Friendly Logics, Fall 2015, Lecture Notes 3

Val Tannen

1 Model checking for FOL

Given a vocabulary $\mathcal{V}$, the **model checking** [1] problem consists of deciding if $\mathcal{A} \models \sigma$ for an FOL $\mathcal{V}$-sentence σ and a finite $\mathcal{V}$-structure (model) $\mathcal{A}$.

Definition 1.1 *We shall be interested in the complexity of the model checking problem, in fact in three different situations:*

Combined complexity *The input is $(\mathcal{A}, \sigma)$, its size is $\mid \mathcal{A} \mid + \mid \sigma \mid$.*

Expression complexity *$\mathcal{A}$ is fixed, the input is just σ.*

Data complexity *σ is fixed, the input is just $\mathcal{A}$.*

For this problem to make sense the finite $\mathcal{V}$-structure $\mathcal{A}$ must admit a finite description. Clearly, the universe of $\mathcal{A}$ is required to be finite. Moreover, we can allow the vocabulary $\mathcal{V}$ to be infinite with the understanding that for the $\mathcal{A} \models \sigma$ model checking problem the description of $\mathcal{A}$ only includes representations for the (finitely many) symbols in $\mathcal{V}$ that actually occur in the sentence σ.

Since Turing machines take strings as inputs we must be clear about describing $\mathcal{A}$ as a string, and the resulting size $|\mathcal{A}|$. We fix an enumeration of the elements of the (non-empty) universe of $\mathcal{A}$: $a_1, a_2, \ldots a_n$ then we represent (encode) the model on Turing machine tapes with respect to this enumeration. One way to do this is to represent a_k by the number k, in *binary*, therefore by using $\log k$ bits. We then encode tuples and then relations using some separating symbols. The tuples in a relation will be ordered lexicographically, based on the ordering of domain elements given by the enumeration. For example, consider the enumerated domain a, b, c. Then $R = \{(c, b), (a, c), (a, b)\}$ $S = \{(c, a, c), (a, a, b)\}$ can be encoded by the string `R/0-1/0-10/10-1/S/0-0-1/10-0-10/`. The interpretation of function symbols of arity m is represented as $m+1$-relations in which the tuples group the arguments and the corresponding result of the function. Another possible encoding observes that the enumeration of the elements of the universe induces, lexicographically, an enumeration of the, say, r-tuples, and then relations of arity r can be encoded as strings of bits of length n^r and functions of arity r can be encoded as strings of bits of length n^{r+1}. In any case, we make the size of the universe, in *unary* (to avoid pathologies) part of the representation. In both representations that we described $|\mathcal{A}|$ is $O(s\, n^r)$ where n is the size of the universe, s is the number of vocabulary symbols that occur in σ and r is the "maximum arity" of these symbols, i.e., the maximum among

[1] This name comes from automated verification. In fact, model checking for many verification logics is a particular case of FOL model checking.

relation arities, function arities +1, and 1 (if the sentence has no symbols). Note that for data complexity the sentence is fixed and therefore $|A|$ is polynomial in n.

We begin with with the quantifier-free case and it will be useful to consider a slightly more general problem, involving formulas and valuations rather than just sentences: decide if $\mathcal{A}, v \models \psi$ where ψ is a quantifier-free formula and v is a valuation defined for a set of variables that includes the free variables of ψ. In this case, we consider v, together with ψ, to be the input for expression complexity.

Theorem 1.1 *The data complexity of quantifier-free formula checking is in LOGSPACE. The expression and combined complexities of the same problem are in PTIME.*

Proof Recall the fixed enumeration of the universe elements and observe that we can lookup information in the description of $\mathcal{A}$ using pointers of size $\log |\mathcal{A}|$. This is polynomial for expression and combined complexity and logarithmic for data complexity.

The procedure uses these pointers to evaluate the functional terms in the formula, determining the truth value of the atoms, and evaluating the resulting boolean expression. Term evaluation is done akin to arithmetic expression evaluation, using a stack, but we make sure it is a stack of pointers, not a stack of values. This gives us the PTIME expression and combined complexities. For data complexity observe that the stack size depends on the sentence but not on the model. This gives the LOGSPACE bound on data complexity. □

Remark It has been shown that boolean expressions can be evaluated in LOGSPACE. (Indeed, to evaluate, e.g., $a \vee b$ we do not need to record both the value of a and b. We would not be evaluating b at all if a resulted in true.) Therefore, in the absence of function symbols, for a *fixed* vocabulary consisting only of relation symbols and constants, the expression and combined complexity of quantifier-free model checking are also in LOGSPACE [2]

Theorem 1.2 *The data complexity of FOL model checking is in LOGSPACE. The expression and combined complexities of FOL model checking are PSPACE-complete.*

Proof First we put the sentence in prenex form [3]. Let $Q_1 x_1 \cdots Q_k x_k \ \psi$ be the result.

