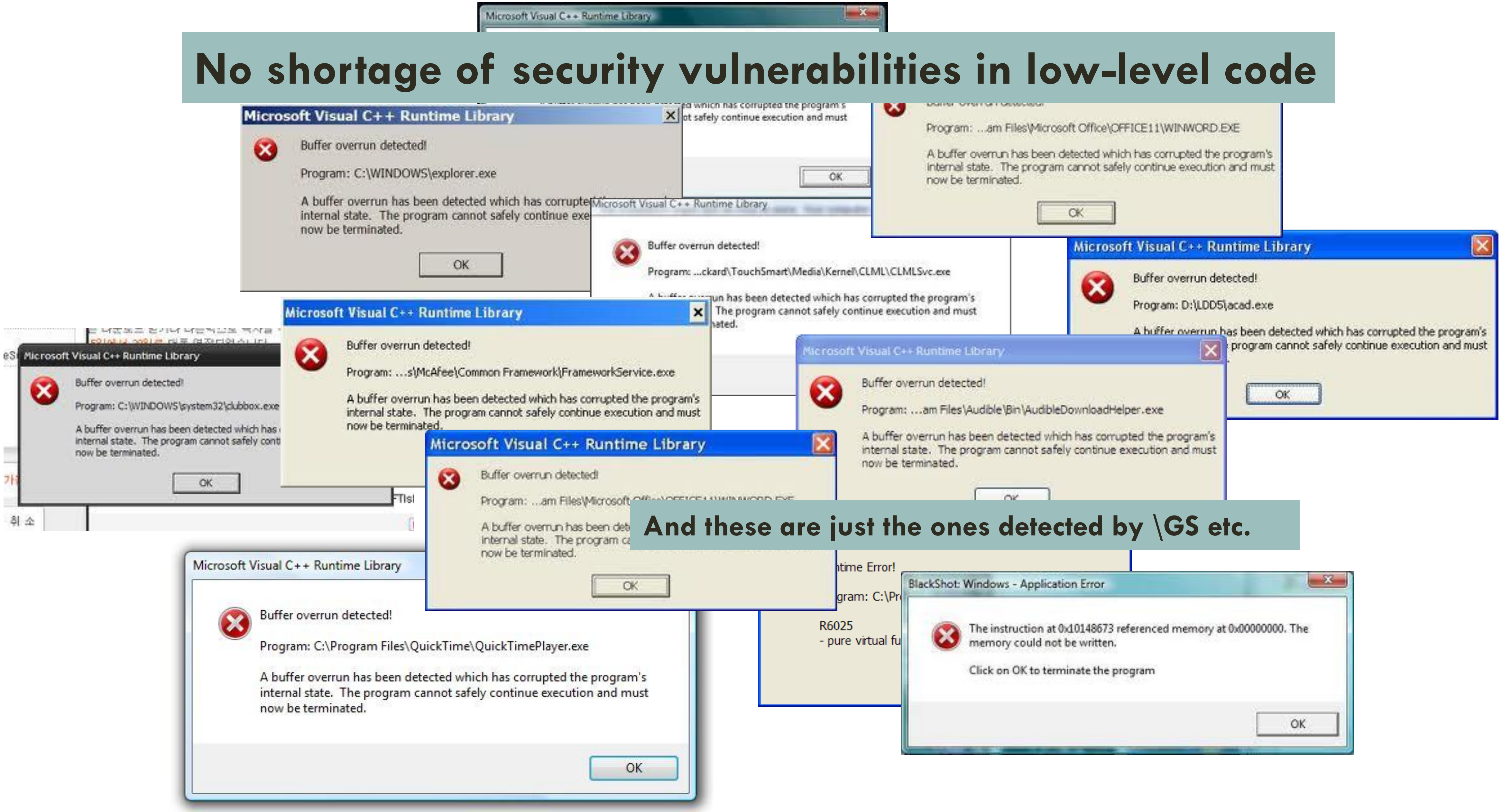
# PROVING PROGRAMS CORRECT WITH

Nikhil Swamy
Microsoft Research

RiSE Research in Software Engineering

Joint work with LOTS of people

# No shortage of security vulnerabilities in low-level code



Microsoft Visual C++ Runtime Library

...d which has corrupted the program's
...ot safely continue execution and must

OK

Program: ...am Files\Microsoft Office\OFFICE11\WINWORD.EXE

A buffer overrun has been detected which has corrupted the program's internal state. The program cannot safely continue execution and must now be terminated.

OK

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: C:\WINDOWS\explorer.exe

A buffer overrun has been detected which has corrupte... internal state. The program cannot safely continue exe... now be terminated.

OK

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: ...ckard\TouchSmart\Media\Kernel\CLML\CLMLSvc.exe

A buffer overrun has been detected which has corrupted the program's ... The program cannot safely continue execution and must ...ated.

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: D:\LDD5\acad.exe

A buffer overrun has been detected which has corrupted the program's ... program cannot safely continue execution and must

OK

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: ...s\McAfee\Common Framework\FrameworkService.exe

A buffer overrun has been detected which has corrupted the program's internal state. The program cannot safely continue execution and must now be terminated.

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: C:\WINDOWS\system32\clubbox.exe

A buffer overrun has been detected which has ... internal state. The program cannot safely cont... now be terminated.

OK

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: ...am Files\Audible\Bin\AudibleDownloadHelper.exe

A buffer overrun has been detected which has corrupted the program's internal state. The program cannot safely continue execution and must now be terminated.

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: ...am Files\Microsoft ...

A buffer overrun has been det... internal state. The program ca... now be terminated.

OK

**And these are just the ones detected by \GS etc.**

...time Error!

...gram: C:\Pr...

R6025
- pure virtual fu...

Microsoft Visual C++ Runtime Library

Buffer overrun detected!

Program: C:\Program Files\QuickTime\QuickTimePlayer.exe

A buffer overrun has been detected which has corrupted the program's internal state. The program cannot safely continue execution and must now be terminated.

OK

BlackShot: Windows - Application Error

The instruction at 0x10148673 referenced memory at 0x00000000. The memory could not be written.

Click on OK to terminate the program

OK

## Fix: Move to higher level languages …

No more buffer overruns … yay!

But, is your program secure? Depends …

Higher abstractions

➔ Improved productivity and fewer bugs

## CORRECT? SECURE?

Your high-speed trading software isn't blowing away billions!

- NASDAQ bugs (Aug 22, 2013), DOW Flash Crash (May 6, 2010), …

Your SSH/TLS library is heartbleed-free, but is it secure?

- TLS renegotiation ➔ man in the middle; No NSA backdoors?

Your national health insurance market place does not crash!

