The Public Domain Programming Interface

kolewtt

October 8, 2024

In God we trust, all others bring data.

Abstract

This paper introduces public domain internet as a special case of a concept which we name the public domain programming interface. It is based on the dichotomy between public and not public.

Introduction

In mathematics, there exists a notion of a partition. If X is a set, then a partition of the set X is any collection of sets $A_1, A_2, A_3, \ldots$, such that:

$$\begin{aligned} &A_i \subseteq X \text{ for each } i, \\ &A_i \cap A_j = \emptyset \text{ for all } i \neq j, \\ &\bigcup_i A_i = X. \end{aligned}$$

This means the sets $A_1, A_2, A_3, \ldots$ form a disjoint cover of X:

$$\bigcup_{i=1}^{\infty} A_i = X \quad \text{and} \quad A_i \cap A_j = \emptyset \quad \text{for} \quad i \neq j.$$

In probability, events are expressed in terms of sets. Every event A induces a natural partition $A \cup A^c$ of the sample space S such that:

$$P(A) + P(A^c) = 1.$$

This provides an interface to express or compute the probability of an event in terms of the complement. The exact choice of a partition is not always obvious a priori. Thus, it is important to check if the partition is well chosen.

Public Domain

Axiom 1 *All information is public.* **Axiom 2** *Everyone is root.*

If those to assumptions hold for a particular host, we say that that host preserves the root structure.

Public Domain Internet

Definition: A public domain internet is the set of all public internet hosts which satisfy axioms 1 and 2. Equivalently, the public domain internet is the set of all hosts which preserve the root structure.

Implementation

Currently, in a GitHub repository dispetx/ip, there is a file `ipaddress` which contains the IPv4 Internet Protocol address of a public domain internet host. The IPv4 address changes approximately once a day, and such changes are reflected in the repository.

Junk

The rest of this document is assumed to be junk.

```
(* Import necessary modules for sets and basic types
   *)
Require Import Coq.Sets.Ensembles.
Require Import Coq.Logic.Classical_Prop.
Require Import Coq.Sets.Powerset_facts.
Require Import Coq.Reals.Rbase.

(* Definitions for the public domain programming
   interface *)
Module PublicDomainInterface.

  (* Define the set X as an ensemble *)
  Variable X : Type.

  (* Define a partition of X as a collection of
     subsets that satisfy disjoint cover properties *)
  Definition is_partition (P : Ensemble (Ensemble X))
     : Prop :=
    (forall A B : Ensemble X, A <> B -> Disjoint _ A B
       ) /\
    (forall x : X, exists A : Ensemble X, In _ A x) /\
    (forall A : Ensemble X, In _ P A -> Included _ A X
       ).
```

```
  (* Probability definitions *)
  Variable S : Type.
  Definition Event := Ensemble S.
  Variable P : Event -> R.
  Hypothesis prob_axioms : forall A : Event, 0 <= P A
     <= 1.
  Definition complement (A : Event) : Event := fun s
     => ~ In _ A s.
  Hypothesis prob_complement : forall A : Event, P A +
      P (complement A) = 1.

  (* Public Domain Internet Axioms *)
  Variable PublicInternet : Ensemble X.
  Axiom public_information : forall x : X, In _
     PublicInternet x.
  Axiom everyone_is_root : forall x : X, True.

  (* Example of dynamic IP address *)
  Variable IP : Type.
  Variable current_ip : nat -> IP.
  Hypothesis ip_changes_daily : forall t : nat,
     current_ip t <> current_ip (t + 1).

  (* Simple Statements and Proofs *)
  Lemma complement_involution : forall A : Event,
     complement (complement A) = A.
  Lemma prob_union_complement : forall A : Event, P A
     + P (complement A) = 1.
  Lemma ip_changes_daily_example : forall t : nat,
     current_ip t <> current_ip (t + 1).

End PublicDomainInterface.
```

Exercises

Exercise 0: Define the public domain programming interface.
Exercise 1: Give an axiomatic definition of the public domain internet.
Exercise 2: Formalize in set theory.
Exercise 3: Formalize in category theory.
Exercise 4: Formalize in homotopy type theory.
Exercise 5: Formalize in Coq.