We evaluate this sentence in $\mathcal{A}$ using k nested loops iterating each of $x_1, \ldots, x_k$ through all the elements of the universe of $\mathcal{A}$. If x_i is universally quantified then the loop corresponding to x_i computes a big conjunction (disjunction if existentially quantified). In the innermost loop we evaluate $\mathcal{A}, v \models \psi$ where v is the valuation recording the current values of $x_1, \ldots, x_k$. If m is the number of elements of $\mathcal{A}$ then the time complexity of doing all this is

$$O(m^k \operatorname{poly}(|\mathcal{A}|, |\psi|))$$

where "poly" is a two-variable polynomial. For the space complexity, note that we can keep track of the current v by using k pointers of size $\log m$. Thus the space complexity adds $O(k \log m)$ to

[2] However, it seems that we cannot evaluate functional terms in LOGSPACE. (Think what goes wrong with the stack-based procedure.) It is known that there exist context-free languages that are NLOGSPACE-complete. I suspect, but I don't know, that such languages can be reduced to the problem of functional term evaluation.

[3] I don't know if this can be done in LOGSPACE but for data complexity the sentence is fixed. It certainly can be done in low degree PTIME hence PSPACE.

the space complexity of the quantifier-free case. This gives LOGSPACE for data complexity and PSPACE for expression and combined complexities.

To show PSPACE-completeness we reduce from TQBF, the problem of checking the truth of a (fully) quantified boolean formula. This is essentially the same reduction performed in the proof of Corollary 2.3 in Lecture Notes 2.

Given a fully (no free propositional variables) quantified boolean formula $\gamma \stackrel{\text{def}}{=} Q_1 p_1 \cdots Q_k p_k \ \beta$ we construct the FO sentence $\sigma \stackrel{\text{def}}{=} Q_1 x_1 \cdots Q_k x_k \, \overline{\beta}$ over a vocabulary with one unary predicate symbol R where $\overline{\beta}$ is obtained by replacing each p_i with $R(x_i)$. Clearly γ is true iff $\mathcal{B} \models \sigma$ where $\mathcal{B}$ is defined in the proof of Corollary **??**. Since $\mathcal{B}$ is a fixed model, this gives a lower bound for both combined complexity and expression complexity. □

2 Database dependencies and the BSR class

Definition 2.1 *A **relational** vocabulary is an FO vocabulary* with *relational symbols of arity* ≥ 1, with *constants (i.e., function symbols of arity 0),* with *equality, but* without *function symbols of arity* ≥ 1.

Integrity constraints in relational databases can be expressed in FOL over relational vocabularies:

Equality-generating dependencies (egd): key constraints and other functional dependencies. Example: $\forall x, y_1, y_2 \, R(x, y_1) \wedge R(x, y_2) \Rightarrow y_1 = y_2$.

Tuple-generating depedencies (tgd): foreign key constraints and other inclusion dependencies, multivalued dependencies, join dependencies, etc. Examples: $\forall x, y \ R(x, y) \Rightarrow \exists z \, S(y, z)$, $\forall x, y, z \, R(x, y) \wedge S(y, z) \Rightarrow T(x, y, z)$.

Database people study the *implication problem* for such depedencies, namely the decidability and complexity of $\Delta \models_{fin} \delta$ where Δ is a finite set of dependencies, δ is a single dependency and where the logical consequence is over all finite models (because databases are finite, you see).

Implication is undecidable in general but there are important decidable cases. A tgd is *full* if there are no existential quantifiers in the conclusion of the implication. Inclusion dependencies are not full but functional and multivalued/join dependencies are. In particular the latter dependencies are expressed by universal sentences over relational vocabulary. Thus, the decidability (but not the exact complexity, because of the special form of the quantifier-free portion) of the implication problem for egds plus full tgds follows from the following.