## OUR GOAL

TO BUILD AND DEPLOY SYSTEMS THAT ARE PROVABLY SECURE, END-TO-END

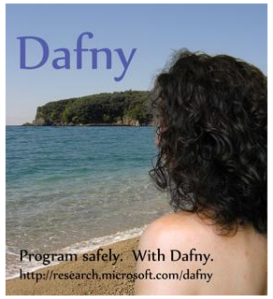
## AN END-TO-END PROGRAM VERIFICATION AGENDA

1. Precisely state application-specific correctness and security criteria

2. Use high-level programming language tools to implement software that can be formally verified to comply with its specification

3. Generate and deploy low-level code that is also proven to meet the same specification.

## Many research projects on program verification, for Pascal-like languages


Dafny
Program safely. With Dafny.
http://research.microsoft.com/dafny

frama C
Software Analyzers

Spec#

Chalice

Vcc

Boogie

VeriFast

Why3

Z3

Alt ERGO

Vampire

Simplify

CVC4

## But, modern languages are not like Pascal! (pervasively higher-order)

Lambdas everywhere!

Delegates, lambdas, LINQ, RX, …

```
delegate B Func<A,B>(A arg)
foreach (var i in L) {…}
```

JavaScript: AJAX, Event handlers, jQuery, DOM,…

```
Element.addEventListener(ev, function(node){…})
$('li').each(function(index) { .. })
```

## HIGHER-ORDER VERIFIERS ~ INTERACTIVE PROOF ASSISTANTS

Coq

Agda

NuPRL …

Very expressive logics! :-)

**Impoverished programming languages**

**Pure total functions only :-(**

## Enter F* …

http://research.microsoft.com/fstar

**An ML-like language designed for program verification**

Since around 2008, many people have worked on it:

Currently: **Bhargavan, Delignat-Lavaud, Fournet, Hritcu, Keller, Rastogi, Strub, Swamy**

Previously: Borgstrom, Chugh, Dagand, Fredrikson, Guha, Yang, Jeannin, Schlesinger, Weinberger

```
val f: x:int -> y:int{y > x}
let f x = x + 1
```

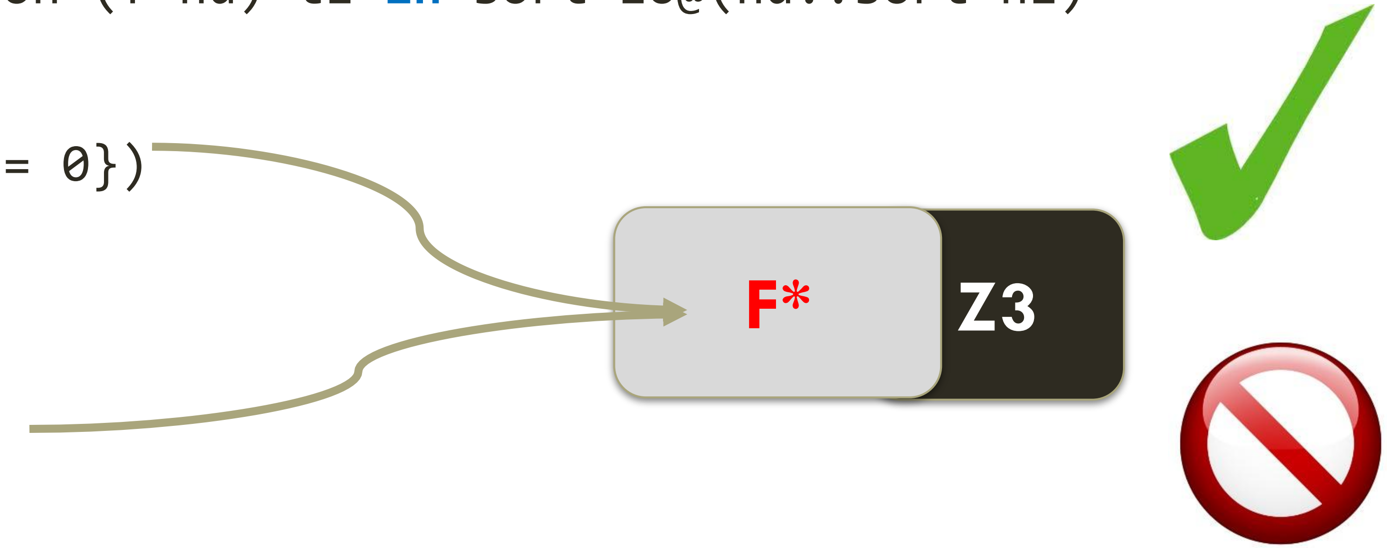
**Term syntax is core-ML, resembling F#/Caml-light**

```
val sort: f:(a -> a -> Tot bool){total_order a f}
      -> l:list a
      -> Tot (m:list a{sorted m /\ forall x. mem x m = mem x l})
let rec sort f = function
  | [] -> []
  | hd::tl -> let hi, lo = partition (f hd) tl in sort lo@(hd::sort hi)
```

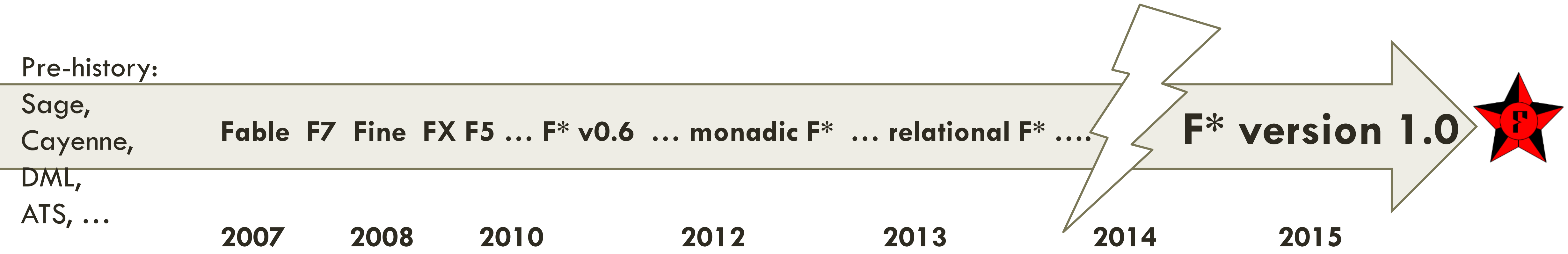
**Types allows expressing precise, functional-correctness properties**

```
val counter: unit -> ST (x:int{x >= 0})
let counter =
    let c = ref 0 in
    fun () -> c := !c + 1; !c
```

