Logical Description of Monotone NP Problems

IAIN A. STEWART, *Department of Computer Science, University of Swansea, Swansea, SA2 8PP, UK.*
E-mail: I.A.STEWART@swansea.ac.uk

Abstract

We introduce the class of problems $NP_{C\text{-}RAT}$ accepted by polynomial time (nondeterministic) conjunctive random-access Turing machines (C-RATs) such machines crash when the oracle answers 'no' (the oracle is always the input structure). We show that $NP_{C\text{-}RAT}$ consists of all monotone problems in NP and we logically characterize this complexity class in two ways the first involves an extension of first-order logic using an operator corresponding to some problem (an approach initiated by Immerman), the second involves a restricted version of existential second-order logic (in the style of Fagin). In both logics, symbols belonging to the problem vocabulary are only allowed to occur positively. We also show that $NP_{C\text{-}RAT}$ possesses complete problems via monotone projection translations It had previously been shown that certain complete problems for NP (via logspace reductions) are unlikely to be complete for NP via monotone projection translations

Keywords: Descriptive complexity theory, monotone problems, computational complexity, NP

1 Introduction

Descriptive complexity theory involves the logical characterization of complexity classes: that is, problems are regarded as sets of finite structures and a complexity class is equated with those problems which can be described by the formulae of some logic. Various well-known complexity classes have been so characterized (see, for example [5, 9, 11, 13, 14, 22, 26, 27, 28, 29]). Not only does the logical characterization of some complexity class provide us with a machine-independent definition of the class but the whole logical approach to complexity theory gives a new environment in which to study traditional complexity theoretic questions and also gives rise to important new concepts which would otherwise have escaped discovery (in particular, the projection translation [13]: see [25] for an occasion where explicit use is made of projection translations and where other stronger reductions do not suffice). This logical approach has led to new results, the most well-known being Immerman's discovery that nondeterministic logspace is closed under complementation [14] (this result was independently proven by Szelepcsényi [31]). As examples of others, the logical characterization of the complexity class NSYMLOG [13] has led to new results concerning the generalized word problem for subgroups of free groups [24] and the structure of the projection translation has enabled related failures in networks of processors to be studied [25].

We show here that the complexity class $NP_{C\text{-}RAT}$, which consists of those problems accepted by polynomial time (nondeterministic) conjunctive random-access Turing machines, is actually equal to the class of all monotone problems in NP (we define our notion of monotone in Section 3). Furthermore, we obtain two logical characterizations of $NP_{C\text{-}RAT}$. These logical characterizations are of different types: the first involves an extension of first-order logic using an operator corresponding to some problem (as initiated by Immerman [13]), and the second involves a

restriction of existential second-order logic (in the style of Fagin [9]). We also show that the complexity class $NP_{C\text{-}RAT}$ has complete problems via monotone projection translations. Note that the fact that the complexity class $NP_{C\text{-}RAT}$ possesses these logical characterizations and also complete problems (via some reduction) lends weight to any assertion that it is a bona fide complexity class worthy of study. The complexity class $NP_{C\text{-}RAT}$ does not sit nicely within the hierarchy $L \subseteq P \subseteq NP$ for it contains problems such as HP(0,max) but not problems such as DTC(0,max) (these problems are defined in Section 2). It has been shown that if HP(0,max) is complete for NP via monotone projection translations then NP = co-NP [25]. The techniques employed in this paper enable us to move from a Fagin-type representation to an Immerman-type representation (moves in the other direction are usually easy).

We should add that monotone problems have been studied in database theory although the notion of monotone is slightly different from ours (see [1]). The basic approach also differs from that adopted here. In database theory the expressiveness of languages such as Datalog and its extensions is of prime concern (the availability of a built-in successor relation is usually disallowed in database theory), and these languages all express monotone queries (monotone in our sense and also in the sense of [1]). Consequently, a language such as Datalog gives rise to a complexity class of (monotone) problems. It has been shown that certain monotone (in the sense of [1]) queries can not be expressed in Datalog and its extensions [1, 17, 18]: these queries are usually NP-complete although some such polynomial time solvable queries also exist [1]. On the other hand, we look at the class of monotone problems contained within a well-known complexity class (namely NP) and derive alternative characterizations (amongst other things).

An important point to note is that our C-RATs are designed to work on finite structures over vocabularies in which there are no constant symbols. However, for various reasons, we sometimes need to include constant symbols in our vocabularies. Consequently, at the beginning of Sections 4, 5 and 6 we state the general rule for that particular section (that is, whether constant symbols are allowed or not).

In Section 2 we give the basic definitions and related results, before introducing our random-access Turing machines in Section 3. In Section 4 we establish some tools required for the main results, before proving these main results in Section 5. Our conclusions and directions for further research are given in Section 6. We mention that references are ordered as to when they appeared in print. Consequently, because time delays between submission and publication are not uniform, it may be the case that this ordering does not reflect the true chronological order.

2 Basic definitions and related results

In this section we describe how we extend first-order logic using an operator (or generalized quantifier) corresponding to some problem and we consider the notion of a logical translation between problems. We also mention some relevant existing results. The reader is referred to [26], for example, for more details.

In general, a *vocabulary* $\tau = \langle \underline{R}_1, \underline{R}_2, \ldots, \underline{R}_r, \underline{C}_1, \underline{C}_2, \ldots, \underline{C}_c \rangle$ is a tuple of *relation symbols* $\{\underline{R}_i : i = 1, 2, \ldots, r\}$, with $\underline{R}_i$ of *arity* a_i, and *constant symbols* $\{\underline{C}_i : i = 1, 2, \ldots, c\}$ although we sometimes disallow constant symbols in our vocabularies (the definitions in this section apply to vocabularies with or without constant symbols). A (*finite*) *structure* of *size* n over τ is a tuple $S = \langle \{0, 1, \ldots, n-1\}, R_1, R_2, \ldots, R_r, C_1, C_2, \ldots, C_c \rangle$ consisting of a *universe* $|S| = \{0, 1, ..., n-1\}$, *relations* $R_1, R_2, \ldots, R_r$ on the universe $|S|$ of arities $a_1, a_2, \ldots, a_r$, respectively, and *constants* $C_1, C_2, \ldots, C_c$ from the universe $|S|$. The size of some structure S is also denoted by $|S|$. We denote the set of all (finite) structures over τ by STRUCT(τ) (henceforth,

we do not distinguish between relations (resp. constants) and relation (resp. constant) symbols, and we assume that all structures are of size at least 2). A *problem of arity* t (≥ 0) over τ is a subset of

$$\mathrm{STRUCT}_t(\tau) = \{(S, \mathbf{u}) : S \in \mathrm{STRUCT}(\tau), \mathbf{u} \in |S|^t\}$$

(we usually refer to a set of strings as a decision problem as opposed to a problem, which we reserve for a set of finite structures). If Ω is some problem then $\tau(\Omega)$ denotes its vocabulary.

The language of the first-order logic $\mathrm{FO}_s(\tau)$ has as its (well-formed) formulae those formulae built, in the usual way, from the relation and constant symbols of τ, the binary relation symbols = and s, and the constant symbols 0 and *max*, using the logical connectives $\vee$, $\wedge$, and $\neg$, the variables $x, y, z_3, \ldots$, and the quantifiers $\forall$ and $\exists$ (we remark that $\mathrm{FO}_s(\tau)$ was denoted $\mathrm{FO}_{\leq}(\tau)$ in earlier papers: we have changed the notation as the symbol $\leq$ implies the existence of a built-in linear order which is not the case). Any formula ϕ of $\mathrm{FO}_s(\tau)$, with free variables those of the t-tuple $\mathbf{x}$, is interpreted in the set $\mathrm{STRUCT}_t(\tau)$, and for each $S \in \mathrm{STRUCT}(\tau)$ of size n and $\mathbf{u} \in |S|^t$, we have that $(S, \mathbf{u}) \models \phi(\mathbf{x})$ if and only if $\phi^S(\mathbf{u})$ holds, where $\phi^S(\mathbf{u})$ denotes the obvious interpretation of ϕ in S except that the binary relation symbol = is always interpreted in S as equality, the binary relation symbol s is interpreted as the successor relation on $|S|$, the constant symbol 0 is interpreted as $0 \in |S|$, the constant symbol max is interpreted as $n - 1 \in |S|$, and each variable of $\mathbf{x}$ is given the corresponding value from $\mathbf{u}$. (We sometimes omit the superscript in some interpreted formula such as ϕ^S if the structure in which ϕ is interpreted is understood, and we usually write $s(x, y)$ as $y = x + 1$ and $\neg(x = y)$ as $x \neq y$.) If we forbid the use of the successor relation in the logic $\mathrm{FO}_s(\tau)$ then we denote the resulting logic by $\mathrm{FO}(\tau)$. Also, $\mathrm{FO}_s = \cup_\tau \mathrm{FO}_s(\tau)$ (with FO defined similarly). The formula $\phi(\mathbf{x})$ *describes* (or *specifies* or *represents*) the problem

$$\{(S, \mathbf{u}) : (S, \mathbf{u}) \in \mathrm{STRUCT}_t(\tau), (S, \mathbf{u}) \models \phi(\mathbf{x})\}$$

of arity t, and two formulae are *equivalent* if and only if they describe the same problem.

