LOGIC COLLOQUIUM '82
G. Lolli, G. Longo and A. Marcja (editors)

NP AND CRAIG'S INTERPOLATION THEOREM

Daniele Mundici

Loc.Romola N.76
50060 Donnini
Florence-Italy

The truth-values of two renowned conjectures about NP (namely, $P \neq NP$ and, NP is not closed under complementation) depend on the difficulty in writing down Craig's interpolants in sentential logic. The general connection between NP and interpolation is studied by blending ideas and techniques from both model theory and computation theory.

0. Introduction.

We fix throughout an alphabet Σ and regard Boolean expressions as particular words over Σ. If $B \longrightarrow C$ is a tautology, then Craig's interpolation theorem yields an interpolant I, that is, a Boolean expression I such that $B \longrightarrow I$ and $I \longrightarrow C$ are tautologies, and the variables occurring in I are exactly those which jointly occur in B and C. In Theorem 2 and Corollary 6 we prove the following result (where $TAUT \subseteq \Sigma^*$ is the set of tautologies and Σ^* is the set of words over Σ):

At least one of the following sentences is true:

(I) TAUT is accepted in deterministic polynomial time (viz., P=NP);

(II) TAUT is not accepted in nondeterministic polynomial time (viz., NP is not closed under complementation);

(III) Interpolation is polynomially intractable, viz., for every function $\Phi : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$, if Φ is computable in deterministic polynomial time, then for some tautology $B \longrightarrow C$, $\Phi(B,C)$ fails to be an interpolant for $B \longrightarrow C$.

For the proof we use a mixture of techniques from computation and model theory. Assume now that the upper bounds for computations are relaxed from the set $\mathfrak{F}_0$ of polynomials to any set $\mathfrak{F} \supseteq \mathfrak{F}_0$ closed under sum and composition. Then the above result still holds relative to the new upper bounds -- perhaps with a different distribution of truth-values among (I)-(III). This is proved in Theorem 7. Thus it might be interesting to find two sets $\mathfrak{F}$ and $\mathfrak{F}'$ (if any) assigning different triplets of truth values to (I)-(III): as a matter of fact, when $\mathfrak{F}$ is the set of all functions, or even, when $\mathfrak{F}$ contains the exponential, then trivially (I) becomes true and (II)

and (III) false. At the opposite extreme, when $\mathfrak{F}$ is restricted to the set of polynomials, it is widely conjectured that (I) is false, hence, either of (II) or (III) is then true.

The above results yield just one more connection between the model-theoretical notion of interpolation, and computation theory. For another such connection, in [6] it is proved that if the rate of growth of the length of interpolants for any tautology $B \longrightarrow C$ can be kept below some polynomial in the length of B and C, then every function which is computable in deterministic polynomial time has circuit depth growing proportionally to the logarithm of the input length.

This as well as other results concerning the complexity of Craig's interpolation theorem in sentential and in first-order logic are surveyed in the final section of this paper. For the general role of Craig's interpolation theorem in (abstract) model theory, see, e.g., [1] , [8] and [9].

1. Preliminaries.

For A an arbitrary set, A^* denotes the set of words over A, i.e., the set of all finite strings of symbols from A. For $w \in A^*$, the length $|w|$ of w is the number of occurrences of symbols in w. For $a \in A$, a^n stands for $aa...a$ (n times). Throughout this paper Σ denotes the following set of symbols:

$$\Sigma = \{ \wedge , \vee , \neg ,) , (, X, 0, 1 \} .$$

Boolean expressions are understood as particular words over Σ, according to the familiar formation rules studied in sentential logic. Propositional variables are words over $\{X,0,1\}$ of the form $Xb_1...b_n$, where the subscript $b_1...b_n$ is the sequence of digits of a number in binary notation. For $B \in \Sigma^*$ an arbitrary Boolean expression,