**Program with state and other effects**



## Brief history of an evolving line of languages …



**An outline of the remainder of this talk:**

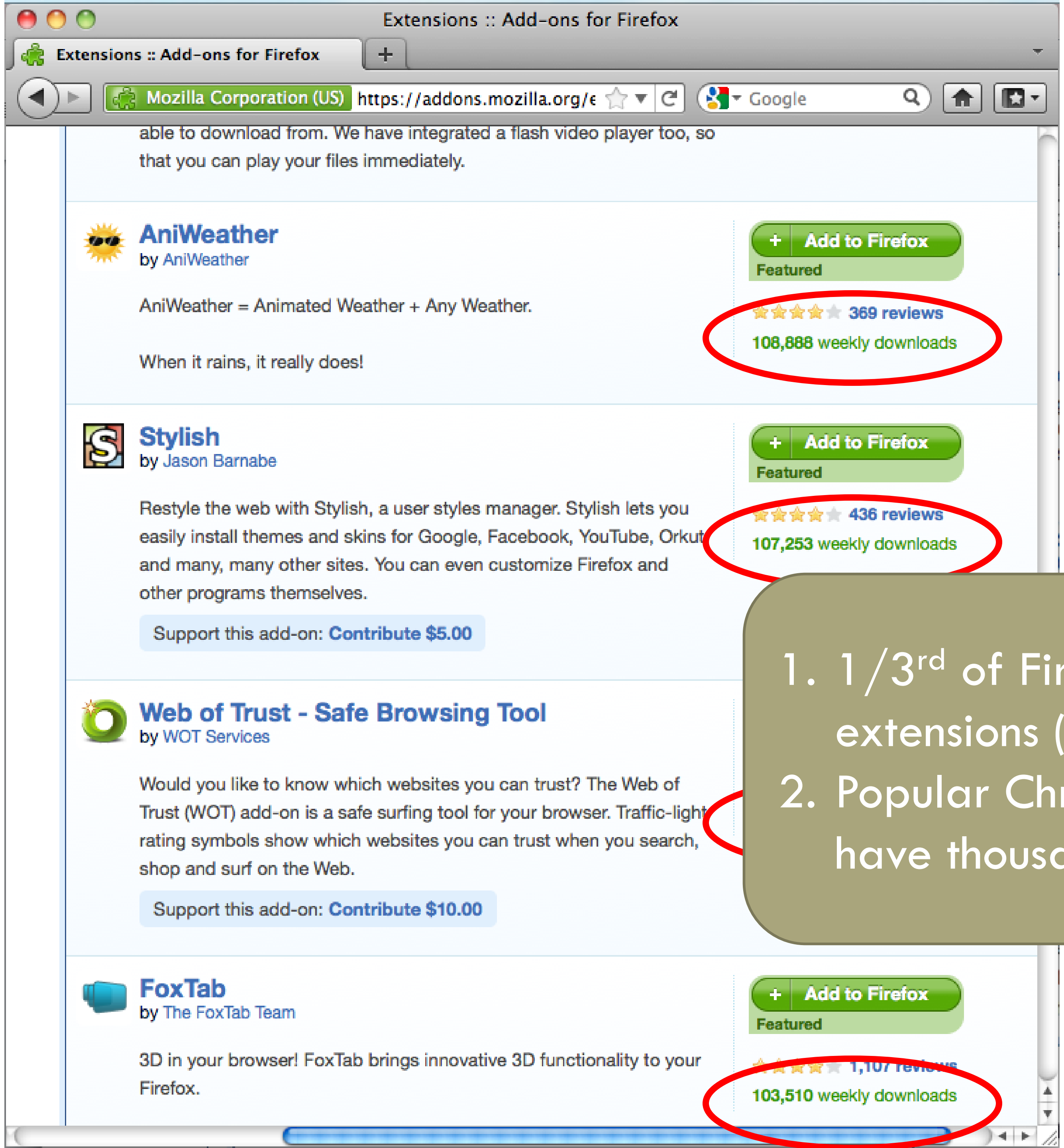
A quick introduction to refinement types, by example

A brief mention of some our past work

F* version 1.0: An outline of the concepts you will learn over the next 2 days

## Web-browser Security

(IEEE S&P (Oakland) 2011)


Extensions :: Add-ons for Firefox

Extensions :: Add-ons for Firefox

Mozilla Corporation (US) https://addons.mozilla.org/e

able to download from. We have integrated a flash video player too, so that you can play your files immediately.

AniWeather
by AniWeather

AniWeather = Animated Weather + Any Weather.

When it rains, it really does!

+ Add to Firefox
Featured
369 reviews
108,888 weekly downloads

Stylish
by Jason Barnabe

Restyle the web with Stylish, a user styles manager. Stylish lets you easily install themes and skins for Google, Facebook, YouTube, Orkut and many, many other sites. You can even customize Firefox and other programs themselves.

Support this add-on: Contribute $5.00

+ Add to Firefox
Featured
436 reviews
107,253 weekly downloads

Web of Trust - Safe Browsing Tool
by WOT Services

Would you like to know which websites you can trust? The Web of Trust (WOT) add-on is a safe surfing tool for your browser. Traffic-light rating symbols show which websites you can trust when you search, shop and surf on the Web.

