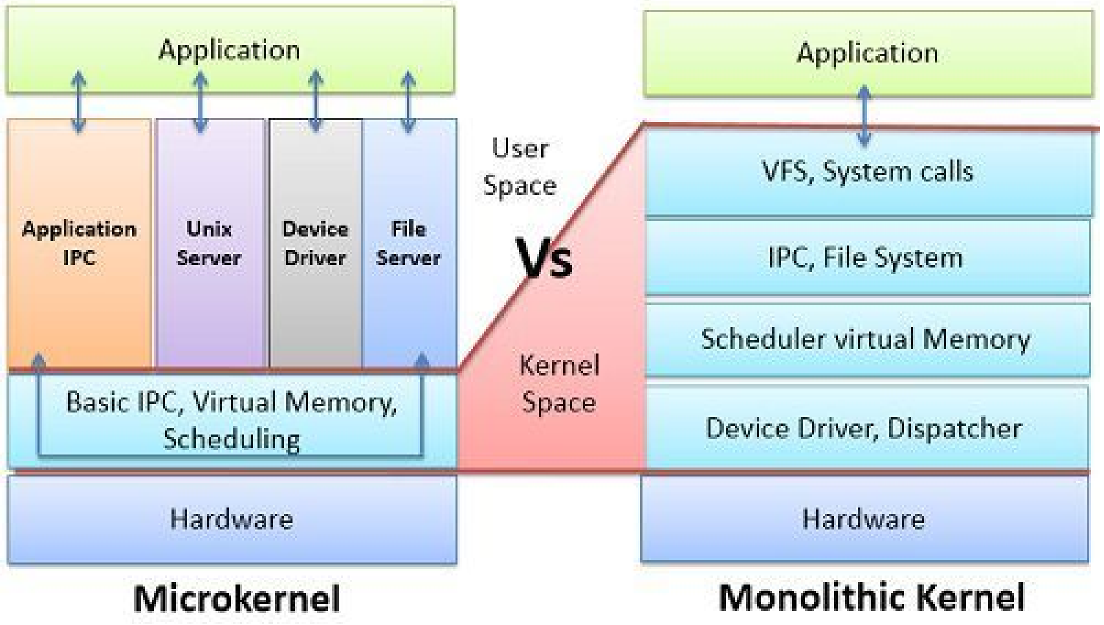
University of Bucharest
Faculty of Mathematics and Computer Science

