\sigma_\equiv$ for a unification closure $\equiv$ for s and t is a most general unifier for s and t.*

Proof Clearly, if there is a most general unifier, s and t are unifiable. For the other direction, Lemma 9.19 establishes first that if s and t are unifiable, then there exists a unification closure $\equiv$ for s and t, so we can consider the substitution $\sigma_\equiv$ as defined above. We first observe that for any two terms u and v in the same equivalence class, by induction on the structure of the terms $u|_{\sigma_\equiv} = v|_{\sigma_\equiv}$: indeed it is immediate for variables and constants. If u and v are function terms, since the relation is homogeneous, $u = f(u'_1, \ldots, u'_k)$ and $v = f(v'_1, \ldots, v'_k)$ where by the unification axiom, each $u'_i \equiv v'_i$. By acyclicity each u'_i and v'_i is a member of a equivalence class for smaller terms than u or v, so we have $u'_i|_{\sigma_\equiv} = v'_i|_{\sigma_\equiv}$ by the induction hypothesis. Therefore indeed $u|_{\sigma_\equiv} = v|_{\sigma_\equiv}$. So $\sigma_\equiv$ in particular is a unifier for s and t.

It now suffices to show that for any other unifier τ for s and t, for any term u, $u|_{\tau \circ \sigma_\equiv} = u|_\tau$. We observe that the equivalence relation given by $u \equiv_\tau v$ iff $u|_\tau = v|_\tau$ is indeed a unification relation, and that moreover, $\equiv$ – being the least unification relation – is

more specific than $\equiv_\tau$. Hence, if for any constants or variables u and v we have $u \equiv v$, we also have $u \equiv_\tau v$. So, if $u \equiv v$, then $u|_\tau = v|_\tau$, and so in particular noting that $u|_{\sigma_\equiv} \equiv u$, $u|_{\tau\circ\sigma_\equiv} = u|_\tau$. So now more generally, by induction on the structure of a term $u = f(u'_1, \ldots, u'_k)$, we know that $u'_i|_{\tau\circ\sigma_\equiv} = u'_i|_\tau$ for $i = 1, \ldots, k$. Thus,

$$u|_\tau = f(u'_1|_\tau, \ldots, u'_k|_\tau) = f(u'_1|_{\tau\circ\sigma_\equiv}, \ldots, u'_k|_{\tau\circ\sigma_\equiv}) = u|_{\tau\circ\sigma_\equiv}$$

completing the induction step. □

Therefore, it suffices to construct a unification closure for s and t to obtain the most general unifier. The core of these fast algorithms is an efficient union-find data structure.

Definition 9.22 A *union-find* data structure maintains a collection of disjoint sets on some universe U using three operations:

1. *Create-element*: takes a representation of an element $u \in U$, creates a new singleton set $\{u\}$, and returns a reference to u.
2. *Find*: takes a reference to an element u and returns the reference to the set currently containing u.
3. *Union*: takes references to two sets and replaces them with a single set containing the union of the two.

A simple union-find data structure represents the sets using linked lists, augmented with two additional fields: one points to the head of the list, and the other field is only used for the head of the list; it contains the total number of elements in the set. References to the head of the list represent the set. Concretely:

Input: Element $u \in U$
Output: Representation of u
begin
  return a new node containing u with *next* set to null, *size* set to 1, and *head* containing a reference to itself.
end

Algorithm 16: Create-element

Input: Node representing u from Create-element
Output: Reference to set containing u
begin
  return contents of *head* for given node.
end

Algorithm 17: Find

Theorem 9.23 *Algorithms 16, 17, and 18 implement a union-find data structure in which Create-element and Find run in $O(1)$ time each, and n Union operations run in total time $O(n \log n)$.*

Input: Reference to sets S_0 and S_1
Output: Reference to new set containing $S_0 \cup S_1$, replacing S_0 and S_1
begin
  Put $i \leftarrow \arg\max_{i \in \{0,1\}} size(S_i)$
  Put $size(S_i) \leftarrow size(S_i) + size(S_{i-1})$
  Put $current \leftarrow S_{1-i}$
  while $next(current)$ *is not null* **do**
    $head(current) \leftarrow S_i$
    $current \leftarrow next(current)$
  end
  $head(current) \leftarrow S_i$
  $next(current) \leftarrow next(S_i)$
  $next(S_i) \leftarrow current$
  return S_i.
end

Algorithm 18: Union

Proof Correctness follows since we maintain the invariant that $head$ contains a reference to the head of the list that represents the set containing the nodes corresponding to the elements in the set: this is immediate from the construction of Create-set and Find. Union maintains this invariant: all of the nodes in the list for S_i already have $head$ set to the head of the list, and all of the nodes in S_{1-i} have $head$ updated to S_i; the list for S_{1-i} is inserted into the list for S_i following the head, so the list now indeed contains the union of the elements, and all of the nodes indeed have $head$ set to the head of S_i.

