Asymptotic Analysis as an Abstract Interpretation

HEEWON LEE, Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea

We formalize the leading-order asymptotic analysis as an elementary example of abstract interpretation. The formalism omits complete partial orders and fixed-point iteration while retaining the Galois connection and assertion checking. This isolates the notion of semantic approximation from the heavier mathematical machinery to handle loops and recursion. We implement and partially mechanize the formalism in Lean. We also implement an interactive shell and fuzzer for experimentation with the key ideas of abstraction, soundness, and overapproximation, making abstract interpretation more approachable. This suggests a pathway toward lightweight pedagogical frameworks for teaching core ideas in program analysis.

CCS Concepts: • **Theory of computation** → **Abstraction**; **Program analysis**; **Assertions**.

Additional Key Words and Phrases: Abstract interpretation, Asymptotic analysis

1 Introduction

Abstract interpretation is a foundational theory of approximating program semantics [3]. It guarantees soundness through the Galois connection of abstraction and concretization functions between complete partial orderings (CPOs), and computability using iterative methods based on fixed-point theorems. It has proven invaluable for program analysis, verification, and optimization [4]. However, it is often perceived as too mathematically demanding for undergraduate-level teaching. Meanwhile, asymptotic analysis is a method for analyzing the limiting behavior of functions [2]. It is central in, among others, algorithm analysis and complexity theory. The leading-order behavior of functions is particularly often used, and is simple enough to be taught in freshman courses.

In this work, we formalize leading-order analysis as an instance of abstract interpretation. This formalism serves as an intuitive, minimal, and structurally complete example of an abstract interpretation, enabling sound assertions about asymptotic behavior through semantic approximations. It significantly reduces the mathematical load on the learner, while faithfully demonstrating key concepts of abstract interpretation. Moreover, the ubiquitous topic of asymptotic analysis further makes the subject more approachable, potentially lowering the barrier to entry for students and educators. We focus on leading-term analysis rather than big-O notation because it retains quantitative information on the coefficient, better illustrating semantic approximation.

2 Formalization

Language. We define a small language to describe natural-to-real functions: $P ::= n^r \mid cP \mid P + P \mid P - P \mid P \cdot P \mid \max\{|P|, |P|\}$. The language is inspired by introductory programming language (PL) courses, which often define similar languages to model simple arithmetic expressions [7]. Here, P is a nonterminal to generate function expressions, c and r denote real numbers, and n is a terminal representing the input variable. When evaluating the function at a particular input, n is interpreted as the function's argument. We assume standard arithmetic precedence for expression evaluation. We include max for its common use in algorithm analysis; we take the absolute values of its arguments to make it compatible with the abstract ordering to be defined later.

Semantics. We define three domains of semantics for this language: denotational, concrete, and abstract. We also define order relations on the respective domains, indicating asymptotic dominance. All omitted formal definitions and proofs are provided in the appendix. The denotational domain is defined as the set of all natural-to-real functions, $\mathbb{D}_d = \mathbb{R}^{\mathbb{N}}$. The denotational semantics $[\![P]\!]_d$ of a

Author's Contact Information: Heewon Lee, heewon.lee@kaist.ac.kr, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea.
Research Advisor: Sukyoung Ryu, sryu.cs@kaist.ac.kr.

function expression is defined compositionally over arithmetic operations. We capture asymptotic dominance as a preorder $\sqsubseteq_d$ on $\mathbb{D}_d$, where $f \sqsubseteq_d g$ if and only if $|g(n)|$ grows asymptotically at least as fast as $|f(n)|$, and $\sim_d$ indicates asymptotic equivalence. These orders, unlike big-O classes, are sensitive to constant multiplications: $2n$ and $2n + 1$ are equivalent, but $2n \sqsubseteq_d 3n$ are not.

The concrete domain is defined as the power set of the denotational domain, $\mathbb{D}_c = \mathcal{P}(\mathbb{D}_d)$, equipped with the subset ordering $\sqsubseteq_c = \subseteq$. The concrete semantics $[\![P]\!]_c$ is defined as the lower closure of its denotational semantics, that is, all functions dominated by $[\![P]\!]_d$. The concrete ordering preserves the denotational structure: $[\![P_1]\!]_d \sqsubseteq_d [\![P_2]\!]_d$ whenever $[\![P_1]\!]_c \sqsubseteq_c [\![P_2]\!]_c$.

