Borodin’s Gap Theorem

In space/time hierarchy we showed that having “little” extra space or time allows us to compute “more” functions/decide more languages. However we needed the requirement that time/space bounds be “fully constructible”. Can we get rid of this requirement?

Not in general!

**Theorem (Borodin):** Suppose $h$ is a recursive function such that $h(n) \geq n$. Then there exists an increasing recursive function $g$ such that, $\text{DTIME}(g(n)) = \text{DTIME}(h(g(n)))$.

Similar Theorem applies for space.
Proof: Suppose $T_k(n)$ denotes the maximum time taken by machine $k$ on any input of length $n$. Note that $T_k(n)$ is partial recursive in $k$ and $n$.
We will construct a recursive function $g$ such that, for each $k$, at least one of the following holds.
(1) $T_k(n) \leq g(n)$ for all but finitely many $n$.
(2) $T_k(n) > h(g(n))$ for infinitely many $n$.
Thus no machine has time complexity between $g(n)$ and $h(g(n))$ for all but finitely many $n$. 
Let \( g(0) = 1 \). Define \( g(n) \), for \( n \geq 1 \) as follows.

\[
g(n).
\]

Search for a \( j > g(n - 1) \) such that, for all \( y < n \), \([T_y(n) > h(j), \text{or } T_y(n) < j]\). When such a \( j \) is found let \( g(n) = j \).

First note that such a \( j \) must exist (note that \( j = 1 + \max\{\{T_y(n) : y \leq n \text{ and } T_y(n) < \infty\}\} \) satisfies the constraints).

Claim: For every \( k \), \( g \) satisfies at least one of (1) and (2) above.

Suppose \( k \) is given. By construction, for all \( n > k \), \( T_k(n) < g(n) \) or \( T_k(n) > h(g(n)) \). Thus, either there are infinitely many \( n \) such that \( T_k(n) > h(g(n)) \), or, for all but finitely many \( n \), \( T_k(n) < g(n) \). Thus either (1) or (2) must hold.
Now, $\text{DTIME}(g(n)) \subseteq \text{DTIME}(h(g(n)))$, since $h(g(n)) \geq g(n)$. Suppose $L$ is a language in $\text{DTIME}(h(g(n)))$, as witnessed by machine $M_k$. Then for all but finitely many $n$, $T_k(n) \leq g(n)$ (since (2) is not true, (1) must be true!). Thus $L$ must also be in $\text{DTIME}(g(n))$ (finitely many inputs on which $M_k$ took more time can be patched). QED
Intuitively what the gap theorem says is that for certain $g(n)$ time bounded computations, it does not matter if we even allowed $h(g(n))$ time!

For example if $h(n) = 2^n$, then at $g(n)$ even allowing exponentially more time does not help. Contrast this with the time hierarchy theorem where we showed that if $T(n)$ is fully time constructible then even slightly more than extra logarithmic factors increases what one can accept. Of course $h(g(\cdot))$ in the above theorem cannot be fully time constructible.
Theorem: Suppose space complexity of $M$ is not bounded by a constant for strings which $M$ accepts.
That is, for every $i$, there exists an input $x$ accepted by $M$, on which $M$ uses space at least $i$.
Then, there exists a constant $c$ such that, for infinitely many $n$, $M$ uses space at least $c \log \log n$, on some input of length $n$.
Proof: We will show:
There exist infinitely many $i$ such that, $M$ uses space at least $i$ on some input (accepted by $M$) of length at most $2^{2^c i}$, for some constant $c'$. 

Space below $\log \log n$
Crossing Sequence:
sequence of (state, work tape contents/head positions, input head move direction), each time the head crosses the boundary between two input cells.

Proposition: Suppose $y = y_1y_2$ and $x = x_1x_2$. Suppose $M$ accepts by moving to the right end of the input. Consider the crossing sequence of $M$ at the boundaries of the cells, for inputs $y$ and $x$ respectively. Suppose $M$ accepts $x$ and the crossing sequence is identical at the boundary of $y_1$ and $y_2$ to that of $x_1$ and $x_2$. Then $M$ also accepts $y_1x_2$. 
Let $s$ be number of states of $M$, $r$ the alphabet size, and $k$ the number of work tapes. Consider $i$ such that $M$ uses space $i$ on some input and accepts. Let $y$ be shortest such string. Since $M$ accepts $y$, no ID is repeated. Thus, at any boundary, no component in the crossing sequence is repeated. Hence, the number of possible crossing sequences is at most $\text{factorial}(1 + 2 \cdot s \cdot i^k \cdot r^i k)$.

As crossing sequence at different boundaries are different, we have:

$$|y| \leq \text{factorial}(1 + 2 \cdot s \cdot i^k \cdot r^i k) = \text{factorial}(2^{c''}i) \leq 2^{c' i},$$

for some constants $c', c''$. QED
**Theorem:** Consider the following language \( L = \{1^k01^n : k, n \geq 2 \text{ and } n \text{ is divisible by each } c \leq k\} \). Then \( L \in \text{DSPACE}(\log \log n) \).

**Proof:** Consider the following \( M \). \( M \) rejects any input not of the form \( 1^k01^n \), for some \( k \) and \( n \geq 2 \).

\( M \) then works as follows.

1. \( c \leftarrow 1 \).

(* \( c \) is a counter and is kept in first work tape *)

Loop

2. Check whether \( n \) is divisible by \( c \).

3. If \( n \) is divisible by \( c \) then let \( c \leftarrow c + 1 \) and go to next iteration of the loop.

4. If \( n \) is not divisible by \( c \), Then check whether \( c > k \). If so accept. Otherwise reject.

Forever
One can implement step 2 above as follows:
(a) Place the input head at the beginning of $1^n$.
(b) Copy $c$ to work tape 2 (call the counter in tape 2, $c'$).
(c) Go on decrementing $c'$ on tape 2 and moving input head right with each decrement until $c'$ becomes zero or end of $1^n$ is reached.
If the end of $1^n$ is reached before $c'$ becomes 0, then $n$ is not divisible by $c$. If head reaches end of $1^n$ exactly when $c'$ becomes 0, then $n$ is divisible by $c$. If end of $1^n$ is not reached when $c'$ becomes 0, then go to (b).

Clearly, the language accepted by $M$ above is $L$. 
The space required by $M$ can be bounded as follows. Suppose $r$ is the maximum value of $c$ in the above computation. (i.e. $r$ is the least number such that $n$ is not divisible by $r$).

Then the space needed by $M$ is $O(\log r)$.

We know that $n$ is divisible by all numbers smaller than $r$, and thus all prime numbers smaller than $r$.

By the prime number theorem, for some constant $c'$, there are at least $c' * \frac{r}{\log r}$ such prime numbers.

Thus $n \geq \text{factorial}(w) \geq 2^w$, where $w = \Omega(\frac{r}{\log r})$.

Thus, $n \geq 2^{\Omega(\frac{r}{\log r})}$, for large enough $n$.

Since space used is $O(\log r)$, space used is bounded by $O(\log \log n)$ and hence by $O(\log \log(n + k + 1))$. 