Having detailed how we use first-order logic to describe problems, we now briefly illustrate how we extend first-order logic with new operators to attain greater expressibility. Let τ_2 be the vocabulary consisting of the binary relation symbol E: so, we may clearly consider structures over τ_2 as digraphs or graphs. Consider the problem DTC of arity 2:

$$\begin{aligned} \mathrm{DTC} \quad = \quad & \{(S, u, v) \in \mathrm{STRUCT}_2(\tau_2): \text{there is a path } p \text{ in the digraph } S \text{ from } u \text{ to } v \\ & \text{such that each vertex on } p, \text{ except perhaps } v, \text{ has out-degree 1 in } S, \\ & \text{i.e. } p \text{ is } \textit{deterministic}\}. \end{aligned}$$

Suppose that $\psi(\mathbf{x}, \mathbf{y})$ is a formula of FO_s, where $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of distinct variables. Then the digraph denoted by $\mathrm{DTC}[\lambda \mathbf{x}\mathbf{y}\psi(\mathbf{x}, \mathbf{y})]$ has its vertices indexed by k-tuples (over the domain of some relevant structure in which it is interpreted) and there is an edge from vertex $\mathbf{x}$ to vertex $\mathbf{y}$ if and only if there is a deterministic path from $\mathbf{x}$ to $\mathbf{y}$ in the digraph described by the relation $\psi(\mathbf{x}, \mathbf{y})$. Consequently, the formula

$$\mathrm{DTC}[\lambda \mathbf{x}\mathbf{y}\psi(\mathbf{x}, \mathbf{y})](\mathbf{0}, \mathbf{max})$$

describes all those structures (over the vocabulary in question) for which there is a deterministic path from vertex $\mathbf{0}$ to vertex $\mathbf{max}$ in the digraph described by $\psi(\mathbf{x}, \mathbf{y})$ ($\mathbf{0}$ (resp. $\mathbf{max}$) is the constant symbol 0 (resp. max) repeated k times). Clearly, we can re-apply the operator DTC to formulae already containing applications of DTC. If we allow an unlimited number of nested applications of the operator DTC then we denote the resulting logic by $(\pm\mathrm{DTC})^*[\mathrm{FO}_s]$ (this is the logic (FO+DTC) of [13]). We write $(\pm\mathrm{DTC})^k[\mathrm{FO}_s]$ (resp. $\mathrm{DTC}^*[\mathrm{FO}_s]$) to denote the sub-logic

of $(\pm\mathrm{DTC})^*[\mathrm{FO}_s]$ where all formulae have at most k nested applications of the operator DTC (resp. where no operator appears within the scope of a negation sign; that is, it appears positively): the sub-logic $\mathrm{DTC}^k[\mathrm{FO}_s]$ of $\mathrm{DTC}^*[\mathrm{FO}_s]$ is defined similarly. Needless to say, first-order logic can be extended by other operators as we shall soon see.

In order to compare logical descriptions of decision problems (that is, problems), we use the notion of a logical translation. Let $\tau' = \langle R_1, R_2, \ldots, R_r, C_1, C_2, \ldots, C_c\rangle$ be some vocabulary, where each R_i is a relation symbol of arity a_i and each C_j is a constant symbol, and let $L(\tau)$ be some logic over some vocabulary τ. Then the formulae of $\Sigma = \{\phi_i(\mathbf{x}_i), \psi_j(\mathbf{y}_j) : i = 1, 2, \ldots, r, j = 1, 2, \ldots, c\} \subseteq L(\tau)$, where

(i) each formula ϕ_i (resp. ψ_j) is over the qa_i (resp. q) distinct variables $\mathbf{x}_i$ (resp. $\mathbf{y}_j$), for some fixed positive integer q;

(ii) for each $j = 1, 2, \ldots, c$ and for each structure $S \in \mathrm{STRUCT}(\tau)$

$$S \models \exists x_1 \exists x_2 \ldots \exists x_q(\psi_j(x_1, x_2, \ldots, x_q) \wedge \forall y_1 \forall y_2 \ldots \forall y_q(\psi_j(y_1, y_2, \ldots, y_q) \Leftrightarrow (x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots \wedge x_q = y_q))),$$

are called τ'-*descriptive*. For each $S \in \mathrm{STRUCT}(\tau)$, the τ'-*translation of* S *with respect to* Σ is the structure $S' \in \mathrm{STRUCT}(\tau')$ with universe $|S|^q$ defined as follows: for all $i = 1, 2, \ldots, r$ and for any tuples $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{a_i}\} \subseteq |S'| = |S|^q$,

$$R_i^{S'}(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{a_i}) \text{ holds if and only if } (S, (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{a_i})) \models \phi_i(\mathbf{x}_i)$$

and, for all $j = 1, 2, \ldots, c$ and for any tuple $\mathbf{u} \in |S'| = |S|^q$,

$$C_j^{S'} = \mathbf{u} \text{ if and only if } (S, \mathbf{u}) \models \psi_j(\mathbf{y}_j)$$

(tuples are ordered lexicographically, with $(0, 0, \ldots, 0) < (0, 0, \ldots, 1) < (0, 0, \ldots, 2) < \ldots$, and so on: note that the condition on each ψ_j ensures that the constant $C_j^{S'}$ is well defined). Let Ω and Ω' be problems over the vocabularies τ and τ', respectively. Let S be a set of τ'-descriptive formulae from some logic $L(\tau)$, and for each $S \in \mathrm{STRUCT}(\tau)$, let $\sigma(S) \in \mathrm{STRUCT}(\tau')$ denote the τ'-translation of S with respect to Σ. Then Ω' is an L-*translation* of Ω if and only if for each $S \in \mathrm{STRUCT}(\tau)$,

$$S \in \Omega \text{ if and only if } \sigma(S) \in \Omega'.$$

Let $\phi \in \mathrm{FO}_s(\tau)$, for some vocabulary τ, be of the form

$$\bigvee\{\alpha_i \wedge \beta_i : i \in I\}$$

for some finite index set I, where

(i) each α_i is a conjunction of the logical atomic relations s, $=$, and their negations, and no symbol of τ appears in any α_i;

(ii) each β_i is atomic or negated atomic;

(iii) if $i \neq j$ then α_i and α_j are mutually exclusive.

Then ϕ is a *projective formula*. If the successor relation symbol s does not appear in ϕ (that is, $\phi \in \mathrm{FO}(\tau)$) then ϕ is a *projective formula without successor*, and if each of the β_i (above) is atomic then ϕ is a *monotone projective formula*. Consequently, we clearly have the notions of one problem being a *first-order translation*, a *quantifier-free translation*, a *projection translation*, and a *monotone projection translation* of another, as well as all of these without successor. If $\Omega_0, \Omega_1, \ldots, \Omega_m$ are problems such that Ω_{i+1} is a projection translation of Ω_i, for each $i = 0, 1, \ldots, m-1$, then there is an *iterated projection translation* from Ω_0 to Ω_m.

Having set up the logical machinery, we now mention how operators have already been added to first-order logic so as to 'capture' well-known complexity classes. Complexity classes are generally considered to be sets of strings over the alphabet $\{0,1\}$ recognized by various resource-bounded computing devices. In order to compare complexity classes and classes of problems, it is necessary to relate a set of strings with a problem and vice versa. Let $S \in \text{STRUCT}(\tau)$, for some vocabulary $\tau = \langle R_1, R_2, \ldots, R_\tau, C_1, C_2, \ldots, C_c \rangle$, where each R_i is a relation symbol of arity a_i and each C_j is a constant symbol. Then the encoding $e_\tau(S)$ of S is defined as follows:

the relations $R_1^S, R_2^S, \ldots, R_\tau^S$ are encoded in order with each R_i^S encoded as a sequence of n^{a_i} 0s and 1s, denoting whether $R_i^S(0,0,\ldots,0)$ holds, whether $R_i^S(0,0,\ldots,1)$ holds, ..., and whether $R_i^S(n-1,n-1,\ldots,n-1)$ holds (where $|S| = n$);

the constants $C_1^S, C_2^S, \ldots, C_c^S$ are encoded in order with each C_j^S encoded as its binary representation.

If Ω is some problem over τ then we define $e_\tau(\Omega) = \{e_\tau(S) : S \in \Omega\} \subseteq \{0,1\}^*$.

So, the set of strings over $\{0, 1\}$ corresponding to the problem Ω, over some vocabulary τ, is $e_\tau(\Omega)$. Note that the set of strings corresponding to Ω depends upon the choice of encoding scheme (we can just as well choose other encoding schemes). Conversely, given a set of strings A over $\{0, 1\}$ there might be many problems Ω, over different vocabularies τ, such that $e_\tau(\Omega) = A$: there is certainly one, namely the problem Ω over the vocabulary τ_1 consisting of one relation symbol M of arity 1 such that for each $S \in \text{STRUCT}(\tau_1)$, $S \in \Omega$ if and only if the binary string described by M^S is in A (the binary string described by M^S has length $|S|$ with the ith symbol being a 1 (resp. 0) if and only if the relation $M^S(i)$ holds (resp. does not hold)). Consequently, a complexity class CC is identified with a set of problems *Prob* if and only if

(i) for each $A \in \text{CC}$, there is some $\Omega \in \mathit{Prob}$ over some vocabulary τ such that $A = e_\tau(\Omega)$;

(ii) for each $\Omega \in \mathit{Prob}$, we have that $e_\tau(\Omega) \in \text{CC}$.

If CC is identified with *Prob* then we write $\text{CC} = \mathit{Prob}$. So, we can speak of a problem being in a complexity class and of a complexity class consisting of a collection of problems.

It is with logics capturing NP that we are concerned here, and, to this end, the following problems are among those which have previously been considered (below, the vocabulary $\tau_{2,2} = \langle P, N \rangle$, where P and N are binary relation symbols, encodes Boolean formulae in c.n.f. via, the literal X_i (resp. $\neg X_i$) is in clause C_j if and only if $P(i,j)$ (resp. $N(i,j)$) holds):

HP	=	$\{(S,u,v) \in \text{STRUCT}_2(\tau_2)$: there is a Hamiltonian path in the digraph S from vertex u to vertex $v\}$;
3COL	=	$\{S \in \text{STRUCT}(\tau_2)$: the graph S can be 3-coloured$\}$;
SAT	=	$\{S \in \text{STRUCT}(\tau_{2,2})$: the c.n.f. formula S is satisfiable$\}$;
$\leq$3SAT	=	$\{S \in \text{STRUCT}(\tau_{2,2})$: S encodes a c.n.f. Boolean formula with at most 3 literals in each clause and there is a satisfying truth assignment for $S\}$;
3SAT	=	$\{S \in \text{STRUCT}(\tau_{2,2})$: S encodes a c.n.f. Boolean formula with exactly 3 literals in each clause and there is a satisfying truth assignment for $S\}$.

The problem HP(0,max) is defined as

HP(0,max)	=	$\{S \in \text{STRUCT}(\tau_2)$: there is a Hamiltonian path in the digraph S, of size n, from 0 to $n-1\}$

(with DTC(0,max), etc., defined similarly).

THEOREM 2.1 ([22, 26, 27, 28, 29])

(i) $\text{NP} = \text{HP}^1[\text{FO}_s] = \text{HP}^*[\text{FO}_s]$.

(ii) $\text{NP} = \text{3COL}^1[\text{FO}_s] = \text{SAT}^1[\text{FO}]$.