We represent the growth rate of cn^r as the tuple $(|c|, r)$ in the abstract domain $\mathbb{D}_a$. To account for possible sub- or super-polynomial behavior, we extend the domain with $\bot$ and $\top$ as well as possibly infinite coefficients. Retaining the coefficient as well as the exponent, unlike the big-O classes, allows for richer quantitative assertion checking and better illustrates the power of abstraction. The ordering of growth rates coincides with the lexicographic order $\sqsubseteq_a$, giving the exponent precedence. The abstract semantics $[\![P]\!]_a$ of a function expression is also defined compositionally over arithmetic operations. To handle term cancellations, we compute the abstract semantics with signed coefficients, and finally take their absolute values.

Galois connection. We define an abstraction function $\alpha : \mathbb{D}_c \to \mathbb{D}_a$ that assigns to each concrete set of functions the tightest representable asymptotic bound, and a concretization function $\gamma : \mathbb{D}_a \to \mathbb{D}_c$ that maps an abstract element back to all functions asymptotically dominated by it. These form a Galois connection: $\alpha(S) \sqsubseteq_a x$ whenever $S \sqsubseteq_c \gamma(x)$, indicating that the abstraction is sound. We thus establish a soundness theorem based on the Galois connection:

Theorem 2.1 (Assertion checking). *The following assertions are sound:*

(1) If $[\![P]\!]_a = \bot$, *then* $\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^r} = 0$ *for any* r.

(2) If $[\![P]\!]_a = (c, r)$ *for a finite number* c, *then* $\limsup_{n\to\infty} \left|\frac{[\![P]\!]_d(n)}{n^r}\right| \le |c|$.

(3) If $[\![P]\!]_a = (\infty, r)$, *then* $\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^s} = 0$ *for any* $s > r$.

It is noted that within our restricted language without sub- or super-polynomial behavior, one can prove stronger assertion theorems about growth rates, stated in the appendix. However, the theorem above holds more generally with possible extensions to the language.

3 Implementation and Evaluation

Implementation. We implemented the formalism in Lean 4 [10] and an interactive shell equipped with an interpreter and a fuzzer.[1] The Lean code mechanizes the three semantic domains, their orderings, the abstraction and concretization functions, and the ordering properties (reflexivity, antisymmetry, transitivity). Full mechanization of the Galois connection and assertion soundness is ongoing work. We also implemented a lightweight fuzzer that generates random function expressions and checks the consistency between the concrete and abstract semantics. Testing 3,000 randomly generated expressions yielded tight bounds in 97% of cases. Loose bounds arose primarily from coefficient cancellation, and incorrect results from floating-point error, both of which the fuzzer was deliberately configured to trigger. This tool provides empirical validation of the soundness properties and serves as a pedagogical aid in exploring the framework.

Illustration of other concepts. Our formalism faithfully illustrates other concepts of abstract interpretation, including overapproximation and abstraction hierarchies. An instructive example of overapproximation is given by the following test case generated by our fuzzer: $P = n^{9.469981} -$

[1]The implementation is available at https://github.com/pingpingy1/asymp_as_absi.

$\left(n^{-6.184874} + n^{9.469981}\right)$. While the true leading term is $n^{-6.184874}$, the structural evaluation discards this information: the subexpression $\left(n^{-6.184874} + n^{9.469981}\right)$ abstracts to $(1, 9.469981)$, causing the outer abstraction to yield "slower than every $cn^{9.469981}$." This is both a shortcoming of the abstraction and an intuitive representation of the general issue of overapproximation. Inviting the student to mitigate this issue, for example by backtracking to find the next leading term, also demonstrates the trade-off between precision and tractability.

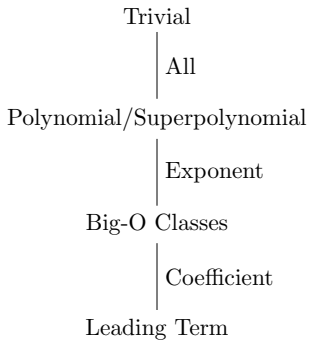

Fig. 1. Hierarchy of abstractions, with each edge indicating the information discarded by the next layer.

This formalism also serves as basis for a simple but instructive hierarchy of abstractions for analyzing the growth of functions, as shown in Figure 1. The pedagogical strength of this hierarchy is that the information lost with each additional abstraction is clear and intuitive. It also emphasizes the different roles of different layers of abstraction: leading-term analysis provides tight complexity bounds, while big-O loses coefficient information but is easier to compute.

Theoretical grounding. We define the abstract domain directly and the abstraction function structurally, which is what Cousot calls the "empirical approach to abstract interpretation" [3]. This was a deliberate choice to make the formalism as approachable as possible. Although fixed-point iterations using the widening/narrowing operators allow algorithmic constructions for general languages, such machinery is not prerequisite to sound semantic approximation. We thus contend that our work satisfies the necessary conditions of an abstract interpretation.