$$\text{var } B \quad , \qquad \text{and} \qquad |\text{var } B|$$

respectively denote the set of variables occurring in B, and the number of elements in this set. The variables in var B inherit the order given by their subscripts. Elements of $\{0,1\}$ are called bits. Letting now $b = |\text{var } B|$ and $x \in \{0,1\}^b$, by

$$x \models B \qquad \text{(read: } x \text{ satisfies } B\text{)}$$

we mean that B, considered as a Boolean function $B: \{0,1\}^b \longrightarrow \{0,1\}$,

takes value 1 on input x $(=(x_1,\ldots,x_b))$. One can recover the familiar semantics of sentential logic upon identifying 1 with "true" and 0 with "false". The set Mod B is defined by

$$\text{Mod } B = \left\{ x \in \{0,1\}^b \mid x \models B \right\}.$$

For arbitrary $n \leqslant b$ we also set

$$\text{Mod } B\restriction(\text{first } n \text{ bits}) = \left\{x \in \{0,1\}^n \mid \exists\, y \in \{0,1\}^b \text{ with } y \models B \text{ and } \bigwedge_{i=1}^{n} y_i = x_i\right\}.$$

In model-theoretical terminology, Mod B $\restriction$(first n bits) is the set of reducts (to the first n bits) of the models of B. B is a tautology iff Mod B $= \{0,1\}^b$; B is satisfiable iff Mod B $\neq \emptyset$. Craig's interpolation theorem in sentential logic (see [1]) yields, for any two Boolean expressions B and C such that $B \longrightarrow C$ is a tautology, an interpolant I, that is a Boolean expression I such that var I = var B $\cap$ var C, and $B \longrightarrow I$ and $I \longrightarrow C$ are tautologies. Here, as usual, $B \longrightarrow C$ is an abbreviation of $((\neg B) \vee (C))$. In case var B $\cap$ var C $= \emptyset$ (never occurring in this paper) Craig's interpolation theorem states that either $\neg$ B or C is a tautology.

Throughout this paper we also use a fixed (but otherwise arbitrary) function $g: \Sigma \longrightarrow \{0,1\}^3$ mapping the symbols of Σ one-one onto the set of triplets of bits; g naturally induces a one-one map:

$$\Gamma : \Sigma^* \longrightarrow \{0,1\}^*, \text{ with } \Gamma(C) = \text{concatenation of } g(C_1),\ldots,g(C_n),$$

for every $C = (C_1,\ldots,C_n) \in \Sigma^*$.

Instead of saying that f is computable by a deterministic (resp., nondeterministic) Turing machine in time bounded by a polynomial in the length of the input, we shall briefly say that f is computable in deterministic (resp., nondeterministic) polynomial time. As usual, P stands for the class of sets of strings which are accepted (that is, whose characteristic function is computable) in deterministic polynomial time; NP is the same, with regard to nondeterministic polynomial time. It is not known whether P=NP or even, whether NP is closed under complementation (that is, for any $S \subseteq A^*$, if $S \in$ NP then $A^* \setminus S \in$ NP). In the following theorem these problems are related to the degree of

difficulty in writing down Craig's interpolants in sentential logic.

2. Theorem. At least one of the following statements holds true:

(i) P=NP;

(ii) NP is not closed under complementation;

(iii) (Intractability of sentential interpolation): for every function $\Phi : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$, if Φ is computable in deterministic polynomial time, then there is a tautology $B \longrightarrow C$ with $B,C \in \Sigma^*$ and $\text{var } B \cap \text{var } C \neq \emptyset$, such that $\Phi(B,C)$ is not an interpolant for $B \longrightarrow C$.