Theorem 2.1 *The Bernays-Schönfinkel-Ramsey (BSR) class of sentences, i.e., sentences over a relational vocabulary with quantifier prefix* $\exists^* \forall^*$*, has the small model property. Specifically, any satisfiable sentence in this class is satisfiable in a model with at most* $\max(1, m)$ *elements, where* m *is the number of existential quantifiers in the sentence.*

Before we prove the theorem we explain how this applies to the implication problem for certain database dependencies, and an even more general class of sentences.

Corollary 2.2 *Let's define (for local purposes only) a* generalized full dependency (gfd) *to be a universal sentence over a relational vocabulary. It is decidable whether* $\Delta \models_{fin} \delta$ *where* Δ *is a finite set of gfds,* δ *is a single gfd and where the logical consequence is over all finite models.*

Proof Because Δ is finite, $\Delta \models_{fin} \delta$ is equivalent to the finite validity of a sentence of the form $\sigma \Rightarrow \delta$ where σ is a gfd hence the finite validity of a sentence $\forall \overline{y}\, \exists \overline{x}\, \varphi \Rightarrow \psi$, ($\varphi, \psi$ quantifier-free), hence the finite unsatisfiability of $\exists \overline{y}\, \forall \overline{x}\, \varphi \wedge \neg\psi$. This last is decidable by Theorem 2.1. (Similarly, logical consequence over all models is also decidable.) □

Proof of Theorem 2.1. We further assume, w.l.o.g., that the relational vocabulary does not contain constants. Indeed, the constants can be eliminated in favor of additional existentially quantified variables: $\sigma(c)$ is satisfiable iff $\exists x\, \sigma(x) \wedge x = c$ is satisfiable iff $\exists x\, \sigma(x)$ is satisfiable. (This also explains why constant symbols are not necessary in the classification presented by the Börger-Grädel-Gurevich monograph.)

In our proof we will make use of the inverse transformation, replacing existentially quantified variables with constants. Essentially the same argument straightforwardly establishes the following:

Exercise 2.1 *Let* σ *be a sentence of the form* $\exists x_1, \ldots, x_m\, \varphi(x_1, \ldots, x_m)$ *over an FO vocabulary* $\mathcal{V}$ *(where* φ *is* not necessarily *quantifier-free). Let* $c_1, \ldots, c_m$ *be fresh constant symbols and let* $\overline{\mathcal{V}} \stackrel{\text{def}}{=} \mathcal{V} \cup \{c_1, \ldots, c_m\}$. *Then, the* $\mathcal{V}$*-sentence* σ *is satisfied in a model* $\mathcal{A}$ *iff the* $\overline{\mathcal{V}}$*-sentence* $\varphi(c_1, \ldots, c_m)$ *is satisfied in some "expansion" of* $\mathcal{A}$ *to a* $\overline{\mathcal{V}}$*-model obtained by keeping the same universe and the same interpretation of the symbols in* $\mathcal{V}$ *and adding interpretations for* $c_1, \ldots, c_m$. *(We call this a* $c_1, \ldots, c_m$*-expansion of* $\mathcal{A}$*).*

Thus, a BSR sentence $\exists x_1, \ldots, x_m\, \forall \overline{y}\, \varphi(x_1, \ldots, x_m, \overline{y})$ is satisfied in some model $\mathcal{A}$ iff $\forall \overline{y}\, \varphi(c_1, \ldots, c_m, \overline{y})$ is satisfied in some $c_1, \ldots, c_m$-expansion of $\mathcal{A}$. Let $a_1, \ldots, a_m$ (not necessarily distinct) be the interpretations of $c_1, \ldots, c_m$ in this expansion. Consider the model, let's call it $\mathcal{B}$, with universe $\{a_1, \ldots, a_m\}$ (eliminate duplicates) in which every symbol in $\mathcal{V}$ is interpreted by restriction to this universe. (**Warning:** this works only because $\mathcal{V}$ is a relational vocabulary without constants. Indeed, the subset $\{a_1, \ldots, a_m\}$ of $\mathcal{A}$ may not be "closed" under functions, nullary or otherwise.) Note that the interpretation of the expansion symbols, $c_1, \ldots, c_m$ also belongs to the universe of $\mathcal{B}$.

