

## SAT<sub>fin</sub> is not recursive

We focus here on the very basic property of finite satisfiability. We show that it is undecidable whether an FO sentence has a finite model.

To prove this result, we use a reduction of the Post Correspondence Problem (PCP) to the finite satisfiability problem for FO.

Before proceeding, we briefly recall the PCP. The input to the PCP consists of two lists

$$u_1, \dots, u_n; \quad v_1, \dots, v_n;$$

of words over some alphabet  $\Sigma$  with at least two symbols. A solution to the PCP is a sequence of indices  $i_1, \dots, i_k$ ,  $1 \leq i_j \leq n$ , such that

$$u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}.$$

The question of interest is whether there is a solution to the PCP. For example, consider the input to the PCP problem:

$$\begin{array}{ccccccccc} u_1 & u_2 & u_3 & u_4 & v_1 & v_2 & v_3 & v_4 \\ aba & bbb & aab & bb & a & aaa & abab & babba \end{array}$$

For this input, the PCP has the solution 1, 4, 3, 1; since

$$u_1 u_4 u_3 u_1 = ababbaababa = v_1 v_4 v_3 v_1.$$

Now consider the input consisting of just  $u_1, u_2, u_3$  and  $v_1, v_2, v_3$ . An easy case analysis shows that there is no solution to the PCP for this input. In general, it has been shown that it is undecidable whether, for given input, there exists a solution to the PCP.

**Theorem 0.0.1** Finite satisfiability of FO sentences r.e. but not recursive.

**Proof:** To see that the problem is r.e., imagine an algorithm that, when given a sentence  $q$  over  $\mathbf{R}$  as input, generates all instances  $\mathbf{I}$  over  $\mathbf{R}$  and tests  $q(\mathbf{I}) = \emptyset$  until a nonempty answer is found.

To show that satisfiability is not recursive, we reduce the PCP to the satisfiability problem. In particular, we show that if there were an algorithm for solving satisfiability, then it could be used to construct an algorithm that solves the PCP.

Let  $\mathcal{P} = (u_1, \dots, u_n; v_1, \dots, v_n)$  be an instance of the PCP, i.e., a pair of sequences of nonempty words over alphabet  $\{0,1\}$ . We describe now an FO sentence  $q_{\mathcal{P}}$  with the property that  $q_{\mathcal{P}}$  is satisfiable iff  $\mathcal{P}$  has a solution.

We shall use a relation schema  $\mathbf{R}$  having relations *ENC(ODING)* with sort<sup>1</sup>  $[A, B, C, D, E]$  and *SYNCH(RONIZATION)* with sort  $[F, G]$ . The sentence  $q_{\mathcal{P}}$  shall use constants

$$\{0, 1, \$, c_1, \dots, c_n, d_1, \dots, d_n\}.$$

<sup>1</sup>In this proof, following database theory terminology, relations have named columns called attributes. The sort of a relation is the set of its attributes.

<i>ENC</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>		<i>SYNCH</i>	<i>F</i>	<i>G</i>
	\$	$a_1$	0	$c_1$	$d_1$			\$	\$
	$a_1$	$a_2$	1	$c_1$	$d_2$			$a_3$	$a_1$
	$a_2$	$a_3$	1	$c_1$	$d_2$			$a_6$	$a_3$
	$a_3$	$a_4$	0	$c_2$	$d_3$			$a_7$	$a_8$
	$a_4$	$a_5$	1	$c_2$	$d_3$				
	$a_5$	$a_6$	1	$c_2$	$d_3$				
	$a_6$	$a_7$	0	$c_3$	$d_3$				
	$a_7$	$a_8$	0	$c_2$	$d_3$				
	$a_8$	$a_9$	1	$c_2$	$d_2$				
	$a_9$	\$	1	$c_2$	$d_2$				

Figure 1: Encoding of a solution to PCP

(The use of multiple relations and constants is largely a convenience; the result can be demonstrated using a single ternary relation and no constants. )

To illustrate the construction of the algorithm, consider the following instance of the PCP:

$$u_1 = 011, u_2 = 011, u_3 = 0; \quad v_1 = 0, v_2 = 11, v_3 = 01100.$$

Note that  $s = (1, 2, 3, 2)$  is a solution of this instance, i.e., that

$$u_1 u_2 u_3 u_2 = 0110110011 = v_1 v_2 v_3 v_2.$$

Figure 1 shows an input instance  $\mathbf{I}_s$  over  $\mathbf{R}$  which encodes this solution, and which satisfies the a sentence  $q_{\mathcal{P}}$  constructed shortly.

In the relation  $ENC$  of this figure, the first two columns form a *cycle*, so that the 10 tuples can be viewed as a sequence rather than a set. The third column holds a listing of the word  $w = 0110110011$  that witnesses the solution to  $P$ ; the fourth column describes which words of sequence  $(u_1, \dots, u_n)$  are used to obtain  $w$ ; and the fifth column describes which words of sequence  $(v_1, \dots, v_n)$  are used. The relation  $SYNCH$  is used to “synchronize” the two representations of  $w$ , by listing the pairs corresponding to the beginnings of new  $u$ -words and  $v$ -words.

