

Proving OS Kernels in Coq (on-going)

Yu Guo

郭宇

2009-09-04

USTC-Yale Joint Research Center for High-Confidence Software

Outline



Background

Previous work

Related & Future work

Motivation



Joshua J. Bloch

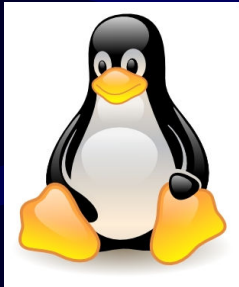
Nearly all binary searches
and merge-sorts are broken.
(June, 2006)

Motivation



Kernel : 3 million LOC

- Sep. 2009



Kernel : 10 million LOC

- Oct. 2008



Whole system :
Surpasses 50 million LOC

- 2003

What about
operating
systems ?



In Real World

- The threatens in real-world (China)
 - Virus, Trojan & Worms
 - 2008, infection rate: 85.5%
 - Buggy OS system (**54.63%**)
 - Windows NT family
 - Linux
 - Malicious code may take over the whole system
 - Even victims were equipped with weapons, e.g. anti-virus software, state-of-the-art secure firewall, and rigorous confidentiality protections



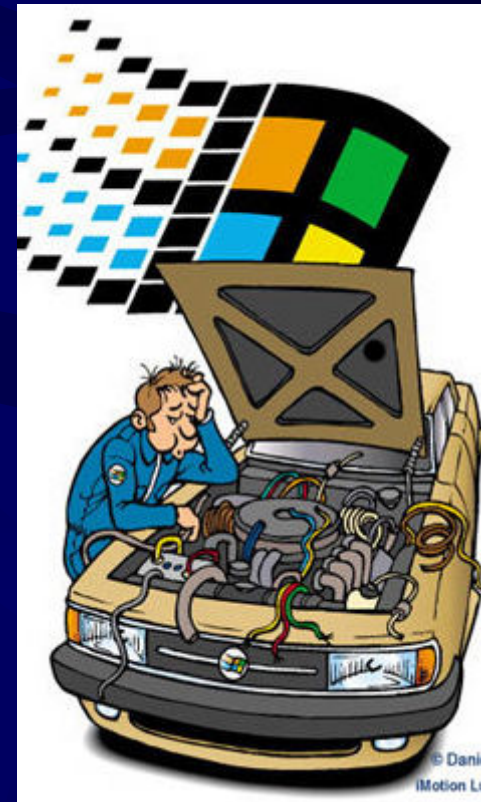
How can we trust the operating system kernels ?

