Monadic Predicate Logic is Decidable

Adam Wyner
CS3518, Spring 2017
University of Aberdeen

Issue

Is Monadic First Order Predicate Logic satisfiability decidable?

In other words, is there an algorithm which decides whether a monadic FOPL formula is satisfied or not.

Why is satisfaction of FOPL important?

- Large scale reasoning on the Semantic Web
- Whatever logic is used to reason, it ought to be decidable
- Description logic fragments – various logics given parts
- Examination of which parts together are decidable fragments of logic. What is sufficient for certain reasoning tasks?
- Giving a flavour of a significant and very useful issue
- Show that monadic FOPL is decidable. Mention that dyadic FOPL is undecidable.

Reminders FOPL

- predicates: P, Q, R
- quantifiers: A (∀), E (∃)
- variables and constants: x, y, z and a, b, c
- logical connectives: & (∧), - (¬), v
- sentences (aka logical formulae, statements)
- Equality statements are atomic formulas: x=y, a=b, x=a, etc
- “sentence” = formula with no free variables

Reminders FOPL

- satisfaction: a model in which a formula is true or false. Must assign values to variables.
- entailment: P entails Q iff every model in which P is true is model in which Q is true.
- Can we decide if there is a model? That is, can we decide (i.e. accept, reject, and no looping) if there an assignment of values to variables and with respect to predicates under which the formula is true or false?

What is the problem?

- If we have a (potentially infinite) domain in which we create a model, can we decide if there is a model?

- Approach:
  - what is the maximal number of assignments of values to variables and with respect to predicates that are needed to have a 'complete' truth table for the formula?
  - how can we 'reduce' the number of assignments with respect to an infinite domain to a finite domain?
  - how can we 'break up' the complex formula to make a finite truth table (with respect to variable assignments and predicates)? Make FOPL more like PL.

Monadic First-Order Predicate Logic (FOPL)

- The fragment of Predicate logic that uses no predicates with more than 1 argument
- In: Ex F(x) & Ey -F(y)
  AxEy -(x=y) (equality statements permitted!)
  G(a) v G(b) , etc.
- Out: AxEy (R(x,y))
  R(a,b,b) v Ex F(x)

  (because they use a dyadic/triadic/etc predicates)

- Showing that monadic FOPL is decidable. Mention that dyadic/ triadic is not.

Some reasoning tasks

- Given sentences φ and ψ, does ψ follow from φ?
  - Does φ have ψ as a logical consequence?”
  - More precisely: is it true that for all models M,

    if M |= φ then M |= ψ?
- Given a sentence φ, is φ satisfiable? We mean:
  - Is there a model that M such that M|= φ?
  - e.g., is Ex F(x) & Ex -F(x) satisfiable? Are there values assigned to variables which make the formula true?
- Analogous for formulas with free variables

Claim

Monadic First Order Predicate Logic satisfiability is decidable.

In other words, there is an algorithm which decides whether a monadic FOPL formula is satisfied or not.

Key theorem (Lőwenheim-Skolem 1915)

- If S is a monadic sentence that has a model, then S is true in some model whose domain consist of at most $2^k * r$ members, where k is the number of predicate letters in S and r is the number of (bound) variables in S.
- Consider S =
  - (Ex F(x) & Ex G(x)) & -(Ex (F(x)&G(x)))
  - Here k=2 (F,G), r=3 (variables/quantifiers), so $2^k * r = 12$
- That is, if S has a model, then that model has a domain of at most 12 members. These are the different possible ways that values need be assigned to variables to make the formula true. Keep alphabetic variation in mind above.

Proof Parts

- Part 1, proof: The Key Theorem
- Part 2, proof: it follows that monadic FOPL is decidable

Proof of Part 1 (= Key Theorem)

- Let S be a sentence of monadic FOPL. Its predicates are $P_1,..,P_k$
- Let M |= S, and let D be the domain of M. D may be infinite.
- Let the signature of d in D (henceforth sig(d)) be the sequence $<j_1, .. ,j_k>$, where $j_i = 1$ if M specifies that P_i is true in D and $j_i = 0$ otherwise
  - sig(d) tells us which predicates in S are true in D
  - given S, there are exactly 2^k different possible signatures
- Similar to the truth tables, where we have all the alternative possible assignments of truth values to predicates. Here we have all the possible alternative values for variables and with respect to the predicates.

