The Termination Problem of Equality Saturation is Undecidable

Termination of Equality Saturation

Theorem 1. The following problem is R.E.-complete:

```
Instance: a set of rewrite rules R, a term t.
Question: does EqSat terminate with R and t?
```

Proof. First, this problem is in R.E. since we can simply run EqSat with R and t to test whether it terminates. To show this problem is R.E.-hard, we reduce the termination problem of Turing machines to the termination of EqSat. We use the technique by [1]. In particular, for each Turing machine M, we produce a string rewriting system R such that the equivalence closure of R, $(\approx_R) = (R \cup R^{-1})^*$, satisfies that each equivalence class of $\approx_R$ corresponds to a trace of the Turing machine. As a result, the Turing machine halts iff its trace is finite iff the corresponding equivalence class in R is finite iff EqSat terminates.

In this proof, we consider a degenerate form of EqSat that works with *string* rewriting systems instead of term rewriting systems. Every string corresponds to a term, and every string rewrite rule corresponds to a rewrite rule. For example, the string uvw corresponds to a term $u(v(w(\epsilon)))$, where $u(\cdot), v(\cdot), w(\cdot)$ are unary functions and ϵ is a constant, and a string rewrite rule $uvw \to vuw$ corresponds to a (linear) term rewriting rule $u(v(w(x))) \to v(u(w(x)))$ where x is a variable.

A Turing machine $M = (K, \Sigma, \Pi, \mu, q_0, \beta)$ consists of a set of states K, the input and the tape alphabet Σ and Π (with $\Sigma \subseteq \Pi$), a set of transitions μ, an initial state $q_0 \in K$, and a special blanket symbol $\beta \in \Pi$. Each transition in μ is a quintuple in $K \times \Pi \times \Pi \times \{L, R\} \times K$. For example, transition $q_i abRq_j$ means if the current state is q_i and the symbol being scanned is a, then replace a with b, move the head to the right, and transit to state q_j. We assume the Turing machine is two-way infinite, so that the head can move in both directions indefinitely. Each configuration of M can be represented as $\rhd u q_i a v \lhd$, where $\rhd, \lhd$ are left and right end markers, u is the string to the left of the read/write head, q_i is the current state, a is the symbol being scanned, and v is the string to the right. We say $w_1 \vdash_M w_2$ if configuration w_1 can transit to configuration w_2 in a Turing machine M, and we omit M when it's clear from the context.

It is useful to define several sets of symbols for our construction. For each Turing machine M, we define $\overline{K} = \{\overline{q} \mid q \in K\}$. We also define $\overline{\Sigma}$, $\overline{\Pi}$ in a similar way. In our encoding, we use $\overline{K}$ to denote states where the symbol being scanned is to the left of the state, and we use $\overline{\Sigma}$ and $\overline{\Pi}$ to denote alphabets that are to the left of the states. Moreover, we introduce two sets of "dummy" symbols L_z and R_z for z ranges over $K \times (\{\lhd\} \cup \Pi)$ and $(\{\rhd\} \cup \overline{\Pi}) \times \overline{K}$. Let D_L and D_R be the set of all L_z and R_z respectively. We use these dummy symbols to make the string rewriting system that we will later define Church-Rosser.

The rewriting system we are going to define works over the set of strings $\mathit{CONFIG} = \rhd(\overline{\Pi} \cup D_L)^*(K \cup \overline{K})(\Pi \cup D_R)^*\lhd$. Strings in CONFIG is in a many-to-one mapping, denoted as π, to configurations of a Turing machine. $\pi(w)$ converts each $\overline{aq}_i$ to $q_i a$, removes dummy symbols L_z and R_z, and replace $\overline{a}$ with a. For example $\pi(\rhd L_{q_0,a}\overline{b}L_{q_1,b}\overline{cq_3}dR_{q_i,\lhd}\lhd) = \rhd b q_3 c d \lhd$

Now, the transitions in M, we define our string rewriting system R as follows.

transitions in M	rewrites in R
$q_i abRq_j$	$q_i a \to_R L_{q_i a}\overline{b}q_j$
	$\overline{aq}_i \to_R L_{\overline{a}q_i}\overline{b}q_j$
$q_i \beta bRq_j$	$q_i\lhd \to_R L_{q_i \lhd}\overline{b}q_j\lhd$

transitions in M	rewrites in R
	$\triangleright\overline{q}_i \to_R \triangleright L_{\triangleright\overline{q}_i}\overline{b}q_j$
$q_i abLq_j$	$q_i a \to_R \overline{q}_j bR_{q_i a}$
	$\overline{aq}_i \to_R \overline{q}_j bR_{\overline{aq}_i}$
$q_i\beta bLq_j$	$q_i\triangleleft \to_R \overline{q}_j bR_{q_i\triangleleft}\triangleleft$
	$\triangleright\overline{q}_i \to_R \triangleright\overline{q}_j bR_{\triangleright\overline{q}_i}$

Moreover, for each z, we have the following two additional (sets of) auxiliary rewrite rules

$$q_i R_z \to_R L_z L_z q_i$$
$$L_z\overline{q}_i \to_R \overline{q}_i R_z R_z$$

for any z.