For the proof we prepare the following lemma:

3.Lemma. For $S \subseteq \{0,1\}^*$ an arbitrary nonempty set the following are equivalent:

(a) $S \in NP$;

(b) there exists a function $F: \{1\}^* \longrightarrow \Sigma^*$ which is computable in deterministic polynomial time, such that, letting $F_n = F(1^n)$, we have, for each $n \geq 1$:

$$|\text{var } F_n| \geq n, \quad \text{and} \quad S \cap \{0,1\}^n = \text{Mod } F_n \restriction (\text{first } n \text{ bits}).$$

Proof of Lemma 3. (b) $\Rightarrow$ (a). A fast nondeterministic algorithm for accepting S is, on input $x \in \{0,1\}^*$, as follows: first compute $n = |x|$ and F_n; then let $r = |\text{var } F_n|$; finally guess a $y \in \{0,1\}^r$ such that $y \models F_n$ and $x_1=y_1, \ldots, x_n=y_n$.

(a) $\Rightarrow$ (b). We shall obtain the required F by modifying Cook's well-known argument to prove that the satisfiability problem for Boolean expressions is NP-complete (see [2]). We shall closely follow the notation and terminology of [5, Theorem 7.3.9]. So let T be a nondeterministic Turing machine accepting S in time bounded by a polynomial p in the length of the input. Let $\{0,\ldots,h\}$ be the set of states of T, with h the halting state. Let $A = \{a_1, a_2, \ldots, a_k\}$ be the alphabet of T, with $a_1=1$, $a_2=0$ and a_k=the blank symbol. Fix $n \geq 1$ and let $m = p(n)$. The variables of F_n are, for each t,i,j,q $(0 \leq t \leq m, \ 0 \leq i \leq 2m, \ 1 \leq j \leq k, \ 0 \leq q \leq h)$, the following:

(1) SYMB(t,i,j), HEAD(t,i), and STATE(t,q), respectively saying that at time t, a_j is printed on tape square i, T has tape position i, and T is in state q.

Consider the conjunction of the following sentences, which uniformly describe the behavior of T over any input of length n:

(A) at each time t, each tape square i has precisely one symbol printed on it, T is scanning precisely one square, and T is in precisely one state;

(B) at each time t, if T is not over square i, then the symbol on i does not change;

(C) at some time t', T halts;

(D) the computation starts in state 0 over the left hand end of the input, with the tape only containing blank symbols (with the possible exception of the squares m through m+n-1);

(E) initially each tape square m through m+n-1 has either 1 or 0 printed on it;

(F) the changes in tape symbols, head position and state obey T's instructions.

Notice that, following [5], we assume that the input is printed on the tape squares m through m+n-1. It is well-known that (A),(B),(C) and (F) can be written down as Boolean expressions only depending on m, h, k and the instructions of T, and whose variables are among those displayed in (1): indeed, such expressions are explicitly written down in [5, p.236,237, expressions (1)-(5),(7),(8),(9)]: our (A) corresponds to (1)∧(2)∧(3), (B) is (4), (C) is (5), and (F) is (7)∧(8)∧(9) therein. The following Boolean expressions take care of (D) and (E):

(D) $\mathrm{STATE}(0,0) \wedge \mathrm{HEAD}(0,m) \wedge (\bigwedge_{i=0}^{m-1} \mathrm{SYMB}(0,i,k)) \wedge (\bigwedge_{i=m+n}^{2m} \mathrm{SYMB}(0,i,k))$;

(E) $\bigwedge_{i=m}^{m+n-1} (\mathrm{SYMB}(0,i,1) \vee \mathrm{SYMB}(0,i,2))$.

Let $F_n = (A) \wedge \ldots \wedge (F)$; let $s = |\mathrm{var}\, F_n| - 1$. It is no loss of

generality to assume that the first n variables of F_n are SYMB(0,m,1),...,SYMB(0,m+n-1,1) in the given order. Assume each variable X_r of F_n is assigned a bit (intuitively, a truth-value) $x_r \in \{0,1\}$, $r = 0,\ldots,s$, in such a way that $x_0,\ldots,x_s$ satisfies F_n; in symbols,

(2) $\qquad x_0,\ldots,x_s \models F_n$.