Now our result follows from the following fact of larger utility:

Exercise 2.2 *This holds for arbitrary vocabularies (so bring the function symbols back in!). A* ***submodel (substructure)*** *of a model* $\mathcal{A}$ *is a model whose universe is a subset of the universe of* $\mathcal{A}$ *that is* closed *under the interpretation of the function symbols (including constants) and thus allows the interpretation of all symbols (not just relation symbols) by restriction. Prove that if a* universal *sentence is true in* $\mathcal{A}$ *then it is true in any of its submodels. Explain why this fails for existential sentences.*

Therefore the universal sentence $\forall \overline{y}\, \varphi(c_1, \ldots, c_m, \overline{y})$ is satisfied in the $c_1, \ldots, c_m$-expansion of $\mathcal{B}$ and it follows that $\exists x_1, \ldots, x_m\, \forall \overline{y}\, \varphi(x_1, \ldots, x_m, \overline{y})$ is satisfied in $\mathcal{B}$. Finally, we note that $\mathcal{B}$ has at most m elements.

What if $m = 0$? Then the definition of $\mathcal{B}$ above is wrong because models are not allowed to have empty universe. However, as the following shows, we can always find a model with one element.

Exercise 2.3 *If a* universal *sentence over a relational vocabulary with* at most one constant *is satisfiable then it is satisfiable in a model with one element.*

Corollary 2.3 *The satisfiability of Bernays-Schönfinkel-Ramsey sentences is decidable in NEXPTIME (in fact it is NEXPTIME-complete but we skip the proof of that).*

Proof Given a BSR sentence $\sigma \equiv \exists x_1, \ldots, x_m \, \forall \overline{y} \, \varphi(x_1, \ldots, x_m, \overline{y})$ let $\mathcal{V}$ be the set of relations symbols that *actually appear* in σ. Note that $\mathcal{V}$ is finite so a finite model over $\mathcal{V}$ has a finite description. The algorithm starts by guessing a $\mathcal{V}$-model $\mathcal{A}$ whose universe has size at most $n = \max(1, m)$ and further guessing an interpretation of the constants $c_1, \ldots, c_m$ in $\mathcal{A}$. If r is the maximum arity of the relation symbols in $\mathcal{V}$ then $\mathcal{A}$ has a representation of size $O(n^r)$. This is polynomial in n but, unfortunately, exponential in r. And r can be as big as the size of φ (give or take a couple of symbols). Therefore, in the worst case the guess is of size exponential in $|\sigma|$.

We now have to check whether $\forall \overline{y} \, \varphi(c_1, \ldots, c_n, \overline{y})$ is true in $\mathcal{A}$. We can see directly that this can be done in EXPTIME by cycling through all the possible valuations of in the model's universe (there are exponentially many such valuations) and for each of them check whether the quantifier-free formula $\varphi(c_1, \ldots, c_n, \overline{b})$ is true. By Theorem 1.1 (for combined complexity) this can be done in PTIME$[|\varphi|, |\mathcal{A}|]$, thus in EXPTIME in $|\sigma|$. □

If you look carefully at the bounds given by the argument above you can see it proves a NTIME$[|\sigma|^{O(|\sigma|)}]$, thus an NTIME$[2^{O(|\sigma| \log |\sigma|)}]$ bound. This is still NEXPTIME of course but it turns out that a more careful analysis gives an NTIME$[2^{O(|\sigma|)}]$ bound.

Note also that these bounds can be improved when the maximum arity of the relation symbols is a constant. This is an eminently reasonable assumption in databases (where we say that the "schema" is fixed). If this is the case then the model $\mathcal{A}$ in the proof has a representation that is of size polynomial in n, hence in $|\sigma|$. Under this assumption, to decide the satisfiability of a BSR sentence we just have to guess a model with a polynomial-size representation and then by Theorem 1.2 (for combined complexity) we can check in PSPACE if the sentence $\forall \overline{y} \, \varphi(c_1, \ldots, c_n, \overline{y})$ is true in that model. This gives an NPSPACE algorithm, but NPSPACE=PSPACE. Is this the best we can do? That is, is the satisfiability of BSR sentences with relation symbols of bounded arity PSPACE-complete? The quantifier prefix suggests otherwise. Indeed:

Theorem 2.4 *Let r be an arbitrary but fixed natural number and let BSR_r be the class of BSR sentences in which all relation symbols have arity $\leq r$. Then, deciding the satisfiability of BSR_r sentences is Σ_2^p-complete.*

Proof To see membership in Σ_2^p we repeat the proof of Corollary 2.3 observing however that now the model that we guess has size $O(n^r)$ and that checking truth in the model was already in coNP, because *each* of the valuations $\overline{y} \longmapsto \overline{b}$ was of polysize.