Proof of Part 1 (cont'd)

- We call d and d' similar if sig(d) = sig(d')
  - This means that d and d' happen to share all their properties P_1, .. ,P_k
- "similar" is an equivalence relation, so each d in D belongs to an equivalence class of similar domain elements
  - Each equivalence class is a subset of D
  - There are at most 2^k equivalence classes

Towards a smaller, finite model M'

- Construct a subset $E \subseteq D$ as follows:
- Choose r elements from each equivalence class
  - If a class has fewer than r elements then choose them all
- E cannot have more than $2^k * r$ elements (r for each equivalence class). This makes E finite (important)
- Define: M' is the restriction of M to E; that is, M' is just like M, but defined for elements of E only.
- To be proven: $M' \models S$

A useful concept: match

- Informally: Two sequences of elements of E that are of the same length match if their elements are similar and differences within each sequence are “respected” in the other.
- Formally: $c_1,..,c_n$ matches $d_1,..,d_n$ iff
  1. c_i is similar to d_i (for every $1 <= i <= n$), e.g. c_i is same thing as d_i
  2. $c_i = c_j$ iff $d_i = d_j$ (for every $1 <= i,j <= n$)

Example

- These 2 sequences of domain objects do not match:
  - $c_1,..,c_n$ = a,b,a
  - $d_1,..,d_n$ = a,b,c
- c_1 and d_1 are similar, so clause 1 is fulfilled there; but clause 2 is not (because $c_1 = c_3$ but d_1 is not equal to d_3)
- Reason behind clause 2: equality statements in FOPL (e.g. in the sentence AxEy -(x = y)

Another useful concept

- A formula φ containing at most the free variables $x_1,..,x_n$ is satisfied by elements $d_1,..,d_n$ in a model M iff

$$M \models \varphi\ (x_1 := d_1, \ldots, x_n := d_n)$$

- $x_1 := d_1$ means substitute d_1 for x_1 in φ
- This is a simple extension of the idea of satisfiability

Lemma

Let

- $G(x_1,..,x_n)$ be any subformula of S, containing at most the free variables $x_1,..,x_n$
- $d_1,..,d_n$ a sequence of elements of D (the original domain)
- $e_1,..,e_n$ a sequence of elements of E (dom constructed above)
- $d_1,..,d_n$ matches sequence $e_1,..,e_n$

Then

$G(x_1,..,x_n)$ is satisfied by $d_1,..,d_n$ in M iff

$G(x_1,..,x_n)$ is satisfied by $e_1,..,e_n$ in M'

$G(x_1,..,x_n)$ is satisfied in the smaller, finite model.

Why does this lemma hold? (informally)

- As far as the predicates $P_1,..,P_k$ occurring in S are concerned, each element d_i is just like e_i
  - Clause 1 of "match"
- The only other thing that can matter (because of equality statements!) is whether two elements in a given sequence are identical
  - Clause 2 of "match"

Sketch of a formal proof (by formula induction)

Base Cases: G is atomic. G is of the form $P_i(t)$ or of the form $t_1 = t_2$ (t, t_1, and t_2 are variables or constants)

1. Let G = $P_i(t)$. We need to prove:

   $P_i(t)$ is satisfied by d_1 in M iff

   $P_i(t)$ is satisfied by e_1 in M'

But d_1 and e_1 are similar, hence the same predicates hold true with respect to d_1 and e_1 (including the predicate P_i). This proves the first Base Case.

Sketch of proof by formula induction

Base Cases: G is atomic. G is of the form $P_i(t)$ or of the form $t_1 = t_2$.