Then in the light of (1) we obtain a collection Δ of statements of the form "at time t, T has such and such position, T is in such and such state, and tape square i has such and such symbol printed on it", for each $0 \leqslant t \leqslant m$ and $0 \leqslant i \leqslant 2m$. Now the definition of F_n, together with (2) ensure that Δ is not a brute collection of incoherent statements, but rather describes a legal accepting computation of T on some input $y \in \{0,1\}^n$ printed on the tape squares m through $m+n-1$. To find the symbol y_j originally printed on tape square $m+j$ ($j = 0,\ldots, n-1$), observe that if $x_j=1$ then SYMB(0,m+j,1) is "true" and, by (1) we have that $y_j= 1$. On the other hand, if $x_j = 0$, then SYMB(0,m+j,1) is "false" and, by (E), SYMB(0,m+1,2) is "true", so that by (1), $y_j = 0$ (recall that $a_1=1$ and $a_2= 0$). In definitive, $y_j = x_j$ for each $j = 0,\ldots,n-1$, so that $x_0,\ldots,x_{n-1}$ is accepted by T. Therefore, $x_0,\ldots,x_{n-1},x_n,\ldots,x_s \models F_n$ implies $x_0,\ldots,x_{n-1} \in S$. In symbols,

(3) $\qquad \mathrm{Mod}\, F_n \restriction (\text{first } n \text{ bits}) \subseteq S \cap \{0,1\}^n$.

Conversely, if $x_0,\ldots,x_{n-1} \in S$ then there is an accepting computation ∇ of T on input $x_0,\ldots,x_{n-1}$ in time at most m, and we can safely assume that the input is printed on tape squares m through $m+n-1$, the whole tape having $2m+1$ many squares, $0,1,\ldots,2m$. Using (1) we can unambiguously assign truth values $x_0,\ldots,x_{n-1}$, $x_n,\ldots,x_s$ to all the variables of F_n, by just coding the pieces of information contained in ∇. Since the latter is a legal computation we then have, by definition of F_n :

(4) $\qquad x_0,\ldots,x_s \models F_n$.

Therefore, $x_0,\ldots,x_{n-1} \in S$ implies that there is $y_0,\ldots,y_s \models F_n$

with $y_0=x_0,\ldots,y_{n-1}=x_{n-1}$. In symbols,

(5) $\quad S \cap \{0,1\}^n \subseteq \text{Mod } F_n \restriction (\text{first } n \text{ bits})$.

Now (3) and (5) jointly yield a first desired conclusion about F. To complete the proof that (a) $\Rightarrow$ (b) we must show that the map $1^n \longmapsto F_n$ described above is computable in deterministic polynomial time. But this is well-known to hold for $(A) \wedge (B) \wedge (C) \wedge (F)$; see [5, p.238] where an upper bound of the form m^8 is claimed to hold, after observing that the length of (1)-(9) therein is at worst proportional to $p^4(n)$, and that (1)-(9) is so simple. The same conclusion holds in the present case, with the same argument. This concludes the proof that (a) $\Rightarrow$ (b) and completes the proof of lemma 3. ■

4.Lemma. Assume statements (ii) and (iii) in Theorem 2 are both false. Then there is a function $I: \{1\}^* \longrightarrow \Sigma^*$ which is computable in deterministic polynomial time, such that, letting $I_n = I(1^n)$, we have for each $n \geq 1$:

$$\text{Mod } I_n = \{x \in \{0,1\}^n \mid \Gamma^{-1}(x) \text{ is satisfiable}\},$$

with Γ the map defined in section 1.

Proof of Lemma 4. Since the set of satisfiable Boolean expressions is in NP (see [5, 7.3.5]) then so is the set S given by

(6) $\quad S = \{x \in \{0,1\}^* \mid \Gamma^{-1}(x) \text{ is satisfiable}\}$.