To explain what these two rules do, let us define two types of strings. Type-A strings are strings where the symbol being scanned is to the immediate right of q_i or to the immediate left of $\overline{q_i}$. In other words, we call a string s a type-A string if s contains $q_i a$ or $\overline{aq_i}$. Type-B strings are strings that are not type-A: they are strings where there are dummy symbols in between the state and the symbol being scanned. The rewrite rules above convert any type-B strings into type-A in a finite number of steps.

Now, we observe that R has several properties:

1. Reverse convergence: the critical pair lemma implies that if a rewriting system is terminating and all its critical pairs are convergent, it is convergent. Define $\leftarrow_R = (\to_R)^{-1}$. $\leftarrow_R$ is terminating since rewrite rules in $\leftarrow_R$ decreases the sizes of terms (that is, rewrite rules in $\to_R$ increases the sizes of terms), and $\leftarrow_R$ has no critical pairs. Therefore, $\leftarrow_R$ is convergent.
2. For each type-A string w, then either
   - there exists no w' with $w \to_R w'$ and $\pi(w)$ is a halting configuration;
   - there exists a unique w' such that $w \to_R w'$ and $\pi(w) \vdash \pi(w')$.
3. For each type-B string w, there exists a unique w' such that $w \to_R w'$, and $\pi(w) = \pi(w')$. Moreover, if $w_0 \to_R w_1 \to_R \ldots$ is a sequence of type-B strings, the sequence must be bounded in length, since the state symbols q_i and $\overline{q}_i$ move towards one end according to the auxillary rules above.
4. By 2 and 3, $w \to_R w_1$ and $w \to_R w_2$ implies $w_1 = w_2$. In other words, $\to_R$ is a function.

These observations allows us to prove the following lemma

Lemma 2. Let $w_0 = \triangleright q_0 s\triangleleft$ be an initial configuration. w_0 is obviously in *CONFIG*. Moreover, given a Turing machine M, construct a string rewriting system R as above. M halts on w_0 if and only if $[w_0]_R$, the equivalence class of w_0 in R, is finite.

Proof. Consider $S : w_0 \to_R w_1 \to_R \ldots$, a sequence of *CONFIG* starting with w_0. By the above observations, S must have a subsequence of type-A strings $w_0 \to_R^* w_{a_1} \to_R^* w_{a_2} \to_R^* \ldots$ with

$$\pi(w_0) = \ldots = \pi(w_{a_1-1}) \vdash \pi(w_{a_1}) = \ldots = \pi(w_{a_2-1}) \vdash \pi(w_{a_2}) = \ldots .$$

An overview of the trace $w_0, w_{a_1}, w_{a_2}, \ldots$ and its properties is shown below:

Rw	w_0	$\to_R$	$\underbrace{w_1 \to_R \ldots \to_R w_{a_1-1}}_{\text{finite}}$	$\to_R$	w_{a_1}	$\underbrace{w_{a_1+1} \to_R \ldots \to_R w_{a_2-1}}_{\text{finite}}$	$\to_R$	w_{a_2}	...
Type	A		B ... B		A	B ... B		A	
Config	$\pi(w_0)$	$=$	$\pi(w_1) = \ldots = \pi(w_{a_1-1})$	$\vdash_M$	$\pi(w_{a_1})$	$\pi(w_{a_1+1}) = \ldots = \pi(w_{a_2-1})$	$\vdash_M$	$\pi(w_{a_2})$	...

Now we prove the claim:

- $\Leftarrow$: Suppose $[w_0]_R$ is finite. We show that there exists a *finite* sequence S of $w_0 \to_R w_1 \to_R \ldots \to_R w_n$ such that there is no w' such that $w_n \to_R w'$. If this is not the case, then an infinite rewriting sequence $w_0 \to_R w_1 \to \ldots$ must exist. Because $[w_0]_R$ is finite, for the sequence to be infinite, there must exist distinct

i, j such that $w_i = w_j$ in the sequence. However, this is impossible, because $\rightarrow_R$ always increases the sizes of terms.

By our observation above, if there is no such w' that $w_n \rightarrow_R w'$ in sequence S, it has to be the case that w_n is type-A and $\pi(w)$ is a halting configuration.

