


References


Theorem 6.22 Our expressibility theorems above can be extended to Buss' bounded oracle model.

Proof: Follows directly, as we never use in the expressibility proofs the unboundedness of the model. However we note that the bounded cases have a simpler proof since here it is possible to encode the contents of the oracle tape into the LogSpace bounded state of the computation. The consequence of this is that the equivalent of the above \(N \text{EXT}(t^0, s^0, t^1, s^1)\) predicate is now much simplified and can use pairs of states of the computations rather than quadruples.

\(\square\)

7 Conclusions

We have shown that, in Buss' unbounded model of oracle computation there is a close relationship between Lindström quantifiers and oracle complexity classes. This supports our 'logical simulation thesis':

All general logic simulations (logics capturing complexity classes) which hold in the absence of an oracle can be extended to hold for every oracle in a properly relativized model via Lindström quantifiers.

We have shown evidence for this for L, NL and P. We think the same to be true for every 'reasonable' complexity class below and including PSpace. L, NL and P are defined via Turing machines whose accessibility relation is well understood, in fact is given by deterministic, non-deterministic and alternating transitive closure respectively. These machines also satisfy Buss' relativization theses. The problem in proving such a connection for every regular complexity class consists of two technicalities: First, it is not clear how to define oracle computations for arbitrary regular complexity classes without recourse to a specific machine model. Second, even for specific machine models, one would need, that the accessibility relation between states for legal computations can be used to define a Lindström quantifier.

We can see ways of overcoming these problems, by defining abstractly the regular complexity classes, for which the technicalities are resolved. But such a solution seems artificial, unless we can show that every regular complexity classes can be represented in such a way. Thus, it remains an open problem how to prove our main theorems for every regular complexity class between L and PSpace.

Acknowledgments:

We are indebted to J. Stewart and N. Immerman for valuable remarks and suggestions on a draft version of this paper. We are also grateful to B. Chor for guidance in the literature.
THEOREM C (i). The case of $D = NL$

**Proof:** The proof is similar to the above except for the following:

(i) One should notice that in claim 6.10 we now use Buss’ requirement that the size of the information on the oracle tape be bounded for an $NL$ machine by a polynomial of some degree $k'$ in the size of the input. Since this $k'$ might not coincide with the $k$ of claim 6.7. We replace $k$ in the remainder of the proof by the maximum of $k$ and $k'$.

(ii) In claim 6.21 the formula is replaced by

$$\exists \bar{s}, \bar{t} ACCEPTING(\bar{s}) \land TC_{r_{init},r_{init},\bar{t},\bar{s}}^{\bar{r},r_{init},\bar{t},\bar{s}} NEX(r_0, s_0, t_1, s_1)$$

THEOREM C (i). The case of $D = AL$ (Alternating LogSpace).

**Proof:** Again similar to the above except that in claim 6.21 the formula is replaced by

$$\exists \bar{s}, \bar{t} ACCEPTING(\bar{s}) \land ATC_{r_{init},r_{init},\bar{t},\bar{s}}^{\bar{r},r_{init},\bar{t},\bar{s}} NEX(r_0, s_0, t_1, s_1)$$

THEOREM C (i). The case of $D = P$.

**Proof:** Follows from the fact that in Buss’ models, for all oracles, relativised-AL equals relativised-P.

THEOREM C (ii). The case of multiple oracles

**Proof:** For each class $D$ as above, the proof proceeds as for the one oracle case except that for each oracle, $K_i$, used we define its own $OId_i$, $\text{nonext}_i$, and $\text{next}_i$ predicates and also its own set of $\Phi$ formulas. The other major difference from the one oracle case is that the vectorization of the $\text{next}$ predicate now increases by $k$ for each of the oracles used, and the definition of $\text{next}$ will be set up so that

$$\text{next}(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n, \bar{s}_0, \bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n, \bar{s}_1)$$

will be true either if each $\bar{a}_i$ equals the corresponding $\bar{b}_i$ and the transition can be taken without oracle consultations or else each $\bar{a}_i$ equals the corresponding $\bar{b}_i$ except for some $\bar{a}_j = \bar{b}_j$ in which case $\bar{b}_j = \bar{s}_0$ and the transition is taken according to the answer of the $j$'th oracle to a query.
Definition 6.18 Let $\text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau)$ be a 4k-ary predicate as follows:

\[ \text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau) \equiv (t_0^\tau = t_1^\tau \land \text{next}(s_0^\tau, s_1^\tau)) \lor \\
( (t_1^\tau = s_1^\tau) \land \text{OLD}(s_0^\tau) \land \nu_{id}(s_0^\tau) \land \\
( Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, t_1^\tau)) \lor (\neg Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, t_1^\tau)) ) \]

Definition 6.19 A string is oracle tape proper for a computation state $s_0^\tau$ and a $\tau$-structure $A$ if there is a state of the computation $t_0^\tau$ such that when $M$ is invoked on $A$ with an empty oracle tape and in $t_0^\tau$ it eventually reaches $s_0^\tau$ with the string written on its oracle tape.

