

A Sequential Model of Computation for First-Order Logic

Steven Lindell

Haverford College 10/94

SUMMARY:

Problems computable in constant time on a uniform parallel model of computation (a type of PRAM) have been elegantly characterized by Immerman as those expressible in first-order logic (FO) on binary strings. It is also known that FO is identified with LH, the logtime alternation hierarchy based on Turing machines. We provide an additional correspondence between FO and those problems computable in constant space on a deterministic sequential model of computation.

The key to this idea is a very careful measuring of work space in a machine. Ordinarily, read-only input and write-only output are not considered part of the read/write work space. This is an essential concept for defining the complexity class LOGSPACE. We go somewhat farther in our model, and do not include in the work space any storage mechanism required to access the input or output, be it memory addressing or tape scanning. By making the access scheme oblivious, we are careful not to let the machine cheat by using the multiple memory addresses or head positions as read/write storage.

Furthermore, we take the step of separating the flow of control of the machine from the computation it is performing. Specifically, we imagine a simple programming language with: a finite set of read/write boolean variables; the operations of AND, OR, NOT; composition of program statements; and a strict form of definite loops. No conditionals are allowed in the programming language (if...then, or while ... repeat), which insure the oblivious nature of the computation. In addition, we impose a read-once (destructive read) condition that prevents a read/write boolean variable from being read more than once without an intervening write. No such restriction applies to the input or output however.

The following is representative of our main theorem:

THEOREM: *A query on binary strings is first-order definable if and only if it is computable by an read-once constant space serial algorithm.*

Binary string structures in logic have an ordering relation on their domain. However, arithmetic (+,*) on the domain is a subtle issue which we explain the nuances of in the paper, together with details of the machine model and proof sketches.

EXAMPLES: Here are two excellent examples which illustrate some of these concepts:

This simple program computes the parity of an input string $a(1)...a(n)$, but the boolean variable p violates the read-once condition, and hence is not a valid program for the machine class we are studying:

```
p := 0                                {parity initially zero}
LOOP i FROM 1 TO n                    {go from LSB to MSB}
p := a(i) <> p                          {exclusive or}
```

Note that p must be read twice in order to form the exclusive-or from the canonical base of boolean operations, although $a(i)$ may be read any number of times. Contrast this with the schoolbook algorithm for serial binary addition of two n -bit numbers, $a(n)...a(1) + b(n)...b(1)$, which yields an n -bit sum $s(n)...s(1)$ and a carry. It requires a single read/write boolean variable c for the carry:

```
c := 0                                {initialize carry}
LOOP i FROM 1 TO n                    {go from LSB to MSB}
s(i) := a(i) <> b(i) <> c                {XORs, see comments}
c := a(i) * b(i) + c * (a(i) + b(i))    {c is read-once}
```

A simple re-parenthesization of the 3-input majority function (which is all the carry really is) results in an algorithm obeying the destructive read restriction. Note well that the reads of c used in assigning $s(i)$ don't count since the destination is write only. So this will be a valid read-once constant space serial algorithm. By our main theorem, this algorithm actually witnesses that n -bit binary addition is in FO, which I find quite surprising by itself, since the standard carry-look-ahead algorithm is not completely trivial. Conversely, the existence of a constant time parallel algorithm for binary addition implies the existence of a (read-once) constant space serial algorithm. This notion of time-space duality is explored in somewhat more detail in the paper.