You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

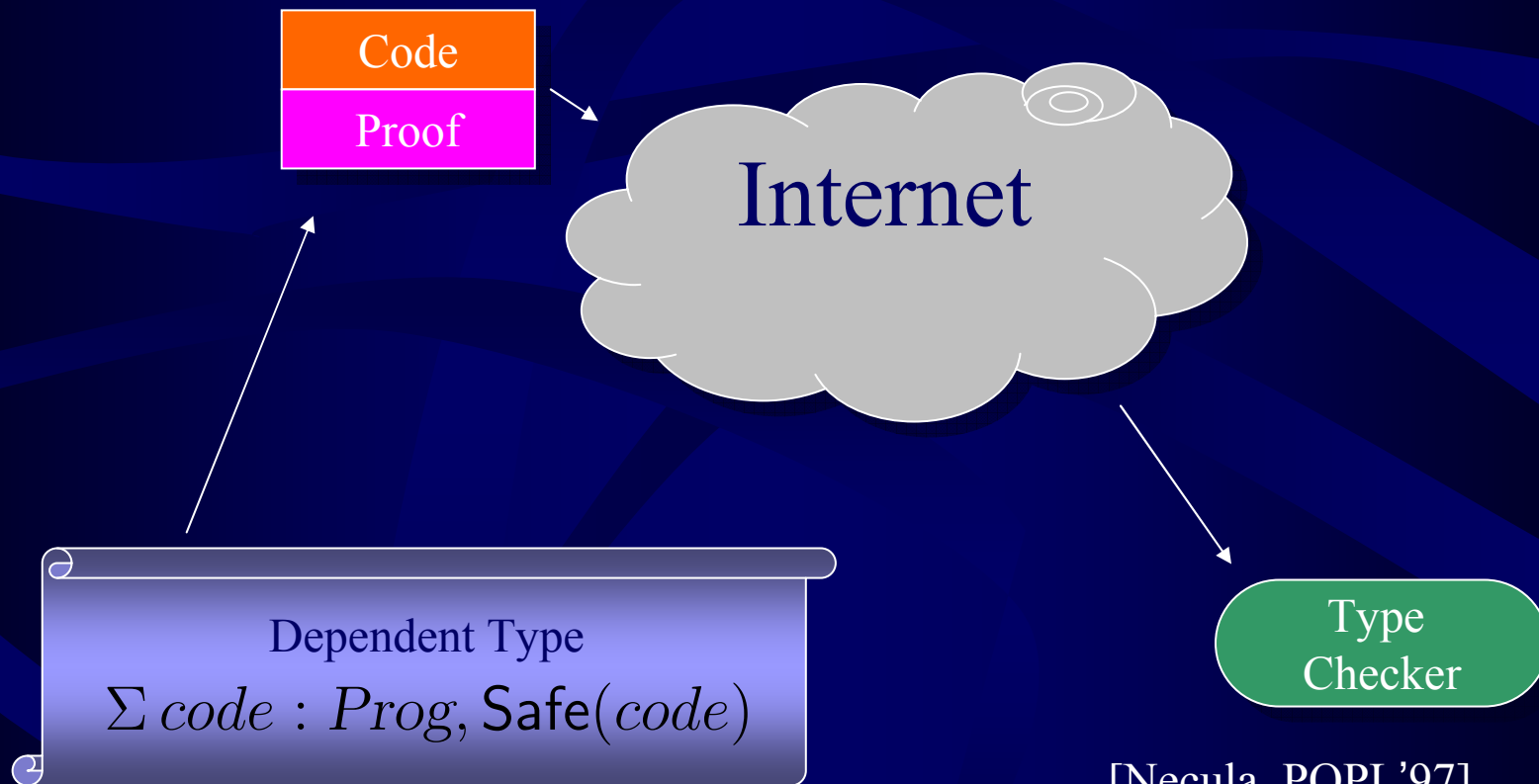
Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。



Proof-Carrying Code

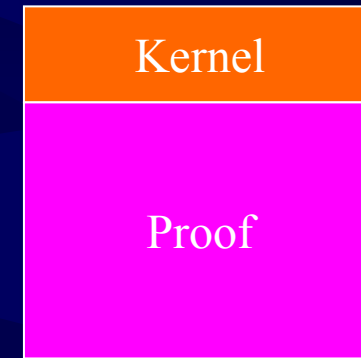


[Necula, POPL'97]

[Appel, LICS'01]