(iii) $L^{NP} = (\pm HP)^1[FO_s] = (\pm HP)^*[FO_s] = 3COL^2[FO_s] = (\pm 3COL)^1[FO_s] = (\pm 3COL)^*[FO_s] = SAT^2[FO] = (\pm SAT)^1[FO] = (\pm SAT)^*[FO]$.

(iv) SAT is complete for NP via projection translations without successor, HP(0,max), $\leq$3SAT, and 3COL are complete for NP via projection translations, and 3SAT is complete for NP via iterated projection translations.

The complexity class L^{NP} consists of all those sets of strings accepted by logspace oracle machines with an oracle in NP.

We have two remarks. First, as we have done in Theorem 2.1, we use L to denote some logic and also the problems describable by the sentences of that logic. Second, Theorem 2.1 holds whether we allow our vocabularies to have constant symbols or not (the built-in constant symbols 0 and max are always present though).

It is open as to whether 3SAT is complete for NP via projection translations, although Dahlhaus has shown that a different logical encoding (not over the vocabulary $\tau_{2,2}$) of the 3-satisfiability problem where all non-empty clauses have exactly 3 literals is not complete for NP via similar logical reductions called interpretative reductions ([8]): using the same techniques, we can show that 3COL is not either [29]). When considering weak logical reductions such as projection translations, different logical encodings seem to give rise to problems with different properties.

For logical characterizations of complexity classes within NP, and related completeness results, the reader is referred to [5, 11, 12, 13, 14, 15, 16, 23].

3 Random-access Turing machines

We now briefly explain how the usual nondeterministic Turing machine model of computation can be adapted so that the input can be randomly accessed. Our adapted model is similar to those in [6, 20] although it is designed to work on inputs in the form of finite structures as opposed to strings of symbols (this is an unimportant detail as strings of 0s and 1s can be viewed as finite structures over the vocabulary consisting of a unary relation symbol).

A *random-access Turing machine* (*RAT*) M over a vocabulary τ consisting entirely of relation symbols is essentially a (one work tape) nondeterministic Turing machine with its input tape and head replaced by oracle tapes and heads, with an oracle tape and head for each relation symbol of τ. The oracle tapes provide M with a means for accessing the input structure which is always the oracle. Let $S \in \text{STRUCT}(\tau)$ be of size n. We assume that when S is input to M, the binary representation of n is initially written in the first $\lfloor \log n \rfloor + 1$ cells of the work tape (and so M knows the size of its input structure). Let R be some relation symbol of τ of arity a. On input S, if M needs to know whether $R^S(x_1, x_2, \ldots, x_a)$ holds, where $x_1, x_2, \ldots, x_a \in |S|$, then M writes the binary representation of $(x_1, x_2, \ldots, x_a)$ on the oracle tape corresponding to R. The oracle answers 'yes' if and only if $R^S(x_1, x_2, \ldots, x_a)$ holds. If ever the queried string on the oracle tape corresponding to R does not correspond to a tuple $(x_1, x_2, \ldots, x_a)$ then the computation crashes (that is, halts without accepting). The input S is accepted if and only if there is some computation of M on S leading to the accepting state. Consequently, the problem over τ accepted by M consists of all those structures of $\text{STRUCT}(\tau)$ accepted by M. We define the time taken and space used by a RAT as usual (for space, we count the number of work tape cells used). The complexity class $\text{NP}_{\text{RAT}}(\tau)$ is the class of all problems over τ accepted by a polynomial time RAT over τ: it is easy to see that $\text{NP}_{\text{RAT}} = \text{NP}$, where NP_{RAT} is the class of all problems accepted by a polynomial time RAT (over some vocabulary consisting entirely of relation symbols).

We are more concerned with when a RAT M is restricted to operate *conjunctively*. By this we mean that if ever the answer to some oracle query is 'no' then the computation crashes. If M is a polynomial time conjunctive RAT (C-RAT) then there is an equivalent polynomial time C-RAT M' with the property that every computation is split into two parts. The first part consists of M' simulating M except that no oracle queries are made: the queries are actually computed and stored on the work tape, with M' assuming that all the answers to the (unmade) oracle queries are 'yes'. If the simulation of M halts in the accepting state then in the second part of its computation, M' queries the oracle for each of the remembered oracle queries and accepts if and only if all the oracle answers are 'yes' (we may assume that the simulation does indeed halt as M is a polynomial time C-RAT and can be augmented with a clock). Consequently, throughout the paper we assume that any polynomial time C-RAT computes all of its oracle queries before actually making any. The complexity class $\mathrm{NP}_{\text{C-RAT}}(\tau)$ is the class of all problems over τ accepted by a polynomial time C-RAT over τ, with the complexity class $\mathrm{NP}_{\text{C-RAT}}$ as expected.

Examples of problems that are in $\mathrm{NP}_{\text{C-RAT}}$ are TC(0,max) and HP(0,max) (TC is the problem analogous to DTC where there should only exist a path in a digraph between two vertices instead of a deterministic one). However, consider DTC(0,max). Suppose that DTC(0,max) $\in \mathrm{NP}_{\text{C-RAT}}(\tau_2)$ and that M is a polynomial time C-RAT over τ_2 accepting DTC(0,max). Let S be the digraph on the vertices $\{0,1,2\}$ with one edge $(0,2)$, and let S' be the digraph on the vertices $\{0,1,2\}$ with edges $(0,1)$ and $(0,2)$. Then $S \in$ DTC(0,max) but $S' \notin$ DTC(0,max). It should be clear that the queries made by any C-RAT depend only on the size of the input structure (given some nondeterministic guesses and our assumption that all oracle queries are computed before they are made): that is, later queries do not depend upon the answers to earlier ones. Consequently, as M accepts S then M must also accept S' which yields a contradiction. So, DTC(0,max) $\notin \mathrm{NP}_{\text{C-RAT}}$.

Consider also a problem such as PLANARTC(0,max), where some structure S over τ_2 is in PLANARTC(0,max) if and only if S encodes a planar digraph in which there is a path from vertex 0 to vertex max. Clearly, there is a polynomial time C-RAT M with the property that amongst the planar digraphs M accepts exactly those with a path from vertex 0 to vertex max. However, M also accepts some structures over τ_2 which do not encode planar digraphs and so PLANARTC(0,max) $\notin \mathrm{NP}_{\text{C-RAT}}$.

We immediately see that $\mathrm{NP}_{\text{C-RAT}}$ is a complexity class similar to the class of problems defined by the sentences of existential monadic second-order logic (see [2]), for $\mathrm{NP}_{\text{C-RAT}}$ contains many ('hard') NP-complete (via logspace reductions) problems but does not contain some ('easy') problems solvable in P. We remark that there are problems in $\mathrm{NP}_{\text{C-RAT}}$ which can not be described by a sentence of existential monadic second-order logic (TC(0,max), for example: see [2]), and there are problems which can be described by a sentence of existential monadic second-order logic that are not in $\mathrm{NP}_{\text{C-RAT}}$ (3COL, for example: the fact that 3COL is not in $\mathrm{NP}_{\text{C-RAT}}$ can be ascertained by reasoning as we did for DTC(0,max) above).

Let $S_1, S_2 \in \mathrm{STRUCT}(\tau)$, for some vocabulary τ consisting entirely of relation symbols. We say that S_2 is a *relational refinement* of S_1 if and only if $|S_1| = |S_2|$ and for every relation symbol R of τ, of arity a, say, and for every $\mathbf{u} \in |S_1|^a = |S_2|^a$

$$\text{if } R^{S_1}(\mathbf{u}) \text{ holds then } R^{S_2}(\mathbf{u}) \text{ holds.}$$

Any problem Ω over the vocabulary τ consisting entirely of relation symbols is *monotone* if for any $S_1, S_2 \in \mathrm{STRUCT}(\tau)$ with S_2 a relational refinement of S_1, $S_1 \in \Omega$ implies $S_2 \in \Omega$. As should be immediately apparent from the above discussion, if Ω is any problem in $\mathrm{NP}_{\text{C-RAT}}$ then Ω is monotone. Conversely, let Ω be some monotone problem in NP and let M be a polynomial time C-RAT defined as follows:

for each relation symbol R of $\tau(\Omega)$, of arity k, guess a set G_R of k-tuples over $|S|$, where S is the input structure;
store these guesses in memory and let S' be the structure over $\tau(\Omega)$ with universe $|S|$ and where for each relation symbol R of $\tau(\Omega)$, of arity k, and each $\mathbf{u} \in |S|^k$, $R^{S'}(\mathbf{u})$ holds if and only if $\mathbf{u} \in G_R$;
check to see whether $S' \in \Omega$;
if $S' \in \Omega$ then query every instance $R^S(\mathbf{u})$ where $\mathbf{u} \in G_R$, and accept only if all the oracle answers are 'yes': otherwise reject.

Suppose some structure S over $\tau(\Omega)$ is accepted by M. Then S is a relational refinement of some $S' \in \Omega$. As Ω is monotone then $S \in \Omega$. Conversely, suppose $S \in \Omega$. For each relation symbol R of $\tau(\Omega)$, of arity k, let the guessed set G_R be exactly those tuples $\mathbf{u} \in |S|^k$ such that $R^S(\mathbf{u})$ holds. Then S is accepted by M. Hence, we have the following result.

THEOREM 3.1
The complexity class $\mathrm{NP}_{\text{C-RAT}}$ consists of all those monotone problems in NP.

Note that as we are working in a logical framework, given some problem we can immediately deduce whether it is monotone or not. However, suppose we are given some decision problem such as the satisfiability problem [10]. It is usually not clear as to whether this decision problem is monotone or not for it is not usually specified whether empty clauses are allowed; and if they are, they are often simply ignored. Our logical framework forces us to be much more explicit with our definitions: for example, the problem SAT is clearly not monotone.

We end by reminding the reader that RATs are designed to accept problems over vocabularies consisting entirely of relation symbols. (Note that if we were to allow arbitrary vocabularies and model constants as relations then no problem over any vocabulary containing a constant symbol would be monotone.)

