

Introduction to the seL4 proofs



Matthew Brecknell, Data61

@mbrcknl <https://m.brck.nl>

To ask questions, please use the discussion board on the virtual hub page for the talk.



@mbrcknl <https://m.brck.nl>

About you:

About you:

- ✓ Involved in the seL4 ecosystem

About you:

- ✓ Involved in the seL4 ecosystem
- ✓ Value that seL4 is formally verified

(But why?)

About you:

- ✓ Involved in the seL4 ecosystem
- ✓ Value that seL4 is formally verified

(But why?)

? Maybe read a bit about the verification

About you:

- ✓ Involved in the seL4 ecosystem
- ✓ Value that seL4 is formally verified

(But why?)

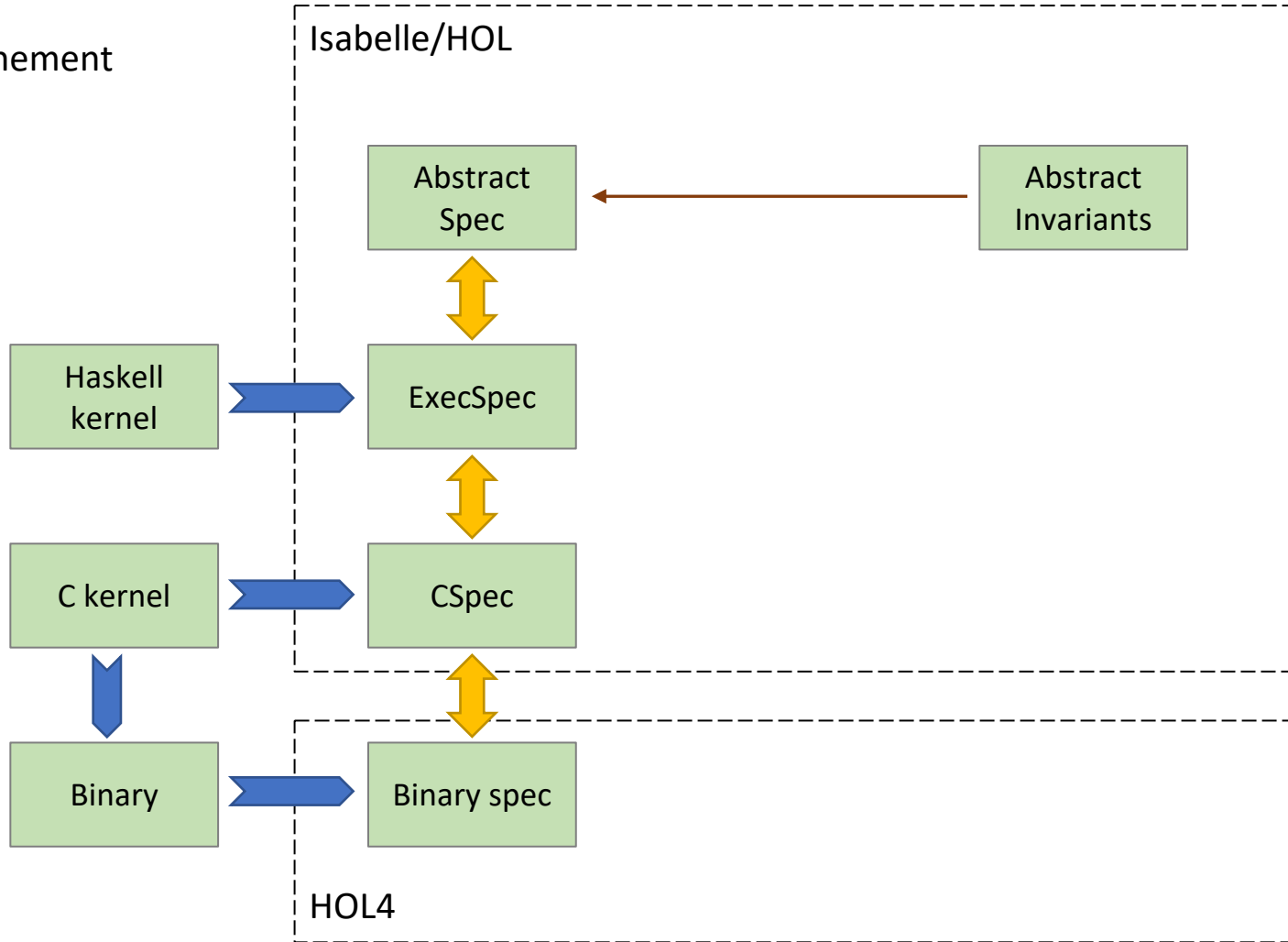
- ? Maybe read a bit about the verification
- x Maybe haven't dug into the proofs


Let's learn about the proofs!