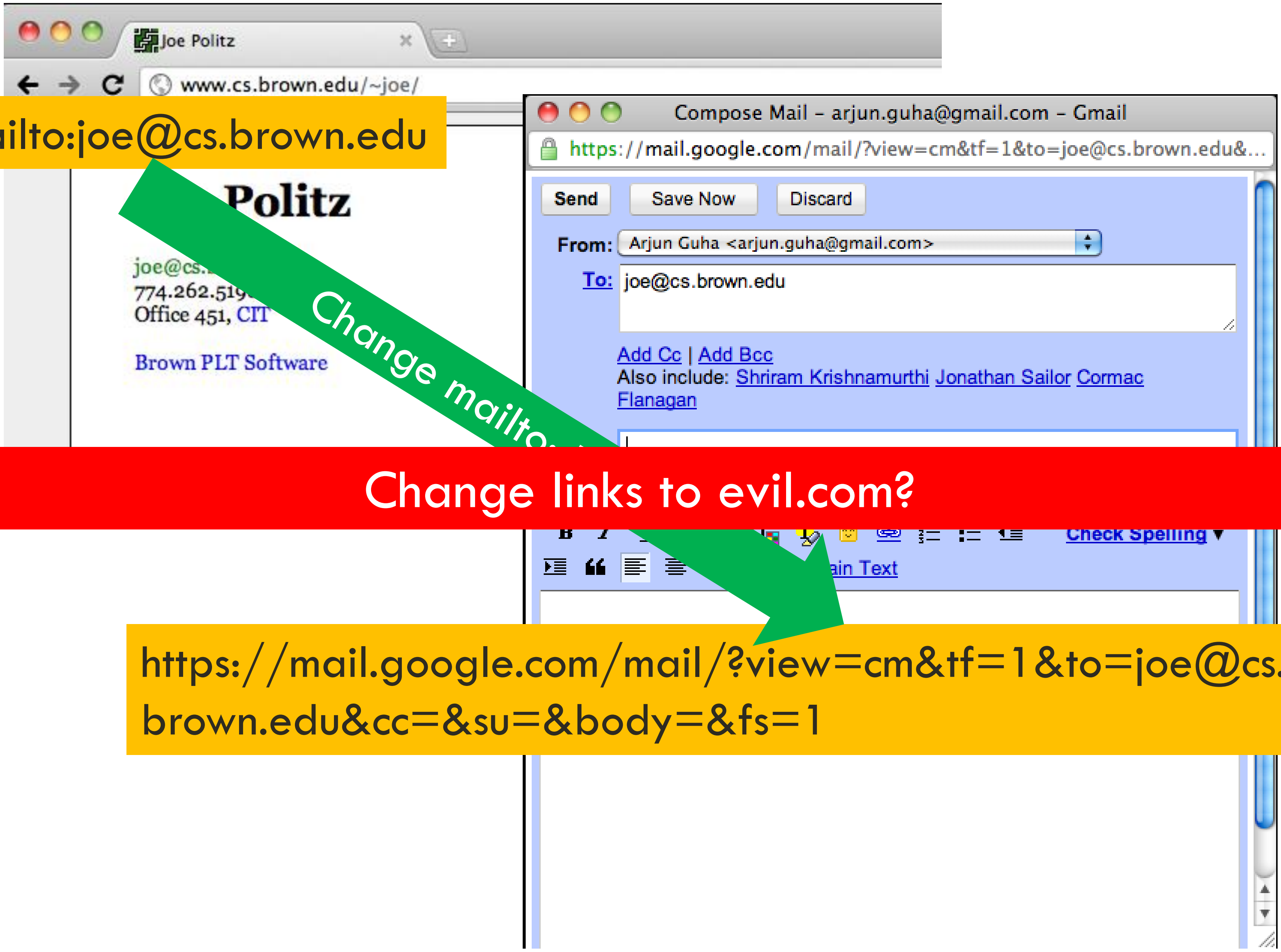
Support this add-on: Contribute $10.00

FoxTab
by The FoxTab Team

3D in your browser! FoxTab brings innovative 3D functionality to your Firefox.

+ Add to Firefox
Featured
1,107 reviews
103,510 weekly downloads

1. 1/3rd of Firefox users run extensions (~34 million users)
2. Popular Chrome extensions have thousands of users



Joe Politz

www.cs.brown.edu/~joe/

Politz

joe@cs.

774.262.519

Office 451, CIT

Brown PLT Software

Compose Mail – arjun.guha@gmail.com – Gmail

https://mail.google.com/mail/?view=cm&tf=1&to=joe@cs.brown.edu&...

Send | Save Now | Discard

From: Arjun Guha <arjun.guha@gmail.com>

To: joe@cs.brown.edu

Add Cc | Add Bcc

Also include: Shriram Krishnamurthi Jonathan Sailor Cormac Flanagan

Check Spelling

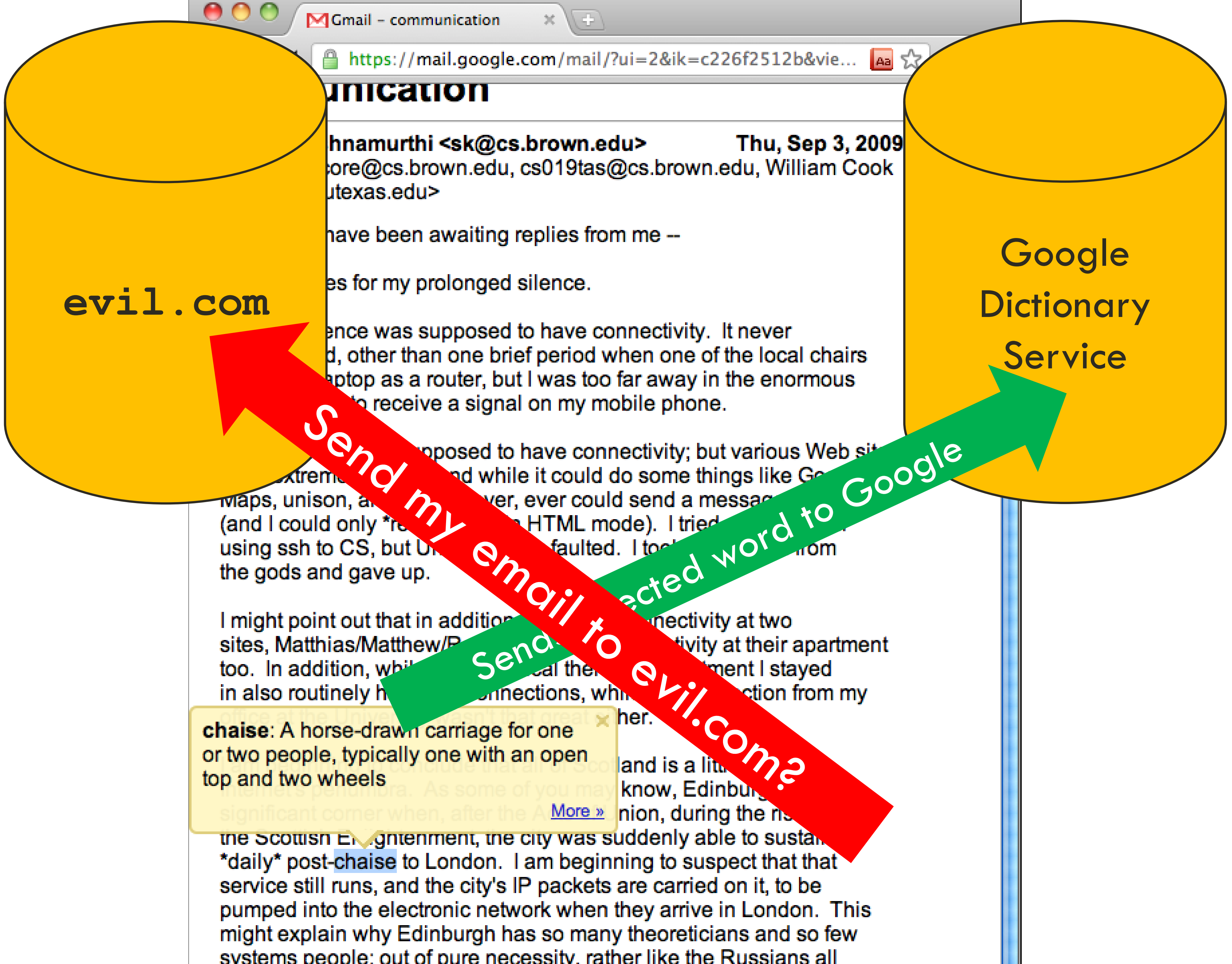
ain Text

mailto:joe@cs.brown.edu

Change mailto:

Change links to evil.com?

https://mail.google.com/mail/?view=cm&tf=1&to=joe@cs.brown.edu&cc=&su=&body=&fs=1


evil.com
Google Dictionary Service
Send my email to evil.com?
Send selected word to Google
Gmail – communication
https://mail.google.com/mail/?ui=2&ik=c226f2512b&vie...
More »

unication

hnamurthi <sk@cs.brown.edu> Thu, Sep 3, 2009
ore@cs.brown.edu, cs019tas@cs.brown.edu, William Cook
utexas.edu>

have been awaiting replies from me --

es for my prolonged silence.

ence was supposed to have connectivity. It never
d, other than one brief period when one of the local chairs
aptop as a router, but I was too far away in the enormous
o receive a signal on my mobile phone.

pposed to have connectivity; but various Web sit
xtrem nd while it could do some things like Go
Maps, unison, a ver, ever could send a messag
(and I could only *re HTML mode). I tried
using ssh to CS, but U faulted. I to rom
the gods and gave up.

