# INTRODUCTION

ROBERT SANDERS

This document is a description of a computer program called *Complexity Zoology*. The program is an *expert system*: it is equipped with a database of information about which complexity classes are subsets of other complexity classes along with an inference engine that the program uses to deduce new conclusions from the existing information. Complexity Zoology then outputs a diagram of the relationships between the complexity classes in the input file.

Complexity Zoology takes its name from the Complexity Zoo, an online wiki of information about complexity classes maintained by Scott Aaronson. This version of Complexity Zoology was written from scratch; however, some of the design choices – particularly the input syntax and the functionality of the output diagram – are based on an earlier version by Greg Kuperberg.

The purpose and motivation of Complexity Zoology is the partial automation of surveying the field of complexity theory. The scope of a field as well-developed as complexity theory is so large that it can be difficult to quickly determine the status of a proposition of the form "the complexity class $\mathsf{C}_1$ is contained in the complexity class $\mathsf{C}_2$." Complexity Zoology aims to help with this problem, at least for the most fundamental complexity classes. More specifically, the project aims to

(1) summarize a large portion of the known complexity class inclusions and oracle separations;
(2) identify redundant results (i.e., the results that follow logically from other known results);
(3) answer questions about complexity class relations automatically when the answer is a corollary of established results.

In the ideal case, Complexity Zoology could determine the truth or falsity of a complexity lass inclusion that had not been considered before. Evern without this outcome, however, the program has already proved itself usefule in identifying when a a result is a corollary of other results and in falsifying conjectures.

To make Complexity Zoology an effective tool, it has been necessary to limit its scope in a few ways. First, it is worth noting that the system's expertise does not lie in reasoning about complexity theory as such. It knows nothing of the standard techniques used to prove results in complexity theory, such as diagonalization. It does not even understand what complexity classes are: even common classes such as $\mathsf{P}$, $\mathsf{NP}$, and $\mathsf{BPP}$ are understood only in terms of their relationships to other complexity classes, Instead, the strength of Complexity Zoology consists of understanding results and open problems in the field. For example, if it is known that $\mathsf{C}_1 \subseteq \mathsf{C}_2$ is proven and $\mathsf{C}_1 \subseteq \mathsf{C}_3$ is an open question, then Complexity Zoology can conclude that $\mathsf{C}_2 \subseteq \mathsf{C}_3$ is unproven – either it has been disproven, or it is itself open. In essence, the system can be thought of as a diligent student conducting a broad overview of the field, drawing connections between results and attempting to fill in all possible gaps but not examining the details too closely.

Second, the project has been deliberately limited to a core collection of important complexity classes. The program has the capacity to identify critical gaps in its knowledge, and by choosing a conservative list of classes we increase the chance that the questions Complexity Zoology asks are of theoretical interest. The system's input syntax makes it easy to add and remove classes as needed, so adjustments can be made as necessary.