4 Related Work

Our primary motivation is pedagogical. Prior introductory material presents abstract interpretation axiomatically, with formal definitions of CPOs and fixed points [1, 11]. We omit these formalities in favor of the accessible example of asymptotic analysis. Asymptotic analysis has been employed in program analysis research, such as space usage [9] or time-complexity-aware type system [8]. Previous work explores the mathematical structures of asymptotic classes [6, 13]; to our knowledge, this is the first formalization of asymptotic analysis in a program analysis context.

5 Conclusion

We have formalized the leading-order asymptotic analysis as an instance of abstract interpretation. This formalism, by omitting fixed-point iterations over CPOs, clearly illustrates the core structure of semantic approximation, using a Galois connection to prove the correctness of assertions. It is a minimal, intuitive, complete example of abstract interpretation that clarifies the key concepts of soundness, overapproximation, and abstraction hierarchy. These concepts can be further explored

using the interactive shell, which demonstrates that this formalism is not only executable but also illuminating. We hope that this work can serve as both a teaching aid and a stepping stone toward intuitive lightweight analyses that can make abstract interpretation more approachable.

References

[1] Samson Abramsky and Chris Hankin. 1987. An introduction to abstract interpretation. In *Abstract Interpretation of declarative languages*, Vol. 1. Ellis Horwood London, 63–102.

[2] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[3] Patrick Cousot. 1996. Abstract interpretation. *ACM Computing Surveys (CSUR)* 28, 2 (1996), 324–328.

[4] Patrick Cousot. 2001. *Abstract Interpretation Based Formal Methods and Future Challenges*. Springer Berlin Heidelberg, Berlin, Heidelberg, 138–156. doi:10.1007/3-540-44577-3_10

[5] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. *ACM POPL* (1977).

[6] Darío García. 2020. Ordered asymptotic classes of finite structures. *Annals of Pure and Applied Logic* 171, 4 (2020), 102776. doi:10.1016/j.apal.2019.102776

[7] Jaemin Hong and Sukyoung Ryu. 2003. *Introduction to Programming Languages*.

[8] Qinheping Hu, John Cyphert, Loris D'Antoni, and Thomas Reps. 2021. Synthesis with asymptotic resource bounds. In *Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I 33*. Springer, 783–807.

[9] Manuel Montenegro, Ricardo Peña, and Clara Segura. 2015. Space consumption analysis by abstract interpretation: Reductivity properties. *Science of Computer Programming* 111 (2015), 458–482. doi:10.1016/j.scico.2014.04.014 Special Issue on Foundational and Practical Aspects of Resource Analysis (FOPARA) 2009 & 2011.

[10] Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 625–635. doi:10.1007/978-3-030-79876-5_37

[11] Xavier Rival and Kwangkeun Yi. 2020. *Introduction to static analysis: an abstract interpretation perspective*. Mit Press.

[12] David A. Schmidt. 1995. Natural semantics and abstract interpretation. In *The Essence of Computation*. Springer.

[13] Atsushi Yoshikawa. 1979. Abstract aspects of asymptotic analysis. *Journal of the Mathematical Society of Japan* 31, 3 (1979), 513–533.

A Supplementary Material

This appendix presents the formal definitions and proof sketches omitted from the main text. The definitions have been implemented in Lean 4, with mechanized proofs of the ordering properties.

A.1 Formal Definitions

Definition A.1 (Language)**.** We denote by $\mathbb{P}$ the language generated by the grammar:

$$P ::= n^r \mid 2^n \mid cP \mid P + P \mid P - P \mid P \cdot P \mid \max\{|P|, |P|\}.$$

The term 2^n is included to introduce exponential behavior.

Definition A.2 (Semantic domains)**.** We define the following three semantic domains:

(1) *Denotational domain*: $\mathbb{D}_d = \mathbb{R}^{\mathbb{N}}$
(2) *Concrete domain*: $\mathbb{D}_c = \mathcal{P}(\mathbb{D}_d)$
(3) *Abstract domain*: $\mathbb{D}_a = ([0, \infty] \times \mathbb{R}) \cup \{\bot, \top\}$

We equip each domain with respective orderings:

(1) *Denotational order* $\sqsubseteq_d$: $f \sqsubseteq_d g \Leftrightarrow (\forall \epsilon > 0)(\exists N \in \mathbb{N})(\forall n \geq N)\ |f(n)| \leq (1+\epsilon)|g(n)|$
(2) *Concrete order* $\sqsubseteq_c$: $\sqsubseteq_c = \subseteq$
(3) *Abstract order* $\sqsubseteq_a$: $\begin{cases} \bot \sqsubseteq_a x \sqsubseteq_a \top \\ (c_1, r_1) \sqsubseteq_a (c_2, r_2) \Leftrightarrow r_1 < r_2 \vee (r_1 = r_2 \wedge c_1 \leq c_2) \end{cases}$

We also define the *signed abstract domain*: $\mathbb{D}_a{}^* = ([-\infty, \infty] \times \mathbb{R}) \cup \{\bot, \top\}$.