The big picture

Proof-carrying OS Kernel

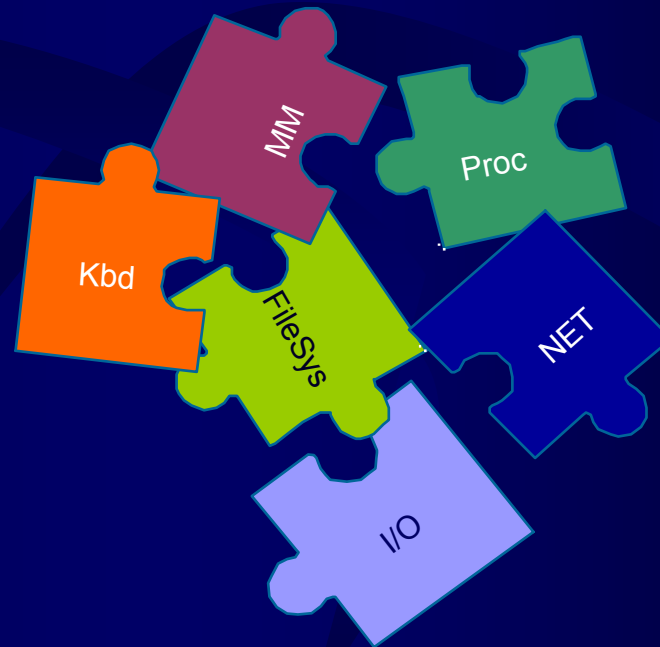


Hard to write code

More harder to specify and prove

The Big Challenge

How to verify the whole kernel with different features, which are always mixed up



Interview of Linus



Richard Morris

"What part of an Operating System do you think is the most difficult to write?"



Linus Torvalds

"That's actually an interesting question, just because my answer is that it's never any particular part. Yes, all the details tend to be complicated too, but **the real job is to make it all work together. ...**" (July, 2008)