4 Path system accessibility

We begin by considering a logical encoding of PATH SYSTEM ACCESSIBILITY ([10]), the well-known decision problem that is defined as follows: an instance (that is, a *path system*) of size n is a set V of n vertices, a relation $R \subseteq V \times V \times V$, a *source* $u \in V$, and a *sink* $v \in V$ (different from u), and a yes-instance is an instance where the sink is accessible, with a vertex w being *accessible* if w is the source or if $R(x, y, w)$ holds for some accessible vertices x and y. In deducing that a vertex w is accessible by exhibiting accessible vertices x and y such that $R(x, y, w)$ holds, we sometimes say that we have *applied the rule* (x, y, w): we remark that x and y need not be distinct. We encode this decision problem as the problem PS over the vocabulary $\tau_3 = \langle R \rangle$, where R is a relation symbol of arity 3, in the obvious manner with 0 (resp. max) representing the source (resp. sink).

REMARK 4.1
Throughout this section, constant symbols are allowed in our vocabularies except where otherwise stated.

THEOREM 4.2
Every formula Φ of the logic $\mathrm{PS}^*[\mathrm{FO}_s]$ is equivalent to one of the form

$$\mathrm{PS}[\lambda \mathbf{x}\mathbf{y}\mathbf{z}\psi]$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are k-tuples of variables, for some k, and where ψ is a projective formula.

PROOF. We proceed by induction on the number of symbols in Φ (as is done, for example, in Theorem 3.3 of [13]). In each case below, any newly introduced variables are assumed to be distinct and different from any other variables occurring in existing formulae (within the case).
Case (i). Φ is atomic or negated atomic. Then Φ is equivalent to

$$\mathrm{PS}[\lambda xyz(x = 0 \wedge y = 0 \wedge z = max \wedge \Phi)].$$

Case (ii). $\Phi \equiv \exists w \mathrm{PS}[\lambda \mathbf{xyz}\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, w)]$, where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are k-tuples of variables, for some k, w is a variable not occurring in $\mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$, and ϕ is projective. Let $\psi(a, b, c, \mathbf{x}, a', b', c', \mathbf{y}, a'', b'', c'', \mathbf{z})$ be defined as

$$\begin{aligned}(a = a' = a'' = 0 \wedge b = b' = b'' = max \wedge c = c' = c'' \wedge \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, c))\\ \vee (a = a' = a'' = 0 \wedge b = b' = 0 \wedge b'' = max \wedge c = c' = 0 \wedge \mathbf{x} = \mathbf{y} = \mathbf{z} = \mathbf{0})\\ \vee (a = a' = 0 \wedge a'' = max \wedge b = b' = b'' = max \wedge c = c' \wedge c'' = max\\ \wedge \mathbf{x} = \mathbf{y} = \mathbf{z} = \mathbf{max})\end{aligned}$$

(the meaning of the 'shorthand' employed above should be obvious). Then ψ is equivalent to a projective formula. Also, given some relevant structure S of size n, ψ^S essentially describes a path system consisting of disjoint copies of the path systems described by

$$\lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, 0), \lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, 1), \ldots, \lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, n-1)$$

joined by adding in a source u, a sink v, and rules

$$\begin{gathered}(u, u, \mathbf{0}_0), (u, u, \mathbf{0}_1), \ldots, (u, u, \mathbf{0}_{n-1}),\\ (\mathbf{max}_0, \mathbf{max}_0, v), (\mathbf{max}_1, \mathbf{max}_1, v), \ldots, (\mathbf{max}_{n-1}, \mathbf{max}_{n-1}, v)\end{gathered}$$

where $\mathbf{0}_i$ (resp. $\mathbf{max}_i$) is the source (resp. sink) of the path system described by $\lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, i)$, for all $i = 0, 1, \ldots, n-1$. Clearly, Φ is equivalent to

$$\mathrm{PS}[\lambda(a, b, c, \mathbf{x})(a', b', c', \mathbf{y})(a'', b'', c'', \mathbf{z})\psi].$$

Case (iii). $\Phi \equiv \forall w \mathrm{PS}[\lambda \mathbf{xyz}\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, w)]$, where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are k-tuples of variables, for some k, w is a variable not occurring in $\mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$, and ϕ is projective. Let $\psi(a, b, \mathbf{x}, a', b', \mathbf{y}, a'', b'', \mathbf{z})$ be defined as

$$\begin{aligned}(a = a' = a'' = 0 \wedge b = b' = b'' \wedge \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, b))\\ \vee (a = a' = a'' = 0 \wedge b = b' \neq max \wedge b'' = b + 1 \wedge \mathbf{x} = \mathbf{y} = \mathbf{max} \wedge \mathbf{z} = \mathbf{0})\\ \vee (a = a' = 0 \wedge a'' = max \wedge b = b' = b'' = max \wedge \mathbf{x} = \mathbf{y} = \mathbf{z} = \mathbf{max}).\end{aligned}$$

Then ψ is equivalent to a projective formula. Also, given some relevant structure $|S|$ of size n, ψ^S essentially describes a path system consisting of disjoint copies of the path systems described by

$$\lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, 0), \lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, 1), \ldots, \lambda \mathbf{xyz}\phi^S(\mathbf{x}, \mathbf{y}, \mathbf{z}, n-1)$$

joined by adding in a sink v and rules

$$\begin{aligned}&(\mathbf{max}_0, \mathbf{max}_0, \mathbf{0}_1), (\mathbf{max}_1, \mathbf{max}_1, \mathbf{0}_2), \ldots\\ &\quad \ldots, (\mathbf{max}_{n-2}, \mathbf{max}_{n-2}, \mathbf{0}_{n-1}), (\mathbf{max}_{n-1}, \mathbf{max}_{n-1}, v)\end{aligned}$$

where the source is $\mathbf{0}_0$ (and the notation is as in Case (ii)). Clearly, Φ is equivalent to

$$\mathrm{PS}[\lambda(a,b,\mathbf{x})(a',b',\mathbf{y})(a'',b'',\mathbf{z})\psi].$$

Case (iv). $\Phi \equiv \mathrm{PS}[\lambda\mathbf{xyz}\mathrm{PS}[\lambda\mathbf{uvw}\phi(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{w})]]$, where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are k-tuples and $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are m-tuples of variables, for some k and m, with all variables distinct, and where ϕ is projective. We may clearly assume that $k = m$. Let $\psi(a,b,\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w},a',b',\mathbf{x}',\mathbf{y}',\mathbf{z}',\mathbf{w}',a'',b'',\mathbf{x}'',\mathbf{y}'',\mathbf{z}'',\mathbf{w}'')$, for new k-tuples of variables $\mathbf{x}',\mathbf{y}',\mathbf{z}',\mathbf{w}',\mathbf{x}'',\mathbf{y}'',\mathbf{z}''$, and $\mathbf{w}''$, be defined as

$$\begin{aligned}
&(a = a' = a'' = 0 \wedge b = b' = 0 \wedge b'' = max \wedge \mathbf{x} = \mathbf{x}'' \wedge \mathbf{x}' = \mathbf{y}'' \wedge \mathbf{y} = \mathbf{y}' = \mathbf{0}\\
&\quad \wedge \mathbf{z} = \mathbf{z}' = \mathbf{0} \wedge \mathbf{w} = \mathbf{w}' = \mathbf{w}'' = \mathbf{0})\\
&\vee(a = a' = a'' = 0 \wedge b = b' = b'' = max \wedge \mathbf{x} = \mathbf{x}' = \mathbf{x}'' \wedge \mathbf{y} = \mathbf{y}' = \mathbf{y}''\\
&\quad \wedge \mathbf{z} = \mathbf{z}' = \mathbf{z}'' \wedge \phi(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w},\mathbf{w}',\mathbf{w}''))\\
&\vee(a = a' = a'' = 0 \wedge b = b' = max \wedge b'' = 0 \wedge \mathbf{x} = \mathbf{x}' \wedge \mathbf{x}'' = \mathbf{z} = \mathbf{z}' \wedge \mathbf{y} = \mathbf{y}'\\
&\quad \wedge \mathbf{y}'' = \mathbf{0} \wedge \mathbf{z}'' = \mathbf{0} \wedge \mathbf{w} = \mathbf{w}' = \mathbf{max} \wedge \mathbf{w}'' = \mathbf{0})\\
&\vee(a = a' = 0 \wedge a'' = max \wedge b = b' = 0 \wedge b'' = max \wedge \mathbf{x} = \mathbf{x}' = \mathbf{x}'' = \mathbf{max}\\
&\quad \wedge \mathbf{y} = \mathbf{y}' = \mathbf{0} \wedge \mathbf{y}'' = \mathbf{max} \wedge \mathbf{z} = \mathbf{z}' = \mathbf{0} \wedge \mathbf{z}'' = \mathbf{max} \wedge \mathbf{w} = \mathbf{w}' = \mathbf{0}\\
&\quad \wedge \mathbf{w}'' = \mathbf{max}).
\end{aligned}$$

Then ψ is equivalent to a projective formula. Also, given some relevant structure S of size n, ψ^S essentially describes a path system P as follows:

for each $\mathbf{x},\mathbf{y},\mathbf{z} \in |S|^k$, let $p(\mathbf{x},\mathbf{y},\mathbf{z})$ be the path system described by

$$\lambda\mathbf{ww'w''}\phi^S(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w},\mathbf{w}',\mathbf{w}'')$$

(all these $p(\mathbf{x},\mathbf{y},\mathbf{z})$'s are disjoint and are described by the second of the four 'conjunctions' of ψ);

there is a disjoint set of vertices indexed by $\mathbf{x} \in |S|^k$; there are rules

$$(\mathbf{x},\mathbf{y},0_{p(\mathbf{x},\mathbf{y},\mathbf{z})}),(max_{p(\mathbf{x},\mathbf{y},\mathbf{z})},max_{p(\mathbf{x},\mathbf{y},\mathbf{z})},\mathbf{z})$$

for each $\mathbf{x},\mathbf{y},\mathbf{z} \in |S|^k$, where $0_{p(\mathbf{x},\mathbf{y},\mathbf{z})}$ (resp. $max_{p(\mathbf{x},\mathbf{y},\mathbf{x})}$) is the source (resp. sink) of $p(\mathbf{x},\mathbf{y},\mathbf{z})$ (the rules of the first type are described by the first 'conjunction' of ψ, whilst those of the second are described by the third);

the source is $\mathbf{0}$, the sink is a new vertex v, and there is a rule $(\mathbf{max},\mathbf{max},v)$ (this rule is described by the fourth 'conjunction' of ψ).