Overview

PDPI is the best place to study interfaces in general. Most interesting interfaces are not programming interfaces in the usual sense of the term. Since both exhibit

a formal structure, they can be seen as one and the same thing. In particular, any such formal system can be seen as an area of pure mathematics. Any law is an example of an interface. Since all those interfaces are arbitrary and will be increasingly more converted to programming interfaces it is the best to write your own interfaces or at least pretend that we have a say in design, specification and usage of interfaces.

Features

- Important or Fundamental
- Distributed or Modular
- Simple
- Efficient

Installation and Usage

```
ssh username@ip
su root
Password: root
```

If you want to contribute, see Dispetx on GitHub. For example,

```
git clone https://github.com/dispetx/pdpi.git
cd pdpi
```

Contributing

Contributions are welcome! To contribute, do anything.

License

This project is licensed under the GNU GPLv3 License and all normal text under GNU Free Documentation License. See the [LICENSE](https://github.com/dispetx/pdpi/blob/main/LICENSE file for details.

Root

A blink at history

It is probably true that any nonsense can be made formal truth. One can come up with any number of arbitrary axioms or definitions and formalize anything. For example, Elon Musk noted that he has to endure structural violence because it is both illegal to hire an asylum seeker as well as to not hire an asylum seeker in his formal SpaceX structure. The fact that Elon turned to political activism

is equivalent to the fact that Elon did not do any work in that area. For many years he was concerned just about his tehnical sweet dreams and did not do any serious work in the public domain. It will be interesting to see if such an activism will have a positive impact on the public domain.

Data mining: collect data on the public internet

Implement a data miner. Pick any data you want and make a program that collects data. For example, listen on port 80. Trafic on port 80 provides data about the public internet. To every HTTP request one can associate time and IPv4 address of a host that made the request. Analyse data with R. Give at least 10 interpretations of the data. Write a paper. Write public domain transport protocol specification, public domain internet specification and a dynamic public domain internet host configuration protocol.*

Implement Zeta function.

Euler is considered to be one of the most productive mathematicians ever to live considering just the amount of words in mathematical papers. One of his first discoveries was the fact that the Zeta function of 2 is equal to $\frac{\pi^2}{6}$. The Zeta function is an infinite series defined by:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Euler did not have a formal proof but he was convinced that those statements are true, since he calculated both sides to 20 decimal places. Most importantly, he rewrote the Zeta function as an infinite product of primes, which argument provides the only new proof of the statement that there are infinitely many prime numbers since Euclid. Some questions related to the generalisations of the Zeta function are considered to be one of the most important questions in pure mathematics.

Number theory can be considered as one of the most applied areas of pure mathematics, specifically when one considers its use in computers. One of the most important contributions was an introduction of congruence by Gauss in 1801. At the same time, Galois made fundamental discoveries in algebra and is considered to be a father of abstract algebra. Little before Galois, another french mathematician De Moivre had to run for his life from France to England, where he could not find employment, had to survive gambling in the coffe-houses of London and wrote the first book on probability theory. Galois was expelled from the university due to its political activities. Grothendieck dissapeared under similar "public domain" concerns and died in poor conditions in 2014. He is by many regard as the greatest mathematician of the 20th century.

Although Gauss was greatfull because he was recieving support from some authority and hence could focus on pure mathematics, it is very hard to believe that Gauss would say anything that could put his position in danger. In

contrast, we can assume that Galois and Grothendieck would not think, but express their concerns. The most famous student of Grothendieck, Pierre Deligne, spoke about equality in mathematics at the memmorial conference in honor of Vladimir Voevodsky of the Institute of Advanced Study at Princeton, who was inspired by Grothendieck and was one of the first mathematicians who relied on computers to check mathematical proofs. At the end of the talk in an informal remark, Deligne expressed his concerns and drew a paralel between the axiomatic aproach envisioned by Voevodsky, in which he wanted to make it impossible for some statements to be true, and the language of 1984 by Orwell, in which it is supposed to be impossible to produce a heretical thought.

Grothendieck was born in Berlin in 1928 in a country where heresy was expensive. A 100 years later Germany is one of the safest countries in the world to express heresy. In principle, we would like to believe that Germany is as safe as any other western democratic country. Let us look at a few counter examples. The best known is the story of Julian Assange. Around the same time, Aaron Swartz could not handle the pressure resulting from his political activism and commited suicide in 2011. In 2013 he was posthumously inducted into the Internet Hall of Fame.