Outline

Background



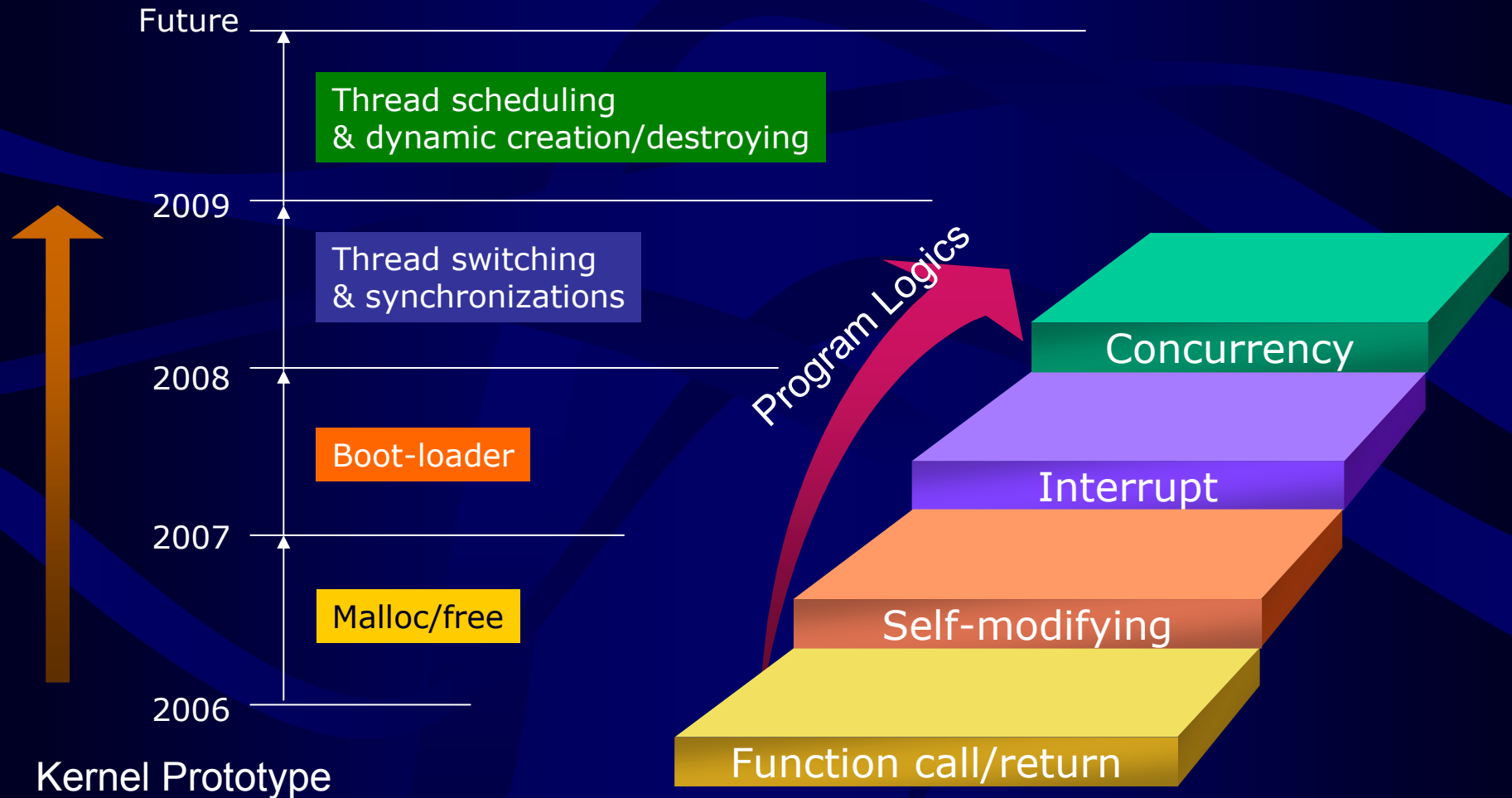
Previous work

Related & Future work

Previous Work: Two Lines

- Design logics
 - Reason about assembly code at system-level
- Certifying kernel prototype
 - Link kernel modules together

