The "Who's Yo Daddy?" Problem
Or: Solvin' Concurren' Axcess Issues wit Computers, Yo

Dr. Donna M. Malayeri, PhD Heather Miller, PhD(c), Esq., PPPoE, P2P
Herr Doctor Doktor Klaus Haller

March 23, 2011

Abstract

Computer Science research can solve many real-world problems. Here we describe our novel research, Uniqueness Types, and how it applies both to multi-threaded programs and to real world scenarios. In particular, we solve issues that commonly arise in popular daytime sociological science documentaries.[1]

1 Introduction

In multi-threaded programs resources such as memory and files are at the same time accessed, i.e., concurrently. This often leads to problems, such as blue screens, not-disappearing hour glasses, the spinning dying color wheel of death and so forth. For example, let's consider a thread program that a file opens, and it accesses. It may happen that yet another threaded process at some later point in time closes the file without the first program knowing about it. If the first program to the file again accesses, then the user may a blue screen experience.

2 Uniqueness Types

To solve the unique problems introduced in the introduction, we introduce a novel language-theoretic type theory called Uniqueness Types. We base our theoretical theory on flow-sensitive linear logic with typestate [**?**]. The idea of our research approach is to assign unique types to pointer variables. A pointer is unique whenever the compiler program knows that all other pointers are pointing towards other datums, or, conversely, if and only if no other pointer is pointing towards it. In predated works computer researchers have published theoretical experiments with unique pointers that sometimes their typestates change.

Unique pointers with uniqueness types give rise to a unique approach to avoid the unique problems of hour glasses and spinning dying color wheel of death. Somehow we can make sure everyone points to the hour glass or something like that. Or no, maybe the thread program must have pointers with unique names.[2]

3 Real-World Scenarios

As interesting as these programming problems are, feel we that it time is to computer science to real world problems apply. How else can we, with a straight face, to funding agencies the claim make that we real problems that affect people's everyday lives solve?

We believe that a good source of real-world problems documentaries is, particularly those highly-rated ones that on broadcast television are shown. These informational programs provide an unprecedented view into the daily life of the everyman. For the purposes of this paper, we shall The Jerry Springer Show use as our primary source, though our solution is applicable to scenarios seen on other esteemed programs, such as The Maury Povich Show or Jersey Shore.

[1] E.g., The Jerry Springer show.

[2] Americanadian translation: if two threads access the same resource, say a file, it would be bad if one thread opened the file, then the other thread closed the file, then the first thread tried to then read from the file. Essentially, threads with pointers to a shared resource need to know about state changes that may affect future operations on that resource. A uniqueness type solves this problem, and you can read more about it in this boring—er, exciting—research paper [**?**].

3.1 Real-World Problem Statement

> So, this one bitch is a real ho fo real and she get wit three playaz, Playa 1, Playa II, and Playa Playaa Playa. And now showty pregnan' and she sez the baby daddy is Playa Playaa Playa and he a pimp.[3] He sho' Playa 1 is da baby daddy fo serious and he don wants to pay no child suppo'. Da ho gots a paternity test dat sez da baby daddy is Play Playaa Playa but he thank it a fake.

Here, the problem is access to the shared resource within a critical timeframe. Since the third man does not trust the results of the paternity test, believing the woman to be a "lyin' ho," we need to provide the parties in this scenario with a fool-proof mechanism for determining parentage.

We believe this problem is widespread, and in the tradition of clever monikers for computer science problems (e.g., Travelling Salesman, Sleeping Barber, Dining Philosophers), we name this problem "Who's Your Daddy?"

4 Applicability to Real-World Scenarios

It is not immediately obvious how the theoretical results of Uniqueness Types to Who's Your Daddy can be applied. We present here an iterative approach to a solution, starting with a naïve solution and refining it to handle all possible scenarios.

4.1 Solution 1: Hëävy M̈ëtäl Locking Device with Physical Key

The problem here is that several pointers the shared resource may access, and her state may at any time change, without it being obvious to any of the parties involved (including the resource herself) that a) the state change has occurred and b) which pointer caused the state change.

We introduce the following protocol:

- An external party outfits the resource with a hëävy m̈ëtäl locking device with a physical key. For instance, the Victorian chastity belt would be quite useful here.
- The physical key is retained by the external party for an incubation period of not less than 9 months.
- The first pointer retrieves the key from the external party and may access the resource.
- When the first pointer is finished with the resource, he or she returns the key to the external party.
- The incubation period is again started and the process repeats itself.

However, this naïve approach is fraught with issues. The pointer who has the key in his or her possession may initiate an ownership transfer, either wanted or unwanted (i.e., in cases of theft). Moreover, borrowing re-introduces the same race condition that the protocol had attempted to eliminate. Thus, anyone with a copy of the key to the hëävy m̈ëtäl locking device may be the future Baby Daddy, which again leaves the question unresolved: Who's Your Daddy?

4.2 Solution 2: Hëävy M̈ëtäl Locking Device with Biometric Key

This solution is similar to the first, except the pointer is the access key. However, this means that once a particular pointer has accessed the resource, she may henceforth never be accessed by any other pointer. This is problematic if the first pointer gets bored and leaves, or is otherwise destroyed.

4.3 Solution 3: Hëävy M̈ëtäl Locking Device with Dynamic Biometric Key

This is similar to Solution 2, except now the locking device can be re-keyed by the external party to any particular pointer, assuming that the resource is in the vacant state (which can always be determined once the incubation period has passed).

[3] This means he has lots of money.

4.3.1 Translation to Lay Speak

Now, dis shit gettin' futuristic and shit yo. Now, to get wit dis bitch, you gots to go to her mama and daddy and get permission and shit. If she ain't wit nobody and she ain't pregan' then they crimp yo junk in computer shit an make it so no other playa get in dem locked panties and shit. Na'i mean? That shit mothafuckin sucks, brace yoself foo' cuz dat shit mothafuckin stings an dey know it was you when shit happen.

5 Conclusions and Future Work

We have shown that research that problems solves in multi-threaded programs can also to serious real-world scenarios be applied. With a minor modification to existing technology (the metal locking device), have we demonstrated how to solve a wide array of problems seen in popular socio-scientific documentaries. In particular, with our solution, can we always definitively that key question answer, "Who's Your Daddy?"

Yo so this is the mothafuckin shit yo. They take the shit they do on computers and shit and fuckin make it so yo ho can't gets with no otha balla. (So she can't claim you're the baby daddy when really it's some other player)

6 Acknowledgements

Herr Doctor Doktor Klaus provided all of the middle-English-like sentences. The other authors thank him highly for this immense contribution, which would not be possible without a native German speaker. Ebonics translation gracefully provided by Frau Hezza (aka Hedaaaiiir).