This clearly follows from the definition of Γ. Since (ii) is assumed to be false, then the set $\bar{S} = \{0,1\}^* \setminus S$ is in NP, too. Without loss of generality there are nondeterministic Turing machines T and T' and a polynomial p such that T accepts S and T' accepts $\bar{S}$ in time bounded by p. By Lemma 3 ((a) $\Rightarrow$ (b)) there exist functions H, H': $\{1\}^* \longrightarrow \Sigma^*$ which are computable in deterministic polynomial time such that, letting $H_n = H(1^n)$, $H'_n = H'(1^n)$, we have for each $n \geq 1$:

(7) $\quad S \cap \{0,1\}^n = \text{Mod } H_n \restriction (\text{first } n \text{ bits})$, and

$\quad\quad \bar{S} \cap \{0,1\}^n = \text{Mod } H'_n \restriction (\text{first } n \text{ bits})$.

Since T and T' both act in time bounded by p, by an easy inspection of the proof of Lemma 3 we can safely stipulate that, in addition,

(8) $\text{var } H_n \cap \text{var } H'_n = \{SYMB(0,m,1),\ldots,SYMB(0,m+n-1,1)\}$;

we just use the same symbols for the first n variables of H_n and H'_n, then rename the other variables of H_n (if necessary) so that condition (8) is satisfied. Consider now the Boolean expression $H_n \wedge H'_n$; if the latter conjunction were satisfiable (absurdum hypothesis), say $x_0,\ldots,x_{n-1},x_n,\ldots,x_z$ satisfies $H_n \wedge H'_n$, then by (7) we have that $x_0,\ldots,x_{n-1} \in S \cap \bar{S}$, which is impossible. Therefore we get:

(9) $H_n \longrightarrow \neg H'_n$ is a tautology, for each $n \geq 1$.

Since we are assuming that (iii) is false, let Φ be a counterexample to (iii) in Theorem 2: that is, Φ is computable in deterministic polynomial time, and misses no interpolants. So let $I_n = \Phi(H_n, \neg H'_n)$; then I_n has the following properties:

(10) $H_n \longrightarrow I_n$ and $I_n \longrightarrow \neg H'_n$ are tautologies;

(11) $\text{var } I_n = \{SYMB(0,m,1),\ldots,SYMB(0,m+n-1,1)\}$;

(12) the mapping $1^n \longmapsto I_n$ is computable in deterministic polynomial time.

Clause (12) is a consequence of our assumptions about Φ together with the fact that the maps $n \longmapsto H_n$ and $n \longmapsto H'_n$ are both computable in deterministic polynomial time. From the first tautology in (10), and from (8) and (11) we have

(13) $\text{Mod } H_n \restriction (\text{first } n \text{ bits}) \subseteq \text{Mod } I_n$.

Hence, by (7) we get

(14) $S \cap \{0,1\}^n \subseteq \text{Mod } I_n$.

Similarly, from the second tautology in (10), written as $H'_n \longrightarrow \neg I_n$, and from (8),(11),(7) we obtain

(15) $\bar{S} \cap \{0,1\}^n \subseteq \text{Mod} \neg I_n$.

From (14) and (15), recalling (6) we get

(16) Mod $I_n = S \cap \{0,1\}^n = \{x \in \{0,1\}^n \mid \Gamma^{-1}(x) \text{ is satisfiable}\}$,

which completes the proof of our Lemma. ■

5. End of the proof of Theorem 2.

We shall prove that if (iii) and (ii) are both false, then P=NP. To this purpose, let $I: \{1\}^* \to \Sigma^*$ be as given by Lemma 4, and let M be a deterministic Turing machine computing each I_n in time bounded by a polynomial q in the length n of the input. A fast(deterministic)decision procedure for satisfiability is, on input $B \in \Sigma^*$, as follows:

(D1) compute $\Gamma(B)$;

(D2) write down $n = |\Gamma(B)| = 3|B|$;

(D3) using M write down I_n;

(D4) check whether $\Gamma(B) \models I_n$.

Notice that (D4) can be carried out in deterministic polynomial time as claimed, since the properties of M ensure that $|I_n|$ is bounded by $q(n)$, and $\Gamma(B)$ is a sequence of n bits. The above process (D1)-(D4) provides the required decision procedure for satisfiability of any Boolean expression B in time bounded by a polynomial in the length of B. Therefore we conclude that, under our assumptions, P=NP holds. This completely proves the Theorem. ■

Let now $TAUT \subseteq \Sigma^*$ denote the set of Boolean expressions which are tautologies.