What we have done

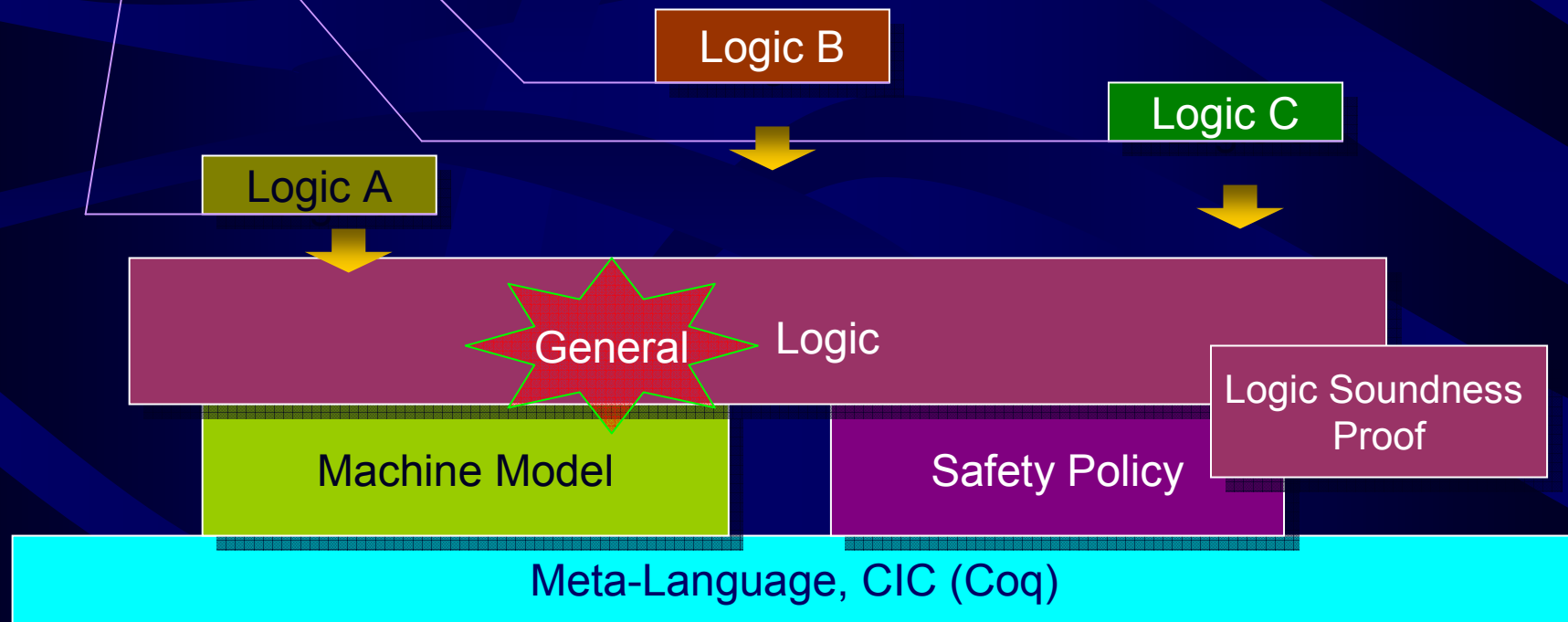


Domain-Specific Logics

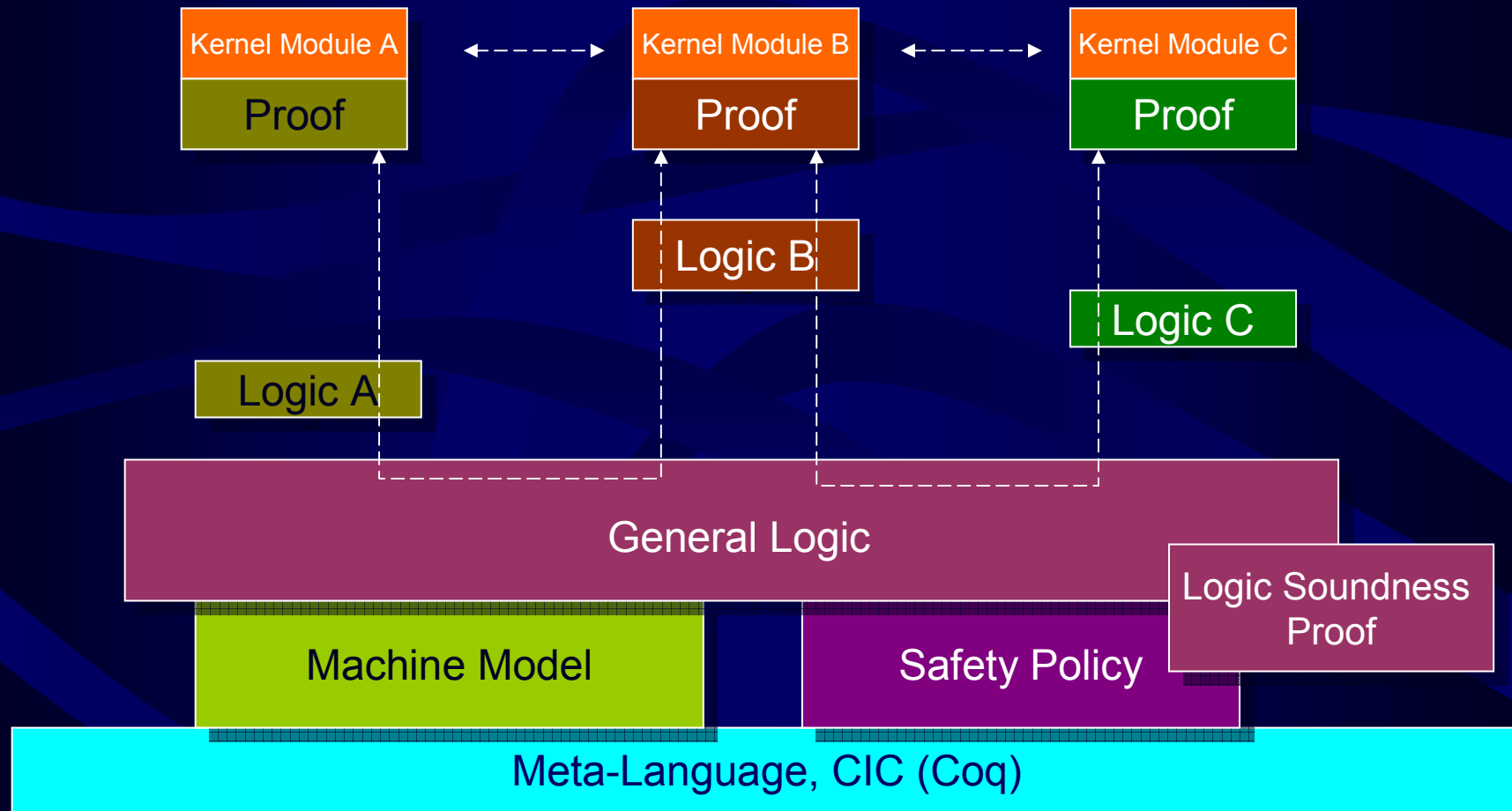
- Domain-specific logics for system code
 - Boot-loader
 - Thread scheduler
 - Interrupt handler
- OCAP, the general logic to glue them together

