Chapter 5
Quantifier-free theory of equality

In this chapter, we consider the problem of deciding the validity of sentences of the kind

$$\forall x_1, \ldots x_n.\ \varphi$$

where φ is quantifier free, and over an arbitrary signature.

Note that our convention for validity for formulas (with free variables) is that a formula is valid if it is true in every structure and every valuation of variables over the structure. Hence validity of the sentence $\forall x_1, \ldots, x_n.\ \varphi$ is the same as the validity of the formula φ. Since this formula has no quantifiers, this fragment is referred to as the *quantifier-free fragment*. Furthermore, since the functions and relations are not restricted in any way, the only relation that has a fixed interpretation is *equality* (interpretation of = symbol). Hence this theory is called the quantifier-free theory of equality.

We saw in the last chapter that general FOL validity is undecidable. However, the proof of that undecidability crucially used existential quantification (in particular, $\forall\exists$ quantification), and hence that proof does not apply for this fragment. We will show in fact that validity for this fragment is decidable.

Let us consider the dual problem of satisfiability. Given a quantifier-free formula $\varphi(\mathbf{x})$, is there a model and interpretation of $\mathbf{x}$ that satisfies φ? Note that this is the same decision problem as validity, as φ is valid iff $\neg\varphi$ is not satisfiable. Hence a decision procedure for satisfiability gives a decision procedure for validity as well.

5.1 Decidability using Bounded Models

We first make the simple observation that a quantifier-free formula φ is satisfiable iff it is satisfied in a finite model, in fact the finite model needs to be only of size n, where n is the number of *terms* mentioned in the formula.

Let $\varphi(\mathbf{x})$ be a satisfiable quantifier-free formula. Let M be a (potentially infinite) model and s be an interpretation of the variables $\mathbf{x}$, under which $\varphi(\mathbf{x})$ is true, i.e., $M \models \varphi(\mathbf{x})$.

Now let us construct a *finite* model M' from M that also satisfies φ. Let T be the terms mentioned in φ; T is finite, and let's say $|T| = n$. Without loss of generality, assume $n > 0$ (if not, add a conjunction $x = x$ to the formula to introduce a term x). Let the universe in M be U_M. Now, under the interpretation s, each term $t \in T$ evaluates to an element $[t]_s \in U_M$.

Let us define the model M' as follows: the universe U' of M' is $\{[t]_s \mid t \in T\}$ (i.e., the finite subset of elements that terms evaluate to. For every relation symbol R, $R(\overline{u})$ is true in M' iff $R(\mathbf{u}$ is true in M. Functions are a bit more complex to define. Let us fix an arbitrary element e in U'. Define $f^{M'}(\overline{u})$ to be $f^M(\overline{u})$ if $f^M(\overline{u}) \in U'$, and e otherwise.

The above construction basically takes the sub-universe corresponding to the terms mentioned in φ, restricts the relations and functions to this subset, and when the function maps a vector of elements to an element outside this subset, map it to e instead.

Our claim now is that M' with the same interpretation s will also satisfy φ. First, we prove that every term in φ maps to the *same* element in M' as it does in M. And hence any atomic formula $R(\overrightarrow{t})$ as well as any atomic formula $t = t'$ will evaluate the same way under both models. It follows that the formula φ will evaluate to *true*.

The above argument shows that satisfiability for quantifier-free formulas is decidable. We can just enumerate all possible models of size n, where n is the set of terms mentioned in φ, choosing all possible interpretations for functions, relations, constants, and variables, and check if any of them satisfy φ.

If the signature (including arities) are *fixed*, and validity of formulas only over the fixed signature is to be decided, we can even do this in NP. We can just non-deterministically guess the model of size n and the interpretation, and check if the formula is satisfied. Since checking whether a formula holds in the model can be done in polynomial time, this gives an NP algorithm.

It is also easy to see that the problem is NP-hard as well, as it essentially includes Boolean logic. Reduction from SAT: Given a propositional formula α, introduce a new variable x, and replace each proposition occurrence p in φ with the atomic formula $(p{=}x)$. This formula is satisfiable iff α is satisfiable. Hence satisfiability of quantifier-free formulas is NP-complete.

5.2 An Algorithm for Conjunctive Formulas

The above proof that checking satisfiability is NP-hard is a bit unsatisfactory, as it shows it is hard because of the Boolean logic within in. What is the precise complexity of reasoning with equality itself?

We can ask the above question more precisely by asking what is the complexity of deciding *conjunctive* formulas. Can they be decided in polynomial time?