For the running time, it is also immediate that Create-set and Find run in time $O(1)$. As for Union, we observe first that we indeed maintain the invariant that the field $size$ in the head of the list contains the size of the set: Create-set sets this to 1, and Union updates the sizes from $size(S_0)$ and $size(S_1)$ to $size(S_0) + size(S_1)$ for the new set.

Now, observe that Union takes $O(1)$ time to update the head of S_i, plus $O(1)$ time to update each node in S_{1-i} where $|S_{i-1}| \leq |S_i|$, so $|S_0 \cup S_1| \geq 2|S_{i-1}|$: that is, the size of the set containing these nodes doubles each time it is updated. At the same time, since all sets initially have size 1, after n calls to Union, the largest set has size at most n. Now observe: no node may have been updated more than $\log_2 n$ times, or else there would be a set of size greater than $2^{\log_2 n} = n$. There are also at most $2n$ distinct elements in these n calls. Therefore, the n calls to union update at most $2n \log_2 n$ nodes in total. We thus see that the total running time is indeed $O(n \log n)$. □

We will use the union-find structure to maintain the equivalence classes of terms corresponding to a candidate unification closure. We extend the node structures to include a symbol sym to be substituted for variables, list of variables $vars$ appearing in the term, and two Boolean flags, $visited$ and $acyclic$, both initialized to false. We will also include a global dictionary $Dict$ that maps terms to their sets.

We then simply need to check that the resulting structure is acyclic, and if so, we can construct the most general unifier recursively.

Input: Terms s and t
Output: Reference to sets containing homogeneous, unification-closed equivalence classes of s and t, or $FAIL$
begin
  if $Dict(s)$ *and* $Dict(t)$ *are nonempty and Find(*$Dict(s)$*)=Find(*$Dict(t)$*)* **then return** $Dict(s)$
  if $Dict(s)$ *is empty* **then**
    Put $Dict(s) \leftarrow$Create-element(s), $sym(Dict(s)) \leftarrow s$
    if s *is a variable* **then** $vars(Dict(s)) \leftarrow \{s\}$
  end
  if $Dict(t)$ *is empty* **then**
    Put $Dict(t) \leftarrow$Create-element(t), $sym(Dict(t)) \leftarrow t$
    if t *is a variable* **then** $vars(Dict(t)) \leftarrow \{t\}$
  end
  Put $S \leftarrow Find(Dict(s)), T \leftarrow Find(Dict(t))$
  if $sym(S)$ *or* $sym(T)$ *are variables* **then**
    if $sym(S)$ *and* $sym(T)$ *are variables* **then** Put $sym \leftarrow sym(S)$
    else Put $sym \leftarrow u \in \{sym(S), sym(T)\}$: u is not a variable
    Put $vars \leftarrow vars(S) \cup vars(T)$
    Put $C \leftarrow$Union(S, T), $sym(C) \leftarrow sym$ and $vars(C) \leftarrow vars$.
    return C
  end
  else if $s = f(s'_1, \ldots, s'_k)$ *and* $t = f(t'_1, \ldots, t'_k)$ **then**
    Initialize $vars \leftarrow \emptyset$, $sym \leftarrow f(\cdots)$
    for $i = 1, \ldots, k$ **do**
      Put $C'_i \leftarrow$Create-closure(s'_i,t'_i)
      if $C'_i = FAIL$ **then return** $FAIL$
      $vars \leftarrow vars \cup vars(C'_i)$
      $sym \leftarrow sym$ with $sym(C'_i)$ as the ith argument
    end
    Put $C \leftarrow$Union(S, T)
    $vars(C) \leftarrow vars$, $sym(C) \leftarrow sym$.
    return C.
  end
  else
    // Symbols clash: $s = f(\cdots)$ and $t = g(\cdots)$ for $f \neq g$
    return $FAIL$
  end
end

Algorithm 19: Create-closure

Theorem 9.24 *If we first invoke Create-closure on a pair of terms s and t and it does not return FAIL, then Find-unifier on s returns a most general unifier of s and t if*