2. Let G = $t_1 = t_2$. We need to prove

$t_1 = t_2$ is satisfied by d_1, d_2 in M iff

$t_1 = t_2$ is satisfied by e_1, e_2 in M'

But the sequences d_1 d_2 and e_1 e_2 match, hence $d_1 = d_2$ iff $e_1 = e_2$. This proves the second Base Case.

Sketch of proof by formula induction

Inductives Cases: [Proofs omitted] It suffices to address -, v, A, as the other connectives and E can be expressed in them.

(1) Assume the Lemma holds for φ. Prove that it holds for –φ.

(2) Assume the Lemma holds for ϕ and ψ. Prove that it holds for ϕ v ψ.

(3) Assume the Lemma holds for ϕ. Prove that it holds for Ax ϕ.

Proof (cont'd)

S is itself a subformula of S, hence it follows directly (with n = 0) from the Lemma that

S is true in M iff S is true in M'

Recall: M may be infinite, but M' is finite, with at most $2^k * r$ elements

Proof of Part 2

- Let S be a FOPL sentence
- Associate S with a quantifier-free formula S' such that S' is satisfiable iff S is satisfiable. (Next page)

If we manage to do this, then we deduce:

- The satisfiability of S can be decided using truth tables (since these suffice for deciding the satisfiability of S')
- Hence the satisfiability of S can be decided

Proof of Part 2

Making use of Part 1, associate with S a quantifier-free formula S' which is satisfiable iff S is satisfiable. As follows:

Inductively associate a quantifier-free H' with each subformula H of S, as follows:

- If H is atomic: H' = H (no change!)
- If H is a truth functional compound: H' = H
- If H = Ex F: H' = $F(a_1)$ v .. v $F(a_m)$ $m = 2^k * r$
- If H = Ax F: H' = $F(a_1)$ & .. & $F(a_m)$ $m = 2^k * r$

S itself is a subformula of S, so this constructs a quantifier-free S' as well. The construction guarantees: S' is satisfiable iff S is satisfiable

Example (using an arbitrary S)

Consider S =

(ExF(x) & ExG(x)) & -(Ex(F(x) & G(x)))

Here k = 2, r = 3, so $2^k * r = 12$. Does not matter the alphabetic variation of variables since all variables take all values from the model.

The following formula S' is constructed:

$F(a_1)$ v .. v $F(a_{12})$ & $G(a_1)$ v .. v $G(a_{12})$ &

- (($F(a_1)$ & $G(a_1)$) v .. v ($F(a_{12})$ & $G(a_{12})$))

Example

$$F(a_1) \vee .. \vee F(a_{12}) \;\&\; G(a_1) \vee .. \vee G(a_{12}) \;\&$$
$$\neg ((F(a_1) \;\&\; G(a_1)) \vee .. \vee (F(a_{12}) \;\&\; G(a_{12})))$$

Propositional formula with 24 atoms

Each can be True or False => truth table has 2^{24} rows (16,777,216). Finite! Each row is a signature sig(e). Try to find a row that is True. Example:

$F(a_1), F(a_2), .. F(a_{12}), G(a_1), G(a_2), G(a_3), .. G(a_{12})$

T F F F T F F

Example

$F(a_1), F(a_2), ..F(a_{12}), G(a_1), G(a_2), G(a_3), ..G(a_{12})$

T F F F T F F

We can read off from this a model with 12 elements that satisfies the formula. The same model must satisfy the original (quantified) formula S too.

Concluding

- The proof suggests an algorithm for deciding whether a formula is satisfiable
  - Satisfiable iff some row of the truth table is True
  - Not satisfiable iff no row of the truth table is True
- Also applicable to logical consequence
- Implementations exist
- 2^k implies Exponential in complexity (though faster methods exist)
- Decidability proofs often tell us something about the worst-case runtime of a program

Other FOPL fragments

- For every n, it is decidable whether a given formula of FOPL has a model of size m <= n. Not proven here
- However, dyadic FOPL is undecidable. The Key Theory does not apply.
- Example: the following FOPL sentence does not have a finite model (why?)

  Ex (x=x) & AxEy (x < y) &

  Axyz ((x < y & y < z) → x < z) &

  Ax -(x < x)

- Description logic fragments