The Immerman-Szelepcsényi Theorem

Preliminaries

Let M be a nondeterministic one tape off-line Turing machine with tape alphabet Γ, running on input $x \in \Sigma^*$. A *configuration*† of M is a formal description of the machine M at a particular time instant in its computation. For definiteness, we can take as configuration of M the word $p\alpha q\beta$, where (i) p is the position of the head on the input tape, encoded as a binary integer of length $\lceil \lg(n+2) \rceil$, (ii) $\alpha \in \Gamma^*$ is the sequence of tape symbols on the work tape up to but not including the one under the tape head, (iii) $q \in Q$ is the current state of M, and (iv) $\beta \in \Gamma^*$ is the sequence of tape symbols on the work tape, starting at the tape head, and ending just before the first blank symbol. If M is $S(n)$ space-bounded, the configurations of M have maximum length $S(n) + \lceil \lg(n+2) \rceil + 1$. Observe that when $S(n) \geq \ln n$ the configuration lengths are $\mathrm{O}(S(n))$.

A *computation* of M on input x is a sequence, σ, of configurations of M; the first configuration of σ is the initial configuration of M when presented with input x, and consecutive pairs of configurations in σ encode valid transitions of M. For $t \in \mathbb{N}$, a configuration is called t*-reachable* if it is the final configuration of a computation of length t (i.e., with $t+1$ configurations). Note that the property of t-reachability is defined only relative to a particular input x.

Nondeterministic space is closed under complement

The following theorem was proved independently, in 1987, by Neil Immerman [2] and Róbert Szelepcsényi [3].

Theorem. $\mathrm{NS{\small PACE}}(S(n)) = \text{co-}\mathrm{NS{\small PACE}}(S(n))$, for any function $S(n) \geq \lg n$.

Proof. Suppose $L \in \mathrm{NS{\small PACE}}(S(n))$, and let M be a nondeterministic, $S(n)$ space-bounded Turing machine which recognises L. By Theorem 12.2 of [1] we may assume that M is a one tape off-line machine. We shall construct an $S(n)$ space-bounded Turing machine $\overline{M}$ which recognises $\overline{L} = \Sigma^* - L$, the complement of L.

Consider the operation of M when presented with an input $x \in \Sigma^*$ of length n. Let $N(t)$ denote the *number* of distinct t-reachable configurations of M, and let $l(t)$ denote the maximum *length* of any such configuration. The simulating machine $\overline{M}$ will pass through a sequence of *epochs* numbered $0, 1, 2, \ldots$; during epoch t, the machine $\overline{M}$ will explore all possible computations of M up to a total of $t+1$ steps.

† *Instantaneous description* is sometimes used as a synonym for *configuration.*

Let Γ_c be an alphabet which $\overline{M}$ uses to represent configurations of the machine M. We assume that configurations can be padded to the right, so that Γ_c^l effectively contains all possible configurations of length less than or equal to l. Of course, not every element of Γ_c^l is a valid configuration, but we can easily recognise the ones which are. The operation of $\overline{M}$ within epoch t is sketched in the accompanying Pascal-like procedure. We assume that the values $N(t)$ and $l(t)$ are available from computations performed during epoch $t-1$; in fact, the values $l(t)$ and $N(t)$ represent the *only* information carried forward from one epoch to the next.

```
⟦ N(t) and l(t) are available from the previous epoch ⟧
high_water := l(t);   count_z := 0;
for each z ∈ Γ_c^{l(t)+1} do begin
    count_y := 0;   hit := false;
    for each y ∈ Γ_c^{l(t)} do begin
        guess a computation σ of length ≤ t;
        if the final configuration of σ is equal to y then begin
            increment count_y;
            if z = y or z is reachable from y in a single step then hit := true
        end
    end;
    if count_y < N(t) then reject;
    if hit then begin
        increment count_z;
        if z is an accepting configuration then reject;
        if the unpadded length of z exceeds high_water then increment high_water
    end
end;
⟦ count_z = N(t + 1) and high_water = l(t + 1) ⟧
if N(t + 1) = N(t) then accept
```

The operation of $\overline{M}$ during epoch t

A commentary on the simulation procedure is called for. The inner for-loop (over y) cycles through all potential configurations of M which are t-reachable. By non-deterministically guessing a computation σ, the procedure verifies that that the configuration y is indeed t-reachable and a counter, *count_y*, is incremented. The non-deterministic test is one sided: it can verify that a configuration is t-reachable, but not that it is t-unreachable. In general, then, the y-loop finds only a *subset* of the set of all t-reachable configurations. The event that the y-loop fails to discover all possible t-reachable configurations is detected by the test immediately following the loop, and the current nondeterministic branch of the computation is terminated. The only nondeterministic branches which survive are those in which the y-loop correctly identifies all $N(t)$ configurations which are t-reachable.