For any vertices $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ of P, we can deduce that v is accessible using only the fact that $\mathbf{x}$ and $\mathbf{y}$ are accessible if and only if we can deduce that $max_{p(\mathbf{x},\mathbf{y},\mathbf{z})}$ (resp. $max_{p(\mathbf{y},\mathbf{x},\mathbf{z})}$) of the path system $p(\mathbf{x},\mathbf{y},\mathbf{z})$ (resp. $p(\mathbf{y},\mathbf{x},\mathbf{z})$) is accessible using only the fact that $0_{p(\mathbf{x},\mathbf{y},\mathbf{z})}$ (resp. $0_{p(\mathbf{y},\mathbf{x},\mathbf{z})}$) is accessible. Hence, Φ is equivalent to

$$\mathrm{PS}[\lambda(a,b,\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w})(a',b',\mathbf{x}',\mathbf{y}',\mathbf{z}',\mathbf{w}')(a'',b'',\mathbf{x}'',\mathbf{y}'',\mathbf{z}'',\mathbf{w}'')\psi].$$

The cases where Φ is a disjunction or a conjunction of two sentences of the required form can be handled as in cases (ii) and (iii), respectively. The result follows by induction. ∎

DEFINITION 4.3

Let Ω and Ω' be problems over the vocabularies τ and τ', respectively. Suppose that there is a logspace transducer M, computing the function f, and some positive integer k such that for each $\omega \in \{0,1\}^*$

(i) $\omega \in e_\tau(\Omega)$ if and only if $f(\omega) \in e_{\tau'}(\Omega')$;
(ii) if $\omega = e_\tau(S)$, for some $S \in \text{STRUCT}(\tau)$ of size n, then $f(\omega) = e_{\tau'}(S')$ for some $S' \in \text{STRUCT}(\tau')$ of size n^k.
Then Ω is said to be *logspace reducible to* Ω' *with exponent* k.

By proceeding almost exactly as in the proof of Proposition 4.2 of [22], we can show that every problem in P is logspace reducible to PS with exponent k, for some k (dependent upon the problem in P), as PS is complete for P via logspace reductions ([7]: see also [10]).

LEMMA 4.4
$(\pm\text{TC})^*[\text{FO}_s] \subseteq \text{PS}^*[\text{FO}_s]$ (as classes of problems).

PROOF. By [13, 14], every problem in $(\pm\text{TC})^*[\text{FO}_s]$ can be described by a sentence of the form

$$\text{TC}[\lambda \mathbf{xy}\psi(\mathbf{x}, \mathbf{y})](\mathbf{0}, \mathbf{max})$$

where $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of variables, for some k, ψ is a projective formula, and $\mathbf{0}$ (resp. $\mathbf{max}$) is the constant symbol 0 (resp. max) repeated k times. Consider the formula $\phi(u, \mathbf{x}, u', \mathbf{x}', u'', \mathbf{x}'')$, where $\mathbf{x}$, $\mathbf{x}'$, and $\mathbf{x}''$ are k-tuples of variables and u, u', and u'' are variables (with all variables distinct), defined as

$$\begin{aligned}(u = u' = 0 \wedge u'' = max \wedge \mathbf{x} = \mathbf{x}' = \mathbf{x}'' = \mathbf{0}) \\ \vee (u = u'' = 0 \wedge u' = max \wedge \mathbf{x}' = \mathbf{0} \wedge \psi(\mathbf{x}, \mathbf{x}'')) \\ \vee (u = u' = 0 \wedge u'' = max \wedge \mathbf{x} = \mathbf{x}' = \mathbf{x}'' = \mathbf{max}).\end{aligned}$$

Clearly, $\text{TC}[\lambda \mathbf{xy}\psi(\mathbf{x}, \mathbf{y})](\mathbf{0}, \mathbf{max})$ and $\text{PS}[\lambda(u, \mathbf{x})(u', \mathbf{x}')(u'', \mathbf{x}'')\phi]$ are equivalent, and so the result follows. ∎

COROLLARY 4.5
$(\pm\text{PS})^*[\text{FO}_s] = \text{PS}^*[\text{FO}_s] = \text{PS}^1[\text{FO}_s] = \text{P}$ and PS is complete for P via projection translations.

PROOF. By Theorem 4.2, $\text{PS}^*[\text{FO}_s] \subseteq \text{P}$, and as P is closed under complementation, a simple induction yields that $(\pm\text{PS})^*[\text{FO}_s] \subseteq \text{P}$. Let Ω be some problem in P. By the remark preceding Lemma 4.4 and Proposition 3.1 of [22], PS is a $(\pm\text{DTC})^*[\text{FO}_s]$-translation of Ω (and so also a $(\pm\text{TC})^*[\text{FO}_s]$-translation). Hence, Ω can be described by a sentence of the form

$$\text{PS}[\lambda \mathbf{xyz}\text{TC}[\lambda \mathbf{uv}\psi(\mathbf{u}, \mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z})](\mathbf{0}, \mathbf{max})]$$

where ψ is a first-order formula. By Lemma 4.4, Ω can then be described by a sentence of the form

$$\text{PS}[\lambda \mathbf{xyz}\text{PS}[\lambda \mathbf{u'v'w'}\phi(\mathbf{u'}, \mathbf{v'}, \mathbf{w'}, \mathbf{x}, \mathbf{y}, \mathbf{z})]]$$

where ϕ is a first-order formula. Hence, by Theorem 4.2, Ω can be described by a sentence of the form

$$\text{PS}[\lambda \mathbf{x'y'z'}\phi'(\mathbf{x'}, \mathbf{y'}, \mathbf{z'})]$$

where ϕ' is a projective formula. So $\text{P} \subseteq \text{PS}^1[\text{FO}_s] \subseteq \text{PS}^*[\text{FO}_s] \subseteq (\pm\text{PS})^*[\text{FO}_s] \subseteq \text{P}$ and the result follows. ∎

Corollary 4.5 gives another problem complete for P via projection translations (the only other one known is the problem ATC of [13]) and we use Corollary 4.5 explicitly in the next section. We also use Corollary 4.8.

REMARK 4.6
Corollary 4.5 holds whether we allow constant symbols in our vocabularies or not.

DEFINITION 4.7
Let τ be some vocabulary. For a logic such as $\mathrm{PS}^*[\mathrm{FO}_s(\tau)]$, we define the sub-logic $\mathrm{PS}^*[\mathrm{FO}_s^+(\tau)]$ to be all those formulae of $\mathrm{PS}^*[\mathrm{FO}_s(\tau)]$ where any relation or constant symbol of τ appears positively; that is, not within the scope of a negation sign (we regard the operator PS as a generalized quantifier). The logic $\mathrm{PS}^*[\mathrm{FO}_s^+] = \cup_\tau \mathrm{PS}^*[\mathrm{FO}_s^+(\tau)]$.

For example, the formula

$$\forall x(\mathrm{PS}[\lambda uvw((R(u,v,w) \vee u \neq w) \wedge v \neq 0)] \vee \exists y(y = x+1 \vee R(x,x,y)))$$

is a formula of $\mathrm{PS}^*[\mathrm{FO}_s^+]$ while the formulae

$$\forall x(\mathrm{PS}[\lambda uvw(\neg R(u,v,w) \wedge x \neq max)] \wedge x = C)$$

and

$$\forall x(\mathrm{PS}[\lambda uvw(R(u,v,w))] \vee x \neq C)$$

are not (above, R is a relation symbol of arity 3 and C is a constant symbol).

COROLLARY 4.8
The problem PS is complete for $\mathrm{PS}^*[\mathrm{FO}_s^+]$ via monotone projection translations.

PROOF. The result follows immediately from the proof of Theorem 4 2 as for every case in that proof, it is never the case that the 'known' projective formula ϕ (or Φ in Case (i)) appears negated in the 'constructed' projective formula ψ (note that the notion of one problem being a monotone projection translation of another is transitive). ∎

As pointed out by a referee, problems describable by sentences of the logic $\mathrm{PS}^*[\mathrm{FO}_s^+]$ can be recognized by sequences of monotone circuits of polynomial size: in fact, they can be recognized by uniform sequences of such circuits (under a suitable uniformity constraint such as logspace computability). Consequently, by Razborov's result that there are monotone problems in P that cannot be recognized by sequences of monotone circuits of polynomial size [19], we can say that there are monotone problems in P that are not expressible by sentences of $\mathrm{PS}^*[\mathrm{FO}_s^+]$. It would be interesting to investigate the complexity class captured by the logic $\mathrm{PS}^*[\mathrm{FO}_s^+]$: for example, does it coincide with the class of problems recognized by uniform sequences of monotone circuits of polynomial size?

5 A logical characterization

We now use the results of the previous section to logically characterize the complexity class $\mathrm{NP}_{\text{C-RAT}}$.

REMARK 5.1
Throughout this section, vocabularies consist entirely of relation symbols except where otherwise stated.

THEOREM 5.2
Let Ω be some problem over the vocabulary τ which can be accepted by some polynomial time C-RAT (over τ). Then Ω can be described by some sentence of the form

$$\exists G \Phi$$

where G is a relation symbol not in τ and where $\Phi \in \mathrm{PS}^*[\mathrm{FO}_s(\tau \cup \langle G \rangle)]$ with all occurrences of any symbol of τ positive.

PROOF. Let Ω be some problem over the vocabulary $\tau = \langle R_1, R_2, \ldots, R_r \rangle$, where each R_i is a relation symbol of arity a_i, which can be accepted by some C-RAT M over τ of time complexity n^k. Let G be a new relation symbol of arity k and let $D_1, D_2, \ldots, D_{a_i}$ be constant symbols, for some $i \in \{1, 2, \ldots, r\}$. Define the vocabulary $\tau_i' = \langle G, D_1, D_2, \ldots, D_{a_i} \rangle$ and the problem Ω_i' over τ_i' as

> $\{S' \in \mathrm{STRUCT}(\tau_i') :$ on input $S \in \mathrm{STRUCT}(\tau)$, where S is any structure of size $|S'|$, the C-RAT M with nondeterministic guesses given by $G^{S'}$ queries whether $R_i^S(D_1^{S'}, D_2^{S'}, \ldots, D_{a_i}^{S'})$ holds$\}$

(recall that a C-RAT defers all oracle queries to the end of the computation). Clearly, Ω_i' can be solved in deterministic polynomial time and so by Corollary 4.5 there is a sentence of $\mathrm{PS}^*[\mathrm{FO}_s(\tau_i')]$ of the form

$$\mathrm{PS}[\lambda \mathbf{xyz} \psi_i'(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$

describing Ω_i', where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are tuples of variables of the same length and $\psi_i'(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a projective formula (remember, we can have constant symbols in our vocabularies for Corollary 4.5 any occurrence of the constant symbol D_j in ψ_i' by the (new) variable u_j, for each j, to get the formula ψ_i (over $\langle G \rangle$).