Now, take the subsequence of S that contains every type-A string:

$$w_0 \rightarrow_R^* w_{a_1} \rightarrow_R^* \dots \rightarrow_R^* w_{a_k} = w_n.$$

We have $\pi(w_{a_i}) \vdash \pi(w_{a_{i+1}})$ for all i and $\pi(w_{a_k})$ is a halting configuration. This implies a finite trace of the Turing machine:

$$w_0 \vdash \pi(w_{a_1}) \vdash \dots \vdash \pi(w_{a_n}).$$

Since we only consider deterministic Turing machines, the Turing machine halts on w_0.

- $\Rightarrow$: Suppose otherwise M halts on w_0 and $[w_0]_R$ is infinite.

  By definition, w_0 is a normal form with respect to $\leftarrow_R$, and because $\leftarrow_R$ is convergent, if there exists a w such that $w \approx_R w_0$, then $w_0 \rightarrow_R^* w$. The fact that $[w_0]_R$ is infinite implies w_0 can be rewritten to infinitely many strings w. Because $\rightarrow_R$ satisfies the functional dependency, it has to be the case that there exists an infinite rewriting sequence: $S : w_0 \rightarrow_R w_1 \rightarrow_R \dots$. Taking the subsequence of S consisting of every type-A strings:

  $$w_0 \rightarrow_R^* w_{a_1} \rightarrow_R^* \dots .$$

  This implies an infinite trace of the Turing machine:

  $$w_0 \vdash \pi(w_{a_1}) \vdash \dots,$$

  which is a contradiction.

We are ready to prove the undecidability of the termination problem of EqSat:

Given a Turing machine M. We construct the following two-tape Turing machine M':

```
M' alternates between the following two steps:
1. Simulate one transition of M on its first tape.
2. Read the string on its second tape as a number, compute the next prime number,
  and write it to the second tape.
M' halts when the simulation of M reaches an accepting state.
```

It is known that a two-tape Turing machine can be simulated using a standard Turing machine, so we assume M' is a standard Turing machine and takes input string (s_1, s_2), where s_1 is the input on its first tape and s_2 is the input on its second tape. Let R' be the string rewriting system derived from M' using the encoding we introduced in the lemma.

Given a string s, let w be the initial configuration $\triangleright q_0(s, 2)\triangleleft$. The following conditions are equivalent to each other:

1. M halts on input s.
2. M' halts on input $(s, 2)$.
3. $[w]_{R'}$ is finite.
4. $[w]_{R'}$ is regular.

Note that (3) implies (4) trivially, and (4) implies (3) because if $[w]_{R'}$ is infinite, it must not be regular since the trace of M' computes every prime number.

Now run EqSat with initial string w and rewriting system $\leftrightarrow_{R'}$. EqSat terminates if and only if M halts on s:

- $\Rightarrow$: Suppose EqSat terminates with output E-graph G. Strings equivalent to w in G is exactly the equivalence class of w, i.e., $[w]_G = [w]_{R'}$. Moreover, every e-class in an E-graph represents a regular language, so $[w]_G$ is regular. Therefore, $[w]_G$ is finite.
- $\Leftarrow$: Suppose M halts on s. This implies $[w]_{R'}$ is finite. Because EqSat monotonically enlarges the set of represented terms, it has to stop in a finite number of iterations.

Because the halting problem of a Turing machine is undecidable, the termination problem of EqSat is undecidable as well. ■

Theorem 3. The following problem is undecidable.

```
Instance: a set of rewrite rules R, a term w.
Problem: Is [w]_R regular?
```

Proof.

To show the undecidability, we reduce the halting problem of Turing machines to this problem. As shown in Theorem 1, given a Turing machine M, M halts on an input s if and only if $[w]_{R'}$ is regular for $w = \triangleright q_0(s, 2)\triangleleft$. ■

For a particular kind of rewrite systems, we show this regularity problem is R.E.-complete.

Theorem 4. The following problem is R.E.-complete.

```
Instance: a set of left-linear, convergent rewrite rules R, a term w.
Problem: Is [w]_R regular?
```

Proof.

$\leftarrow_R$, the inverse of $rightarrow_R$ defined in Theorem 1, is convergent. Moreover, every string rewriting system is a linear term rewriting system and therefore a left-linear term rewriting system, so the regularity of a left-linear, convergent term rewriting system is undecidable. Additionally, we show this problem is in R.E. by showing a semi-decision procedure for this problem.

We enumerate E-graphs and for each E-graph G, we check the following three conditions:

- $w \in L(G)$.
- $[w]_R \subseteq [w]_G$: to check this, we can check if $[w]_G$ is saturated with respect to $\leftrightarrow_R$.
- $[w]_G \subseteq [w]_R$: to check this, we can check if $[w]_G$ has only one normal form.

References

[1] Narendran, P., Ó'Dúnlaing, C. and Rolletschek, H. 1985. Complexity of certain decision problems about congruential languages. *Journal of Computer and System Sciences*. 30, 3 (1985), 343–358. DOI:https://doi.org/https://doi.org/10.1016/0022-0000(85)90051-0.