Our Certification Framework

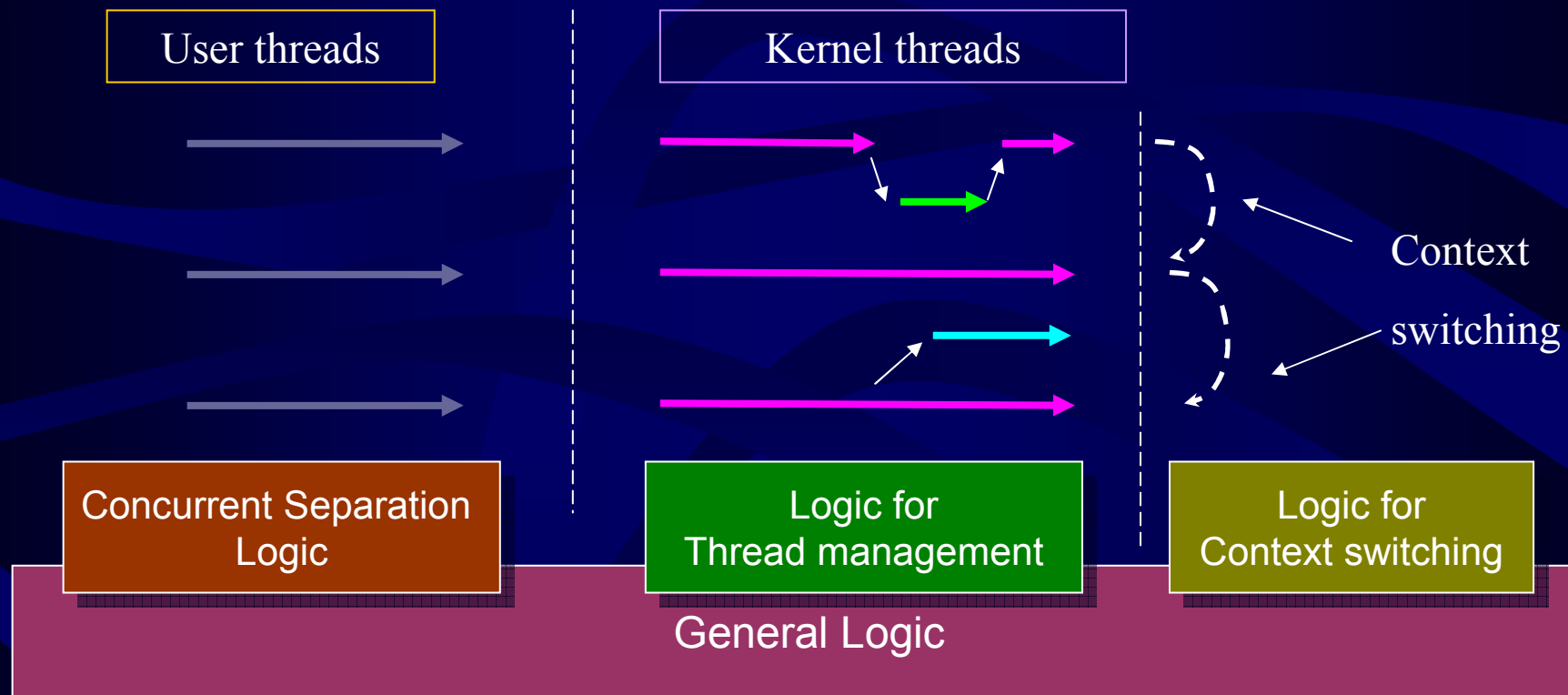
Different abstraction levels



Our Certification Framework



Kernel Prototype



Kernel Prototype

- 150 lines of assembly code
 - miniOS
- 82,000 lines of coq scripts
 - Machine model
 - Safety policy
 - Specifications
 - Logics & their soundness proof
 - Code Safety Proof