Input: Term s with created unification closure
Output: Unifier for terms in s, or $FAIL$
begin
  Put $S \leftarrow Dict(s)$
  if *s is a variable* **then return** substitution $s \leftarrow sym(\text{Find}(S))$
  else if $acyclic(Find(S))$ **then return** identity substitution
  else
    // $s = f(s'_1, \ldots, s'_k)$
    Initialize $\sigma \leftarrow$ identity substitution
    if $visited(Find(S))$ **then return** $FAIL$
    Put $visited(Find(S)) \leftarrow true$
    for $v \in vars(Find(S))$ **do**
      if $\sigma_v \leftarrow$*Find-unifier(v)*$= FAIL$ **then return** $FAIL$
      else Put $\sigma \leftarrow \sigma_v \circ \sigma$
    end
    Put $visited(Find(S)) \leftarrow false, acyclic(Find(S)) \leftarrow true.$
    return σ
  end
end

Algorithm 20: Find-unifier

one exists, and returns FAIL otherwise. The total running time of these algorithms is $O(n \log n)$ where n is the total length of s and t.

Proof Recall that by Theorem 9.21, if there is a unifier for s and t, then there is a least unification closure $\equiv$, and then $\sigma_{\equiv}$ is a most general unifier. We first claim that in this case, Create-closure indeed creates a least unification closure for the terms appearing in s and t: indeed, by induction on the total size of s and t, it creates disjoint classes for individual constants and variables – as needed for a least unification closure – for which sym contains a substitution meeting the requirements for $\sigma_{\equiv}$. For the induction step, assuming s and t are indeed unifiable, it recursively identifies each s_i and t_i (which by hypothesis are unifiable) so that the unification axiom holds, and then places s and t in the same equivalence class as required for a unification closure of s and t. Since by hypothesis s and t were unifiable, the resulting classes are indeed homogeneous and acyclic. Since each equivalence class only contains terms required by the unification axiom for a unifier of s and t, it is indeed a least unification closure.

Then, since the relation is acyclic by hypothesis, Find-unifier does not encounter a visited class during its recursion; it therefore returns the substitution $\sigma_{\equiv}$ for a unifier of the class containing the input term s, which (by Theorem 9.21 again) is a most general unifier.

Conversely, if s and t are not unifiable, then either the equivalence class would be inhomogeneous/needs to unify distinct constants (so Create-closure returns $FAIL$) or else the closure under the unification axiom is cyclic. In this latter case, when the

recursion in Find-unifier finds a sub-term that is equivalent to a given term, since *visited* is set for the class, it returns $FAIL$.

As for the running time, note that the number of invocations of Create-closure and Find-unifier are bounded by the number of symbols in s and t, where Create-closure calls Union once per symbol, and otherwise each does $O(1)$ work per symbol. Hence, by Theorem 9.23, the total running time is indeed $O(n \log n)$, dominated by the n calls to Union. □

9.6 Simple relational learning problems are NP-hard.

We now turn to the problem of learning relational representations. We will begin by considering the full-information learning setting. Here, we follow the natural generalization from the propositional learning setting, in which the examples consist of the truth values of all of the atomic formulas for some set of objects, together with a label indicating whether or not an unknown property from some class of representations C holds for these valuations. As usual, there is an unknown distribution D on these examples, and the definition of learning is essentially identical to Definition 4.15, where the possible worlds are taken to be given by these valuations. Needless to say, this entails that the set of objects is finite, and for all practical purposes, that the arity of the atomic formulas is small.

Occam's Razor (Theorem 4.2) may still be applied, and establishes that it is sufficient to find a representation that is consistent with a set of examples that is of size similar to the representations we hope to learn. Unfortunately, this turns out to be quite challenging: Even when the language consists of only unary relations and the universe has only three objects, we cannot learn universally quantified clauses or existentially quantified conjunctions.

Theorem 9.25 *Unless NP has randomized polynomial-time algorithms, there is no efficient proper PAC-learning algorithm for proper+ existentially quantified conjunctions with unary predicates and universes with at most three objects.*

Proof By Theorem 4.11, a proper PAC-learning algorithm entails a randomized polynomial-time algorithm for finding a consistent existentially quantified proper+ conjunction. We give a reduction from 3COLORING to the problem of finding such a representation, which will establish the claim.