Claim 6.20 Let $s_0^\tau$ and $s_1^\tau$ be two computation states. Then there is a proper oracle tape string for $s_0^\tau$ and $A$ such that $M$ can go from $s_0^\tau$ to $s_1^\tau$ in one transition iff there are two computation states $t_0^\tau$ and $t_1^\tau$ such that $\text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau)$.

Proof: Assume there is such a string. If the operation of $M$ does not include an oracle consultation then next($s_0^\tau, s_1^\tau$) and hence for any $t_0^\tau$ we have that $\text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau)$. Assume the transition includes an oracle consultation. This implies that there is a string proper for $s_0^\tau$ and $A$ which encodes some $\sigma$-structure $B$. Hence by claim 6.17 there is some $t_0^\tau$ such that $\nu_{id}(s_0^\tau)$ and $B$ is defined by $\Phi^{t_0^\tau}$. If the transition from $s_0^\tau$ to $s_1^\tau$ occurs by a positive answer of the oracle, then $(Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, s_1^\tau))$ and if it occurs by a negative answer then $(\neg Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, s_1^\tau))$. In either case $\text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau)$ holds.

Now assume $\text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau)$. If $t_0^\tau = t_1^\tau$ then next($s_0^\tau, s_1^\tau$) and regardless of the contents of the oracle tape $M$ can go from $s_0^\tau$ to $s_1^\tau$ in one transition. Otherwise $t_0^\tau = s_1^\tau$ and $s_0^\tau$ is an oracle consultation state. From claim 6.17 and the fact that $\nu_{id}(s_0^\tau)$ we have that if $M$ is invoked with an empty oracle tape in computation state $t_0^\tau$ it can eventually reach the state $s_0^\tau$ with the $\sigma$-structure $\Phi^{t_0^\tau}$ on the oracle tape. Hence an encoding of the structure is oracle tape proper for $s_0^\tau$ and $A$. Now either $(Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, s_1^\tau))$ or $(\neg Q(\Phi^{t_0^\tau}) \land \text{next}(s_0^\tau, s_1^\tau))$ holds. In either case the answer of the oracle will cause the machine to move to state $s_1^\tau$.

Claim 6.21 The set of $\tau$ structures accepted by $M$ can be expressed by a formula of $\mathcal{L}[Q_{\kappa^+}]$.

Proof: Let $s_{init}$ be an encoding of the initial computation state of $M$ then the following is the required formula:

\[ \exists \vec{s}, \vec{t} \mathop{\text{ACCEPTING}}(\vec{s}) \land \text{DTC}_{s_{init}, \vec{t}_{init}, \vec{t}_{1}^\tau, \vec{t}_{2}^\tau, \vec{t}_{3}^\tau, \vec{t}_{4}^\tau} \text{NEXT}(t_0^\tau, s_0^\tau, t_1^\tau, s_1^\tau) \]

\[ \square \]
Notation 6.13 Let \( W_\tau \) be the set of \( \tau' \)-structures for which there is a \( \sigma' \)-structure fitting for \( M \) and let \( W_\sigma \) be the set of \( \sigma' \)-structures for which there is a \( \tau' \)-structure fitting for \( M \).

Then:

Claim 6.14 \( W_\tau \) is \( L \)-reducible to \( W_\sigma \).

Proof: Consider the machine \( M' \) which operates as follows: First it checks that the input structure is proper. Then it decodes the computation state encoded in the relation \( \text{id}_\tau \). Then it simulates \( M \) from this computation state, as follows: \( A \) the \( \tau \)-structure part on the original input tape is used as input for \( M \). The characters which \( M \) writes on its oracle tape are written on the output tape of \( M' \). When \( M \) either terminates or enters an oracle consultation state, \( M' \) encodes the "current" computation state of \( M \) on the output tape and halts.

It is easy to verify that if \( M \) is LogSpace so is \( M' \), and that an input \( \tau' \)-structure is in \( W_\tau \) iff the output of \( M' \) is a \( \sigma' \)-structure in \( W_\sigma \). \( \square \)

From our THEOREM A we immediately get:

Corollary 6.15 \( W_\tau \) is \( L \)-reducible to \( W_\sigma \).

Notation 6.16 The reduction formulas

(i) Let \( \Phi' = < \phi, \psi_1, \psi_2...\psi_n, \psi_{\text{id}_\tau} > \) be the set of formulas \( k \)-feasible for \( \sigma' \) over \( \tau' \) such that a given \( \tau' \)-structure \( A \in W_\tau \) iff the structure described by \( \Phi \) is in \( W_\sigma \).

(ii) Let \( \Phi \) be the set of formulas \( \phi, \psi_1, \psi_2...\psi_n \) of \( \Phi' \).

(iii) Let \( \Phi^\tilde{y}, \Phi^\tilde{y}, \psi^\tilde{y}_{\text{id}_\tau} \) be the sets of formulas formed from \( \Phi' \), \( \Phi \) and \( \psi^\tilde{y}_{\text{id}_\tau} \) by replacing all appearances of the relation symbol \( \text{id}_\tau (\tilde{x}) \) by the formulas \( \tilde{x} = \tilde{y} \) where \( \tilde{y} \) is a vector of free variables not appearing in any formula of \( \Phi, \Phi^\tilde{y}, \psi^\tilde{y}_{\text{id}_\tau} \).