6.Corollary. At least one of the following statements holds true:

(I) TAUT is accepted in deterministic polynomial time;

(II) TAUT is not accepted in nondeterministic polynomial time;

(III) same as statement (iii) in Theorem 2.

Proof. It is well-known that TAUT is in P iff P=NP (see [2]). Similarly, TAUT is in NP iff NP is closed under complementation (see, e.g., [3, 1.1]). Now apply Theorem 2. ■

The above Corollary is stable under relaxation of the upper bounds for (deterministic and nondeterministic) computations, as we shall see in Theorem 7 below. As usual, ${}^{\mathbb{N}}\mathbb{N}$ denotes the set of all functions $f: \mathbb{N} \longrightarrow \mathbb{N}$. A set $\mathfrak{F} \subseteq {}^{\mathbb{N}}\mathbb{N}$ is closed under composition iff the composition of any two functions in $\mathfrak{F}$ is still a function in $\mathfrak{F}$; closure under sum is similarly defined.

7. Theorem. Let $\mathfrak{F} \subseteq {}^{\mathbb{N}}\mathbb{N}$ be an arbitrary set containing the polynomials and closed under composition and sum. Then at least one (perhaps depending on $\mathfrak{F}$) of the following statements holds true:

($I_{\mathfrak{F}}$) TAUT is accepted by some deterministic Turing machine in time bounded by a function of $\mathfrak{F}$ (in the length of the input);
($II_{\mathfrak{F}}$) TAUT is not accepted by any nondeterministic Turing machine in time bounded by a function of $\mathfrak{F}$;
($III_{\mathfrak{F}}$) For every $\Phi : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$, if Φ is computable by a deterministic Turing machine in time bounded by a function of $\mathfrak{F}$, then there is a tautology $B \longrightarrow C$, with $B, C \in \Sigma^*$, var $B \cap$ var $C \neq \emptyset$, such that $\Phi(B,C)$ is not an interpolant for $B \longrightarrow C$.

For the proof we modify Lemmas 3 and 4 as follows:

8. Lemma. Assume $\emptyset \neq S \subseteq \{0,1\}^*$, and S is accepted by a nondeterministic Turing machine T in time bounded by a function $f \in \mathfrak{F}$. Then there exists $F: \{1\}^* \longrightarrow \Sigma^*$ which is computable by a deterministic Turing machine in time bounded by a function of $\mathfrak{F}$, such that letting $F_n = F(1^n)$ we have for each $n \geq 1$:

$$|\text{var } F_n| \geq n, \quad \text{and} \quad S \cap \{0,1\}^n = \text{Mod } F_n \restriction (\text{first } n \text{ bits}).$$

Proof. Argue exactly as in the proof of Lemma 3 ((a) $\Rightarrow$ (b)); the upper bound for T is now given by f (instead of p therein). Write down F_n and note that F_n satisfies the second requirement of the present Lemma. To see that the mapping $n \longmapsto F_n$ is computable by a deterministic Turing machine in time bounded by a function of $\mathfrak{F}$, by analogy with the final observation in the proof of Lemma 3, first note that $|F_n|$ is at most proportional to m^4, that is, proportional to $f^4(n)$. Again, F_n can be written down in time not much greater than $|F_n|$, say for definiteness $|F_n|^2$. But the function $f^8(n)$

$\geq |F_n|^2$ still is in $\mathfrak{F}$, by the assumed closure properties of $\mathfrak{F}$. ■

9.Lemma. Assume $(II_{\mathfrak{F}})$ and $(III_{\mathfrak{F}})$ are both false. Then there is a function $I: \{1\}^* \longrightarrow \Sigma^*$ which is computable by a deterministic Turing machine in time bounded by a function of $\mathfrak{F}$, such that letting $I_n = I(1^n)$ we have for each $n \geq 1$:

$$\text{Mod } I_n = \left\{ x \in \{0,1\}^n \mid \Gamma^{-1}(x) \text{ is a tautology} \right\}.$$