The formula  $\varphi_{\mathcal{P}}$  constructed now includes subformulas to test whether the various conditions just enumerated hold on an input instance. In particular,

$$\varphi = \varphi_{ENC\text{-key}} \wedge \varphi_{cycle} \wedge \varphi_{SYNCH\text{-keys}} \wedge \varphi_{u\text{-encode}} \wedge \varphi_{v\text{-encode}} \wedge \varphi_{u\text{-v-synch}}$$

where, speaking informally,

$\varphi_{ENC\text{-key}}$  states that the first column of  $ENC$  is a *key*, that is, each value occurring in the  $A$  column occurs in exactly one tuple of  $ENC$ .

$\varphi_{cycle}$  states that the first two columns of  $ENC$  hold a cycle, with length  $> 1$ , which has the constant  $\$$  occurring in it.

$\varphi_{SYNCH-keys}$  states that both the first and second columns of  $SYNCH$  are keys.

$\varphi_{u-encode}$  states that for each value  $x$  occurring in the first column of  $SYNCH$ , if tuple  $\langle x_1, y_1, z_1, c_i, d_{j_1} \rangle$  is in  $ENC$  then there are at least  $|u_i| - 1$  additional tuples in  $ENC$  “after” this tuple, all with value  $c_i$  in the fourth coordinate, and if these tuples are

$$\langle x_2, y_2, z_2, c_i, d_{j_2} \rangle \dots \langle x_k, y_k, z_k, c_i, d_{j_k} \rangle$$

then:  $z_1 \dots z_k = u_i$ ; none of  $x_2, \dots, x_k$  occurs in the first column of  $SYNCH$ ; and if  $y_k \neq \$$  then the  $A$  value “after”  $x_k$  occurs in the first column of  $SYNCH$ .

$\varphi_{v-encode}$  is analogous to  $\varphi_{u-encode}$ .

$\varphi_{u-v-synch}$  states that (i)  $\langle \$, \$ \rangle$  is in  $SYNCH$ , (ii) if a tuple  $\langle x, y \rangle$  is in  $SYNCH$ , then the associated  $u$ -word and  $v$ -word have the same index, and (iii) if a tuple  $\langle x, y \rangle$  is in  $SYNCH$ , and either  $x$  or  $y$  are not the “maximum”  $A$  value occurring in  $F$  or  $G$ , then there exists a tuple  $\langle x', y' \rangle$  in  $SYNCH$ , where  $x'$  is the first  $A$  value “after”  $x$  occurring in  $F$  and  $y'$  is the first  $A$  value “after”  $y$  occurring in  $G$ . Finding the  $A$  values “after”  $x$  and  $y$  is done as in  $\varphi_{u-encode}$ .

The constructions of these formulas are relatively straightforward; we give two of them here and leave the others for the reader. In particular, we let

$$\psi(x, y) = \exists p, q, r \text{ } ENC(x, y, p, q, r)$$

and set

$$\begin{aligned} \varphi_{cycle} = & \exists x(\psi(x, \$) \wedge \neg(x = \$)) \wedge \exists y(\psi(\$ , y) \wedge \neg(y = \$)) \wedge \\ & \forall x((\exists y\psi(x, y)) \rightarrow (\exists z\psi(z, x))) \wedge \\ & \forall x((\exists y\psi(y, x)) \rightarrow (\exists z\psi(x, z))) \wedge \\ & \forall x, y_1, y_2(\psi(y_1, x) \wedge \psi(y_2, x) \rightarrow y_1 = y_2) \end{aligned}$$

If  $ENC$  satisfies  $\varphi_{ENC-key} \wedge \varphi_{cycle}$  then the first two coordinates of  $ENC$  hold one or more disjoint “cycles”, exactly one of which contains the value  $\$$ .

Parts (i) and (ii) of  $\varphi_{u-v-synch}$  are realized by the formula:

$$\begin{aligned} & SYNCH(\$ , \$) \wedge \\ & \forall x, y(SYNCH(x, y) \rightarrow \\ & \quad \exists s, p, r, t, p', q((ENC(x, s, p, c_1, r) \wedge ENC(y, t, p', q, d_1)) \vee \\ & \quad \quad (ENC(x, s, p, c_2, r) \wedge ENC(y, t, p', q, d_2)) \vee \\ & \quad \quad \quad \vdots \\ & \quad \quad \quad (ENC(x, s, p, c_n, r) \wedge ENC(y, t, p', q, d_n)))) \end{aligned}$$

Verifying that the a sentence  $q_P$  described above is satisfiable if and only if  $P$  has a solution is left to the reader.  $\square$