The outer loop (over z) cycles through all potential configurations of M which are $(t+1)$-reachable. (Note that, in one step, the configuration length can grow by at most one.) For every configuration z which is $(t+1)$-reachable, there is *some* configuration y which witnesses $(t+1)$-reachability; when such a witness is encountered within the inner loop, a 'hit' is recorded. The variable *count_z* counts the number of hits, and hence the number, $N(t+1)$, of $(t+1)$-reachable configurations. The maximum length, $l(t+1)$, of a $(t+1)$-reachable configuration is similarly recorded in the variable *high_water*. These values are ready for use by the procedure during epoch $t+1$. Note that the surviving nondeterministic branches, which may be numerous, all agree on the values for $N(t+1)$ and $l(t+1)$.

Since M is space bounded, there must exist a smallest $t_{\max}$ for which $N(t) = N(t_{\max})$ whenever $t > t_{\max}$. If the machine M does not accept the input x, then the simulating machine, $\overline{M}$, will reach epoch $t_{\max}$, halt, and accept. However, if M *does* accept x then $\overline{M}$, which is forced inevitably to examine *all* configurations, will eventually find an accepting computation, halt, and reject. (Note that *every* nondeterministic branch of $\overline{M}$ is forced to reject.) Thus, as required, $\overline{M}$ accepts if and only if M rejects.

We now briefly consider some of the implementation details. The simulating machine $\overline{M}$ has several work tapes: three of these record configurations of M and the remainder are binary counters. Two of the configuration tapes are used to hold the program variables y and z and the third is used to guess the computation sequence σ. Note that $\overline{M}$, being nondeterministic, can guess the configurations of σ in sequence, recording only the most recent one, and using one of the counter tapes to halt the guessing cycle after at most t steps. Other counter tapes are used to record the program variables *count_y* and *count_z*, and to pass the value of $N(t)$ from one epoch to the next. The program variable *high_water* can be implemented as a special marker symbol on each of the three configuration tapes.

We have already argued that the configurations of M, and hence the configuration tapes of $\overline{M}$, remain bounded in length by $\mathrm{O}(S(n))$. (It is here that the condition $S(n) \geq \lg n$ is used.) Furthermore, it is clear that the binary encodings of *count_y*, *count_z* and $N(t)$ are all $\mathrm{O}(S(n))$ bounded. Finally, since $N(t+1) \geq N(t)+1$ for all $t < t_{\max}$, we conclude that $t_{\max} \leq N(t)$, and hence that the binary encoding of the epoch number t is $\mathrm{O}(S(n))$ bounded. By the tape compression theorem for nondeterministic Turing machines (Corollary to Theorem 12.1 of [1]) $\overline{M}$ can be adapted to run within space $S(n)$ $\square$

Note. Immerman's statement of the theorem requires $S(n)$ to be space-constructible; it will be seen that this condition is inessential.

Consequences of the theorem

The theorem immediately settles an open question which was first posed in 1964.

Corollary 1. The complement of a context sensitive language is context sensitive.

Proof. The class of all context sensitive languages is precisely $\text{NS}\textsc{pace}(n)$. (Refer to Theorems 9.5 and 9.6 of [1]) $\square$

The theorem also yields a tight hierarchy for nondeterministic space, analogous to the deterministic space hierarchy described in Theorem 12.8 of [1].

Corollary 2. Let $S_1(n), S_2(n) \geq \lg n$ be space-bounds such that $S_2(n)$ is (fully) space-constructible, and the ratio $S_2(n)/S_1(n)$ is unbounded. Then $\text{NS}\textsc{pace}(S_2(n)) - \text{NS}\textsc{pace}(S_1(n))$ is non-empty.

Proof. We use the notation of [1]. The stumbling block to carrying across the proof of Theorem 12.8 to the nondeterministic case lies at the following point in the specification of the diagonalising machine M: the machine M is required to accept the word w iff (i) M can complete the simulation in $S_2(n)$ space, and (ii) M_w halts *without* accepting w. The crucial point is that, whereas a determinstic machine can easily perform this inversion of acceptance and rejection, a nondeterministic machine cannot.

We therefore modify the construction of the simulating machine M by changing the second criterion to: (ii) M_w halts, accepting w. To complete the proof, we assume that L, the language accepted by M, is in $\text{NS}\textsc{pace}(S_1(n))$. By the main theorem, there exists a nondeterministic $S_1(n)$ space-bounded Turing machine $\overline{M}$ which accepts $\overline{L}$, the complement of L. Now choose a a word $w \in \Sigma^*$ such that $M_w = \overline{M}$ and the ratio $S_2(|w|)/S_1(|w|)$ is sufficiently large. (Recall that M_w is the Turing machine which is encoded by the word w.) The machine M accepts the word w if and only if $M_w = \overline{M}$ does; however, this is a contradiction since the languages recognised by M and $\overline{M}$ are complementary $\square$

References

1. J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley 1979.
2. N. Immerman, *Nondeterministic space is closed under complement*, Yale University Department of Computer Science Technical Report TR 552, July 1987.
3. R. Szelepcsényi, The method of forcing for nondeterministic automata (extended abstract), *Bulletin of the European Association for Theoretical Computer Science*, October 1987, pp. 96–100.

Mark Jerrum, Edinburgh, 25th October 1988