I might point out that in additio nectivity at two
sites, Matthias/Matthew/R ivity at their apartment
too. In addition, whi al the ment I stayed
in also routinely h nections, wh ction from my
office at the Unive her.

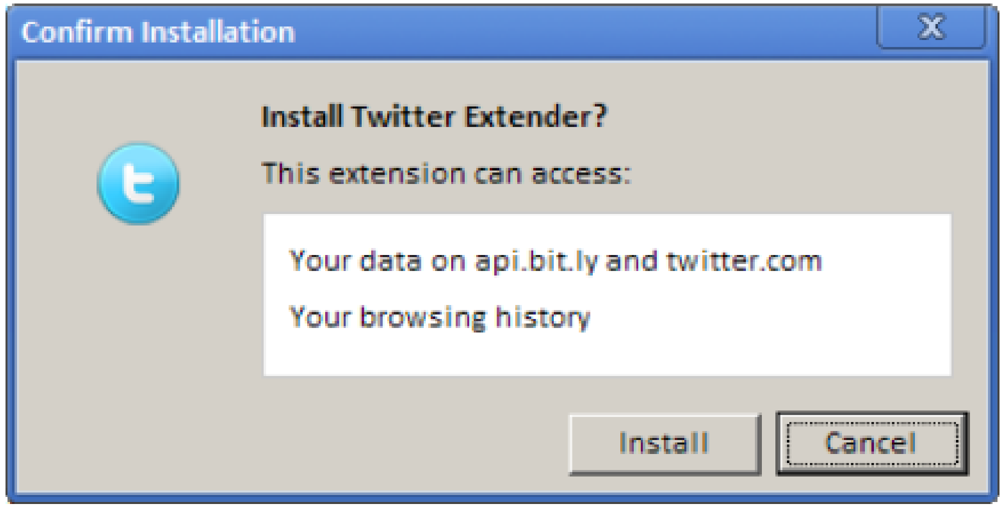
land is a lit
know, Edinbu
nion, during the ri
the Scottish E gntenment, the city was suddenly able to susta
*daily* post-chaise to London. I am beginning to suspect that that
service still runs, and the city's IP packets are carried on it, to be
pumped into the electronic network when they arrive in London. This
might explain why Edinburgh has so many theoreticians and so few
systems people: out of pure necessity, rather like the Russians all

**chaise**: A horse-drawn carriage for one or two people, typically one with an open top and two wheels

## ACCESS CONTROL IN CHROME

```
"permissions": [
  "tabs",
  "http://www.twitter.com/*",
```

1. Sensitive APIs



Confirm Installation

**Install Twitter Extender?**

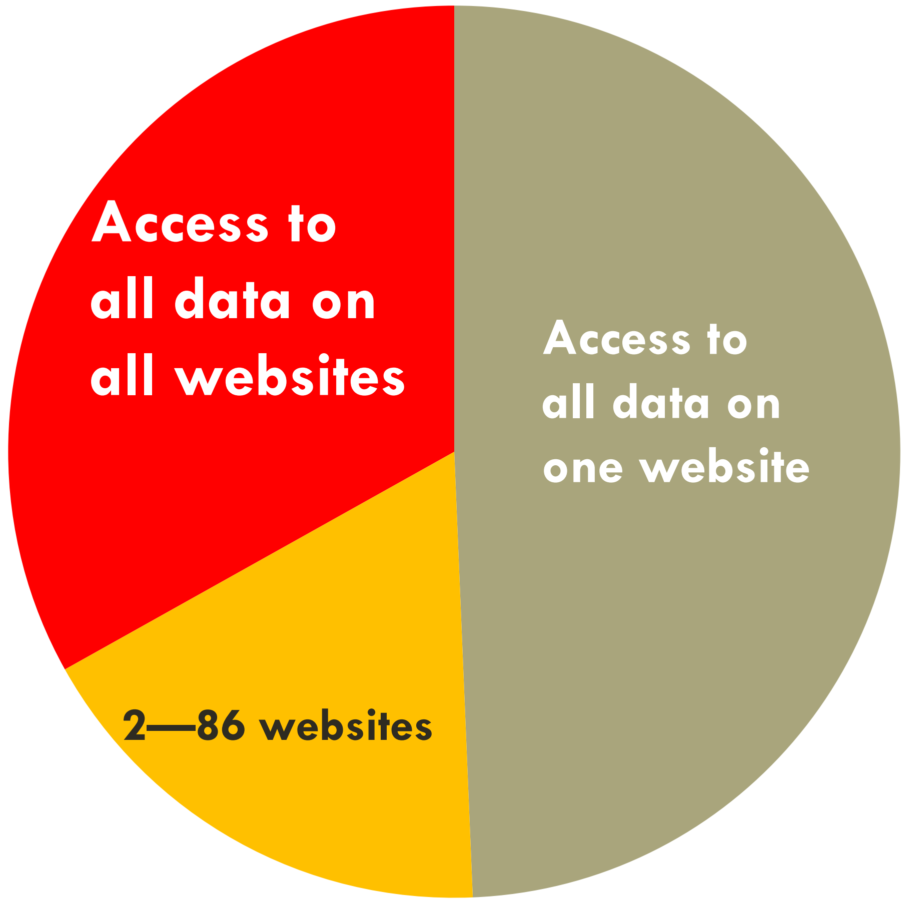
This extension can access:

Your data on api.bit.ly and twitter.com

Your browsing history

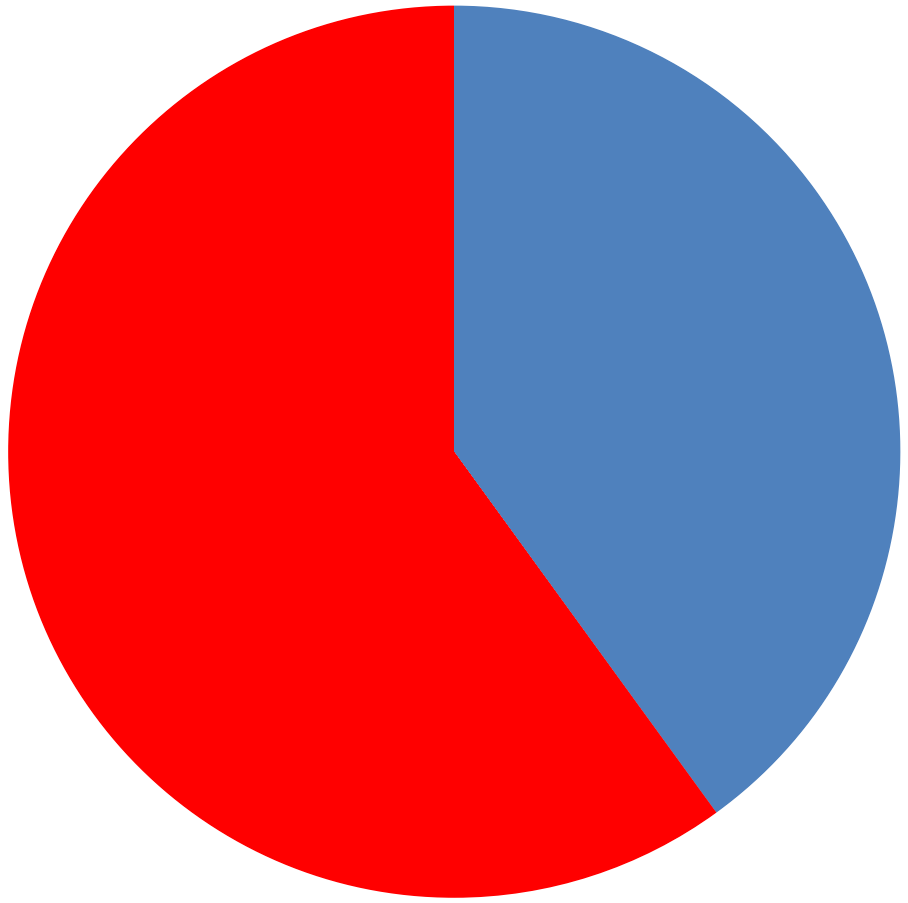
Extension runs on these URLs

## POLICY ANALYSIS
### ACCESSIBLE URLS

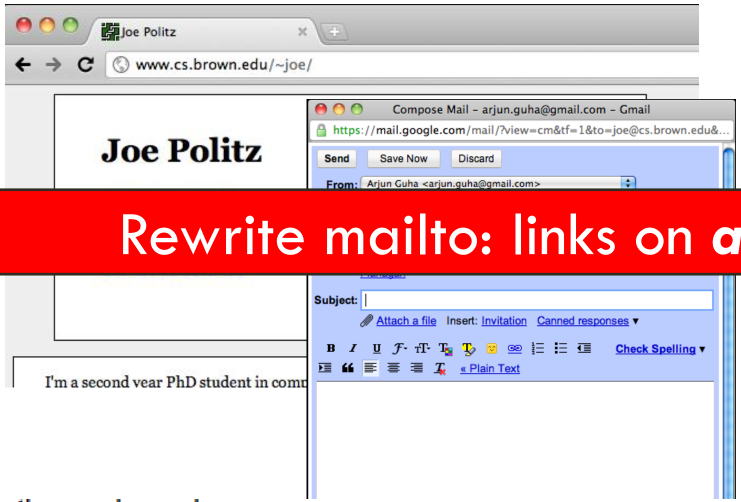


1,137 extensions policies

## POLICY ANALYSIS
### ACCESS TO HISTORY



1,137 extension policies

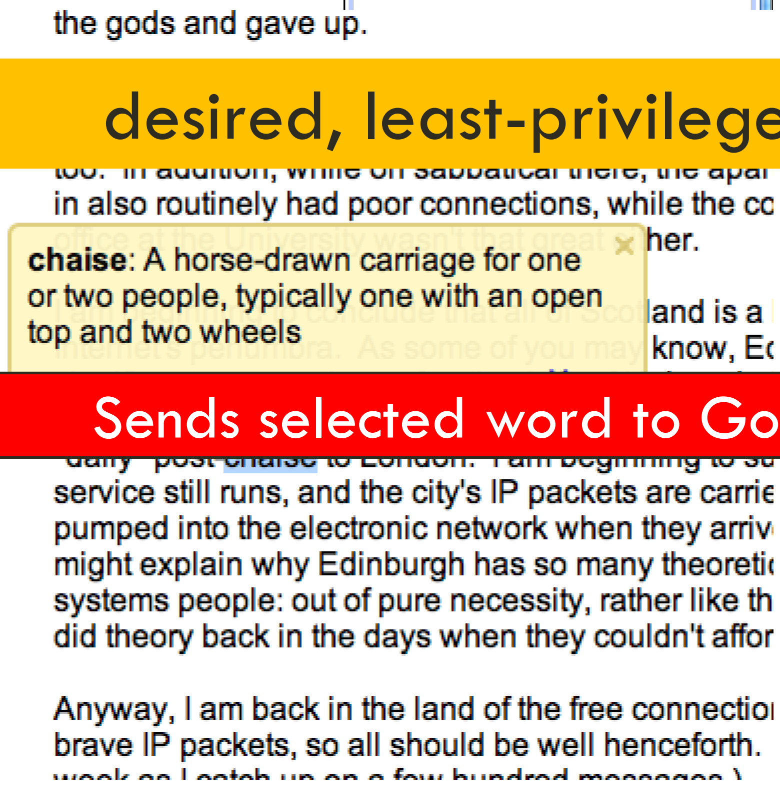

Joe Politz
www.cs.brown.edu/~joe/
Compose Mail – arjun.guha@gmail.com – Gmail
https://mail.google.com/mail/?view=cm&tf=1&to=joe@cs.brown.edu&...
Send
Save Now
Discard
From:
Arjun Guha <arjun.guha@gmail.com>
Subject:
Attach a file
Insert: Invitation
Canned responses ▾
Check Spelling ▾
« Plain Text

Joe Politz

I'm a second year PhD student in comp

Rewrite mailto: links on ***all sites***

```
"permissions": [
  "http://*/*"
]
```



the gods and gave up.

in also routinely had poor connections, while the co

her.