It turns out that the problem is indeed solvable in polynomial time. We will consider an algorithm in this section that clearly works in polynomial time if the arity of functions/relations are fixed (i.e., bounded by k). However, it turns out that one can implement this algorithm using clever data-structures to get a polynomial time algorithm even when arities are not fixed (see Calculus of Computation, Chapter 9, Section 9.3, for example).

To simplify the algorithm, let us first get rid of predicates (other than equality) from the formula. This can be done easily. Let us fix a constant $\mathbb{T}$. We can model a predicate $p \subseteq U^n$ as a *function* $f_p : U^n \rightarrow U \cup \{T\}$, with the understanding that $p(\overline{u})$ is true iff $f_p(\overline{u}) = \mathbb{T}$. Hence, given a formula φ with functions and relations, we can replace each occurrence of $p(\bar{t})$ with $f_p(\bar{t}) = \mathbb{T}$, to get a formula φ' such that φ is satisfiable iff φ' is satisfiable. (We leave this as an exercise.)

Let φ be a conjunctive formula that is quantifier-free, that uses function symbols and =, but no relation symbols. Then $\varphi \equiv \alpha_1 \wedge \alpha_2 \wedge \ldots \alpha_n$, where each α_i is a literal of the form $t = t'$ or $\neg(t = t')$.

Since the formula is conjunctive, let us look upon the formula as a set of conjuncts: $\{\alpha_1, \ldots, \alpha_n\}$. In fact, let us divide these formula into two sets $E \cup D$, where E consists of equalities of the form $t = t'$ and D consists of disequalities of the form $\neg(t = t')$. Let us look upon the elements of E and D as pairs of terms of the form (t, t').

Our key idea is now to build a model, if the formula is satisfiable, just using the terms T mentioned in the formula φ, which is finite and linear in $|\varphi|$. Furthermore, our idea is to find the *smallest* set of equality constraints that are imposed by the set of equalities in E. It turns out that such a smallest set always exists, and is called the *congruence closure of* E, denoted $CC(E)$, and is easy to compute in polynomial time. We then check whether any of the disequalities are violated in $CC(E)$. Then there is a violation iff the formula is not satisfiable.

The *congruence closure of* E, $CC(E)$ is the smallest set such that:

Includes E: For every $(t, t') \in E$, $(t, t') \in CC(E)$
Reflexive closure: For every $t \in T$, $(t, t) \in CC(E)$
Symmetric closure: For every $t, t' \in T$, if $(t, t'), (t', t) \in CC(E)$.
Transitive closure For every $t, t', t'' \in T$, if $(t, t') \in CC(E)$ and $(t', t'') \in CC(E)$, then $(t, t'') \in CC(E)$.
Congruence closure For every function symbol R of arity n, for every $f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n) \in T$, if $(t_1, t'_1), (t_2, t'_2), \ldots (t_n, t'_n) \in CC(E)$, then $(f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)) \in CC(E)$

Intuitively, $CC(E)$ is the reflexive, symmetric and transitive closure of E, and also congruence-closed, in the sense that if two tuples of terms are deemed equal by it, then the function expressions applied on those terms are also deemed equal.

It turns out that $CC(E)$ exists (i.e., a smallest set of such equalities exists). Here is a simple proof. Define

$$
\begin{aligned}
CC_0 =& E \\
CC_{i+1} =& CC_i \\
& \cup \ \{(t,t) \mid t \in T\} \\
& \cup \ \{(t,t') \mid (t',t) \in CC_i\} \\
& \cup \ \{(t,t') \mid \exists t'' \in T, (t,t''), (t'',t') \in CC_i\} \\
& \cup \ \{(f(t_1,\dots,t_n), f(t'_1,\dots,t'_n)) \mid (t_1,t'_1),\dots,(t_n,t'_n) \in CC_i, f(t_1,\dots,t_n), f(t'_1,\dots,t'_n) \in T\}
\end{aligned}
$$

Now, let $CC = \cup_{i\in\mathbb{N}} CC_i$. Then it is easy to prove that CC has the required properties and is the smallest set that has these properties (readers should prove this for themselves). Since T is finite, the above procedure in fact terminates, i.e., there will be an i, such that $CC_i = CC_{i+1}$, in which case we can stop, as the future sets will all be the same. It's easy to see that this is in fact a polynomial time algorithm, since CC_i monotonically increase and can have at most $|T|^2$ pairs. We will see later in this section a concrete algorithm that does this computation a bit better.