Note that \( \Phi \) is \( k \)-feasible for \( \sigma \) over \( \tau' \) and \( \Phi^\tilde{y} \) (\( \Phi^\tilde{y} \)) is \( k \)-feasible for \( \sigma' \) (\( \sigma \)) over \( \tau \). Also note that from these definitions we have the following:

Let \( t_0^\tau \) and \( s_0^\sigma \) be two states of the computation of \( M \) with input \( A \), then

Claim 6.17 If \( M \) is invoked in computation state \( t_0^\tau \) with an empty oracle tape it can eventually reach the computation state \( s_0^\sigma \) with the \( \sigma \)-structure \( B \) written on its oracle tape, without passing any oracle consultations in between, iff the set of formulas \( \Phi^\tilde{y} \) defines \( B \) and the formula \( \psi^\tilde{y}_{\text{id}_\tau} \) is satisfied by \( s_0^\sigma \).
• \(O I D(\bar{x})\) - A predicate that \(\bar{x}\) is an encoding of an oracle consultation state of the computation of \(M\).

• \(ACCEPTING(\bar{x})\) - A predicate that \(\bar{x}\) is an encoding of an accepting state of the computation of \(M\).

• \(next(\bar{x}, \bar{y})\) - A predicate that \(I D(\bar{x}), ID(\bar{y}), \neg O I D(\bar{x})\) and that \(\bar{y}\) is reachable in one transition from \(\bar{x}\).

• \(nonext(\bar{x}, \bar{y})\) - A predicate that \(O I D(\bar{x}), ID(\bar{y})\), and that \(\bar{y}\) is reachable in one transition from \(\bar{x}\) provided the oracle answered positively to the query on its tape.

• \(nonext(\bar{x}, \bar{y})\) - A predicate that \(O I D(\bar{x}), ID(\bar{y})\), and that \(\bar{y}\) is reachable in one transition from \(\bar{x}\) provided the oracle answered negatively to the query on its tape.

**Claim 6.10** For an input \(\tau\)-structure \(\mathcal{A}\) the size of any \(\sigma\)-structure written on the oracle tape by \(M\) is at most \(|A|^k\).

**Proof:** Follows from the fact that \(M\) is deterministic and has at most \(|A|^k\) possible states of computation and from the oracle model which demands that the oracle tape is erased every time an oracle query is answered.

**Notation 6.11** Let \(\tau^\prime\) be an extension of \(\tau\) with a new \(k\)-ary relation symbol \(id^\tau\). Similarly let \(\sigma^\prime\) be an extension of \(\sigma\) with a new unary relation symbol \(id^\sigma\). Let \(\mathcal{A}\) be a \(\tau\)-structure and let \(\mathcal{A}'\) be a \(\tau^\prime\)-structure extending it. Let \(\mathcal{B}\) be a \(\sigma\)-structure and let \(\mathcal{B}'\) be a \(\sigma^\prime\)-structure extending it. (Intuitively one should think of \(\mathcal{B}\) and \(\mathcal{B}'\) as structures with a universe of size \(|A|^k\)).

**Definition 6.12** Proper structures for \(M\).

(i) \(\mathcal{A}'\) is called proper for \(M\) if the relation \(id^\tau\) contains a single \(k\)-tuple and this \(k\)-tuple is an encoding of a state of the computation of \(M\).

(ii) \(\mathcal{B}'\) is called proper for \(M\) if the relation \(id^\sigma\) contains a single element and this element is an encoding of an oracle consultation state of the computation of \(M\).

(iii) The pair \((\mathcal{A}', \mathcal{B}')\) is called fitting for \(M\) if both \(\mathcal{A}'\) and \(\mathcal{B}'\) are proper and when \(M\) is simulated from the computation state encoded by the single \(k\)-tuple in the relation \(id^\tau\), with an input tape containing \(\mathcal{A}\) and an empty oracle tape, it can eventually reach, without passing any intermediate oracle consultation state, the oracle computation state encoded by the single element in the relation \(id^\sigma\) with \(\mathcal{B}\) written on its oracle tape.

Note that the the sets of proper \(\tau^\prime\)-structures and \(\sigma^\prime\)-structures are in \(L\) (and hence definable in \(L\)).
Part (iii) of our THEOREM B follows from the observation that for regular complexity classes $D \subseteq C$, $D^C = C$.

### 6.2 Expressibility Proofs

**Lemma 6.5 (Immerman)** [15] The DTC (TC, ATC) predicate transformer is expressible in any logic which captures $L (\mathcal{NL}, \mathcal{P})$.

**THEOREM C (i).** The case of $D = L$.

**Proof:** Let $M$ be some Turing machine which recognizes a set of $\tau$-structures $K_\tau$, such that $M$ is of complexity $L$ and uses an oracle for $K_\tau$. We show a formula of $L[Q_{K_\tau}]$ which defines $K_\tau$.

**Claim 6.6** Without loss of generality we can assume that $M$ enters an oracle consultation state only when the string on its input tape is an encoding of a $\sigma$-structure.