**chaise**: A horse-drawn carriage for one or two people, typically one with an open top and two wheels

land is a

know, Ec

service still runs, and the city's IP packets are carrie
pumped into the electronic network when they arriv
might explain why Edinburgh has so many theoreti
systems people: out of pure necessity, rather like th
did theory back in the days when they couldn't affor

Anyway, I am back in the land of the free connectio
brave IP packets, so all should be well henceforth.

desired, least-privilege security policy is ***inexpressible***

Sends selected word to Google from ***any website***

```
"permissions": [
  "http://*/*"
]
```

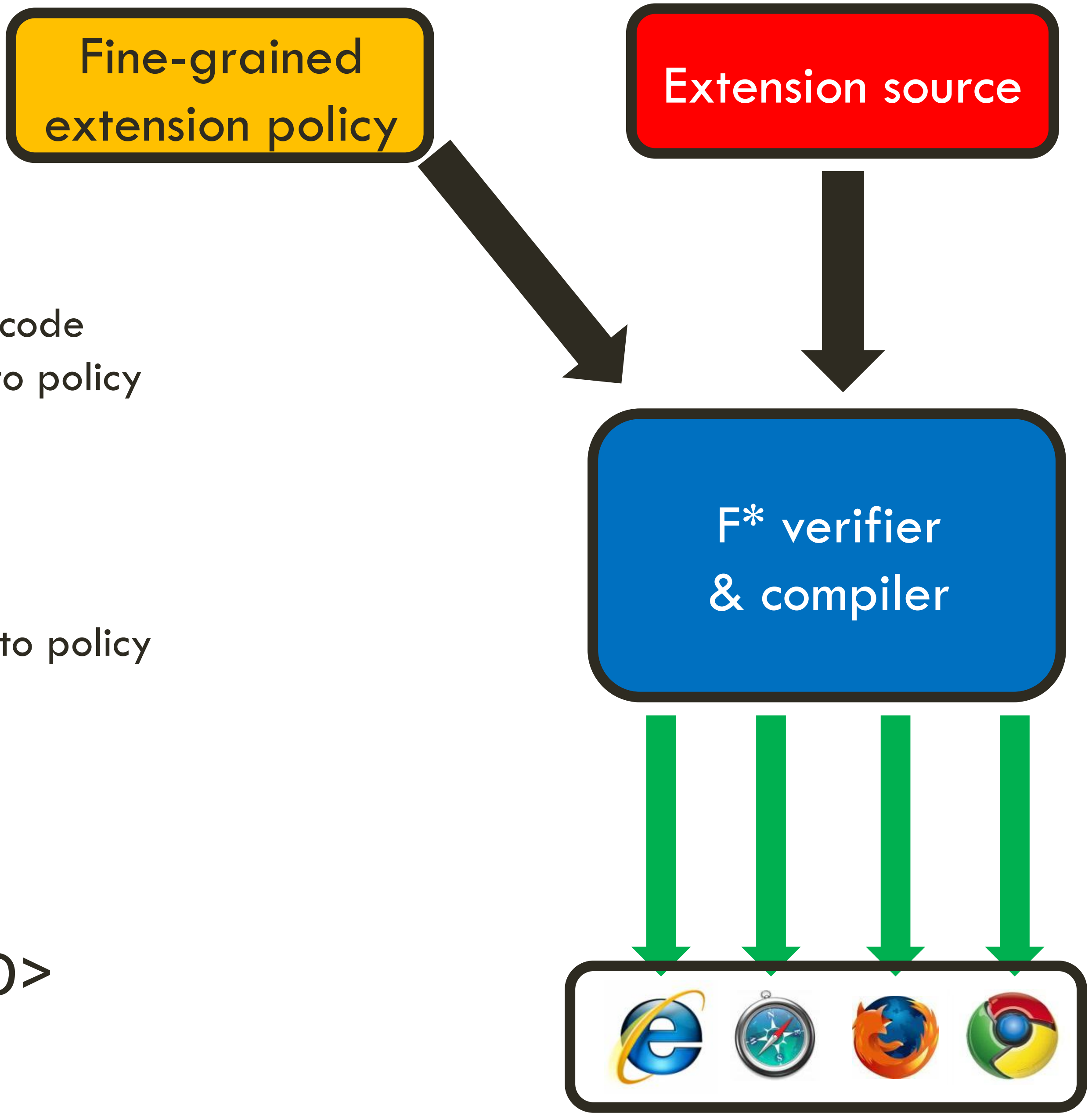
## Developers

- Write extension policy along with their code
- Use tools to ensure extension conforms to policy

## App store and users

- Uses tools to ensure extension conforms to policy
- Host and install approved extensions

## Example:
## Only Read text in <HEAD>



Secure DOM API

```
type elt

assume val getInnerText :
      e:elt { canRead e }
   -> string

assume val tagName :
      e:elt
   -> string
```

Native DOM elements, abstract to F*, API implemented by browser

**A refinement type:**
Only those elts e for which canRead e = true

F* checks pre- and post-conditions ***statically.***
***No need for manual code audit; only policy review***

Policy

```
      tagName e = "head" || hasAttribute e "public" || …
```

ONLY READ TEXT IN <HEAD>, OR NODES TAGGED "PUBLIC"

Code

```
let safeRead e =
  if canRead e = "head"
  then getInnerText e
  else "not allowed"
```

UNTRUSTED CLIENT CODE:
VERIFIED BY F* TO MAKE SURE THAT DOM API FUNCTIONS ARE NEVER ACCESSED EXCEPT AS ALLOWED BY THE POLICY

## Some more examples of refinement types:

```
val factorial: x:int{x >= 0} -> y:int{y >= 1}


val append: l1:list 'a -> l2:list 'a -> l3:list 'a {length l3 = length l1 + length l2}



val mac:    k:key -> t:text{key_property k t} -> tag
val verify: k:key -> t:text -> m:tag -> b:bool{ b ==> key_property k t}

…
```

| Extension Name | Access control using refinement types |
|---|---|
| Gmail checker | Rewrites “mailto:” links to open Gmail compose page |
| Dictionary lookup | Queries online dictionary with selection; displays definition in a popup |
| PrintNewYorker | Rewrites internal links to go directly to print view |
| Bookmarking | Sends selection to delicious.com |
| Google Reader client | Sends RSS feed links to Google Reader |
| Facebook miner | Sends friends’ Web addresses to delicious.com |
| JavaScript toolbox | Edits selected text |
| Password manager | Stores and retrieves passwords on each page |
| Magnify under mouse | Modifies the CSS on the page |
| Short URL expander | Sends URLs to longurlplease.com |
| Typography | Modifies <input> elements |