Proof. Let $Z \subseteq \{0,1\}^*$ be defined by $Z = \left\{ x \in \{0,1\}^* \mid \Gamma^{-1}(x) \text{ is a tautology} \right\}$. By assumption, and by definition of Γ, Z is accepted by a nondeterministic Turing machine W in time bounded by a function $w \in \mathfrak{F}$. Clearly, $\Sigma^* \setminus \text{TAUT}$ is in NP, hence the set $\bar{Z} = \{0,1\}^* \setminus Z$ is accepted by a nondeterministic Turing machine W' in time bounded by a function $w' \in \mathfrak{F}$. Since $\mathfrak{F}$ is closed under sum, we can safely assume $w = w'$. By Lemma 8 there are functions $H, H' : \{1\}^* \longrightarrow \Sigma^*$ which are computable by deterministic Turing machines in time bounded by the same function $u \in \mathfrak{F}$ (using the closure properties of $\mathfrak{F}$) in such a way that

$$Z \cap \{0,1\}^n = \text{Mod } H_n \restriction (\text{first } n \text{ bits}), \text{ and}$$

$$\bar{Z} \cap \{0,1\}^n = \text{Mod } H'_n \restriction (\text{first } n \text{ bits}).$$

Arguing now as in the proof of Lemma 4 one shows that $H_n \longrightarrow \neg H'_n$ is a tautology. Since $(III_{\mathfrak{F}})$ is assumed to be false, let Φ be computable by a deterministic Turing machine in time bounded by some function $b \in \mathfrak{F}$, with the property that whenever $B \longrightarrow C$ is a tautology, $\Phi(B,C)$ is an interpolant for $B \longrightarrow C$. Let $I_n = \Phi(H_n, \neg H'_n)$; the mapping $n \longmapsto I_n$ is computable by a deterministic Turing machine in time bounded by some function $d \in \mathfrak{F}$ (d can be obtained as a suitable composition of the functions u,b together with some polynomial). The mapping $n \longmapsto I_n$ is now proved to satisfy all our requirements by the same argument as in the end of the proof of Lemma 4. This completes the proof of Lemma 9. ■

Arguing now as in section 5, using Lemmas 8 and 9 and the closure properties of $\mathfrak{F}$, one easily produces a deterministic Turing machine accepting TAUT in time bounded by a function of $\mathfrak{F}$. Thus Theorem 7 is proved. ■

10. Further Topics.

In this final section we survey what is known on the complexity of Craig's interpolation theorem. We shall state a number of results concerning the rate of growth of interpolants, both in sentential and in first-order logic. Boolean expressions for sentential logic are particular words over alphabet Σ as defined in section 1. Sentences of first-order logic are understood as particular words over some suitable alphabet Σ', according to the familiar formation rules (see [1]). In sentential logic the precise determination of the rate of growth of interpolants is an open (and important) problem. The following Theorem states that if sentential interpolants turn out to grow polynomially, then every function which is computable in deterministic polynomial (Turing) time, has circuit depth proportional to the logarithm of the input length. This would provide a positive solution to a central open problem of computation theory (see [11]). See [10, 2.2] for the necessary background:

10.1 Theorem. Assume there exists a polynomial p such that whenever $B \longrightarrow C$ is a tautology in sentential logic, one can find an interpolant I with $|I| \leq p(|B| + |C|)$. Then for every function $f: \{0,1\}^* \longrightarrow \{0,1\}$ which is computable in deterministic polynomial time there is a sequence $F_1, F_2, \ldots$ of circuits, with F_n computing the restriction of f to $\{0,1\}^n$, such that, for some $c > 0$,

$$\text{depth } F_n \leq c \cdot \log_2 n \text{ , for each } n = 1,2,\ldots .$$

Proof. See [6 , Theorem 2.1] . ■

Recall (from [10 , 2.3.2]) that the delay complexity of a Boolean function $f: \{0,1\}^n \longrightarrow \{0,1\}$ is the depth of the smallest depth circuit for f (over our fixed basis $\{\wedge, \vee, \neg\}$). As usual, any Boolean expression B is regarded as a Boolean function over its own variables, via the identification 1 = "true" and 0 = "false"; the delay complexity of B is, roughly, the time needed for inputs to propagate to the output, in the fastest circuit computing B. The following Theorem then states that the time needed by the fastest

circuit to compute (the Boolean function corresponding to) any interpolant I for $B \to C$ may happen to be greater than the time needed to compute either of B or C:

10.2 Theorem. For infinitely many $d \in \mathbb{N}$ (and starting with some $d < 620$) there is a tautology $B \to C$ in sentential logic, with B and C having their delay complexity $\leq d$, such that every interpolant I has a delay complexity $d_I > d + (1/3)\log_2(d/2)$.

Proof. See [7, Theorem 2.5]. ■

As remarked above, in sentential logic there is at present no definitive estimate of the rate of growth of $|I|$ as a function of $|B|$ and $|C|$, where I is a smallest length interpolant for $B \to C$. (See [6, Theorem 1.9] for an upper bound, and try to improve it). By contrast in first-order logic we have:

10.3 Theorem.

(i) In the arithmetical hierarchy there is a Π_1-function $b: \mathbb{N} \to \mathbb{N}$ giving an upper bound for the length of first-order interpolants, i.e.,

(*) whenever $B \to C$ is valid in first-order logic, there is an interpolant I for $B \to C$ with $|I| \leq b(|B| + |C|)$.

(ii) No Σ_1-function (i.e., no recursive function) b can give an upper bound for $|I|$ as in (*).

Proof. (i) See [6, Theorem 3.1] . (ii) This can be extracted from [4 , Theorem 1] . ■

Due to its asymptotic character, the above Theorem 10.3 (ii) gives no information on the possible lengths of interpolants for short implications. The following is a non-asymptotic result:

10.4 Theorem. We can write down a valid implication in first-order logic, $B \to C$ with $|B|, |C| < 1145$ such that whenever I is an interpolant we have:

$$|I| > 2^{\cdot^{\cdot^{\cdot^{2}}}} \Big\uparrow \text{ seven two's .}$$

Proof. See [6, Theorem 3.5] . ■

References

[1] Chang C.C. and Keisler H.J., Model Theory (North-Holland, Amsterdam, second edition 1977).

[2] Cook S.A., The complexity of theorem proving procedures, In: Proceedings of the Third Annual ACM Symp. on the Theory of Computing, May 1971, pp.151-158.

[3] Cook S.A. and Reckhow R.A., The relative efficiency of propositional proof systems, Journal of Symb.Logic 44 (1979) 36-50.

[4] Friedman H., The complexity of explicit definitions, Advances in Mathematics 20 (1976) 18-29.

[5] Machtey M. and Young P., An Introduction to the General Theory of Algorithms (North-Holland, Amsterdam, third printing 1979).

[6] Mundici D., Complexity of Craig's interpolation, Annales Soc. Math.Pol., Series IV: Fundamenta Informaticae V.3-4 (1982) 261- 278.

[7] Mundici D.,A lower bound for the complexity of Craig's interpolants in sentential logic, Archiv math.Logik (1983) to appear.

[8] Mundici D., Duality between logics and equivalence relations, Transactions Amer.Math.Soc. 270 (1982) 111-129.

[9] Mundici D., Compactness, interpolation and Friedman's third problem, Annals of Mathematical Logic 22 (1982) 197-211.

[10] Savage J.E., The Complexity of Computing (Wiley, New York 1976).

[11] Schnorr C.P., The network complexity and the Turing machine complexity of finite functions, Acta Informatica 7 (1976) 95-107.