COMP2009-ACE:

ADE Lec05

Rules for little-oh proofs and other advanced usages

Lecturer: Andrew Parkes
andrew.parkes@Nottingham.ac.uk

from http://www.cs.nott.ac.uk/~pszajp/

NOTES

- These slides are meant for advanced understanding.
  - Some parts have been stated elsewhere
  - The rules are methods for little-oh that might be useful. However, if a question asks for “from the definition” then such rules should not be used.

“Multiplication Rule” for big-Oh * little-Oh ?

Suppose a, c, d are all strictly positive numbers

If $\quad c < d$

then we can multiply by a to get

$\quad a*c < a*d$

Given the mnemonic “<=“ ~ “Big-Oh” and “<“ ~ “little-oh”, it is reasonable to expect to be able to multiply

“a function, and an o to get an o”.

But is this actually correct?

"Multiplication Rule" for function*little-Oh

- Suppose h(n) is (strictly) positive,
  and

  f(n) is o(g(n))   [[little-oh]]

- Then, from the definition, we know

  Forall c >0 exists n0 f(n) < c g(n) for all $n \geq n_0$
- Then multiplying by h(n) gives

  Forall c > 0   exists n0.   h(n) f(n) < c h(n) g(n) for all $n \geq n_0$
- Which is precisely: h(n) f(n) is o(h(n) g(n))

- E.g. can multiply "1 is o(n)" by h(n)=n to get "n is o(n^2)"

- (Note: for brevity, we may suppress extra conditions, such as needing h(n) > 0, as they will be obvious.)

“Multiplication Rule” for big-Oh * little-Oh ?

Suppose a, b, c, d are all strictly positive numbers

If a <= b
and c < d

then we can multiply to get

a*c < b*d

Given the mnemonic “<=“ ~ “Big-Oh” and “<“ ~ “little-oh”, it is reasonable to expect to be able to multiply “O and o to get an o”.

But is this actually correct?

“Multiplication Rule” for big-Oh * little-Oh

- Suppose
  - $f_1(n)$ is O($g_1(n)$) [[Big-Oh]]
  - $f_2(n)$ is o($g_2(n)$) [[little-oh]]
- Then, from the definition, there exist positive constants $c_1>0$, n_1 such that
  1. Exists c1 >0 exists n1 $f_1(n) \le c_1\, g_1(n)$ for all $n \ge n_1$
  2. Forall c2 >0 exists n2 $f_2(n) < c_2\, g_2(n)$ for all $n \ge n_2$
- Then multiplying gives
  - Exists c1 forall c2. exists n1 exists n2, $n_0 = \max(n_1, n_2)$,
    $f_1(n)\, f_2(n) < c_1\, c_2\, g_1(n)\, g_2(n)$ for all $n \ge n_0$
- But for any (strictly) positive value of c1, then “forall c2>0” captures the same as “forall c1*c2”.
- Hence: $f_1(n)\, f_2(n)$ is o($g_1(n)\, g_2(n)$)
- E.g. n is O(n), and 1 is o(log n), hence n is o(n log n)
- E.g. n is O(n), and log(n) is o(n), hence n log n is o(n^2)

Advanced Usages of Big-Oh

- **Some of the following slides on just a reminder of work covered before**
- They are included for
  - The ways of thinking about big-oh
  - Usages that might be seen in advanced academic literature, and in mathematical literature

Usages as sets of functions

Sometimes in (advanced theory) papers might see expressions such as

$$n^{O(1)}$$

If "big-Oh" only means "worst case of an algorithm" then this is nonsensical.

But if the big-Oh family are used to refer to sets of functions then we can take it to mean

$$\{ n^{f(n)} \mid f(n) \text{ is } O(1) \}$$

e.g. including f(n)=1, f(n)=2, f(n)=3, …. So $n^{O(1)}$ includes n^1, n^2, n^3, …

So $n^{O(1)}$ is interpreted as

"any function that is no worse than (Big-Oh of) some power law"

Usages as sets of functions

Another example can be seen in:
https://en.wikipedia.org/wiki/Stirling%27s_approximation

$$\ln(n!) = n \ln n - n + O(\ln(n))$$

Where "ln" is the "natural log" using the base e:
I.e. $\ln(y) = x$ iff $y = e^x$ $(e=2.7128\ldots)$

It means that the extra term, which one might think of as the "error", is "some function that is O(ln n)"
It means that the error is small compared to the other two terms.

This approximation of n! is relevant to later lectures on "Bounds on Comparison based sorting".

Usages as sets of functions

$1 - o(1)$ would then mean a set of functions:

$$\{ 1 - f(n) \mid f(n) \text{ is } o(1) \text{ (and } f(n) \text{ is positive) } \}$$

Hence

forall c > 0. exists n0. s.t. forall n >= n0. 0 <= f(n) < c

That is:

However small we pick c, then eventually f(n), becomes, and stays, smaller than c.

This is the same meaning as

f(n) tends to zero as n goes to infinity.

So 1-o(1) is interpreted as "any function which approaches 1 (from below) in the limit of n becoming large" – e.g. (1 – 1/n)