SeL4 verification in Isabelle/HOL

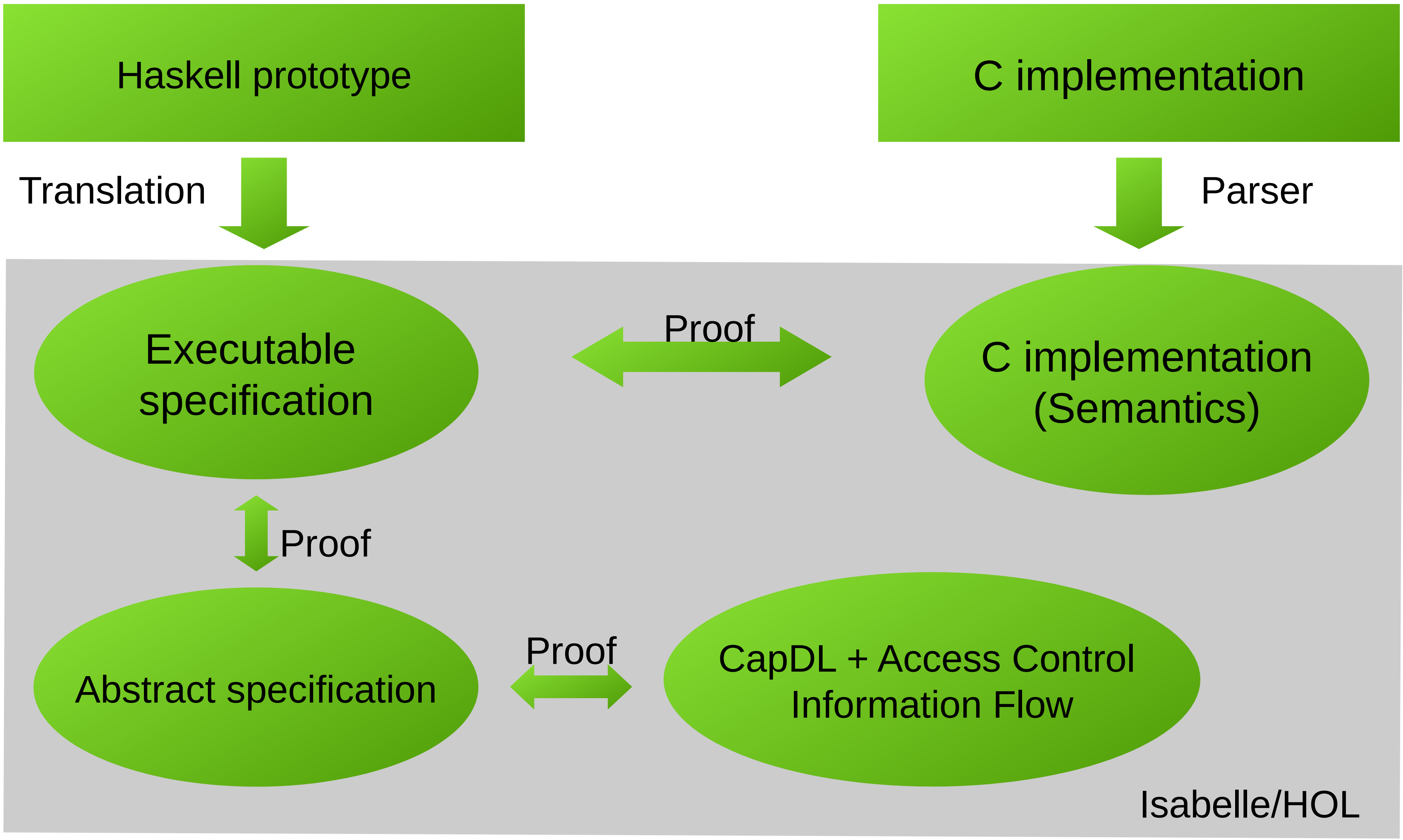
What is SeL4

- Operating system microkernel (vs monolithic kernels)
- Developed by National Information and Communications Technology Australia – start in 2009
- Short syscalls, interrupts are
- disabled while in kernel mode
- Capability-based access control
  - Capability = “fat pointer” immutable +
    - access rights
- Functional correctness proof



https://techdifferences.com/difference-between-microkernel-and-monolithic-kernel.html

How kernel verification is done



OS Verification projects

Project	Highest Level	Lowest Level	Specs	Proofs	Prover	Approach	Year
UCLA Secure Unix	security model	Pascal	90%	20%	XIVUS	Alphard	(?) - 1980
PSOS	application level	source code	17 layers	0%	SPECIAL	HDM	1973 - 1983
KIT	isolated tasks	assembly	100%	100%	Boyer Moore	interpreter equivalence	(?) - 1987
VFiasco/ Rodin	does not crash	C++	70%	0%	PVS	semantic compiler	2001 - 2008
EROS/ Coyotos	security model	BitC	security model	0%	ACL2 (?)	language based	2004 - (?)
Verisoft	application level	gate level	100%	75%	Isabelle	fully pervasive	2004 - (2008)
L4.verified	security model	C/assembly	100%	70%	Isabelle	performance, production code	2005 - (2008)

G. Klein - Operating System Verification - An Overview, NICTA

Sel4 based OSes

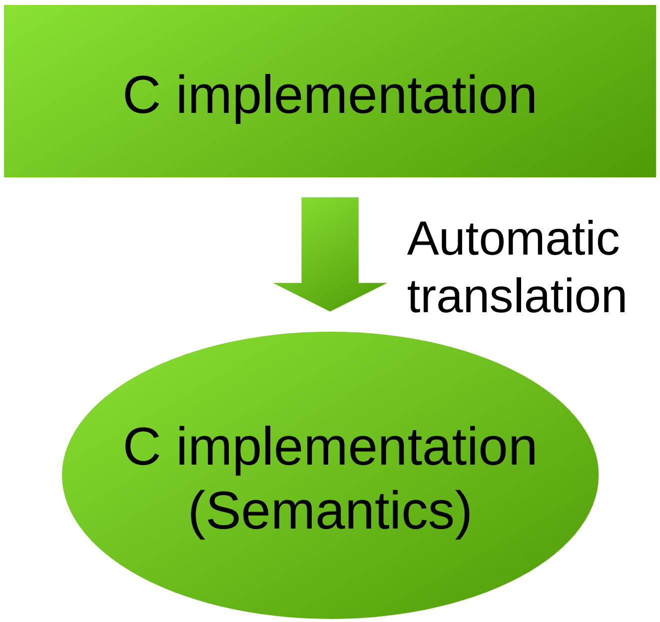
https://genode.org/documentation/articles/sel4_part_1