Finally, let Ω' be the problem over $\tau' = \langle G \rangle$ defined as

> $\{S' \in \mathrm{STRUCT}(\tau') :$ on input $S \in \mathrm{STRUCT}(\tau)$, where S is any structure of size $|S'|$, the C-RAT M with nondeterministic guesses given by $G^{S'}$ accepts S so long as all the answers to the oracle queries are 'yes'$\}$.

Then Ω' can be solved in deterministic polynomial time and so by Corollary 4.5 a sentence of $\mathrm{PS}^*[\mathrm{FO}_s(\tau')]$ of the form

$$\mathrm{PS}[\lambda \mathbf{xyz} \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$

describing Ω', where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are tuples of variables of the same length and $\phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a projective formula.

Consequently, for any $S \in \mathrm{STRUCT}(\tau)$, $S \in \Omega$ if and only if

$$\begin{aligned} S \models\ & \exists G(\mathrm{PS}[\lambda \mathbf{xyz} \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})] \wedge \forall \mathbf{u}(\mathrm{PS}[\lambda \mathbf{xyz} \psi_1(\mathbf{x}, \mathbf{y}, \mathbf{z}; \mathbf{u}_1)] \Rightarrow R_1(\mathbf{u}_1) \\ & \wedge \ldots \wedge \mathrm{PS}[\lambda \mathbf{xyz} \psi_r(\mathbf{x}, \mathbf{y}, \mathbf{z}; \mathbf{u}_r)] \Rightarrow R_r(\mathbf{u}_r))) \end{aligned}$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_t)$, with $t = max\{a_i : i = 1, 2, \ldots, r\}$, and $\mathbf{u}_i = (u_1, u_2, \ldots, u_{a_i})$, for $i = 1, 2, \ldots, r$ (we may clearly assume that the lengths of the tuples $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ in each formula ψ_i or ϕ are the same). ∎

It should be clear from the proof of Theorem 5.2 why we sometimes need constant symbols in our vocabularies.

DEFINITION 5.3

The problem NES (standing for non-empty satisfiability) consists of all those structures over the vocabulary $\tau_{2,2}$ encoding satisfiable Boolean formulae in conjunctive normal form (as detailed prior to Theorem 2.1) where every clause is non-empty.

DEFINITION 5.4
A *labelled path system* P of size n is a path system $(V, R, \text{source}, \text{sink})$ of size n where if $R(x, y, z)$ holds then the rule (x, y, z) has an associated boolean literal from the set $\{X_0, \neg X_0, X_1, \neg X_1, \ldots, X_{n-1}, \neg X_{n-1}, \mathit{True}\}$. The sink is accessible in this labelled path system if there is a truth assignment t on the boolean variables $\{X_0, X_1, \ldots, X_{n-1}\}$ such that the sink is accessible in the path system obtained from P by retaining only those rules whose associated literal is set at *True* under t. LABELLED PATH SYSTEM ACCESSIBILITY is the decision problem of deciding whether the sink is accessible in some labelled path system. We encode this decision problem as the problem LPS over the vocabulary $\tau = \langle L_p, L_n, T\rangle$, where L_p and L_n are relation symbols of arity 4 and T is a relation symbol of arity 3, by stipulating that for any labelled path system

$L_p(x, y, z, u)$ holds if and only if the rule (x, y, z) is labelled with X_u;
$L_n(x, y, z, u)$ holds if and only if the rule (x, y, z) is labelled with $\neg X_u$;
$T(x, y, z)$ holds if and only if the rule (x, y, z) is labelled with *True*

(notice that not every structure over τ corresponds to a labelled path system but we can describe exactly those structures that do using a first-order formula: this is analogous to the situation in traditional complexity theory where not every string is the encoding of some instance of a decision problem). Of course, if $S \in \text{STRUCT}(\tau)$ does not encode a labelled path system then $S \notin \text{LPS}$.

THEOREM 5.5
Let Ω be some problem over the vocabulary τ which can be described by some sentence of the form

$$\exists G \Phi$$

where G is a relation symbol not in τ and where $\Phi \in \text{PS}^*[\text{FO}_s(\tau \cup \langle G\rangle)]$ with all occurrences of any symbol of τ positive. Then Ω can be described by a sentence of $\text{NES}^*(\text{FO}_s^+(\tau))$ of the form

$$\text{NES}[\lambda \mathbf{x}\mathbf{y}\psi_p, \mathbf{x}\mathbf{y}\psi_n]$$

where $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of variables, for some k, and ψ_p and ψ_n are monotone projective formulae (over τ).

PROOF. Let P be some path system on a set V of n vertices with rules given by the relation R of arity 3. Consider the path system P' defined as follows:

the vertices of P' are arranged in n rows, each row being a copy of V (so there are n^2 vertices in total);

the source (resp. sink) of P' is the vertex of row 1 (resp. n) corresponding to the source (resp. sink) of V;

for each $\imath = 1, 2, \ldots, n-1$, let $x_\imath$ be the vertex on the $\imath$th row of P' corresponding to the vertex x of P: then for each vertex x of V, there is a rule $(x_\imath, x_\imath, x_{\imath+1})$;

if there is a rule (x, y, z) in P, then there is a rule $(x_\imath, y_\imath, z_{\imath+1})$ in P' for each $\imath = 1, 2, \ldots, n-1$.

Clearly, the sink of P is accessible if and only if the sink of P' is accessible. Also, P' can be described in terms of P by a monotone projective formula and so by the proof of Theorem 4.2 we may assume that Φ (in the statement of the theorem) is of the form

$$\text{PS}[\lambda \mathbf{x}\mathbf{y}\mathbf{z}\psi(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are k-tuples of variables, for some k, ψ is a projective formula over $\tau \cup \langle G\rangle$, with all occurrences of any symbol of τ positive, and for every $S \in \text{STRUCT}(\tau \cup \langle G\rangle)$, the path

system described by $\lambda \mathbf{xyz}\psi^S(\mathbf{x},\mathbf{y},\mathbf{z})$ has the layered form of P' above (see, for example, [29] where some monotone projection translations are given explicitly).

By amending ψ if necessary, we may assume that G is of arity k. In particular, suppose that ψ is of the form

$$(\alpha_1 \wedge G(\mathbf{w}_1)) \vee \ldots \vee (\alpha_i \wedge G(\mathbf{w}_i)) \vee (\alpha_{i+1} \wedge \neg G(\mathbf{w}_{i+1})) \vee \ldots$$
$$\ldots \vee (\alpha_j \wedge \neg G(\mathbf{w}_j)) \vee (\alpha_{j+1} \wedge \beta_{j+1}) \vee \ldots \vee (\alpha_m \wedge \beta_m)$$

where each α_p is a conjunction of atomic and negated atomic formulae involving only the relation symbols s and $=$, the α_p's are mutually exclusive, each β_p is an atomic formula over τ, and each $\mathbf{w}_p$ is a k-tuple of variables from amongst those of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ (that is, a projection of the tuple $(\mathbf{x},\mathbf{y},\mathbf{z})$).

Let $\mathbf{u}$ be a k-tuple of new variables. Define the formulae

$$\begin{aligned}
\phi_1(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}) &\equiv (\alpha_1 \wedge \mathbf{u} = \mathbf{w}_1) \vee \ldots \vee (\alpha_i \wedge \mathbf{u} = \mathbf{w}_i) \\
\phi_2(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}) &\equiv (\alpha_{i+1} \wedge \mathbf{u} = \mathbf{w}_{i+1}) \vee \ldots \vee (\alpha_j \wedge \mathbf{u} = \mathbf{w}_j) \\
\phi_3(\mathbf{x},\mathbf{y},\mathbf{z}) &\equiv (\alpha_{j+1} \wedge \beta_{j+1}) \vee \ldots \vee (\alpha_m \wedge \beta_m).
\end{aligned}$$

Then given some $S \in \mathrm{STRUCT}(\tau)$, $\lambda \mathbf{xyzu}\phi_1^S(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u})$, $\lambda \mathbf{xyzu}\phi_2^S(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u})$, and $\lambda \mathbf{xyz}\phi_3^S(\mathbf{x},\mathbf{y},\mathbf{z})$ describe a labelled path system where the rule $(\mathbf{x},\mathbf{y},\mathbf{z})$ is labelled with the Boolean literal $X(\mathbf{u})$ (resp. $\neg X(\mathbf{u})$, *True*) if and only if $\phi_1^S(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u})$ (resp. $\phi_2^S(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u})$, $\phi_3^S(\mathbf{x},\mathbf{y},\mathbf{z})$) holds (remember, the α_p's are mutually exclusive). Moreover, S clearly satisfies $\exists G(\mathrm{PS}[\lambda \mathbf{xyz}\psi(\mathbf{x},\mathbf{y},\mathbf{z})])$ if and only if there is some truth assignment on the Boolean variables involved as labels of the labelled path system making the sink accessible in the resulting path system (note also that the labelled path system has the same layered structure as the path system described by $\lambda \mathbf{xyz}\psi(\mathbf{x},\mathbf{y},\mathbf{z})$). Consequently, the sentence $\exists G(\mathrm{PS}[\lambda \mathbf{xyz}\psi(\mathbf{x},\mathbf{y},\mathbf{z})])$ is equivalent to

$$\mathrm{LPS}[\lambda \mathbf{xyzu}\phi_1(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}), \mathbf{xyzu}\phi_2(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}), \mathbf{xyz}\phi_3(\mathbf{x},\mathbf{y},\mathbf{z})].$$

Let us say that a labelled path system is *acyclic* if there exists some total ordering of its vertices such that the source (resp. sink) is the first (resp. last) vertex in the ordering and if (x,y,z) is a labelled rule then $x \leq z$ and $y \leq z$, with respect to this ordering (we can also talk of an acyclic path system). Clearly, the labelled path system above described by the formulae

$$\lambda \mathbf{xyzu}\phi_1(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}), \lambda \mathbf{xyzu}\phi_2(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}), \lambda \mathbf{xyz}\phi_3(\mathbf{x},\mathbf{y},\mathbf{z})$$

is acyclic. Let ALPS be the problem consisting of all those structures over $\langle L_p, L_n, T\rangle$ encoding acyclic labelled path systems in which the sink is accessible.