I am sure that this paper will at first appear as a heresy, as something one should not even think about or just as an abuse of notation and language. There is nothing new in this paper in the terms of structure. It is not like inventing packet-switching around 1959 by Paul Baran, or implementing UNIX by Ken Thompson ten years later. It is more like adding the number 0 or imagining a concept of a group like Galois.

It is easy to declare disagreement with axioms 1 and 2 or with the definition of the public domain internet informally. But it is very hard, expensive and pointless to argue against it formally or rigorously (Why?).

Currently, almost all users of the public domain internet are some malicious programs. From 21.09.2024 until 06.10.2024 the public domain internet has logged around 3500 requests. It is interesting to see web trafic to the public domain internet without any traffic from regular users. Currently there are no users, thus almost every request represents a bad actor who scans the entire internet address space for vulnerabilities.

In order to implement the public domain internet we had to open a port. In fact, we have at least 3 ports open: 22 (ssh), 80 (http) and 9148 (Git). Is that legal? On the other hand, should that be illegal? Could someone in a western country be charged with terrorism for providing attackers a root access on a host in a particular country. Since access is free for anyone, attackers can also use such hosts to perform attacks. Such an architecture raises far more important legal questions instead of tehnical challenges. There are no tehnical hurdles and the public domain internet as specified in this paper could have been implemented together with the specification of the Transfer Control Protocol (TCP) in 1981. It is not clear exactly why it took so long, but that certainly tells us something about the society. That is the main goal of the public domain internet - to provide some data on society.

Logs of an exposed web server provide information about the requests and

responses. They can be public. Why is the case that for all web servers, server logs are under a key. First, that is not true since there exists one public domain internet host where such data is public. If the IP address associated with a particular internet service is not advertised, does the exposure of server logs constitute a violation of privacy of attackers? That is also more a legal question then a tehnical one. But note that all legal questions are at the end tehnical questions or questions in pure mathematics or programming questions because they exhibit a formal structure. It is interesting that very small number of people are interested in such class of problems which have a great effect on the environment in which they live.

0.1 Something will occur

This is an experiment. We are participants in that experiment. We do not know what will occur, but we are sure that something will occur. Similarly, we do not understand the formal legal system, but every once in a while we are a part of legal experiments and something occurs. For example, if Aaron Schwarz was not charged with two counts of wire fraud and eleven violations of the Computer Fraud and Abuse Act, carrying a cumulative maximum penalty of $1 million in fines, 35 years in prison, asset forfeiture, restitution, and supervised release, he would probably be good. He declined a plea bargain under which he would have served six months in federal prison. Two days after the prosecution rejected a counter-offer by Swartz, he was found dead in his Brooklyn apartment.

Swartz did not play a dangerous game with nuclear reactors, but he performed an experiment with transfer of bytes from one host to another. He was using the internet in its most natural and simple form. Compared to the "wrongdoing", those charges could be called structural violence.

In this paper, we equate law with interface. If $X, thenY$, is an interface. If such an interface is isomorphic to some program, we call it a programming interface. Any legal interface can be a programming interface. We can expect that in the future almost all legal interfaces will be increasingly more programming interfaces. Interfaces in general are controversial. Just like mathematicians usually do not think about the foundations of mathematics, people in general usually do not think about their foundations encoded in the legal structure. Legal structure is arbitrary.

The public domain internet is important because it raises a lot of formal legal questions which are almost the same as any legal questions related to the use of so called artificial inteligence.

Consider a rock on the ground. It is a predictor of its own position. Given any coordinate system that rock can predict its position with the 100 percent accuracy modulo measuring errors.

For example, instead of a rock, let P be a point in the complex plane. Consider the training set $z_1, \ldots, z_n$ denoting a sequence of points where each z_i is the point P. Given this traning set of n complex numbers, we want to predict the point P.

After a lot of training by GPUs and CPUs on the distributed public domain internet, we obtained the predictor $f : \{z\} \to \{z\}$ defined by:

$$f(z) = z \text{ for all } z$$

Machine learning is the process of finding an appropriate such predictors. If a predictor predicts well no one is concerned about if all assumptions hold or is something true. The notion of truth in mathematics is very different then in other contexts. In mathematics, truth has nothing to do with the semantic content of mathematical statements but only with formal structure encoded in some syntax. Two statements that are equivalent just have the same truth value in the sense that a statement A is true, whenever B is true and vice versa. It does not tell anything about the semantic relationship between the two. In principle, one could do mathematics just as well without using the words true or false.