Given a undirected graph on vertices $v_1, \ldots, v_n$, we create a set of unary predicates corresponding to the assignments of colors to vertices $X_{1,1}, X_{1,2}, X_{1,3}, \ldots, X_{n,1}, X_{n,2}, X_{n,3}$. We then construct the following set of examples with three objects each: first, we have an example in which all of the predicates are true for all three objects, labeled true:

$$\left(\begin{array}{l} (1, \ldots, 1) \\ (1, \ldots, 1) \\ (1, \ldots, 1) \end{array} , 1 \right)$$

Next, for each ith attribute, we create four examples:

$$\left(\begin{array}{c} \\ (1,\ldots,1,\overbrace{0,0,1}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \\ (1,\ldots,1) \\ (1,\ldots,1) \end{array} ,0 \right),$$

$$\left(\begin{array}{c} \\ (1,\ldots,1,\overbrace{0,1,0}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \\ (1,\ldots,1) \\ (1,\ldots,1) \end{array} ,0 \right),$$

$$\left(\begin{array}{c} \\ (1,\ldots,1,\overbrace{1,0,0}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \\ (1,\ldots,1) \\ (1,\ldots,1) \end{array} ,0 \right),$$

and

$$\left(\begin{array}{c} (1,\ldots,1,\overbrace{0,1,1}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \\ (1,\ldots,1,\overbrace{1,0,1}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \\ (1,\ldots,1,\overbrace{1,1,0}^{V_{i,1},V_{i,2},V_{i,3}},1,\ldots,1) \end{array} ,1 \right),$$

Finally, for each edge $\{v_i, v_j\}$, and each color c we have one example

$$\left(\begin{array}{c} \\ (1,\ldots,1,\overbrace{0}^{X_{i,c}},1,\ldots,1,\overbrace{0}^{X_{j,c}},1,\ldots,1) \\ (1,\ldots,1) \\ (1,\ldots,1) \end{array} ,0 \right).$$

Observe that this set of examples can be constructed in linear time in the size of the graph. Suppose that there is an existentially quantified proper+ conjunction $\exists x_1,\ldots,\exists x_k e \;\Rightarrow\; \bigwedge_j L_j(x_{i_j})$ that is consistent with this set of examples, where each L_j is a literal, i.e., either some $X_{i,c}$ or some $\neg X_{i,c}$. We first note that since the representation must be consistent with the first example, actually, no literal of the form $\neg X_{i,c}$ may appear, or else the conjunction would be false for this example. Second, we must have three distinct objects x_1, x_2, and x_3 all containing at least two of $X_{i,1}$, $X_{i,2}$, and $X_{i,3}$ for some object or else some example in the second set of examples would be labeled true. We note that for each i and object x_j, we cannot include all three $X_{i,c}(x_j)$, or else the example for the ith attribute would not be labeled true in the fourth set for i. Thus, for each object, precisely one is omitted. Finally, we note that for each edge $\{v_i, v_j\}$, we cannot have both $X_{i,c}$ and $X_{j,c}$ omitted for one of the objects, or else the final set of examples would be labeled true. Therefore, consider one of the objects x;

we know that for each vertex i, one of the colors c, $X_{i,c}$ cannot appear. Suppose we assign each vertex i the color of the omitted $X_{i,c}$. Since we cannot omit c for both v_i and v_j, the graph is properly colored.

On the other hand, conversely, if there exists a proper coloring χ assigning vertex v_i color $\chi(i)$, then the proper+ conjunction

$$\exists x_1 \exists x_2 \exists x_3 \, [x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3] \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{3} X_{i,\chi(i)+j \bmod 3}(x_j) \wedge X_{i,\chi(i)+j+1 \bmod 3}(x_j)$$

is indeed consistent with the examples. □

Corollary 9.26 *Unless NP has randomized polynomial-time algorithms, there is no efficient proper PAC-learning algorithm for universally quantified proper+ clauses.*

Proof We invoke essentially the same reduction as used in Section 4.3: we negate the labels and attributes in the examples used in Theorem 9.25. Observe that for any existentially quantified conjunction $\exists x_1 \cdots \exists x_k e \wedge \bigwedge_j R_j(x_{i_j})$, we have

$$\exists x_1 \cdots \exists x_k e \wedge \bigwedge_j R_j(x_{i_j}) = \neg \forall x_1 \cdots \forall x_k [\neg e] \Rightarrow \bigvee_j \neg R_j(x_{i_j})$$

so, if we could find a consistent universally quantified proper+ clause that is consistent with this new set of examples, the existentially quantified proper+ conjunction on the same set of atoms is consistent with the original set of examples. Hence, a PAC-learning algorithm for universally quantified proper+ clauses would, by Theorem 9.25, give a randomized algorithm for any problem in NP. □

Thus, we see that (assuming NP is intractable) we cannot have efficient algorithms to identify proper+ expressions that are consistent with sets of examples – either expressions for the knowledge base or for the query – even given full information about only three objects with only unary predicates. We will need to seek an easier variant of the learning task for relational logics.

9.7 Implicit learning of proper+ KBs.

We have some tractable methods for reasoning with relational representations, even in large or infinite domains via the grounding trick, but as we saw in the previous section, learning presents two new forms of our main challenges:

1. How can we set up learning to permit generalization in large domains?
2. How can we achieve computationally tractable learning?