PROPOSITION 5.6
There is a monotone projection translation from ALPS to NES.

PROOF. Let $S \in \mathrm{STRUCT}(\tau)$ be of size n, where $\tau = \langle L_p, L_n, T\rangle$, with L_p and L_n relation symbols of arity 4 and T a relation symbol of arity 3. Furthermore, suppose that S encodes an acyclic labelled path system. Consider the set of clauses

$$\begin{aligned}
\Sigma = \{&C(z), C_{i,i}(x,y,z), C_{p,j}(x,y,z,u), C_{n,j}(x,y,z,u), D_0 \\
&: x,y,z,u \in |S|, z \neq 0, i \in \{1,2,3\}, j \in \{1,2,3,4\}\}
\end{aligned}$$

over the Boolean variables

$$Y = \{X(x), X_t(x,y,z), X_p(x,y,z,u), X_n(x,y,z,u), L(x) : x,y,z,u \in |S|\}$$

defined as follows:

$\neg X(z) \in C(z)$;
if $T^S(x,y,z)$ holds then $X_t(x,y,z) \in C(z)$;
if $L_p^S(x,y,z,u)$ holds then $X_p(x,y,z,u) \in C(z)$;
if $L_n^S(x,y,z,u)$ holds then $X_n(x,y,z,u) \in C(z)$;
$C_{t,1}(x,y,z) = \{\neg X_t(x,y,z), X(x)\}$;
$C_{t,2}(x,y,z) = \{\neg X_t(x,y,z), X(y)\}$;
$C_{t,3}(x,y,z) = \{\neg X_t(x,y,z), X(z)\}$;
$C_{p,1}(x,y,z,u) = \{\neg X_p(x,y,z,u), X(x)\}$;
$C_{p,2}(x,y,z,u) = \{\neg X_p(x,y,z,u), X(y)\}$;
$C_{p,3}(x,y,z,u) = \{\neg X_p(x,y,z,u), X(z)\}$;
$C_{p,4}(x,y,z,u) = \{\neg X_p(x,y,z,u), L(u)\}\}$;
$C_{n,1}(x,y,z,u) = \{\neg X_n(x,y,z,u), X(x)\}$;
$C_{n,2}(x,y,z,u) = \{\neg X_n(x,y,z,u), X(y)\}$;
$C_{n,3}(x,y,z,u) = \{\neg X_n(x,y,z,u), X(z)\}$;
$C_{n,4}(x,y,z,u) = \{\neg X_n(x,y,z,u), \neg L(u)\}$;
$D_0 = \{X(n-1)\}$,

where $x,y,z,u \in |S|$ with $z \neq 0$.

Suppose that f is a truth assignment which satisfies every non-empty clause of S. Then f induces a truth assignment f' on the Boolean variables $\{Z_0, Z_1, \ldots, Z_{n-1}\}$ involved in the labelling of the rules of the acyclic labelled path system S as follows:

$$f'(Z_u) = \textit{True} \text{ if and only if } f(L(u)) = \textit{True}$$

Let P be the (acyclic) path system resulting from S with the truth assignment f'. We say that a vertex x of P is marked if and only if $f(X(x)) = \textit{True}$. We claim that if a vertex of P is marked then it is accessible in P.

Suppose, as our induction hypothesis, that the result holds for all vertices x such that $x < z$, for some fixed vertex z where z is not the source (the base case of the induction clearly holds). If z is not marked then we are done; so suppose that z is marked. Hence, $f(X(z)) = \textit{True}$. So we must have that $X_t(x,y,z)$, $X_p(x,y,z,u)$, or $X_n(x,y,z,u)$ is in $C(z)$ and $f(X_t(x,y,z))$, $f(X_p(x,y,z,u))$, or $f(X_n(x,y,z,u))$ is *True*, respectively, for some $x,y,u \in |S|$ (as the clause $C(z)$ is satisfied by f).

Case (i). If $X_t(x,y,z) \in C(z)$ and $f(X_t(x,y,z)) = \textit{True}$ then (x,y,z) is a rule of S labelled *True* and $f(X(x)) = f(X(y)) = f(X(z)) = \textit{True}$. So (x,y,z) is a rule of P and by the induction hypothesis (as $x < z$ and $y < z$) z is accessible.

Case (ii). If $X_p(x,y,z,u) \in C(z)$ and $f(X_p(x,y,z,u)) = \textit{True}$ then (x,y,z) is a rule of S labelled Z_u and $f(X(x)) = f(X(y)) = f(X(z)) = f(L(u)) = \textit{True}$. So (x,y,z) is a rule of P and by the induction hypothesis (as $x < z$ and $y < z$) z is accessible.

Case (iii). If $X_n(x,y,z,u) \in C(z)$ and $f(X_n(x,y,z,u)) = \textit{True}$ then (x,y,z) is a rule of S labelled $\neg Z_u$ and $f(X(x)) = f(X(y)) = f(X(z)) = \textit{True}$ and $f(L(u)) = \textit{False}$. So (x,y,z) is a rule of P and by the induction hypothesis (as $x < z$ and $y < z$) z is accessible.

Hence, our claim follows by induction. As $f(X(n-1)) = \textit{True}$ then the sink is accessible in P.

Conversely, suppose that $S \in$ ALPS via the truth assignment f' on the Boolean variables $\{Z_0, Z_1, \ldots, Z_{n-1}\}$, and let $r_1, r_2, \ldots, r_m$ be all the rules used to establish that the sink is accessible in the path system P resulting from S with the truth assignment f'. Applying these rules establishes the accessibility of various vertices of P: call these the marked vertices of P. Consider the following truth assignment f on the Boolean variables of Y:

if x is marked in P then set $f(X(x)) = \mathit{True}$;
for each $i = 1, 2, \ldots, m$
  if r_i is the rule (x, y, z) labelled *True* then set $f(X_t(x, y, z)) = \mathit{True}$;
  if r_i is the rule (x, y, z) labelled Z_u then set $f(X_p(x, y, z, u)) = \mathit{True}$;
  if r_i is the rule (x, y, z) labelled $\neg Z_u$ then set $f(X_n(x, y, z, u)) = \mathit{True}$;
if $f'(Z_u) = \mathit{True}$ then set $f(L(u)) = \mathit{True}$;
set $f(W) = \mathit{False}$ for all other Boolean variables W of Y.

It is easy to see that the truth assignment f satisfies every non-empty clause of S.

We can easily describe the set of clauses Σ, in terms of S, using monotone projective formulae (see, for example, the constructions in [29]) and we can also easily ensure that every clause is non-empty (by adding the literals $X(0)$ and $\neg X(0)$ to empty clauses) without ruining the monotone projective formulae. Hence the proposition follows. ∎

As the sentence

$$\mathrm{LPS}[\lambda \mathbf{xyzu}\phi_1(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyzu}\phi_2(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyz}\phi_3(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$

is equivalent to

$$\mathrm{ALPS}[\lambda \mathbf{xyzu}\phi_1(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyzu}\phi_2(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyz}\phi_3(\mathbf{x}, \mathbf{y}, \mathbf{z})],$$

then the result of the theorem follows by applying Proposition 5.6. ∎

Note that we need to ensure that our labelled path systems are acyclic in the proof of Theorem 5.5 so that we can use induction in the proof of Proposition 5.6. Note also that we have moved from a Fagin-type representation to an Immerman-type representation by labelling structures with Boolean literals.

COROLLARY 5.7
Let Ω be some problem over τ. Then $\Omega \in \mathrm{NP}_{\text{C-RAT}}$ if and only if Ω can be described by a sentence of $\mathrm{NES}^*[\mathrm{FO}^+_s(\tau)]$ of the form

$$\mathrm{NES}[\lambda \mathbf{xy}\psi_p, \mathbf{xy}\psi_n]$$

where $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of variables, for some k, and ψ_p and ψ_n are monotone projective formulae.

PROOF. Immediate from Theorems 5.2 and 5.5, together with the fact that any problem described by a sentence of $\mathrm{NES}^*[\mathrm{FO}^+_s(\tau)]$ of the form in the statement of the corollary can clearly be solved by a polynomial time C-RAT over τ. ∎

COROLLARY 5.8
$\mathrm{NP}_{\text{C-RAT}} = \mathrm{NES}^*[\mathrm{FO}^+_s]$.

PROOF. Immediate from Corollary 5.7 as any problem in $\mathrm{NES}^*[\mathrm{FO}^+_s]$ can be solved by a polynomial time C-RAT (this follows from an easy induction on the length of the describing sentence of $\mathrm{NES}^*[\mathrm{FO}^+_s]$). ∎

We need the version (NES) of SAT where every clause is known to contain at least one literal so that the problem can be solved by a polynomial time C-RAT. Note that in proving Theorem 5.5, we firstly describe Ω using operators whose corresponding problems are not in $\mathrm{NP}_{\text{C-RAT}}$ (namely LPS and ALPS) before describing Ω using the operator NES (whose corresponding problem is in $\mathrm{NP}_{\text{C-RAT}}$).

COROLLARY 5.9
For any vocabulary τ, $\mathrm{NP}_{\text{C-RAT}}(\tau)$ is captured by the sub-logic of existential second-order logic with sentences of the form

$$\exists T \phi$$

where T is some relation symbol not in τ and ϕ is a first-order sentence over $\tau \cup \langle T \rangle$ with all occurrences of any relation symbol of τ positive.