Now let M be any model that satisfies the equalities in E. Then it is clear that M will satisfy the equalities in $CC(E)$ as well, since what we demand of $CC(E)$ are properties satisfied by equality. Consequently, the equalities in $CC(E)$ are logically implied by the equalities in E.

Let us now prove:

Lemma 5.1 *There exists a model satisfying the equalities in E and the disequalities in D exist iff* $CC(E) \cap D = \emptyset$

Proof In the forward direction, assume M is a model satisfying the equalities in E and the disequalities in D. Then, as we argued above, the equalities in $CC(E)$ must be satisfied in M as well, as they are logically implied by the equalities in E. It follows that since every disequality in D is satisfied by M, $CC(E)$ and D cannot have a common pair of terms.

In the other direction, assume $CC(E)$ and D are disjoint. Let us define the equivalence relation $\sim$ over T defined by $CC(E)$ as $t \sim t'$ iff $(t,t') \in CC(E)$. It is easy to see that $\sim$ is indeed an equivalence relation.

Let us construct a model M, where the universe U of M is $T/\!\sim$, i.e., the universe is the set of equivalence classes of $\sim$. Interpret each constant symbol c occurring in φ as the equivalence class $[\![c]\!]$, and each variable x occurring in φ as the equivalence class $[\![x]\!]$. The interpretation for a function symbol f on a vector of elements $e_1, \dots e_n$ is defined as follows (for the definition below, fix a particular element e^* arbitrarily):

- If there are some terms $t_1 \in e_1, \dots t_n \in e_n$ such that $f(t_1,\dots,t_n)$ is a term in T, then map $f(e_1,\dots,e_n)$ to $[\![f(t_1,\dots t_n)]\!]$.
- Else, map $f(e_1,\dots,e_n)$ to e^*.

Note that the above model construction is well defined only because $CC(E)$ satisfies the congruence-closure condition. For example, if $t_1 \sim t_2$, then we would need $f(t_1) \sim f(t_2)$ in order for the definition of f to be well-defined.

It is now straightforward to argue, by induction on structure of terms, that for every term t, the term evaluates to the element $[t]$ in this model. Consequently all the equalities E are satisfied. Also, every disequality $(t, t') \in D$, we are guaranteed $[t] \neq [t']$, since $CC(E) \cap D = \emptyset$. Hence the formula holds in the model, and is hence satisfiable. $\square$

The above shows that in order to check whether a quantifier-free conjunctive formula φ is satisfiable, we just need to compute $CC(E)$ and check if $CC(E) \cap D = \emptyset$, where E and D are the equalities and disequalities occurring in φ.

5.2.1 Computing $CC(E)$

One simple method for computing the congruence closure, especially on paper manually, is to compute successive equivalence relation using its *equivalence* classes.

An equivalence relation $\sim$ over T can be represented as a set of sets that form a partition of T, i.e., as $\{E_1, \ldots E_k\}$ where each $E_i \subseteq T$, the sets are all disjoint, and their union is E_i.

Given such a representation of $\sim$, let us define a *Merge* operation on them: $Merge(\sim, E_i, E_j)$, where E_i, E_j are two equivalence classes of $\sim$ simply merges the two equivalence classes into one and results in a new equivalence relation. More precisely, if $\sim$ is $\{E_1, \ldots, E_k\}$, then $Merge(\sim, E_i, E_j)$ is $\sim'$ whose representation is $\{E_r \mid r \neq i, r \neq j\} \cup \{E_i \cup E_j\}$.

The algorithm for computing congruence closure is then as follows. :

- Initialize $\sim$ to $\{\{t\} \mid t \in T\}$. In other words, we start with each term in its own equivalence class.
- For every $(t, t') \in E$, merge $[t]$ and $[t]'$.
- Do the following till $\sim$ stabilizes:
  - If there are any terms $t_1, \ldots, t_n, t'_1, \ldots, t'_n \in T$ such that both $f(t_1, \ldots, t_n)$ and $f(t'_1, \ldots t'_n)$ are in T, and if $[f(t_1, \ldots, t_n)]$ is not equal to $f(t'_1, \ldots, t'_n)$, then merge them.
- Check whether there is any disequality $(t, t') \in D$ such that $[t] = [t']$; if there is, report formula is unsatisfiable; otherwise, report formula is satisfiable.

Let us illustrate through an example:

Example 5.2 This example is taken from the book Calculus of Computation.