To show Σ_2^p-hardness we reduce from the Σ_2SAT problem (check the truth of (closed) quantified boolean formulas of the form $\exists p_1, \ldots, p_m \, \forall q_1, \ldots, q_n \; \varphi(p_1, \ldots, p_m, q_1, \ldots, q_n)$ in the same way in which we reduced from TQBF in the proof of Theorem 1.2. □

3 Essentially finite classes of sentences

You might have noticed the sentences in the decidable classes have either an unbounded number of quantifiers or an unbounded number of relation symbols. Moreover, as we have seen, assuming an unbounded number of constants leads us to sentences with unboundedly many existential quantifiers. But what is the status of, say, sentences with quantifier prefix, say, $\forall^2\exists^{13}$, with a binary relation symbol, with, say, 17 unary relation symbols, and with equality? These sentences do not belong to the maximal decidable Gödel/Kalmár/Schütte class, because of equality and they do not belong to one of the minimal undecidable (Goldfarb) classes.

It turns out that if we put finite bounds on the parameters like these, we get classes of sentences for which the satisfiability problem is described as "trivial". But maybe not obviously trivial :) and certainly not trivial in a practical sense, as we shall see.

Definition 3.1 *A class of sentences is* ***essentially finite*** *if it is defined by a fixed finite quantifier prefix and a fixed finite relational vocabulary (no function symbols of arity* ≥ 1*).*

Proposition 3.1 *Let* $\mathcal{C}$ *be an essentially finite class of sentences. Then, there exists a* finite *set of sentences* $\mathcal{F} \subseteq \mathcal{C}$ *and a total computable function* $f : \mathcal{C} \to \mathcal{F}$ *such that* σ *is equivalent to* $f(\sigma)$*.*

Proof Fix an essentially finite class of sentences $\mathcal{C}$. Suppose the prefix of the sentences in $\mathcal{C}$ has n quantifiers, then fix n variable names $x_1, \ldots, x_n$. Since the vocabulary is relational and finite there are only finitely many atomic formulae that can be build from it and the variables $x_1, \ldots, x_n$. Out of these atomic formulas we can build only finitely many quantifier-free formulas in *irredundant CNF*. This gives our finite set of sentences $\mathcal{F}$. □

Corollary 3.2 *The satisfiability of sentences in an essentially finite class can be decided in LOGSPACE.*

Proof The decidability is immediate because $\mathcal{F}$ is finite. Indeed, consider a table in which we associate with each sentence in $\mathcal{F}$ the information whether it is satisfiable or not. This finite table exists and therefore can be hard-coded into the decision procedure. (**Warning:** it is *not* computed by the decision procedure, and, in fact, it cannot be, as it this would contradict the undecidability of satisfiability for some of the minimal reduction classes.) Then, given $\sigma \in \mathcal{C}$ compute $f(\sigma) \in \mathcal{F}$ and use it to look up the status of its satisfiability in the table.

The membership in LOGSPACE is really a corollary to the *proof* of the proposition and requires the observation that the ordering $x_1, \ldots, x_n$ of the variables and a total ordering of the vocabulary induces (lexicographically) a total ordering on the atomic formulas, hence a total ordering of CNF clauses of these, hence a total ordering of the sentences in $\mathcal{F}$. This total ordering can be used to look up the satisfiability status for each sentence. We also need to verify that the index of the sentence in the ordering, as well as the lookup can be computed in LOGSPACE. □.

4 Horn and Krom logics (unfinished)

Theorem 4.1 *HORNSAT is in PTIME. (In fact, it is PTIME-complete under LOGSPACE reductions but we skip the proof of that.)*

Theorem 4.2

(a) The satisfiability of BSR-Horn sentences is in EXPTIME. (In fact, it is EXPTIME-complete but we skip the proof of that.)

(b) The satisfiability of BSR sentences without constants and with prefix $\exists\forall^*$ (just one existential) is NP-complete.

(c) The satisfiability of BSR-Horn sentences without constants and with prefix $\exists\forall^*$ (just one existential) is PTIME-complete.

$\exists^2\forall^*$-KROM-HORN is PSPACE-complete

$\forall\exists^*\forall$-KROM-HORN is a reduction class

$\forall\exists\forall$-KROM sat is decidable and NLOGSPACE-complete

$\exists^*\forall^*\exists^*$-KROM sat is decidable and EXPTIME-complete