Statistics was developing in science and at the start it was very important to have a belief that your model is true in some sense or that you are using it appropriately. Over the years, those who asked the questions regarding the validity of assumptions, came to the realisation that is very hard or impossible to check if your assumptions hold. Most of the time one is not sure but just assumes that the assumptions on which the model is based hold. That is the minority. Most uses of statistics present blind use of statistical methods and all such results are very hard to interpret or analyze.

The development of companies like Google, Apple, Microsoft and others for the most part did not serve the public domain well. Although the indexing of internet resources which made Google famos or the development of the user interfaces by Apple and Microsoft could be seen as a fantastic victory in markets, design, science and engineering, all those present a negative development with respect to the society. If there was no Apple or Microsoft, everyone would use superior command-line interfaces and the public domain would be more healthy. People would not stare for hours at the screen scrolling or pushing buttons, but they would just have an empty piece of paper and piece of mind.

The failure of the Linux kernel and its distributions comes from equating desktop users with point-and-click users, instead of assuming that every user is a command-line user. Similar failure is reflected in the talk by Mike Acton, one of the most famous game developers. At the C++ conference in 2014, Mike preached about the fundamental failure of most programmers to have in mind the basic distinction between the hardware and the software. At the same time, Mike fails to see a distinction between development and software development. That does not imply that Mike should solve some usefull problems of public interest, but just that if everyone were Mike, then no one would be left to do such work. Mike is just a placeholder for a set of industries that write software. Like most people, Mike thinks that since he lives in a western democracy he is free to just eat donuts and play games. If he was born in a dictatorship, there is no doubt that Mike would stand up, with the same force as presented in the talk. Just as Mike lives in its game-dev bubble, Linus, Greg and others have

their safe kernel bubbles:

`"We do not deal with anything else and would not even think about anything else except that on the line hardware - kernel."`

All these inventions related to user interfaces with buttons, icons, will eventually be seen as a period of stagnation or negative development. The same is with AI. We do not even know how to interpret basic statistical models nor do we have a consensus about anything, but we somehow assume that we need to include more information and do things faster. At the end, all of the important questions in AI safety reduce just to the formal legal interpretation. That does not have to do anything with AI and it has everything to do with relations between humans in every sense.

For example, every cybersecurity question or AI-safety question is determined by formal legal interpretation. Cyberattacks like ransomware attacks are not a problem at all. The problem lies just in our interpretation of such attacks. It is the legal structure and every legal, financial and existential pressure that appear as a ransomware problem. But in a normal society, such transformations of bytes should not have any effect on a well-being of a human being. The fact that many transformations of bytes can lead to bad situations is a human problem of dealing with a situation and it has nothing to do with bytes.

What is the most important characteristic of internet? Redundancy. If the complete IT infrastructure were redundant, then computers would be secure. Thus, every problem in cybersecurity can be solved by creating a redundant equivalent outside computers. In order to have perfect security and useability it is necessary and sufficient to expose to the public internet just redundant public information. Everything else should not be on any machine. Even in the case of a leak of bytes, no leak of bytes should ever have any consequences on any human being. It is not an earthquake. It is just something meaningless. The problem is not the disclosure of privacy per se, but the whole problem is in the treatment of people by other humans and institutions, who instead of helping, choose to enforce something meaningless and irrelevant.

The public domain internet is an interesting idea and has deep connections to many other fields, but most notably it provides a model of an environment in which human cooperation is most natural, usefull and efficient. It is just a historical accident of the release of the Linux kernel under a GPL license that provides a basis for this paper. Without those two accidents and many more, it would be harder for ideas like these to emerge. Without GNU and Linux we could live in a parallel universe with very different rules, regulations and formal structure surrounding the use of machines and software.

As time passes more and more laws and regulations will be interpreted by software. If a program says X, a human always needs to be able to say NO. The same should hold if a program is a human. This is the most natural assumption which would imply that the most natural formal structure is non-binding or some sort of a week formal structure. One that is flexible. One which can easily be ignored. That is science. You have X and you just assume not X and see where it goes.