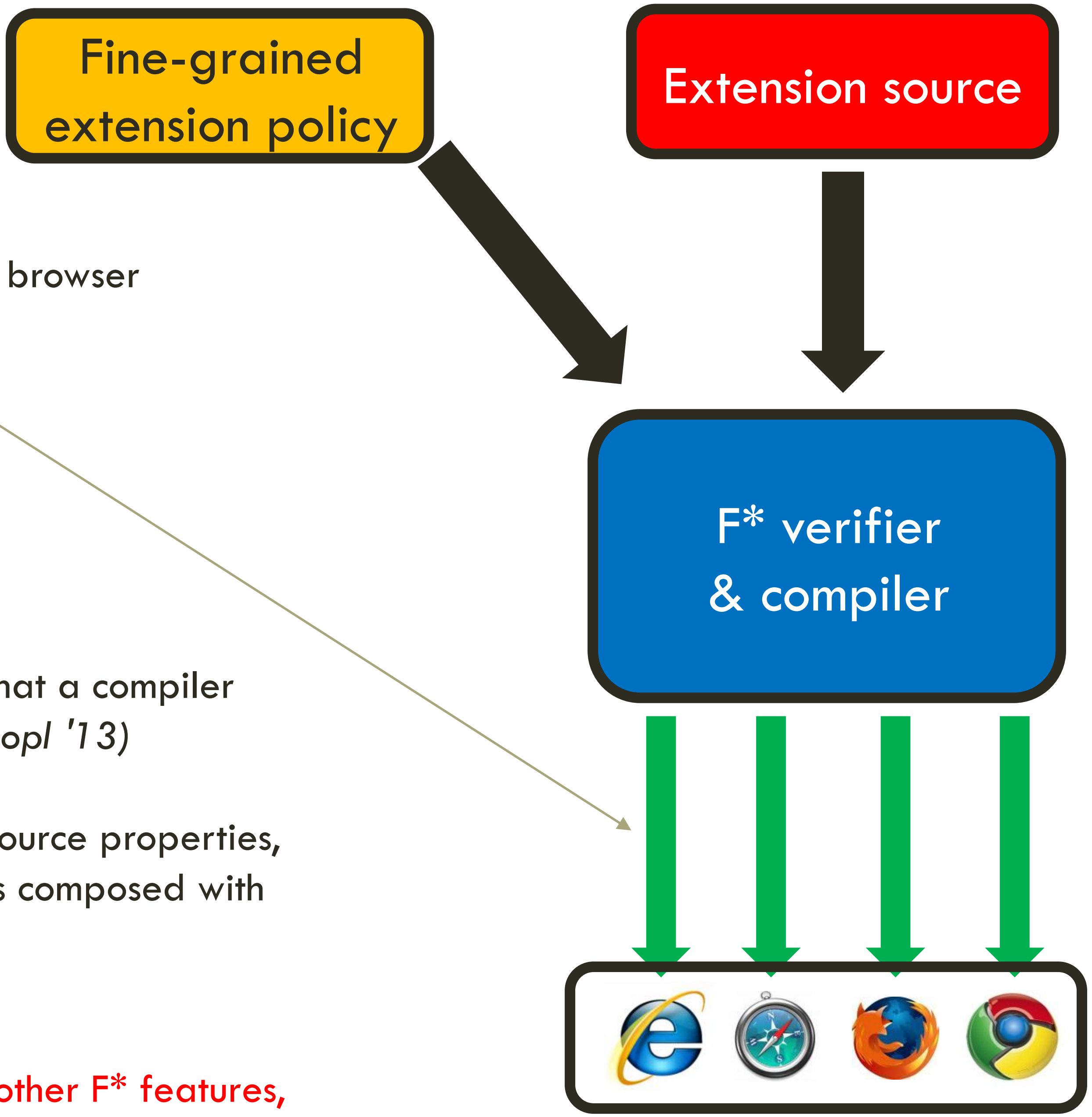
Write and verify F* code
**Compile it to JavaScript** and deploy in browser

But, how do you know that the code running in the browser behaves exactly like the verified source code?

So, we used F*'s type system to prove that a compiler from F* to JavaScript is *fully abstract (popl '13)*

- The compiler precisely captures all source properties, even when a compiled F* program is composed with arbitrary JavaScript

Refinement types, when combined with other F* features, can be used to prove highly non-trivial properties



## Refinement types, when combined with other F* features, can be used to prove highly non-trivial properties

- Security of an implementation of the TLS 1.2 standard (Cedric and Antoine, tomorrow)
- Self-certification: Proving the correctness of the F* type-checker itself using F*, and bootstrapping it in Coq (brief mention tomorrow)
- Proving the safety of an embedded, security-oriented sublanguage of TypeScript, a JavaScript dialect
- Probablistic relational Hoare logic, i.e., a logic similar to EasyCrypt's, encoded in F*'s type checker and used to prove several small crypto constructions
- …

## F* v1.0: Refinements and beyond

- A new version, based on a fresh code base
- Consolidating, then significantly improving, many of our prior efforts
- Written entirely in F* itself, bootstraps to multiple platforms
  - Caml done, almost. F# and JavaScript on the way!
- Why a new version?
  - Partly motivated by wanting to build a new, high-efficiency, certified implementation of TLS

## F* v1.0: Refinements and beyond

But, it's still under heavy development:

Completing and polishing the implementation:

- Code generations to multiple backends
- Error reporting
- Test, test, test! Then test some more.

And with more research:

- Formal certification of the implementation
- Formally certified proofs from an SMT solver

## F* v1.0: Refinements and beyond

A sampling of new features that you will see in the next couple of days …

## A logic including total, recursively defined higher-order functions

Here's what F* infers for the type of max:

A *total* function from two integers to an integer

```
val max : int -> int -> Tot int
let max i j = if i > j then i else j
```

Tot: this is an *effect* label, meaning that max is a total function.

Allows you to rely on *computation* to state and prove specifications

```
assert (map (fun x -> x + 1) [0;1;2] = [1;2;3])
```

## Extrinsic and intrinsic proofs

An intrinsic refinement of the ML type of reverse

```
val reverse: l:list 'a -> Tot (m:list 'a{length m = length l})
let reverse l = match l with
    | [] -> []
    | hd::tl -> reverse tl @[hd]
```

```
val reverse_involutive: l:list 'a -> Lemma (reverse (reverse l) = l)
```

An "after the fact" (aka extrinsic) proof about reverse

## Semantic proofs of program termination

```
val ackermann: m:int{m>=0} -> n:int{n>=0} -> Tot (a:int{a>=0})
let ackermann m n =
    if m=0 then n+1
    else if n=0 then ackermann (m – 1) 1
    else ackermann (m – 1) (ackermann m (n – 1))
```

## Other effects

A function that may read or write the heap, or diverge, when called, returning a stateful function itself

```
val counter: unit -> ST (unit -> ST int)
let counter () =
   let c = ST.alloc 0 in
   fun () -> c := c + 1; !c
```

ST: this is an *effect* label, meaning that counter may have state effects or diverge

Plus, a customizable lattice of user-defined effects

## Type inference with indexed effects / verification condition generation

```
val swap: x:ref 'a
       -> y:ref 'a
       -> ST unit
            (requires (fun h -> contains h x && contains h y))
            (ensures (fun hold _ hnew ->
                               hnew=(hold[x] <- hold[y])[y] <- hold[x]))
            (modifies {x,y})
let swap x y = let tmp = x in x := !y; y := tmp
```

## Plan for today

- Tutorial 1: F* basics.
  - Simple stateless access control
  - Functions on integers and basic refinement types and lemmas
  - Functions on lists and lemmas

- Tutorial 2: More F* basics
  - Proving termination
  - A full example: A verified implementation of quicksort

## Plan for tomorrow

- Lecture 1: Advanced F*, higher-kinds, state, and other effects
  - Stateful access control
  - Hiding local state
- Lecture 2: Attacks on TLS and a verified implementation in F7
- Tutorial 1: Type-based cryptography in F*
- Tutorial 2: Programming language metatheory in F*
  - Syntactic type soundness for the simply typed lambda calculus