Let's suppose that we have fixed a function-free first-order relational language. We consider a distribution D on (possibly countably infinite) interpretations, which recall, consist of a pair, a set of objects O and an interpretation mapping ι. We hope to reason with the following semantics

Definition 9.27 We say that a formula φ is p-valid w.r.t. D if φ is true in an interpretation drawn from D with probability at least p.

We would like to be able to learn and use universally quantified expressions. At first glance, this seems to place us in a bind: how can we hope to infer a universally quantified expression is true without examining the truth values of every possible binding of the expression? If a domain is large or infinite, this seems hopeless. We can dodge this issue by using an implicit learning strategy, as we saw previously in Section 8.3 for propositional representations.

In the first place, we can naturally tackle learning in an infinite domain by considering *partial examples* that will specify truth values for (only) a subset of the atomic formulas.

Definition 9.28 A *masking function* m for a first-order relational language is a mapping that takes an interpretation (O, ι) to a finite assignment ρ of truth values to ground atomic formulas α on O such that if α is given a truth value by ρ, that value is $\iota(\alpha)$. A *masking process* is a masking-function-valued random variable.

As before, our partial examples will be obtained by drawing an interpretation (O, ι) from D, independently drawing a masking function m from M, and applying $m(O, \iota)$ to obtain ρ. With a slight abuse of notation, we can interpret one of these partial examples as partial examples for the propositional possible worlds corresponding to one of our groundings $\mathsf{ground}_{\hat{O}}(KB)$ of a proper+ knowledge base KB with the set of objects $\hat{O}$. Then our notions of partial evaluation and testability for propositional formulas can be directly carried over to this setting, when applied to a grounding. Using the grounding trick, we will be able to "lift" implicit learning of proper+ knowledge bases from the propositional setting as well. We first extend lifted resolution to proper+:

Definition 9.29 *Proper+ resolution* is a proof system for which the lines consist of proper+ clauses, universally quantified expressions of the form $\forall x \cdots e \Rightarrow c$ where e is an equality expression and c is a clause. For any proper+ clauses $\forall x \cdots e_1 \Rightarrow A \vee C$ and $\forall x \cdots e_2 \Rightarrow B \vee D$, and any substitution on variables such that for some atomic formula α $A|_\sigma$ simplifies to α and $B|_\sigma$ simplifies to $\neg\alpha$, we may infer the proper+ clause $\forall x \cdots [e_1 \wedge e_2]|_\sigma \Rightarrow C|_\sigma \vee D|_\sigma$. We call the substitution σ a *unifier* for A and B. In a *proper+ resolution refutation*, the final line consists of the empty equality expression and empty clause, hence uses a unifier that satisfies the equality expression.

Motivated by the grounding trick, we consider the following definition:

Definition 9.30 We say that a set of resolution refutations Π is *closed under groundings* if whenever there is a (lifted) refutation $\pi \in \Pi$ of a set of clauses η, then for any set of objects $\hat{O}$ that contains the constants appearing in Π as well as r additional objects where r is the quantifier rank of η, the hypotheses used in π, there is a ground refutation of $\mathsf{ground}_{\hat{O}}(\eta)$ in Π.

Proposition 9.31 *The set of space-s treelike lifted resolution refutations is closed under groundings. The same holds for proper+.*

Proof Observe that in a treelike refutation π of a set of clauses η, each line is used exactly once. Hence, if π_i is used with a substitution σ and obtained from π_j and π_k using the substitution τ, i.e., π_i is the result of the resolution rule on $\pi_j|_\tau$ and $\pi_k|_\tau$, then we can obtain $\pi_i|_\sigma$ from $\pi_j|_{\sigma\circ\tau}$ and $\pi_k|_{\sigma\circ\tau}$: $\sigma\circ\tau$ is a unifier since we had $\pi_j = A\vee C$, $\pi_k = B\vee D$ where $A|_\tau = \neg B|_\tau$ and so $A|_{\sigma\circ\tau} = \neg B|_{\sigma\circ\tau}$. Indeed, recursively, we can simply apply the sequence of substitutions $\sigma_1\circ\cdots\circ\sigma_h$ to a hypothesis η_i where $\sigma_1,\ldots,\sigma_h$ is the sequence of substitutions appearing on the path in the graph of π from the root empty clause to η_i. Observe that the most general substitutions used for unification only mention constants already appearing in the hypotheses of π (ultimately, in η) and do not increase the quantifier rank. Hence, we can apply any grounding of the remaining free variables with the remaining r distinct objects from $\hat{O}$ to obtain a refutation of $\mathsf{ground}_{\hat{O}}(\eta)$ with the same graph: in particular, it is treelike and has the same rank and thus uses the same clause space.