**Proof:** Verifying that a string is a valid $\sigma$-structure can be done in $L$ with a single pass on the input string. As the oracle tape is writeonly - we can assume without loss of generality that it is written sequentially. Hence if $M$ is not as required we can replace it with an $M'$ which simulates it and also simulates a one pass $\sigma$-structure verifier. Whenever $M$ would have consulted the oracle with a string which does not encode a $\sigma$-structure $M'$ continues as if the oracle answers negatively. \hfill $\square$

**Claim 6.7** There is a number $k$ such that for every $\tau$-structure $A$ (with a universe of size $|A|$) the number of all possible “states of the computation” of $M$ for the input $A$ is less than $|A|^k$. (Where a “state of the computation” encodes the state of the Turing machine, the information on the work tape and the position of the head on the work tape - but not the information on the oracle tape which is not bounded by the LogSpace restriction). Furthermore there is an encoding scheme by which all possible states of the computation are mapped into integers between 0 and $|A|^k$.

**Proof:** The existence of $k$ follows from the fact that $M$ is LogSpace. For an example of an encoding scheme see [15]. \hfill $\square$

**Notation 6.8** We denote by $\vec{x}$ a vector of size $k$ of variables $x_1, x_2, \ldots, x_k$.

In a way similar to Immerman’s proof [15] we have:

**Claim 6.9** The following predicates can be written as formulas of $L$:

- $ID(\vec{x})$ - A predicate saying that $\vec{x}$ is an encoding of a state of the computation of $M$.  


on whether the mechanism for recognizing membership in $K$ is an oracle or a Turing machine.

If it is a Turing machine, $M_K$, we modify $M_\Phi$ so that it will receive an additional binary number $b$ as input and produce as output only the $b$th digit of the string coding of $A_\Phi$. We then modify $M_K$ so that whenever it wants to read some $b$th digit from the input, it will instead write $b$ on an auxiliary tape, invoke the modified $M_\Phi$ and use its output digit. It should be clear that the modified $M_K$ (together with the modified $M_\Phi$) perform the required computation, but use only space logarithmic in the size of the input structure.

For the case of oracles this approach does not work as we are not allowed to modify the oracle (as we did with $M_K$). However we can use Buss’ unbounded model of oracle computation which allows $\log$-Space bounded machines to write a polynomial number of digits on their oracle tapes.

We note that this is the only purpose for which we need Buss’ unbounded model. Were we allowed to modify the oracle (that is have it ask $M_\Phi$ for its input digits one by one, as we did with $M_K$) Buss’ bounded model for oracle computation would have been sufficient for all our needs. □

Applying lemma 6.2 to the case where $\Phi = \langle \phi, \psi_1, \ldots, \psi_m \rangle$ are formulas in $\mathcal{L}$ (and hence have model checker in $D$), and noting that a $\mathcal{D}^C$ computation which asks only one oracle query and then immediately terminates is equivalent to a computation in $C$ (even when $C$ is only semi-regular), we get part (i) of our THEOREM B.

**Definition 6.3** Let $\chi$ be a formula of $\mathcal{L}[Q_K]$ over $\tau$ which contains a subformula $\theta$ of the form $\theta = Q_K x, \bar{y}_1, \ldots, \bar{y}_m \Phi$. Let $\tau_1 = \tau \cup \{R\}$ be the vocabulary formed by adding to $\tau$ a relation symbol $R$ of arity $i$, where $i$ is the number of free variables in $\theta$. We define $\chi'$ - the $\theta$-reduced equivalent of $\chi$ as the $\tau_1$-formula which is formed by replacing every appearance of the sub-formula $\theta$ in $\chi$ by the relation symbol $R$ (over the free variables of $\theta$).

**Lemma 6.4** Let $\chi, \theta$ and $\chi'$ be as above. If both $\theta$ and $\chi'$ have model checkers of complexity $\mathcal{D}^C$ then so does $\chi$.

**Proof outline:** The proof of this lemma is similar in spirit to the proofs of proposition 3.2 and lemma 6.2 above: A simple extension of the model checker of $\theta$ performs a reduction of the input $\tau$-structure to the appropriate $\tau_1$-structure on which the model checker of $\chi'$ is then invoked. The possible problem that the size of the $\tau'$-structure might be larger than the space bound is solved as in lemma 6.2.

We can now prove part (ii) of our THEOREM B, and show that every formula of $\mathcal{L}[K]$ has a model checker in $\mathcal{D}^C$, by a simple induction on the number of quantifiers which appear in the formula. Both the basis of the induction and the inductive step are direct applications of the two lemmas above. (Lemma 6.2 for subformulas where the quantifier is syntactically outermost and lemma 6.4 when it is not).
Note that we do not need that HEX is PSpace-complete for First order Reductions. In fact our proof does not give this and hence we do not know if $FOL[HEX]$ has a normal form.

Using a problem which is $\Sigma_n^P$ complete via P-reductions we can also capture $\Delta_{n+1}^P$ of the polynomial hierarchy with Lindström quantifiers:

**APPLICATION 2.** Let $K_n$ be a problem which is suitably coded and $\Sigma_n^P$ complete via P-reductions. Then $FOL[ATC, K_n]$ captures $\Delta_{n+1}^P$.