The specific meanings of each abstract element are as follows:

- $\bot$: Subpolynomial growth.
- $(0, r)$: Slower than n^r but faster than n^s for any $s < r$.
- (c, r) $(0 < c < \infty)$: The growth rate of cn^r.
- (∞, r): Faster than n^r, but slower than n^s for every $s > r$.
- $\top$: Superpolynomial growth.

Definition A.3 (Abstract operations)**.** We define the following operations on $\mathbb{D}_a{}^*$:

(1) *Addition* $+_a$ and *subtraction* $-_a$:

$$\bot \pm_a x = x \pm_a \bot = x \quad \top \pm_a x = x \pm_a \top = \top \quad (c_1, r_1) +_a (c_2, r_2) = \begin{cases} (c_1, r_1) & (r_1 > r_2) \\ (c_1 \pm c_2, r_1) & (r_1 = r_2) \\ (\pm c_2, r_2) & (r_1 < r_2) \end{cases}$$

(2) *Scaling* $\cdot_a$:

$$c \cdot_a \bot = 0 \quad c \cdot_a \top = \begin{cases} \bot & (c = 0) \\ \top & (c \neq 0) \end{cases} \quad c \cdot_a (c_1, r_1) = \begin{cases} \bot & (c = 0) \\ (cc_1, r_1) & (c \neq 0) \end{cases}$$

(3) *Multiplication* $\cdot_a$:

$$\bot \cdot_a x = x \cdot_a \bot = \bot \quad \top \cdot_a x = x \cdot_a \top = \begin{cases} \bot & (x = \bot) \\ \top & (x \neq \bot) \end{cases} \quad (c_1, r_1) \cdot_a (c_2, r_2) = (c_1 c_2, r_1 + r_2)$$

(4) *Maximum* $\max_a$:

$$\max_a\{\bot, x\} = \max_a\{x, \bot\} = x \quad \max_a\{\top, x\} = \max_a\{x, \top\} = \top$$

$$\max_a\{(c_1,r_1),(c_2,r_2)\} = \begin{cases}(|c_1|,r_1) & (r_1 > r_2)\\ (\max\{|c_1|,|c_2|\},r_1) & (r_1 = r_2)\\ (|c_2|,r_2) & (r_1 < r_2)\end{cases}$$

Note that we abuse the notation $\cdot_a$; which operation it denotes is inferred from context.

Definition A.4 (Semantics)**.** We define the following three semantics for the language $\mathbb{P}$:

(1) *Denotational semantics*: $[\![\cdot]\!]_d : \mathbb{P} \to \mathbb{D}_d$

$$[\![n^r]\!]_d(n) = n^r \quad [\![2^n]\!]_d(n) = 2^n \quad [\![cP]\!]_d(n) = c[\![P]\!]_d(n)$$

$$[\![P_1 \pm P_2]\!]_d(n) = [\![P_1]\!]_d(n) \pm [\![P_2]\!]_d(n) \quad [\![P_1 \cdot P_2]\!]_d(n) = [\![P_1]\!]_d(n) \cdot [\![P_2]\!]_d(n)$$

(2) *Concrete semantics*: $[\![\cdot]\!]_c : \mathbb{P} \to \mathbb{D}_c$

$$[\![P]\!]_c = \left\{f \in \mathbb{D}_d : f \sqsubseteq_d [\![P]\!]_d\right\}$$

(3) *Abstract semantics*: $[\![\cdot]\!]_a : \mathbb{P} \to {\mathbb{D}_a}^*$

$$[\![n^r]\!]_a = (1,r) \quad [\![2^n]\!]_a = \top$$

$$[\![P_1 \pm P_2]\!]_a = [\![P_1]\!]_a \pm_a [\![P_2]\!]_a \quad [\![c \cdot P]\!]_a = c \cdot_a [\![P]\!]_a \quad [\![P_1 \cdot P_2]\!]_a = [\![P_1]\!]_a \cdot_a [\![P_2]\!]_a$$

$$[\![\max\{|P_1|,|P_2|\}]\!]_a = \max_a\{[\![P_1]\!]_a, [\![P_2]\!]_a\}$$

We also define the *postprocessing function* $\sigma : {\mathbb{D}_a}^* \to \mathbb{D}_a$:

$$\sigma(\bot) = \bot \quad \sigma(c,r) = (|c|,r) \quad \sigma(\top) = \top.$$

Definition A.5 (Abstraction and concretization)**.** We define the *abstraction function* $\alpha : \mathbb{D}_c \to \mathbb{D}_a$:

$$\alpha(S) = \begin{cases}\bot & (\alpha_1(S) = -\infty)\\ (\alpha_2(S),\alpha_1(S)) & (-\infty < \alpha_1(S) < \infty)\\ \top & (\alpha_1(S) = \infty)\end{cases},$$

where $\alpha_1 : \mathbb{D}_c \to [-\infty,\infty]$ and $\alpha_2 : \mathbb{D}_c \rightharpoonup [0,\infty)$ are defined as follows:

$$\alpha_1(S) = \inf\{r \in \mathbb{R} : (\forall f \in S)\ f \sqsubseteq_d n^r\},$$

$$\alpha_2(S) = \inf\left\{c \in [0,\infty) : (\forall f \in S)\ f \sqsubseteq_d cn^{\alpha_1(S)}\right\}.$$

Note that $\alpha_2(S)$ is defined only when $\alpha_1(S)$ is finite.

We also define the *concretization function* $\gamma : \mathbb{D}_a \to \mathbb{D}_c$:

$$\gamma(\bot) = \{f \in \mathbb{D}_d : (\forall r \in \mathbb{R})\ f \sqsubseteq_d n^r\},$$

$$\gamma(0,r) = \{f \in \mathbb{D}_d : (\forall c \in (0,\infty))\ f \sqsubseteq_d cn^r\},$$

$$\gamma(c,r) = \{f \in \mathbb{D}_d : f \sqsubseteq_d cn^r\},$$

$$\gamma(\infty,r) = \{f \in \mathbb{D}_d : (\forall s > r)\ f \sqsubseteq_d n^s\},$$

$$\gamma(\top) = \mathbb{D}_d.$$

In prose, α maps a set of concrete functions to the tightest asymptotic bound representable in $\mathbb{D}_a$, while γ inverts this mapping by collecting all functions dominated by the abstract element.

A.2 Proof of Galois Connection

This subsection proves the Galois connection (α, γ). We first prove some properties of the semantics and abstraction/concretization functions.

Lemma A.6 (Alternate formulation of denotational ordering)**.** If $g(n) \neq 0$ for any natural number n, then $f \sqsubseteq_d g$ if and only if $\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \leq 1$.

Proof. By definition, $f \sqsubseteq_d g$ is equivalent to $\left|\frac{f(n)}{g(n)}\right| \leq 1 + \epsilon$ for any positive ϵ and sufficiently large n. The lemma then follows from the definition of lim sup. □

Lemma A.7 (Compatibility of the different semantics)**.** For any $P \in \mathbb{P}$, $\alpha\left(\llbracket P \rrbracket_c\right) \sqsubseteq_a \sigma\left(\llbracket P \rrbracket_a\right)$.

Proof sketch. The proof proceeds by structural induction on the syntax of P. We provide representative cases here, as the full proof is lengthy and routine.

(1) $P = n^r$
In this case, $\llbracket P \rrbracket_d(n) = n^r$, so the theorem holds with $\alpha(\llbracket P \rrbracket_c) = \sigma(\llbracket P \rrbracket_a) = (1, r)$.

(2) $P = P_1 \cdot P_2$, $\llbracket P_1 \rrbracket_a = (c_1, r_1)$, $\llbracket P_2 \rrbracket_a = (c_2, r_2)$
Then $\sigma(\llbracket P \rrbracket_a) = (|c_1 c_2|, r_1 + r_2)$. For any $f \sqsubseteq_d \llbracket P \rrbracket_d$,

$$\begin{aligned}\limsup_{n\to\infty} \left|\frac{f(n)}{n^{r_1+r_2}}\right| &\leq \limsup_{n\to\infty} \left|\frac{\llbracket P \rrbracket_d(n)}{n^{r_1+r_2}}\right| \\ &= \limsup_{n\to\infty} \left|\frac{\llbracket P_1 \rrbracket_d(n)}{n^{r_1}} \cdot \frac{\llbracket P_2 \rrbracket_d(n)}{n^{r_2}}\right| \\ &\leq |c_1 c_2|\end{aligned}$$

and thus $f \sqsubseteq_d |c_1 c_2|\, n^{r_1+r_2}$. Therefore, $\alpha(\llbracket P \rrbracket_c) \sqsubseteq_a (|c_1 c_2|, r_1 + r_2)$ and the theorem holds.

The proofs for other cases follow similarly.