Example : Specifications

- Condition variable (wait)

```
Definition Ast_cond_wait_pre (lk cv bp sp pc : Label) (mP : MPred)
  (S : State) : Prop :=
  (stkSpacePrec sp _STK_SIZE_cond_wait
   *** (sp |->w pc)
   *** (sp + 2 |->w cv)
   *** (sp + 4 |->w lk)
   *** (Gamma lk)
   *** mP) (S_Mem S)
  /\ S_RF S BP = bp
  /\ S_RF S SP = sp
  /\ sp >= _STK_SIZE_cond_wait
  /\ In cv cvSet_M
  /\ In lk lockSet_H
  /\ S_Flag S_IF = true
  /\ S_ISR S = false.
```

```
Definition Ast_cond_wait_post := Ast_cond_wait_pre.
```

Example : Verification

- `<cond_wait+6> movw 4(%bp), %bx`

Lemma `wff_cmd_cond_wait6`:

`NH_WFcmd`

```
(SCAP_consCT (cond_wait + 1 + 1 + 1 + 1 + 1 + 1)
  (Ast p_cond_wait_6 g_cond_wait_6 1) psi_cond_wait)
(cond_wait + 1 + 1 + 1 + 1 + 1) (Ast p_cond_wait_5 g_cond_wait_5 1)
(comm1 (_movwld (A_reg 4 BP) BX)).
```

Proof.

.....