Our next application generalizes and extends results of Stewart [26]:

**APPLICATION 3.** Let $K$ be a problem.

(i) If $K$ is NP-complete via P-reductions, $FOL[ATC][Q_K]$ captures $P^{NP}$ and $FOL[ATC][Q_K]$ captures NP (over ordered structures).

(ii) If $K$ is NP-complete via L-reductions, $FOL[DTC][Q_K]$ captures $L^{NP}$ and $FOL[DTC][Q_K]$ captures NP (over ordered structures).

(iii) $L^{NP} = NL^{NP} = P^{NP}$.

In particular $FOL[ATC](3COL)$ and $FOL[ATC](HAM)$ capture NP, where 3COL is the class of 3 colourable graphs, and HAM is the class of graphs with a hamiltonian cycle. Using the fact that 3COL and HAM are NP-complete for first order reductions ([24, 25]), we also get that $FOL(3COL)$ and $FOL(HAM)$ capture NP.

6 Proofs

6.1 Model Checking

**Notation 6.1** In the following $D$ is a regular complexity class and $C$ is a semi-regular complexity class such that $L \subseteq D \subseteq C$. $K$ is a set of $\sigma$-structures in $C$ and $L$ is a regular logic which has a model checker in $D$.

**Lemma 6.2** Let $\Phi = \langle \phi, \psi_1, \ldots, \psi_m \rangle$ be a $\sigma$-feasible set of formulas in $L[K]$ having model checkers of complexity $D^C$. Then the formula $Q_K x, y_1, \ldots, y_m \Phi$ has a model checker in $D^C$.

**Proof outline:** If $P \subseteq D^C$ the proof is a direct application of the Turing machine $M_\Phi$ outlined in proposition 3.2, followed by the machine or oracle which checks membership in $K$.

For other cases this direct approach might not work as the size of the $\sigma$-structure we have to generate can be polynomial in the size of the input and hence larger than our space bound. The way to by-pass this problem depends
THEOREM C. (Expressive power)
Under the hypotheses of theorem B we have for both the bounded and the unbounded model of oracle computation of Buss:

Let \( K_1, \ldots, K_n \) be any classes of \( \tau \)-structures closed under isomorphisms and \( D \) one of \( L, NL, P \).

(i) If a problem \( K \) is in \( D^{K_1} \) then \( K \) is definable in \( L[Q_{K_1}] \).

(ii) For any finite \( n \), if a problem \( K \) is in \( D^{K_1, K_2, \ldots, K_n} \) (The class of Turing machines of complexity \( D \) which use the \( n \) oracles \( K_1, \ldots, K_n \), then \( K \) is definable in \( L[Q_{K_1}, Q_{K_2}, \ldots, Q_{K_n}] \) (the extension of \( L \) with the appropriate \( n \) families of quantifiers).

(iii) If furthermore \( K_1 \) is \( C \)-complete for \( L \)-reductions, then all problems \( K \) in \( D^C \) are definable in \( L[Q_{K_1}] \).

COROLLARY D. (Turing reducibility)
Let \( L \) be a regular logic capturing a relational (regular) complexity class \( D \) such that \( L \subseteq D \subseteq PH \). Let \( K \) be a set \( C \)-complete for \( D \)-reductions. Let \( Q_K \) be the family of quantifiers based on \( K \), and let \( K_1, K_2 \) be classes of structures closed under isomorphisms (not necessarily in \( D \) or \( C \)). Then \( K_1 \) is \( C \)-Turing \( D \)-reducible to \( K_2 \) iff \( K_1 \) is \( L[Q_K] \)-reducible to \( K_2 \).

Theorems B and C imply D and give (a) and (b) from the introduction for the case \( D = L, NL \) and \( P \) respectively, assuming we use the unbounded model of oracle computations of Buss. For the following section we assume, therefore, that this unbounded model be used.

5 Applications
Theorems A, B and C allow us now to construct many logics capturing regular complexity classes stepwise, without having to prove that the problem, on which the construction is based, is complete via First Order Reductions. For example, with \( D = P \) and \( C = PSPACE \), we can turn the generalized HEX problem into a quantifier and show

APPLICATION 1. \( FO\; L[HEX] \) captures \( PSPACE \) over ordered structures.

To see this we observe that \( FO\; L[ATC] \) captures \( P \) by [16], \( HEX \) is \( PSPACE \)-complete for \( P \)-reductions by [10, 12], so by theorems A, B and C \( FO\; L[ATC][HEX] \) captures \( PSPACE \). To show that \( FO\; L[HEX] \) captures \( PSPACE \) we prove that \( ATC \) is first order reducible to \( HEX \).

This result is the first characterization of \( PSPACE \) in terms of a Lindström Logic based on a single quantifier and its vectorizations. Previous characterizations by Vardi and Immerman [29, 14], where in terms of second order logics.
such that $M'(A)$ is a $\tau_3$ structure with universe $A^k$ and $M'(A)$ relativized to $P$ is isomorphic to $M(A)$.