As for proper+, we note that the substitution in the final clause in π must satisfy the conjunction of all equality expressions used in the proof. Following Lemma 9.4, the grounding remains satisfiable if we add the r objects we used in $\hat{O}$, and in turn the grounding using $\hat{O}$ is also satisfiable. Hence, there is a substitution that satisfies these expressions and only uses constants from our set of r objects. Hence, the grounding we obtain also satisfies the equality expressions appearing in all of the hypotheses, so we indeed obtain a propositional space-s treelike resolution refutation using this grounding. □

We remark that the proof shows, more strongly, that there is a ground proof that is of the same size as the original lifted refutation.

Input: Proper+ knowledge base KB, existentially quantified proper+ conjunctive query q, parameter r such that r is at least the quantifier rank of KB and q, set of objects O_0 containing the constants of KB and q, partial examples $\rho^{(1)},\ldots,\rho^{(m)}$
Output: A lower bound $\hat{p}$ for the validity of $KB \Rightarrow q$ w.r.t. D
begin
  Initialize $COUNT \leftarrow 0$
  for *i=1,…,m* **do**
    foreach *r-tuple of objects* $(o_1,\ldots,o_r)$ *in* $\rho^{(i)}$ *that are* not *in* O_0 **do**
      Put $\hat{O} \leftarrow O_0 \cup \{o_1,\ldots,o_r\}$.
      if $A(\mathsf{ground}_{\hat{O}}(KB\wedge\neg q)|_{\rho^{(i)}})$ *returns "unsat"* **then** Increment $COUNT$ and skip to the next i
    end
  end
  return $COUNT/m$.
end

Algorithm 21: Reasoning with implicit learning from partial examples for proper+ knowledge bases

Theorem 9.32 *Suppose A decides provability w.r.t. Π, a set of resolution refutations that is closed under groundings and restrictions. For any $\delta, \gamma \in (0,1)$, proper+ knowledge base KB, existentially quantified proper+ query q, integer rank parameter r that is at least the rank of KB and q, set of objects O_0 containing the constants of KB and q, distribution D over interpretations, and masking process M, given $m \geq \frac{1}{2\gamma^2} \ln \frac{2}{\delta}$ partial examples from $M(D)$, Algorithm 21 returns a value $\hat{p}$ such that with probability at least $1 - \delta$*

- ***Soundness:*** *if $KB \Rightarrow q$ is p-valid, then $\hat{p} \leq p + \gamma$*
- ***Completeness:*** *if there is a proper+ KB I such that*
  1. *there is a refutation of $I \wedge KB \wedge \neg q$ in Π*
  2. *the quantifier rank of I is also at most r, and all constants in I appear in O_0 and*
  3. *with probability at least p over partial examples $\rho \in M(D)$, there exist objects $o_1, \ldots, o_r$ not corresponding to the constants in O_0, such that every clause in $\mathsf{ground}_{O_0 \cup \{o_1,\ldots,o_r\}}(I)$ is witnessed true in ρ*

  then $\hat{p} \geq p - \gamma$.

Proof **Soundness:** We first note that since resolution is sound, if A contains a refutation, then indeed $\mathsf{ground}_{\hat{O}}(KB \wedge \neg q)|_{\rho^{(i)}} \models \bot$. Now, since $\rho^{(i)}$ is consistent with the ith interpretation drawn from D, there is some grounding of a clause in KB or a grounding of $\neg q$ that is falsified in that interpretation (or else any grounding of $KB \cup \{\neg q\}$ would be satisfiable). Hence, $KB \cup \{\neg q\}$ was falsified in this interpretation, i.e., $KB \Rightarrow q$ is true. Now, if $KB \Rightarrow q$ is p-valid, Hoeffding's inequality gives us that with probability at least $1 - \delta/2$, this occurs in at most $p + \gamma$ of the interpretations drawn from D. Hence, $\hat{p} \leq p + \gamma$ with probability at least $1 - \delta/2$.
Completeness: For any $\hat{O}$ that contains the constants of KB, I, and q together with an additional r objects where r is at least the quantifier rank of KB, I, and q, if there is a refutation of $KB \cup I \cup \{\neg q\}$ in Π, then there is also a refutation of $\mathsf{ground}_{\hat{O}}(KB \wedge I \wedge \neg q)$ since Π is closed under groundings. Moreover, when I is witnessed true on a partial example $\rho^{(i)}$, we obtain a refutation of $\mathsf{ground}_{\hat{O}}(KB \wedge \neg q)|_{\rho^{(i)}}$, which is also in Π since it is closed under restrictions. Thus, since there such a set of objects O for which I is witnessed true with probability at least p, Hoeffding's inequality gives us that $\hat{p} \geq p - \gamma$ with probability at least $1 - \delta/2$.