PROOF. By Corollary 5.7, any problem of $\mathrm{NP}_{\text{C-RAT}}(\tau)$ can be described by a sentence of the form

$$\exists T \forall \mathbf{y} \exists \mathbf{x}((\psi_p(\mathbf{x}, \mathbf{y}) \wedge T(\mathbf{x})) \vee (\psi_n(\mathbf{x}, \mathbf{y}) \wedge \neg T(\mathbf{x})))$$

where $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of variables, for some k, ψ_p and ψ_n are monotone projective formulae, and T is a relation symbol of arity k ($\forall \mathbf{y}$, for example, is shorthand for $\forall y_1 \forall y_2 \ldots \forall y_k$ if $\mathbf{y} = (y_1, y_2, \ldots, y_k)$). The converse containment is obvious. ∎

Corollary 5.9 should be compared with Fagin's result that NP is captured by existential second-order logic [9]. We should add that Corollary 5.9 could probably have been proven by resorting to Fagin's methods [9]. However, we prefer our proof as it shows how we can move from Immerman-type descriptions to Fagin-type ones, we do not need to drop down to the level of Turing machines, and we can use existing results. Also, we obtain a particularly nice normal form (see the proof of Corollary 5.9) for formulae of the sub-logic L involved in the statement of Corollary 5.9. Bearing this normal form in mind, it might be worthwhile looking at the complexity classes captured when we restrict the first-order quantifier prefix in the formulae of the logic L: for example, we might insist that the first-order quantifier prefix consists entirely of universal quantifiers.

We close this section with a corollary of Theorems 5.2 and 5.5 and their proofs.

COROLLARY 5.10
The problem NES (resp. LPS, ALPS) is complete (resp. hard, hard) for $\mathrm{NP}_{\text{C-RAT}}$ via monotone projection translations. Moreover the restricted version of ALPS consisting of all structures in ALPS where the acyclic labelled path system is such that the rules labelled with Boolean literals, apart from *True*, depend only on the size of the structure is hard for $\mathrm{NP}_{\text{C-RAT}}$ via monotone projection translations.

PROOF. The results follow from the proofs of Theorems 5.2 and 5.5. With the notation as in the proof of Theorem 5.5, note that any problem in $\mathrm{NP}_{\text{C-RAT}}$ can be described by a sentence of the form

$$\mathrm{ALPS}[\lambda \mathbf{xyzu}\phi_1(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyzu}\phi_2(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}), \mathbf{xyz}\phi_3(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$

and that ϕ_1 and ϕ_2 are devoid of symbols of τ. ∎

We remark that any problem of $\mathrm{NP}_{\text{C-RAT}}$ which happens to be a monotone projection translation of NES is also complete for $\mathrm{NP}_{\text{C-RAT}}$ via monotone projection translations.

6 Conclusion

In conclusion, let us begin by reminding the reader of the main theorem of this paper.

THEOREM 6.1
As classes of problems over vocabularies consisting entirely of relation symbols, the following are identical.

(i) $\mathrm{NP}_{\text{C-RAT}}$.

(ii) The class of all monotone problems in NP.

(iii) $\mathrm{NES}^*[\mathrm{FO}^+_s]$.

(iv) The class of problems represented by sentences of existential second-order logic of the form

$$\exists T_1 \exists T_2 \ldots \exists T_k \phi$$

where each T_i is a relation symbol and ϕ is a first-order sentence in which all relation symbols, apart from $T_1, T_2, \ldots, T_k, s$, and $=$, occur positively.

Attention is drawn to the particular normal forms for the sentences of the logics in (ii) and (iii) above: see Corollaries 5.7 and 5.9.

It is appropriate that we mention the papers [21] and [32]. In those papers, a complexity theory based on Boolean algebra was developed. However, this theory involves non-uniform complexity classes based upon families of Boolean functions (whereas our theory, based upon the Turing machine model of computation, involves uniform complexity classes). In order to compare families of Boolean functions, reductions called (*monotone*) *p-projections* were introduced (in fact our (monotone) projection translations were so named by Immerman as he regarded them as uniform versions of (monotone) p-projections [13]). It was shown that various (non-uniform) complexity classes have complete problems via (monotone) p-projections.

Because the complexity classes of [21] and [32] are defined with respect to families of Boolean functions, it is difficult to tie them in with $\mathrm{NP}_{\text{C-RAT}}$. Although it is remarked in [21] that under a uniformity constraint such as the stipulation that all p-projections should be computable in logspace, the results still hold, we still have complexity classes defined with respect to uniform families of Boolean functions in contrast to $\mathrm{NP}_{\text{C-RAT}}$ which is defined with respect to the Turing machine model. Moreover, whereas our complexity class $\mathrm{NP}_{\text{C-RAT}}$ can be considered as a monotone version of NP, no non-uniform monotone version of NP was actually explicitly defined in [21] and [32] (although monotone p-projections between functions such as the Hamiltonian circuit function and the Clique function were shown to exist). Consequently, while our approach and that of Skyum and Valiant are similar in that they both consider monotone reductions (of different sorts), the environments are so different that they should be treated separately. It would be interesting to consider the non-uniform version of $\mathrm{NP}_{\text{C-RAT}}$ obtained by allowing a polynomial amount of advice.

As a final remark, let us draw the readers attention to the paper [3] where it is shown that there are monotone first-order describable problems which cannot be described by first-order sentences in which the relation symbols are positive. Also, as remarked at the end of section 3, there are monotone problems solvable in polynomial time that cannot be described by sentences of $\mathrm{PS}^*[\mathrm{FO}^+_s]$. However, we have shown that every monotone problem solvable in NP can be described by a sentence of the logic $\mathrm{NES}^*[\mathrm{FO}^+_s]$ where the relation symbols are positive. It would be interesting to know whether the class of monotone problems solvable in P can be captured by a logic of the form $\Omega^*[\mathrm{FO}^+_s]$.

Acknowledgements

The author would like to thank two anonymous referees and an editor for their comments, corrections, and suggestions which have gone towards improving substantially the results in and the presentation of this paper.

References

[1] F. Afrati, S S Cosmadakis and M. Yannakakis On Datalog vs. polynomial time. *Proceedings of the Tenth ACM Annual Symposium on Principles of Database Systems*, pp. 13–23, 1991.

[2] M Ajtai and R Fagin Reachability is harder for directed than for undirected finite graphs *J. Symbolic Logic*, **55**, 113–150, 1990

[3] M Ajtai and Y Gurevich Monotone versus positive *J Assoc. Comput Mach.*, **34**, 1004–1015, 1987

[4] J L Balcazar, J Diaz and J Gabarro *Structural Complexity Theory* I Springer-Verlag, Berlin, 1988

[5] D A M Barrington, N. Immerman and H Straubing. On uniformity within NC^1 *J Comput System Sci* , **41**, 274–306, 1990

[6] A. K Chandra, D.C Kozen and L.J Stockmeyer Alternation *J Assoc Comput Mach.*, **28**, 114–133, 1981

[7] S. A Cook An observation on time-storage trade off *J Comput System Sci.*, **9**, 308–316, 1974

[8] E. Dahlhaus. Reduction to NP-complete problems by interpretation *Lecture Notes in Computer Science* Vol 171, pp 357–365, Springer-Verlag, 1984

[9] R Fagin Generalized first-order spectra and polynomial-time recognizable sets In *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, R M Karp, ed., pp 43–73, 1974

[10] M R Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness* W H. Freeman and Co., San Francisco, 1979

[11] E Gradel Capturing complexity classes by fragments of second-order logic *Theoret Comput Sci* , **101**, 35–57, 1992

[12] Y. Gurevich Logic and the challenge of computer science In *Current Trends in Theoretical Computer Science*, E Börger, ed , pp 1–57 Computer Science Press, 1987.

[13] N Immerman Languages that capture complexity classes *SIAM J Comput* , **16**, 760–778, 1987

[14] N Immerman Nondeterministic space is closed under complementation *SIAM J Comput.*, **17**, 935–938, 1988.

[15] N Immerman Descriptive and computational complexity. In *Computational Complexity Theory, Proc of AMS Symp in Applied Mathematics*, Vol. 38, J. Hartmanis, ed., pp 75–91, 1989

[16] N Immerman and S Landau The complexity of iterated multiplication In *Proc Fourth IEEE Ann. Symp. on Structure in Complexity Theory*, pp 104–111, 1989

[17] P G. Kolaitis and M Y Vardi On the expressive power of Datalog tools and a case study. In *Proc. Ninth ACM Ann. Symp on Principles of Database Systems*, pp 61–71, 1990.

[18] V S Lakshmanan and A O Mendelzon Inductive pebble games and the expressive power of Datalog In *Proc 8th ACM Ann. Symp on Principles of Database Systems*, pp 301–310, 1989

[19] A. A. Razborov A lower bound on the monotone network complexity of the logical permanent, *Mat. Zametki*, **41**, 598–607, 1987 (In Russian English translation in: *Mathematical Notes*, **41**, 333–338, 1987)

[20] W L. Ruzzo. On uniform circuit complexity *J Comput System Sci* , **22**, 365–383, 1981.

[21] S Skyum and L. G Valiant. A complexity theory based on Boolean algebra. *J Assoc Comput. Mach.*, **32**, 484–502, 1985

[22] I. A Stewart. Comparing the expressibility of languages formed using NP-complete operators. *J Logic Computat* , **1**, 305–330, 1991

[23] I. A Stewart. Complete problems for logspace involving lexicographic first paths in graphs In *Lecture Notes in Computer Science* Vol 570, pp 198–208. Springer-Verlag, 1991.

[24] I A Stewart. Complete problems for symmetric logspace involving free groups. *Inform. Process. Lett* , **40**, 263–267 1991.

[25] I. A Stewart Complete problems involving Boolean labelled structures and projection translations *J Logic Computat.*, **1**, 861–882 1991

[26] I A. Stewart. Using the Hamiltonian path operator to capture NP. *J. Comput. Systems Sci.*, **45**, 127–151, 1992.

[27] I A. Stewart. Logical characterizations of bounded query classes I logspace oracle machines. *Fund. Informat* , **18**, 65–92, 1993.

[28] I A Stewart Logical characterizations of bounded query classes II polynomial-time oracle machines *Fund. Informat.*, **18**, 93–105, 1993.

[29] I. A. Stewart. On completeness for NP via projection translations. In *Lecture Notes in Computer Science* Vol. 626, pp 353–366. Springer-Verlag, 1993 To appear, *Math. Systems Theory*

[30] I A Stewart Logical descriptions of monotone NP problems. 1993, *J Logic Computat*, to appear

[31] R Szelepcsényi The method of forced enumeration for nondeterministic automata. *Acta. Informat*, **26**, 279–284, 1988.

[32] L G Valiant. Completeness classes in algebra. In *Proc. Eleventh ACM Ann. Symp on Theory of Computing*, pp. 249–262, 1979.

Received 16 March 1992