Let

$$K_P = \{ (A, \tilde{a}) : \tilde{a} \in M'(A)(P) \}$$

and

$$K_{R_i} = \{ (A, \tilde{a}) : \tilde{a} \in M'(A)(R_i) \}.$$ 

Clearly, $K_P$ and $K_{R_i}$ are recognizable by some Turing machines in $T$. As $\mathcal{L}$ captures $D(T)$ there is $\Phi = (\psi_1, \ldots, \psi_m)$ in $\mathcal{L}$ $k$-feasible for $\tau_3$ over $\tau_1$ such that $K_P = M_\tau(\phi)$ and $K_{R_i} = M_\tau(\psi_i)$. (We allow here tacitly the passage from uninterpreted constant symbols to free variables). It now follows from the definitions that $A \in K_1$ iff $A \tilde{a} \in K_2$, which shows that $K_1$ is $\mathcal{L}$-$k$-reducible to $K_2$.

We note that our theorem is true for deterministic as well as nondeterministic reductions (for example $\text{NL}$-reductions), however a nondeterministic class has to be regular (for $\text{NL}$-reductions, for example, we need the fact that $\text{NL} = \text{Co-NL}$ for proposition 3.2). By a slight modification of the proof we can extend it to reductions via non-deterministic classes not known to be regular (such as $\text{NP}$-reductions) provided the quantifier we use is based on a set $K$ which is monotonic in the sense of [20]. (For example, The set of Hamiltonian graphs is monotonic, as adding edges to a graph does not disrupt its Hamiltonicity, however the set of three colourable graphs 3CO $L$ is not). It can be shown that the specific example of $\text{NP}$-reductions via monotonic problems is equivalent to the $\gamma$-reductions of [12].

A similar theorem for $\text{C-Turing-D}$-reducibility (that is reducibility via Turing machines in $D$ which use oracles in $C$) will come out as a corollary of our main results below.

4 Main Results

THEOREM B. (Model checking)

Let $\mathcal{L}$ be a regular logic which has a model-checker in a regular complexity class $D$. Let $K$ be a class of $\tau$-structures closed under isomorphism in $C$, $D \subseteq C$. Then

(i) $\mathcal{L}(Q_K)$, the sub-logic obtained from $\mathcal{L}$ by one application of a quantifier (predicate transformer) obtained from $K$, has a model-checker in $C$.

(ii) $\mathcal{L}[Q_K]$, the regular logic obtained from $\mathcal{L}$ by adding a family of quantifiers (predicate transformers) obtained from $K$, has a model-checker in $D^K \subseteq D^C$, (the class of $D$ machines using $K$ as the oracle - under Buss’ unbounded model).

(iii) If, additionally, $C$ is regular, $\mathcal{L}[Q_K]$ has a model-checker in $C$. 


We say that $K_1$ is $L$-reducible to $K_2$ ($K_1 \preceq L K_2$), if $K_1 \preceq_{L-k} K_2$ for some $k \in \mathbb{N}$.

Our definition of $L$-reducibility is justified by the following:

**Proposition 3.2** Let $L$ and $\Phi$ be as above and $D$ be a regular complexity class containing $L$. If $L$ is $D$-computable, then there is a Turing machine $M_\Phi \in D$ such that for every $\tau_1$-structure $A$ \[ M_\Phi (A) = A_\Phi. \]

Or more accurately: \[ w^{-1} \circ M_\Phi (w(A)) = A_\Phi. \] where $w$ is a reasonable encoding scheme from structures to strings and $w^{-1}$ is its inverse encoding from strings to structures (see [20] for a discussion of reasonable encoding schemes).

**Proof outline:** Let $M_\Phi$ operate as follows: First it writes down the universe of $A_\Phi$. This is done by generating one by one, in binary, according to the order, all possible tuples of $A^k$. The model checker of $\phi$ is invoked using the generated tuple as a substitution for the $k$ free variables of $\phi$. If the model checker accepts, a '1' is written on the output tape. Then the relations are written down one by one as follows: For a formula $\psi_i$ having $kn_i$ free variables, all possible tuples of $A^{kn_i}$ are generated. For each of the $n_i$ $k$-tuples generated, the model checker of $\phi$ is invoked. If it accepted all of them, the model checker of $\psi_i$ is then invoked using the whole $kn_i$-tuple as a substitution. If this model checker also accepts, the $kn_i$-tuple is written down (in binary) on the output tape.

Since at each step at most $kn_i$ elements of the universe are written on the auxiliary tape (in binary) the whole procedure can be completed in log-space the size of the input string (a polynomial in $|A|$) plus $D$ (the complexity of doing a polynomial number of model checkings).

We can now state and prove the following:

**THEOREM A.** Let $L$ be a regular logic capturing a relational (regular) complexity class $D$ such that $L \subseteq D \subseteq \text{PH}$. Let $K_1, K_2$ be classes of $\tau$-structures closed under isomorphisms (not necessarily in $D$). Then $K_1$ is $D$-reducible to $K_2$ iff $K_1 \preceq L K_2$ if $K_1 \preceq_{L-k} K_2$ for some $k \in \mathbb{N}$.

**Proof:** The observation that $K_1 \preceq L K_2$ implies $K_1 \preceq_{D} K_2$ is a direct corollary of proposition 3.2.