`apply enable_movwld.`

`intros S HS.`

`destruct S as [[[M R] fl] isr].`

`unfold p_cond_wait_5 in HS.`

`destruct HS as [lk [cv [bp [sp [pc HS]]]]].`

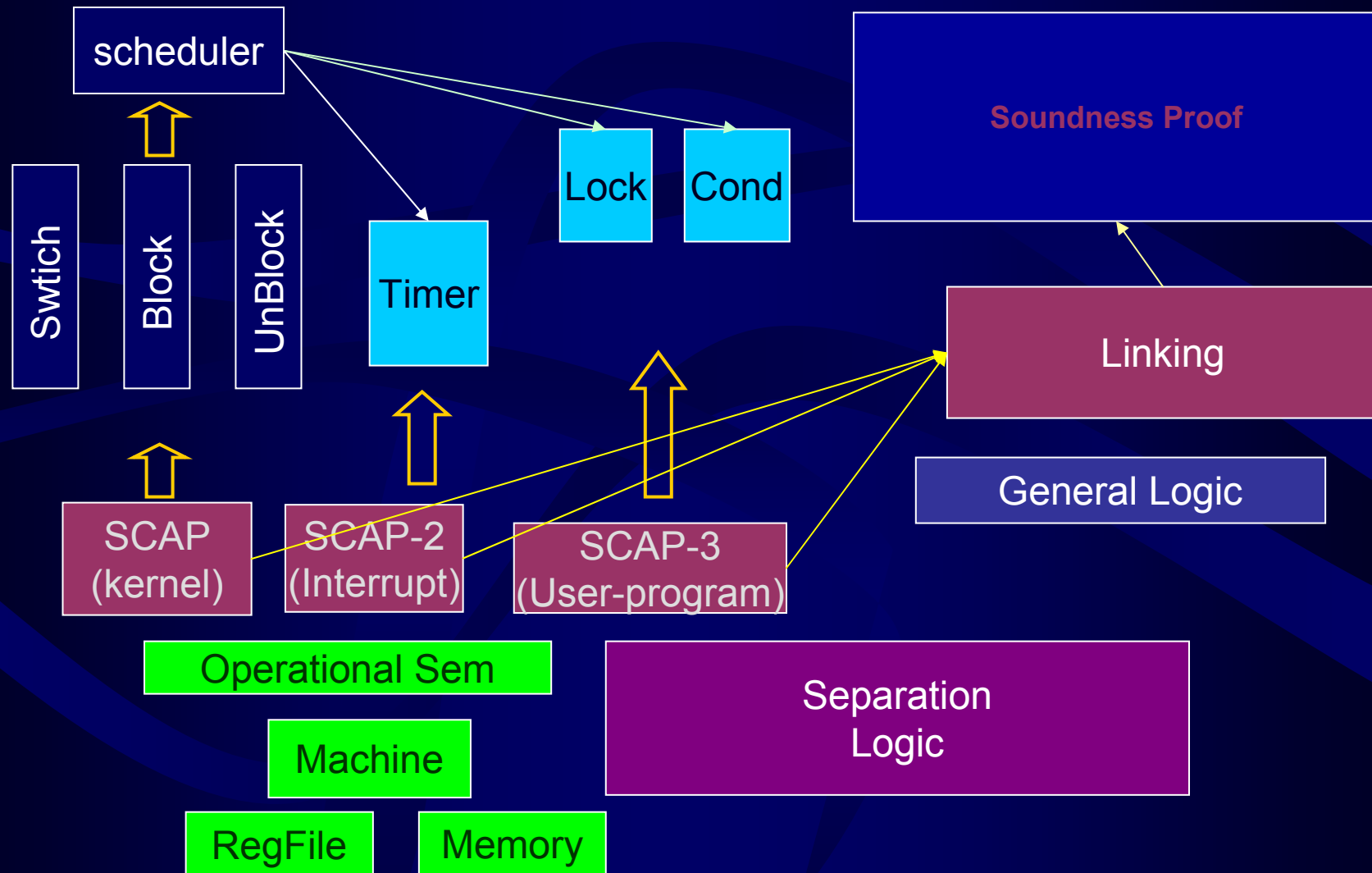
`unfold Ast_cond_wait_5 in HS.`

`destruct HS as [HM [HBP [HSP [Hsp [Hcv [Hlk [Hif Hisr]]]]]]].`

.....

`Qed.`

Coq Implementation



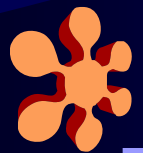
Discussion

- But,
 - At assembly level
 - Why not C ?
 - The logic (semantics) of kernel is independent with languages
 - The semantics of assembly language are simple
 - Some parts of kernel code have to be written in assembly lang.
 - Manual specifications and proving in hands
 - Why not automation tools ?
 - Higher-level code semantics VS. Low-level automation tools

Outline

Background

Previous work



Related & Future work

Related Work : seL4

- A verified realistic OS micro kernel
 - Prove at C level
 - Support virtual memory
 - Security properties
- However,
 - Some assembly code is trusted to be correct
 - Avoid complex control flows
 - Interrupts
 - Concurrency
 - Exceptions

Related Work : Verisoft

- Pervasive verification
 - Hardware, compiler, C0 programming language
 - Verify assembly code and C0 code at different level
- Not realistic
 - For specific embedded system
 - The verification framework cannot be applied to modern OS kernels

Future work

- Design more general logic system
 - Reason about program traces
 - Reason about higher-order code pointers
 - Better modularity
- Certify more realistic kernel prototype
 - Thread management (in progress)
 - Virtual memory management
 - Paging
 - Process abstraction
 - Temporal properties
 - Liveness, deadlock-freedom, fairness
 - I/O, Interaction
- Certifying compilers & Theorem prover (Zhaopeng Li)

The background is a dark blue color with several lighter blue, wavy, ribbon-like patterns that flow across the frame. On the right side, there is a small, light blue rectangular shape that looks like a piece of tape or a sticker.

Thanks !