Thus, if we take a union bound over the two cases, we find that both hold with probability at least $1 - \delta$. □

Chapter notes

First-order logic was developed independently by Frege (1879) and in a series of works by Peirce (1870; 1931; 1885), although a logic of relations was considered in a much more limited way earlier by De Morgan (1847). The formal semantics of the logic that we now use are due to Tarski (1956).

Compactness was first established by Gödel (1930). The language proper+ was first proposed by Lakemeyer and Levesque (2002), building on an earlier language of "proper" knowledge bases proposed by Levesque (1998). The version of the grounding trick we use here was studied by Belle (2017), building on the above.

Church's Theorem is due to Church (1936), although as mentioned, Turing's proof of the theorem (Turing, 1936) introduced the Turing machine formalism and proved it equivalent in expressive power to Church's family of computable functions. Skolemization was introduced by Skolem (1928).

The first algorithm for unification, and for lifted resolution more generally, was proposed by Robinson (1965), although Robinson's original algorithm did not run in polynomial time. The approach we present here, based on constructing a unification relation, was originally proposed by Huet (1976). The problem of constructing the equivalence relation was first considered by Arden, Galler, and Graham (1961), and most subsequent algorithms follow the structure of the algorithm of Galler and Fisher (1964). The rule for merging the smaller set into the larger is credited to M. D. McIlroy, while its analysis was essentially first given by Hopcroft and Ullman (1971). (Hopcroft and Ullman originally claimed a linear-time bound, but the argument was flawed.) The optimal running time bound for this problem was first obtained by Tarjan (1975), analyzing the "path compression" algorithm that was first proposed by A. Tritter. (Huet actually used this improved algorithm to obtain his "almost" linear time algorithm for unification.) The analysis was later simplified by Kozen (2012, Chapter 11) and is featured in standard algorithms texts, e.g., Cormen et al. (2022, Section 19.4). Actually, for unification, the algorithm can be modified to obtain a truly linear time bound rather than "almost" linear time, cf. Paterson and Wegman (1978) or Martelli and Montanari (1982). A nice overview of unification is provided by Baader and Snyder (2001).

The intractability of proper PAC-learning of relational formulas was originally established by Haussler (1989). Strictly speaking, however, Haussler established the intractability of learning a representation in which the existential quantifiers are required to bind to distinct objects; for this language, he proves that even learning in a universe containing only two objects is NP-hard. Proper+ is a strictly more expressive language; but, recall that in contrast to improper learning, the intractability of proper learning of one representation does not imply that learning a more expressive language is also intractable. Thus, strictly speaking, Theorem 9.25 is new, and indeed, it is open whether or not efficient learning of proper+ conjunctions is possible in universes with only two objects. In contrast to these negative results, Valiant (1985) proposed an approach to *improper* learning of small existentially quantified conjunctions in small domains using propositionalization: the resulting formula is a ground k-DNF if the conjunction has k literals. Beyond this, Valiant (2000b; 2006) considers learning knowledge bases of such formulas, again based on propositionalization. Although a ground formula is simpler in a sense, note that the ground formula may be much larger (polynomial in the domain size for fixed k, cf. Section 9.2) and moreover, the learned ground formulas may not possess the symmetries that enable lifted reasoning to be independent of the domain size.

Section 9.7 is adapted from Belle and Juba (2019). Again, strictly speaking, that work referred only to ground queries and used a logic of limited belief, proposed by Liu et al. (2004), for its tractable fragment. We have extended it slightly, to bounded-space lifted resolution for existentially quantified queries. This is indeed a strict strengthening, see Exercise 9.6. (The full characterization of space-bounded treelike resolution in terms of the failed literal rule is again due to Ansótegui et al. (2008).)

Exercises

9.1 *Setting up a relational domain.* Consider the following simple package-delivery domain. There are packages, locations, trucks, and time points. Packages may be at locations at particular times, trucks may be at locations at particular times, and a truck may carry a package at a particular time. Trucks can move between adjacent locations.
(a) Identify natural types of objects for this domain. For each type, give at least two example constants.
(b) Define a first-order vocabulary for the domain. Your vocabulary should include at least four predicate symbols, each with its arity and intended meaning. It may also include function symbols if useful.
(c) Write a first-order formula expressing that each package is at at most one location at any given time.
(d) Write a first-order formula expressing that if a truck carries a package at a time t, then the package and the truck are at the same location at time t.
(e) Write a first-order formula expressing one possible effect of a truck moving from one location to an adjacent location between time t and the next time point.
(f) Explain how your first-order representation avoids the need to introduce separate propositional attributes for every package-location-time combination.