+ =

Robigalia OS
https://rbg.systems/

C-to-Isabelle Parser

- Implemented by Michael Norrish
- Translates C into its representation in Simpl (imperative programs in Isabelle/HOL)
- Only a subset of C language is used
- Proof[*] for a large number of programs
- that output matches the binary
- compiled with gcc



*T. Sewell, M. Myreen, G. Klein. *Translation Validation for a Verified OS Kernel*

C-to-Isabelle Parser

```c
int max(int a, int b) {
    if (a <= b)
        return b;
    return a;
}
```

$$\mathsf{max}\ a\ b \equiv \quad \mathsf{if}\ a \le b\ \mathsf{then}\ b\ \mathsf{else}\ a$$

```
TRY
    IF {|´a ≤s ´b|} THEN
        ´ret__int :== ´b;;
        ´global_exn_var :== Return;;
        THROW
    ELSE
        SKIP
    FI;;
    ´ret__int :== ´a;;
    ´global_exn_var :== Return;;
    THROW;;
    GUARD DontReach ∅
        SKIP
CATCH
    SKIP
END
```

D. Greenaway *Automated proof-producing abstraction of C code*

Deeply and shallowly embedded

2 + 2 = 4

Deep embedding – representation of program *structure:*

C-to-Isabelle parser

Shallow embedding – representation of program *semantics:*

AutoCorres tool

In order to prove two programs are equivalent, we need a *shallow embedded* representation.

AutoCorres generates that model through monadic representation bound to program state.

C subset and limitations

- Explicit guards against undefined behaviour (division by zero, dereferencing null pointer)
- Allowed only expressions without side effects
- Functions that return values may be called only as the right-hand side of an assignment
- No function pointers, *goto* or *switch* statements, unions or bit-fields also unsupported

```c
int i =0;
int a[2] = {0,0};
int f(void)
{
    i++;
    return i;
}

int main()
{
    a[i]=f();
    return a[0];
}
```

Octrng driver demo

```c
void
octrng_rnd(void)
{
  unsigned int value;

  rand_value = get_register(OCTRNG_ENTROPY_REG);
  add_task(octrng_rnd, 10);
}
```

A function from driver adaptation

```
octrng_rnd' ≡ do ret' <- get_register' 0;
                  modify (rand_value_''_update (λa. ret'));
                  add_task' (PTR(unit) (symbol_table ''octrng_rnd'')) 10
               od
```

The representation of function in Isabelle

Octrng driver demo

```
octrng_rnd' ≡ do ret' <- get_register' 0;
                 modify (rand_value_''_update (λa. ret'));
                 add_task' (PTR(unit) (symbol_table ''octrng_rnd'')) 10
              od
```

The representation of function in Isabelle

```
thm octrng_rnd'_def
lemma octrng_rnd:
 "⦃ λs. timer_'' s = a ∧ running_tasks_'' s < MAX_QUEUE ∧ current_task_'' s < MAX_QUEUE ∧
   control_addr_C (rng_regs_'' s) && OCTRNG_ENABLE_OUTPUT ≠ 0 ∧
   control_addr_C (rng_regs_'' s) && OCTRNG_ENABLE_ENTROPY ≠ 0 ⦄
  octrng_rnd'
  ⦃ λ_s. rand_value_'' s = a⦄!"
  unfolding octrng_rnd'_def
  unfolding get_register'_def add_task'_def
  unfolding get_time'_def
  unfolding MAX_QUEUE_def
  unfolding OCTRNG_ENABLE_OUTPUT_def OCTRNG_ENABLE_ENTROPY_def
  apply (wp; auto)+
  done
```

Proof of lemma stating that
octrng_rnd function returns current time

Bibliography

- D. Greenaway, "*Automated proof-producing abstraction of C code*", Ph.D. Dissertation, School of Computer Science and Engineering, University of NSW, Sydney, Australia, (2014);
- G. Heiser, "*The seL4 Microkernel – An Introduction*", White paper, The seL4 Foundation, rev 1.2 from 2020-06-10;
- H. Tuch, G. Klein, M. Norrish, "*Types, Bytes, and Separation Logic*" in Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Nice, France, 97–108, (2007);
- D. Greenaway, J. Andronick, K. Gerwin, "*Bridging the Gap: Automatic Verified Abstraction of C*" (2012);
- G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewel, R. Kolanski, G. Heiser, NICTA and UNSW, "*Comprehensive Formal Verification of an OS Microkernel*", Sydney, Australia, ACM Transactions on Computer Systems, Vol. 32, No. 1, Art. 2, (2014).