1. Asymptotic Analysis - Big O and Little o

1.1. Overview

This module formalizes asymptotic comparisons between functions using Big O, Little o, and related notations. These tools are fundamental for analyzing algorithm complexity, approximation quality, and limiting behavior in analysis. The framework works for functions between arbitrary normed spaces, not just real-valued functions.

1.2. Core Relations

1.2.1. IsBigOWith

The foundational relation `IsBigOWith c l f g` means that eventually along filter `l`: ‖f(x)‖ ≤ c dot ‖g(x)‖

This captures the idea that f is bounded by a constant multiple of g near the limit point of filter `l`.

Key properties:
- The constant c is explicit
- Works for any filter (at infinity, at a point, etc.)
- Domain and codomain can be different types

1.2.2. Big O Notation

The relation `f =O[l] g` means there exists some constant c such that: ∃ c in ℝ, ∀^F x in l, ‖f(x)‖ ≤ c dot ‖g(x)‖

Equivalent formulations:
- $\exists c > 0$, eventually $\|f(x)\| \leq c \cdot \|g(x)\|$
- $\exists c > 0$, eventually $c \cdot \|f(x)\| \leq \|g(x)\|$
- The ratio $\|f(x)\| / \|g(x)\|$ is eventually bounded

1.2.3. Little o Notation

The relation `f =o[l] g` means that for every positive constant: ∀ c > 0, ∀^F x in l, ‖f(x)‖ ≤ c dot ‖g(x)‖

This captures that f becomes negligible compared to g:
- The ratio $\|f(x)\| / \|g(x)\| \to 0$ along `l`
- f is dominated by arbitrarily small multiples of g

1.3. Relationships Between Notations

1.3.1. Hierarchy

The relations form a strict hierarchy:
1. `f =o[l] g` implies `f =O[l] g`
2. `f =o[l] g` implies `IsBigOWith c l f g` for any $c > 0$
3. `IsBigOWith c l f g` implies `f =O[l] g`

1.3.2. Conversions

- Little o to Big O: Every little o relation is also big O
- Big O to IsBigOWith: If $f = O(g)$, then $\exists c$, `IsBigOWith c l f g`
- IsBigOWith to Big O: Any `IsBigOWith` relation gives big O

1.4. Filter Flexibility

The framework works with any filter `l`:

1.4.1. Common Filters

- At infinity: `l = atTop` for $x \to \infty$
- At a point: `l = 𝓝 a` for $x \to a$

- Within a set: `l = 𝓝[s] a` for approach within s
- Along a sequence: `l = atTop.map u` for sequences

1.4.2. Examples

- $\sin(x) = O[\text{atTop}](1)$ - sine is bounded
- $x^2 = o[\mathcal{N}\ 0](x)$ - quadratic vanishes faster than linear near 0
- $e^x = o[\text{atTop}](e^{2x})$ - exponential growth rates

1.5. Special Cases

1.5.1. Functions to Normed Fields

When $g : \alpha \to \mathbb{k}$ where $\mathbb{k}$ is a normed field and $g(x) \neq 0$: f = o[l](g) $\iff$ "Tendsto"(x $\mapsto$ f(x)/g(x), l, $\mathcal{N}(0)$)

This connects little o to the familiar notion of limit.

1.5.2. Real-Valued Functions

For $f, g : \alpha \to \mathbb{R}$:

- The norm is just absolute value
- $f = O(g)$ means $|f| \leq c|g|$ eventually
- Captures growth rates in analysis of algorithms

1.5.3. Vector-Valued Functions

The framework handles vector spaces naturally:

- Compare magnitudes via norms
- No need to work component-wise
- Preserves geometric intuition

1.6. Algebraic Properties

1.6.1. Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$
- If $f = o(g)$ and $g = o(h)$, then $f = o(h)$
- If $f = o(g)$ and $g = O(h)$, then $f = o(h)$

1.6.2. Arithmetic Operations

The relations interact well with arithmetic:

- $O(f) + O(f) = O(f)$
- $o(f) + o(f) = o(f)$
- $O(f) \cdot O(g) = O(f \cdot g)$
- Constants: $c \cdot O(f) = O(f)$ for $c \neq 0$

1.6.3. Composition

For continuous functions:

- If $f = O(g)$ and h continuous, then $h \circ f = O(h \circ g)$ (under conditions)

1.7. Applications

1.7.1. Algorithm Analysis

- Time complexity: $T(n) = O(n \log n)$
- Space complexity: $S(n) = o(n^2)$
- Average vs worst case analysis

1.7.2. Numerical Analysis

- Truncation error: $e_n = O(h^p)$ for p-th order methods
- Convergence rates: $\|x_n - x^*\| = o(r^n)$

- Condition numbers and stability

1.7.3. Asymptotic Expansions

- Taylor series: $f(x) = \sum_{\{k=0\}}^{n} a_k x^k + o(x^n)$
- Stirling's formula: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$
- Stationary phase approximations

1.7.4. Probability Theory

- Law of large numbers: $\frac{S_n}{n} - \mu = o(1)$ a.s.
- Central limit theorem rates
- Large deviation principles

1.8. Design Philosophy

1.8.1. Generality

The module is designed for maximum generality:

- Arbitrary normed spaces (not just reals)
- Any filter (not just limits at infinity)
- Separate types for domain and codomain

1.8.2. Irreducibility

Core definitions are marked `irreducible`:

- Prevents unwanted unfolding
- Explicit lemmas for working with definitions
- Better proof performance

1.8.3. Notation

Standard mathematical notation:

- `f =O[l] g` for big O
- `f =o[l] g` for little o
- Filter annotation `[l]` makes limit point explicit

1.9. Related Concepts

The module connects to:

- Theta notation (tight bounds): $f = \Theta(g)$ means $f = O(g)$ and $g = O(f)$
- Asymptotic equivalence: $f \sim g$ means $\frac{f}{g} \to 1$
- Growth rates: Logarithmic, polynomial, exponential hierarchies
- Regularity theory: Hölder and Lipschitz continuity as special cases