Now assume $K_1 \preceq_{D} K_2$ and let $M \in T$ be a Turing machine transforming $\tau_1$-structures into $\tau_2$-structures. As $M$ is polynomial in time there is $k \in \mathbb{N}$ such that $|M(A)| < |A|^k$. Let $\tau_3 = \tau_2 \cup \{P\}$, where $P$ is a unary relation symbol. Without loss of generality we can assume that there is a Turing machine $M' \in T$
This framework is set up such as to ensure that a relational complexity class \( C(T) \) is regular iff it consists exactly of the definable classes of structures of some computable logic.

For our purposes we restrict our attention to logics which are extensions of First Order Logic, \( FOL \), by vectorized families of Lindström quantifiers. A Lindström quantifier is defined by a class \( K \) of finite structures over some vocabulary (similarity type) \( \tau \). The vocabulary determines uniquely the syntax of the quantifier, and \( K \) determines its semantics. For \( n \) a natural number, \( n \)-vectorization of \( K \) is obtained by thinking of the universe of the structure as an \( n \)-fold cartesian product of some set \( A \) and of a, say, binary relation as a \( 2n \)-ary relation over \( A \). We denote the quantifier defined by \( K \) by \( Q_K \) and by \( FOL[Q_K] \) the smallest regular extension of first order logic which contains all the vectorized versions of \( Q_K \). Similarly, for a regular logic \( L \), \( L[Q_{K_1}, \ldots, Q_{K_m}] \) is the smallest regular extensions of \( L \) containing all the vectorized versions of each \( Q_{K_i}, i \leq m \). The details may be found in [20].

Logics based on Lindström quantifiers are inherently first order in their formation rules. In contrast to this, logics based on fragments of second order logic, such as fixed point operators or existential second order quantification, are still inherently second order in their formation rules. To find a Lindström logic equivalent to, say, fixed point logic, one has to add an infinite family of vectorized quantifiers.

Next we generalize the notion of first order reductions as introduced by Immerman and Dahlhaus [7, 8, 15] to any computable logic \( L \). Our definition is very close to Rabin’s notion of interpretability as described in [21]. A similar notion is also used in [6].

**Definition 3.1 (\( L \)-reducibility)** Let \( K_1, K_2 \) be classes of \( \tau_1(\tau_2) \)-structures closed under isomorphisms and \( L \) be a regular logic.

(i) Let \( \tau_2 = \{ R_1, \ldots, R_m \} \) and let \( \rho(R_i) \) be the arity of \( R_i \). Let \( \Phi = \langle \phi, \psi_1, \ldots, \psi_m \rangle \) be formulas of \( L(\tau_1) \). \( \Phi \) is \( k \)-feasible for \( \tau_2 \) over \( \tau_1 \) if \( \phi \) has exactly \( k \) distinct free variables and each \( \psi_i \) has \( k \rho(R_i) \) distinct free variables.

(ii) Let \( A \) be a \( \tau_1 \)-structure and \( \Phi \) be \( k \)-feasible for \( \tau_2 \) over \( \tau_1 \). The structure \( A_\Phi \) is defined as follows:

(ii.a) The universe of \( A_\Phi \) is the set \( A_\Phi = \{ \bar{a} \in A^k : A \models \phi(\bar{a}) \} \); 

(ii.b) the interpretation of \( R_i \) in \( A_\Phi \) is the set

\[
A_\Phi(R_i) = \{ \bar{a} \in A_\Phi^{\rho(R_i)k} : A \models \psi_i(\bar{a}) \};
\]

Note that \( A_\Phi \) is a \( \tau_2 \)-structure of cardinality at most \( |A|^k \).

(iii) Let \( k \in \mathbb{N} \). We say that \( K_1 \) is \( L \)-\( k \)-reducible to \( K_2 \) (\( K_1 \preceq_L \neg_k K_2 \)), if there is a \( k \)-feasible \( \Phi \) for \( \tau_2 \) over \( \tau_1 \) such that \( \mathcal{A} \in K_1 \iff \mathcal{A}_\Phi \in K_2 \) for every \( \tau_1 \)-structure \( \mathcal{A} \).
and Lynch [19] have shown that there are oracles $B$ such that $\text{NL}^B$ is not contained in $\text{P}^B$. But for logics we have $\text{FOL}[\text{TC}][Q_B] \subseteq \text{FOL}[\text{ATC}][Q_B]$ and (b) would ensure that $\text{NL}^B \subseteq \text{P}^B$. So this model of oracle computation cannot accommodate (b). The reason for this discrepancy stems from the fact that $\text{NL}$-oracle machines, in contrast to $\text{L}$-oracle machines may submit queries whose size are not polynomially bounded in the input.

There are two more models of oracle computations, introduced by Simon [23] and Ruzzo, Simon and Tompa [22], both of which are not suitable to accommodate (b). In both these models there are oracles $B$ such that $\text{P}^B$ is not contained in $\text{AL}^B$, where $\text{AL}$ stands for alternating logarithmic space. As, without oracles, we have $\text{P} = \text{AL}$, and $\text{FOL}[\text{ATC}]$ captures $\text{P}$, (b) would ensure that for every oracle $B$ we have that $\text{P}^B = \text{AL}^B$.

Buss, [2], proposes the following ‘relativization thesis’

> All general machine simulations which hold in the absence of an oracle can be extended to hold for every oracle in a properly relativized model.