Lemma A.8 (Properties of α and γ)**.** For all $S \in \mathbb{D}_c$ and $x \in \mathbb{D}_a$:

(1) $S_1 \sqsubseteq_c S_2 \Rightarrow \alpha(S_1) \sqsubseteq_a \alpha(S_2)$ (i.e., α is monotone.)
(2) $x_1 \sqsubseteq_a x_2 \Rightarrow \gamma(x_1) \sqsubseteq_c \gamma(x_2)$ (i.e., γ is monotone.)
(3) $S \sqsubseteq_c \gamma(\alpha(S))$
(4) $\alpha(\gamma(x)) \sqsubseteq_a x$

Proof.

(1) If $S_1 \sqsubseteq_c S_2$, then $\{r \in \mathbb{R} : (\forall f \in S_1)\ f \sqsubseteq_d n^r\} \supseteq \{r \in \mathbb{R} : (\forall f \in S_2)\ f \sqsubseteq_d n^r\}$. It follows that $\alpha_1(S_1) \leq \alpha_1(S_2)$.
If $\alpha_1(S_1) < \alpha_1(S_2)$, then the result follows immediately from lexicographical ordering. The same holds if $\alpha_1(S_1) = -\infty$ or $\alpha_1(S_2) = \infty$. Otherwise, assume $\alpha_1(S_1) = \alpha_1(S_2) = r \in \mathbb{R}$. In such a case, $\{c \in [0, \infty) : (\forall f \in S_1)\ f \sqsubseteq_d cn^r\} \supseteq \{c \in [0, \infty) : (\forall f \in S_2)\ f \sqsubseteq_d cn^r\}$, implying $\alpha_2(S_1) \leq \alpha_2(S_2)$ and consequently $\alpha(S_1) \sqsubseteq_a \alpha(S_2)$.

(2) (a) $x_1 = (c_1, r_1)$, $x_2 = (c_2, r_2)$, $r_1 < r_2$
Suppose $f \in \gamma(x_1)$, so $f \sqsubseteq_d c_1 n^{r_1}$. Since $c_1 n^{r_1} \sqsubseteq_d c_2 n^{r_2}$, the transitivity of $\sqsubseteq_d$ yields $f \sqsubseteq_d c_1 n^{r_1} \sqsubseteq_d c_2 n^{r_2}$, so $f \in \gamma(x_2)$, and thus $\gamma(x_1) \sqsubseteq_c \gamma(x_2)$.
(b) $x_1 = (c_1, r)$, $x_2 = (c_2, r)$, $c_1 \leq c_2$
Suppose $f \in \gamma(x_1)$, so $f \sqsubseteq_d c_1 n^r$. Since $c_1 n^r \sqsubseteq_d c_2 n^r$, the transitivity of $\sqsubseteq_d$ yields $f \sqsubseteq_d c_1 n^r \sqsubseteq_d c_2 n^r$, so $f \in \gamma(x_2)$, and thus $\gamma(x_1) \sqsubseteq_c \gamma(x_2)$.

(c) $x_1 = \bot$
Here $\gamma(x_1)$ is the set of subpolynomial functions, which is contained in every $\gamma(x)$.
(d) $x_2 = \top$
Here $\gamma(x_2) = \mathbb{D}_d$, which contains every $\gamma(x)$.
In all cases, $x_1 \sqsubseteq_a x_2 \Rightarrow \gamma(x_1) \sqsubseteq_c \gamma(x_2)$.

(3) (a) $\alpha_1(S) = -\infty \Rightarrow \alpha(S) = \bot$
This means that for every $r \in \mathbb{R}$ and $f \in S$, $f \sqsubseteq_d n^r$, i.e., f is subpolynomial. Since $\gamma(\bot)$ is defined as the set of all subpolynomial functions, $S \sqsubseteq_c \gamma(\bot)$.
(b) $\alpha_1(S) = r \in \mathbb{R}$, $\alpha_2(S) = 0$
For every $c > 0$ and $f \in S$, $f \sqsubseteq_d cn^r$. Therefore, $S \sqsubseteq_c \gamma(0, r)$.
(c) $\alpha_1(S) = r \in \mathbb{R}$, $\alpha_2(S) = c \in (0, \infty)$
The condition of $\alpha_2(S) = c$ can be expressed equivalently as $\limsup_{n\to\infty} \left|\frac{f(n)}{n^r}\right| \leq c_1$ for any $f \in S$ and $c_1 > c$. This implies that the limit supremum exists for each f, and that it does not exceed c_1. Hence, we have $f \sqsubseteq_d cn^r$ for all $f \in S$. Therefore, $S \sqsubseteq_c \gamma(c, r)$.
(d) $\alpha_1(S) = r \in \mathbb{R}$, $\alpha_2(S) = \infty$
For every $s > r$ and $f \in S$, $f \sqsubseteq_d n^s$. Therefore, $S \sqsubseteq_c \gamma(\infty, r)$.
(e) $\alpha_1(S) = \infty \Rightarrow \alpha(S) = \top$
In this case, S must be contained in $\mathbb{D}_d = \gamma(\top)$.
In all cases, $S \sqsubseteq_c \gamma(\alpha(S))$.