9.2 *Representing natural language text as a relational scene.* We continue with the package-delivery domain, vocabulary, and rules (background knowledge) established in Exercise 9.1. For this exercise, you may introduce a small number of additional predicate symbols if they are needed to represent the passage below; for each new predicate, state its arity and intended meaning.
Consider the following report:

> Yesterday morning, a delivery truck was at the depot. One package was already inside the truck, and another package was waiting at the warehouse. The depot is directly connected to the warehouse, and the warehouse is directly connected to the customer desk. Yesterday afternoon, the same truck was at the warehouse. The package that had been inside the truck was signed for there. The report does not say whether the package waiting at the warehouse was ever loaded onto the truck.

(a) Introduce constants for the objects, locations, and time points mentioned in the report. State the intended referent of each constant. In particular, resolve the phrases "one package," "another package," "the same truck," "the package that had been inside the truck," and "there."
(b) Translate the explicitly stated information in the report into ground atoms using your relational vocabulary.

(c) List at least two ground atoms whose truth values are not determined by the report.
(d) Using only the rules from Exercise 9.1, infer any additional ground atoms that follow from the facts in part (b). Do not infer facts that require background assumptions you have not stated.
(e) Give two completions of the relational scene that agree with the report and with your rules, but differ on at least one fact that the report leaves open.

9.3 Writing a lifted resolution proof (computing MGUs by hand).
TO DO

9.4 Propositionalization example.
TO DO

9.5 *A family of efficiently automatable first-order proofs.* As with propositional resolution, we can identify families of first-order resolution proofs that are automatable in polynomial time. One such family is based on a generalization of bounded-width resolution (recall Exercise 5.9). For nonnegative integers j and k, a (j,k)-*clause* is an OR of at most k literals, each of which has an atom of arity at most j. Recalling that propositional variables may be represented as relations of arity 0, we note that $(0,k)$-clauses (with $j=0$) are precisely the propositional clauses of width k. We will refer to a set of universal closures of (j,k)-clauses as a (j,k)-*clausal knowledge base*. Consider the set of lifted resolution proofs in which each line is the universal closure of a (j,k)-clause. We will show, moreover, that this family is also suitable for reasoning with implicit learning via Theorem 9.32.
(a) Prove that for a given j and k, the set of lifted resolution proofs in which each line is the universal closure of a (j,k)-clause is closed under groundings.
(b) Prove that for a given j and k, there is an algorithm that takes as input a (j,k)-clausal knowledge base and decides whether or not it has a lifted resolution refutation in which each line is the universal closure of a (j,k)-clause, and runs in time polynomial in the size of the input knowledge base. (Thus, resolution with (j,k)-clauses is automatable in polynomial time.) Note that the exponent in the running time bound may depend on j and k, which are assumed to be fixed.

9.6 *Space-bounded treelike resolution simulates limited belief semantics.* An alternative to our approach to tractable reasoning by considering proofs of limited complexity is to limit the use of disjunctive information. Precisely, given a knowledge base of ground clauses, we will say that a clause c^* is *believed at level 0* if it is the weakening of a clause that can be refuted by unit propagation with the knowledge base. Then, for any $i>0$, the a clause c^* is *believed at level i* if it is either believed at level $i-1$, or if for some clause $c'=\ell_1\vee\cdots\vee\ell_k$ in the knowledge base, for each $j=1,\ldots,k$, when we add ℓ_j to the knowledge base, c^* is believed at level $i-1$ (with the expanded knowledge base). This approach may be extended to proper+ knowledge bases using the grounding trick. In this exercise, we will relate the approaches by showing that the clauses believed at level i can be proved by treelike resolution using clause space $i+2$.
(a) Let a knowledge base of ground clauses KB be given. For any $i>0$, the *level i failed literal* inference rule is: suppose that for some atom α and assignment b, there

is a refutation of $KB|_{\alpha \leftarrow b}$ using only the level $i-1$ failed literal rule (if $i > 1$) and unit propagation. Then we can infer the clauses of $KB|_{\alpha \leftarrow \neg b}$.
Prove that there is a treelike clause space $i+2$ resolution refutation of a knowledge base of ground clauses KB if and only if KB can be refuted using the level i failed literal rule. *(**Hint:** use the characterization of unit propagation as space-2 treelike resolution from Exercise 5.7.)*
(b) Now, using part (a), prove that whenever a clause c^* is believed at level i from some knowledge base of ground clauses KB, it can be proved by a treelike clause space $i+2$ resolution refutation.