In this spirit, he introduces two different models of nondeterministic and alternating oracle computation, bounded and unbounded, exactly for the purpose of avoiding anomalies as the above.

Among other things both his models ensure that the queries submitted to the oracle are polynomially bounded in the input, provided the machine asking the queries is within the polynomial hierarchy. It is for Buss’ unbounded model that we can prove (a) and (b). This model coincides with Wilson’s, [30], oracle stack model in the case of deterministic and nondeterministic space bounded machines.

### 3 The General Framework

First we introduce the notion of a relational regular complexity class, similar in spirit to both Lindström’s abstract definition of logics and the computable queries of A. Chandra and D. Harel [3, 4]. A relational complexity class $C(T)$ is the family of all problems $K$, i.e. of classes of finite ordered structures, closed under isomorphisms and renaming of basic symbols, each of which is recognizable by some Turing machine in $T$. A relational complexity class $C(T)$ is regular if $C(T)$ is closed under boolean operations, projections, $T$-reductions, substitution and $k$-relativization of predicates.

Next we introduce the notion of computable logics over finite structures. These are logics in the sense of abstract model theory with the property that checking whether a given sentence holds in a finite structure is uniformly computable from the structure. In computer science one speaks of computable model checkers. Additionally to the usual definition in model theory, we require that our logics are closed under finite vectorization. We call such logics regular.
1 Introduction

In the last 10 years several logics were exhibited which capture complexity classes such as $L$ (LogSpace), $NL$ (Non-deterministic LogSpace), $P$ (Polynomial Time), $NP$ (Non-deterministic Polynomial Time), $PH$ (the polynomial hierarchy), $[11, 16, 17, 28, 24]$. In mathematical logic the theory of abstract model theory and generalized quantifiers is well established [1]. The purpose of this paper is to show in this framework of abstract model theory how to construct, in general, logics capturing complexity classes. In particular, we explore the analogy between the extension of a logic $L$ by a (or a family of) generalized quantifier(s) and the passage from a complexity class $D$ to the class $D^A$ for some oracle $A$.

Stewart [26, 27] found that if $L$ is $FOL$ (first order logic) then its extension $FOL[HAM]$ by the quantifier $HAM$, which expresses Hamiltonicity of a graph, captures $L^{NP}$. His result uses heavily the particular definition of $HAM$. We would like to replace $HAM$ by any problem which is NP-complete via $D$ reductions, where $D$ can be $L$, $NL$ or $P$. In our context, Stewart’s results are viewed as follows: As deterministic transitive closure $DTC$, transitive closure $TC$ and alternating transitive closure $ATC$ are all in $L^{NP}$, we have that the logics $FOL[HAM]$, $FOL[DTC, HAM]$, $FOL[TC, HAM]$ and $FOL[ATC, HAM]$ have all the same expressive power (over ordered structures). As by Immermann [16] $FOL[DTC]$, $FOL[TC]$, $FOL[ATC]$ capture $L$, $NL$, $P$ respectively, we would like to conclude that

(a) $FOL[DTC, HAM]$, $FOL[DTC, HAM]$, $FOL[ATC, HAM]$ capture $L^{NP}$, $NL^{NP}$, $P^{NP}$ respectively.

(b) More generally, if $L$ captures a complexity class $D$ and $Q_A$ is a generalized quantifier, whose meaning is defined by a class of finite structures $A$, then $L[Q_A]$ captures $D^A$, where $A$ is viewed as an oracle.

We assume the reader is familiar with the basics of complexity theory as presented in [13, 12] or the excellent surveys [28, 18]. The necessary definitions from abstract model theory are given in section 3. A full account of the general framework may be found in [20]. The basics of abstract model theory may be found in [9, 5] or in the first chapters of [1].

2 Models of Oracle Computations

It turns out that our theorems will, if at all true, depend on the model of oracle computations. First we note that (a) implies that

$$L^{NP} = NL^{NP} = P^{NP}.$$ 

The model of oracle computation chosen in [26] uses deterministic machines and unbounded oracle tape. If we allow non-deterministic machines, Ladner
Oracles and Quantifiers

J.A. Makowsky* and Y.B. Pnueli
Department of Computer Science
Technion - Israel Institute of Technology, Haifa, Israel
e-mail: janos@cs.technion.ac.il, yachin@csc.cs.technion.ac.il

September 1993

And he shall stand before Eleazar the priest, who shall ask counsel
for him after the judgment of Urim [...]. (Numbers 27.21)

Abstract

We describe a general way of building logics with Lindstrom quantifiers which capture complexity classes based on oracle Turing machines. Our approach is sensitive to the oracle computation model. Our results hold for the unbounded model introduced by Buss [2] in support of his 'relativization thesis'. They do not hold for the nondeterministic and alternating oracle computations studied by Ladner and Lynch [19], Simon [23] and Ruzzo, Simon and Tompa [22]. Our results generalize and extend previous results of Stewart [26] and Makowsky and Pnueli [20].

*partially supported by a Grant of the French-Israeli Binational Foundation and by the Fund for Promotion of Research of the Technion-Israeli Institute of Technology