(4) (a) $x = \bot$
$\gamma(\bot)$ is the set of all subpolynomial functions. This implies $f \sqsubseteq_d n^r$ for every $f \in \gamma(\bot)$ and that $\alpha_1(\gamma(\bot)) = -\infty$. Therefore, $\alpha(\gamma(\bot)) = \bot$.
(b) $x = (0, r)$
Every $f \in \gamma(0, r)$ satisfies $f \sqsubseteq_d n^r$. Thus, $\alpha_1(\gamma(0, r)) \leq r$. Meanwhile, $n^s \in \gamma(0, r)$ for any $s < r$, so $\alpha_1(\gamma(0, r)) \geq s$. Thus, $\alpha_1(\gamma(0, r)) = r$. Next, for any $f \in \gamma(0, r)$ and $c > 0$, $f \sqsubseteq_d cn^r$, which implies $\alpha_2(\gamma(0, r)) = 0$. Therefore, $\alpha(\gamma(0, r)) = (0, r)$.
(c) $x = (c, r)$, $c \in (0, \infty)$
For every $f \in \gamma(c, r)$ and $s > r$, $f \sqsubseteq_d cn^r \sqsubseteq_d n^s$. Hence, $\alpha_1(\gamma(c, r)) = r$. Next, every $f \in \gamma(c, r)$ satisfies $f \sqsubseteq_d cn^r$, which implies $\alpha_2(\gamma(c, r)) = c$. Therefore, $\alpha(\gamma(c, r)) = (c, r)$
(d) $x = (\infty, r)$
For every $f \in \gamma(\infty, r)$ and $s > r$, $f \sqsubseteq_d n^s$. Also, $n^r \in \gamma(\infty, r)$. Hence, $\alpha_1(\gamma(\infty, r)) = r$. Next, for every $c > 0$, $cn^r \in \gamma(\infty, r)$. Therefore, $\alpha_2(\gamma(\infty, r)) = \infty$ and $\alpha(\gamma(\infty, r)) = (\infty, r)$.
(e) $x = \top$
Since $\gamma(\top) = \mathbb{D}_d$ contains every single natural-to-real functions, it clearly contains superpolynomial ones. Thus, $\alpha_1(\gamma(\top)) = \infty$ and $\alpha(\gamma(\top)) = \top$.
In every case, $\alpha(\gamma(x)) \sqsubseteq_d x$; in fact, they are equal in every case. □

It is a well-known fact that Lemma A.8 is equivalent to a Galois connection between partially ordered sets. We refer to standard treatments of abstract interpretation for a proof [5, 12].

Theorem A.9 (Galois connection). *α and γ form a Galois connection; that is, $\alpha(S) \sqsubseteq_a x \Leftrightarrow S \sqsubseteq_c \gamma(x)$ for any $S \in \mathbb{D}_c$ and $x \in \mathbb{D}_a$.*

A.3 Correctness of Assertion Checking

Proof sketch of Theorem 2.1. Lemma A.7 and Theorem A.9 together imply $[\![P]\!]_d \in [\![P]\!]_c \sqsubseteq_c \gamma\left(\sigma\left([\![P]\!]_a\right)\right)$. The assertions follow directly from the definition of the concretization function γ.

For instance, if $[\![P]\!]_a = (c, r)$, then $[\![P]\!]_d \in \gamma(|c|, r)$, so $\left|[\![P]\!]_d(n)\right| \leq (1+\epsilon)\,|c|\,n^r$ for any positive ϵ and sufficiently large n. This establishes the upper bound in assertion (ii). The other assertions follow analogously from the structure of γ.

A.4 Stronger Assertions for the Simple Language

Theorem A.10 (Stronger assertions for the simple language). *The following assertions are sound for function expressions in $\mathbb{P}$:*

(1) $[\![P]\!]_a = \bot \Rightarrow (\forall n \in \mathbb{N})\ [\![P]\!]_d(n) = 0$

(2) $[\![P]\!]_a = (c, r) \Rightarrow \lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^r} = c.$

Proof. We proceed by structural induction on the expression $P \in \mathbb{P}$.

(1) $P = n^r$
In this case, $[\![P]\!]_a = (1, r)$ and $[\![P]\!]_d(n) = n^r$, so (ii) holds with $\lim_{n\to\infty} \frac{n^r}{n^r} = 1$.

(2) $P = 2^n$
In this case, $[\![P]\!]_a = \top$, so the theorem does not apply.