We want to know whether the following formula is satisfiable:

$$f(a, b) = a \wedge f(f(a, b), b) \neq a$$

Equalities E are $\{(f(a, b), a)\}$.
Disequalities D are $\{(f(f(a, b), b), a)\}$
The set of terms T is $\{a, b, f(a, b), f(f(a, b), b)$.

We start with the equivalence relation:

$$\{\ \{a\}, \{b\}, \{f(a, b)\}, \{f(f(a, b), b)\}\ \}$$

Since $(f(a,b),a)$ are in E, we merge their equivalence classes to get:

$$\{\ \{a, f(a,b)\}, \{b\}, \{f(f(a,b),b)\}\ \}$$

Since a and $f(a,b)$ are in the same equivalence class, and since b and b are in the same equivalence class, we merge $f(a,b)$ and $f(f(a,b),b)$ to get:

$$\{\ \{a, f(a,b), f(f(a,b),b)\}, \{b\}\}\ \}$$

It is easy to verify that the equivalence classes have stabilized.

We can now check whether the disequalities are all satisfied. But since $f(f(a,b),b)$ and a are in the same equivalence class, we report that the formula is unsatisfiable.

Let us now illustrate an example where the formula is satisfiable, and also illustrate the model construction involved in the Lemma above.

Example 5.3 We want to know whether the following formula is satisfiable:

$$f(a) = b \wedge f(b) = a \wedge f(f(a)) = c \wedge \neg(f(a) = a)$$

The equalities are: $E = \{(f(a), b), (f(b), a), (f(f(a)), c)\}$.
The disequalities are: $D = \{(f(a), a)\}$
The terms are $T = \{a, b, c, f(a), f(b), f(f(a))$.

We start with the equivalence relation:

$$\{\ \{a\}, \{b\}, \{c\}, \{f(a)\}, \{f(b)), \{f(f(a))\}\ \}$$

Since $(f(a), b)$ is in E, we merge their equivalence classes to get:

$$\{\ \{a\}, \{b, f(a)\}, \{c\}, \{f(b)\}, \{f(f(a))\}\ \}$$

Since $(f(b), a)$ is in E, we merge their equivalence classes to get:

$$\{\ \{a, f(b)\}, \{b, f(a)\}, \{c\}, \{f(f(a))\}\ \}$$

Since $(f(f(a)), c)$ is in E, we merge their equivalence classes to get:

$$\{\ \{a, f(b)\}, \{b, f(a)\}, \{c, f(f(a))\}\ \}$$

Now, since b and $f(a)$ are in the same equivalence class, we must merge the equivalence classes of $f(b)$ and $f(f(a))$. We get:

$$\{\ \{a, f(b), c, f(f(a))\}, \{b, f(a)\}\ \}$$

The equivalence class has now stabilized. We note that there is only one disequality, $(f(a), a)$, and $f(a)$ and a are in different equivalence classes. So we report the formula to be satisfiable.

Let us examine now how the proof of the Lemma above constructs a model. We have two elements in our model, e_1 standing for the class $\{a, f(b), c, f(f(a))$ and e_2 for the class $\{b, f(a)\}$.

Since a is in e_1 and $f(a)$ is in e_2, we interpret that $f(e_1) = e_2$. Also, since $f(a)$ is in e_2 and $f(f(a))$ is in e_1, we interpret $f(e_2) = e_1$. The constants a and c are interpreted as e_1 (since they belong to e_1) and the constant b is interpreted as e_2 (as b belongs to e_2. This model satisfies the formula.

There are even faster ways to do congruence closure. Notice that the key operations above have to manipulate disjoint sets and support union (merge) and find (which equivalence class does an element belong to?). This turns out to be a well-studied data structure called *disjoint-set datastructure* (or *union-find datastructure* for which efficient algorithms are known. Note that the number of equivalence classes is n, where n is the number of terms in the formula, which is of course linear in the size of the formula. Now if the signature is finite and fixed, or if the signature had functions of fixed arity (the former implies the latter), then it is easy to see that the above algorithm can be implemented in polynomial time. If the maximum arity is k, then there are only n^k possible considerations of terms to consider for identifying candidates of equivalence classes to merge. If the signature consists of functions of arbitrary arity, it turns out that one can still effect a polynomial time algorithm, but we skip this here.

SMT solvers also implement fast algorithms for congruence closure. In particular, given a quantifier-free formula (not necessarily conjunctive), they look upon the formula as a Boolean formula over propositions (each proposition being an atomic formula), and call a SAT solver to find a satisfying valuation (if the SAT solver finds it unsatisfiable, then clearly the original formula is also unsatisfiable). The satisfying valuation can now be interpreted as a *conjunctive* set of equality and disequality constraints, which can then be checked for satisfiability using congruence-closure algorithms. If this conjunctive formula is unsatisfiable, it would return a *core* set of atomic formulas that already make it unsatisfiable, and the SAT engine will add that as a clause, and continue its search for valuations.