(3) $P = rQ$, $r \neq 0$

(a) $r = 0$ or $[\![Q]\!]_a = \bot$
In this case, $[\![P]\!]_a = \bot$ and $[\![P]\!]_d(n) = 0$, so (i) holds.

(b) $r \neq 0$, $[\![Q]\!]_a = (c_1, r_1)$
In this case, $[\![P]\!]_a = (rc_1, r_1)$, so (ii) holds with

$$\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^{r_1}} = rc_1.$$

(c) $r \neq 0$, $[\![Q]\!]_a = \top$
In this case, $[\![P]\!]_a = \top$, so the theorem does not apply.

(4) $P = P_1 \pm P_2$
Assume without loss of generality that $[\![P_1]\!]_a \sqsubseteq_a [\![P_2]\!]_a$.

(a) $[\![P_1]\!]_a = \bot$
In this case, $[\![P_1]\!]_d(n) = 0$, so

$$[\![P]\!]_a = \begin{cases} \bot & ([\![P_1]\!]_a = \bot) \\ (-c, r) & ([\![P_1]\!]_a = (c, r)) \\ \top & ([\![P_1]\!]_a = \top) \end{cases}$$

and $[\![P]\!]_d(n) = \pm[\![P_1]\!]_d$. Hence, the theorem holds due to the inductive hypothesis.

(b) $[\![P_1]\!]_a = (c_1, r_1)$, $[\![P_2]\!]_a = (c_2, r_2)$, $r_1 < r_2$
In this case, $[\![P]\!]_a = (\pm c_2, r_2)$. Therefore, (ii) holds with

$$\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^r} = \lim_{n\to\infty} \frac{[\![P_1]\!]_d(n) \pm [\![P_2]\!]_d(n)}{n^r} = 0 \pm c_2 = \pm c_2.$$

(c) $[\![P_1]\!]_a = (c_1, r)$, $[\![P_2]\!]_a = (c_2, r)$, $c_1 \leq c_2$
In this case, $[\![P]\!]_a = (c_1 \pm c_2, r)$. Therefore, (ii) holds with

$$\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^r} = \lim_{n\to\infty} \frac{[\![P_1]\!]_d(n) \pm [\![P_2]\!]_d(n)}{n^r} = c_1 \pm c_2.$$

(d) $[\![P_2]\!]_a = \top$
In this case, $[\![P]\!]_a = \top$, so the theorem does not apply.

(5) $P = P_1 \cdot P_2$
Assume without loss of generality that $[\![P_1]\!]_a \sqsubseteq_a [\![P_2]\!]_a$.

(a) $[\![P_1]\!]_a = \bot$
In this case, $[\![P]\!]_a = \bot$ and $[\![P]\!]_d(n) = 0$, so (i) holds.
(b) $[\![P_1]\!]_a = (c_1, r_1)$, $[\![P_2]\!]_a = (c_2, r_2)$
In this case, $[\![P]\!]_a = (c_1 \cdot c_2, r_1 + r_2)$, so (ii) holds with

$$\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^{r_1+r_2}} = \lim_{n\to\infty} \left(\frac{[\![P_1]\!]_d(n)}{n^{r_1}} \cdot \frac{[\![P_2]\!]_d(n)}{n^{r_2}} \right) = c_1 \cdot c_2.$$

(c) $[\![P_2]\!]_a = \top$, $[\![P_1]\!]_a \neq \bot$
In this case, $[\![P]\!]_a = \top$, so the theorem does not apply.

(6) $P = \max\{|P_1|, |P_2|\}$
Assume without loss of generality that $[\![P_1]\!]_a \sqsubseteq_a [\![P_2]\!]_a$.
(a) $[\![P_1]\!]_a = [\![P_2]\!]_a = \bot$
In this case, $[\![P]\!]_a = \bot$ and $[\![P]\!]_d(n) = 0$, so (i) holds.
(b) $[\![P_2]\!]_a = (c, r)$
In this case, $[\![P]\!]_a = (|c|, r)$, and

$$\left| \limsup_{n\to\infty} \frac{[\![P_1]\!]_d(n)}{n^r} \right| \leq c, \quad \lim_{n\to\infty} \frac{[\![P_2]\!]_d(n)}{n^r} = c.$$

Therefore, (ii) holds with

$$\lim_{n\to\infty} \frac{[\![P]\!]_d(n)}{n^r} = \lim_{n\to\infty} \frac{\left|[\![P_2]\!]_d(n)\right|}{n^r} = |c|.$$

(c) $[\![P_2]\!]_a = \top$
In this case, $[\![P]\!]_a = \top$, so the theorem does not apply.

Therefore, the theorem holds in all cases. □