5.3 Axioms for The Theory of Equality

In this book/course, we will treat the equality symbol (=) as an interpreted relation throughout— it is interpreted as equality of elements in the universe. However, in this section, we are briefly going to suspend that in order to understand what properties equality really satisfies.

Let us fix a FO signature, and for clarity, let us not have = as a symbol, but instead have the symbol $\doteq$. If $\doteq$ was uninterpreted, what properties would we like it to satisfy?

Here are some properties (which we will call the *congruence axioms*) that equality clearly satisfies (we continue to write relations as $t \doteq t'$)), instead of $\doteq(t, t')$:

Reflexivity" $\forall x.\ x \doteq x$
Symmetry: $\forall x.\ (x \doteq y \Rightarrow y \doteq x)$
Transitivity: $\forall x, y, z.\ (x \doteq y \wedge y \doteq z) \Rightarrow x \doteq z$
Congruence wrt relations: For any relation R of arity n,

$$\forall x_1, \dots x_n, y_1, \dots, y_n.\ \left(\left(\bigwedge_{i \in [1,n]} x_i \doteq y_i\right) \Rightarrow R(x_1, \dots, x_n) \Leftrightarrow R(y_1, \dots, y_n)\right)$$

Congruence wrt functions: For any function f of arity n,

$$\forall x_1, \dots x_n, y_1, \dots, y_n.\ \left(\left(\bigwedge_{i \in [1,n]} x_i \doteq y_i\right) \Rightarrow f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n)\right)$$

First, notice that the above doesn't ensure that $\doteq$ will be interpreted as *true* equality on the model. For example, if there were two elements e_1 and e_2 such that all functions and relations behaved identically on them and no constant was interpreted as either of them, then we could relate them with $\doteq$ and satisfy all the properties above. In fact, FO with $\doteq$ (and without $=$) will not be able to distinguish this model from one where we removed e_2 and just had e_1. The above properties in fact only insist that $\doteq$ is an equivalence relation that is also a congruence with respect to the relations and functions.

However, it turns out that the above properties are sufficient to capture equality as far as satisfiability/validity of logical formulae are concerned. It doesn't matter that the properties above capture only congruence and not true equality.

Let us formalize this. Let M be a model with universe U an interpretation of the relation $\doteq$ that satisfies the congruence axioms given above. Then let $M/\doteq$ be the model where we take as the universe the equivalence classes $U/\doteq$, and interpret relations and functions as follows:

- For any constant c,
$$c^{M/\doteq} = [c^M]$$
- For any n-ary relation R,
$$R^{M/\doteq}([e_1], [e_2], \dots [e_n]) \textit{ holds iff } R^M(e1, \dots, e_n) \textit{ holds}$$
- For any n-ary function f,
$$f^{M/\doteq}([e_1], [e_2], \dots [e_n]) = [f^M(e_1, \dots, e_n)]$$

The above says that constants are interpreted in $M/\doteq$ as the equivalence class of their interpretation in M, and relations and functions are interpreted in $M/\doteq$ depending on how the relations and functions are interpreted on the elements in their equivalence classes. The reason the above is well-defined is because $\doteq$ satisfies the congruence axioms.

We can now show the following:

Lemma 5.4 *Fix a signature S without $=$ and $\doteq$. Let φ be a formula over $S \cup \{=\}$. Let φ' be φ, where $=$ is replaced with $\doteq$, and hence is over the signature $S \cup \{\doteq\}$.*

- *If φ holds in a model M, where $=$ is interpreted as equality in the model, then φ' holds in M' where $\doteq$ is interpreted as equality, and the interpretation of $\doteq$ does satisfy the congruence axioms.*
- *If φ' holds in a model M, where $\doteq$ satisfies the congruence axioms, then φ holds in $M/\doteq$ with $=$ interpreted as equality in the model.*

We leave the above as an exercise for the reader.

A consequence of the above lemma is that a formula with $=$ is satisfiable (or valid) iff the formula, with $=$ replaced by $\doteq$ is satisfiable (or valid) over the class of models that satisfy the congruence axioms.

Hence the above congruence axioms *define* equality as far as logic goes. Logically, there is no real difference between true equality and a congruence.