- What are the proofs, and what do they mean?
- Where do I look for them, and how do I read them?
- How do they actually provide strong assurances?

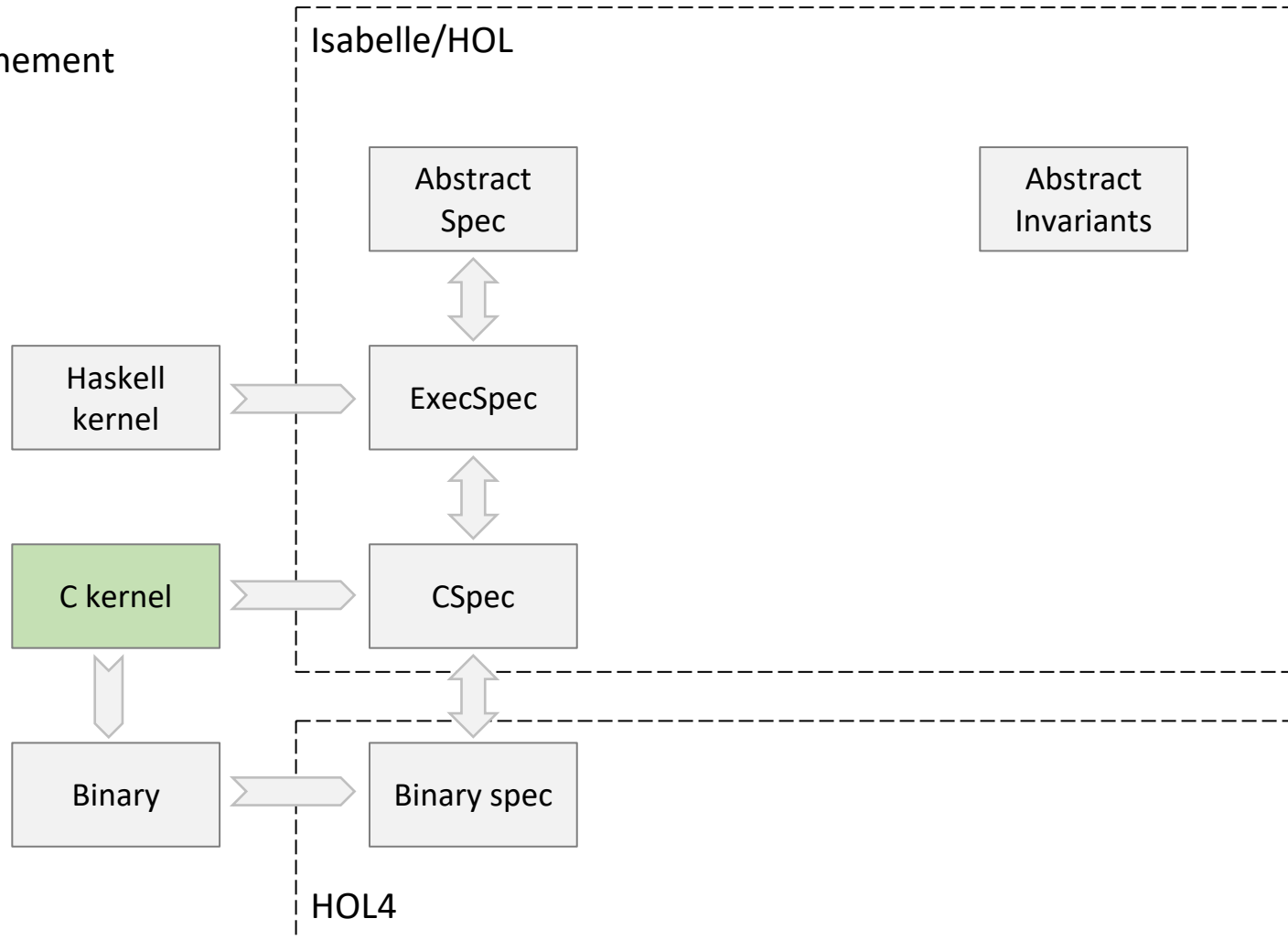
➡ = translation


↕ = refinement




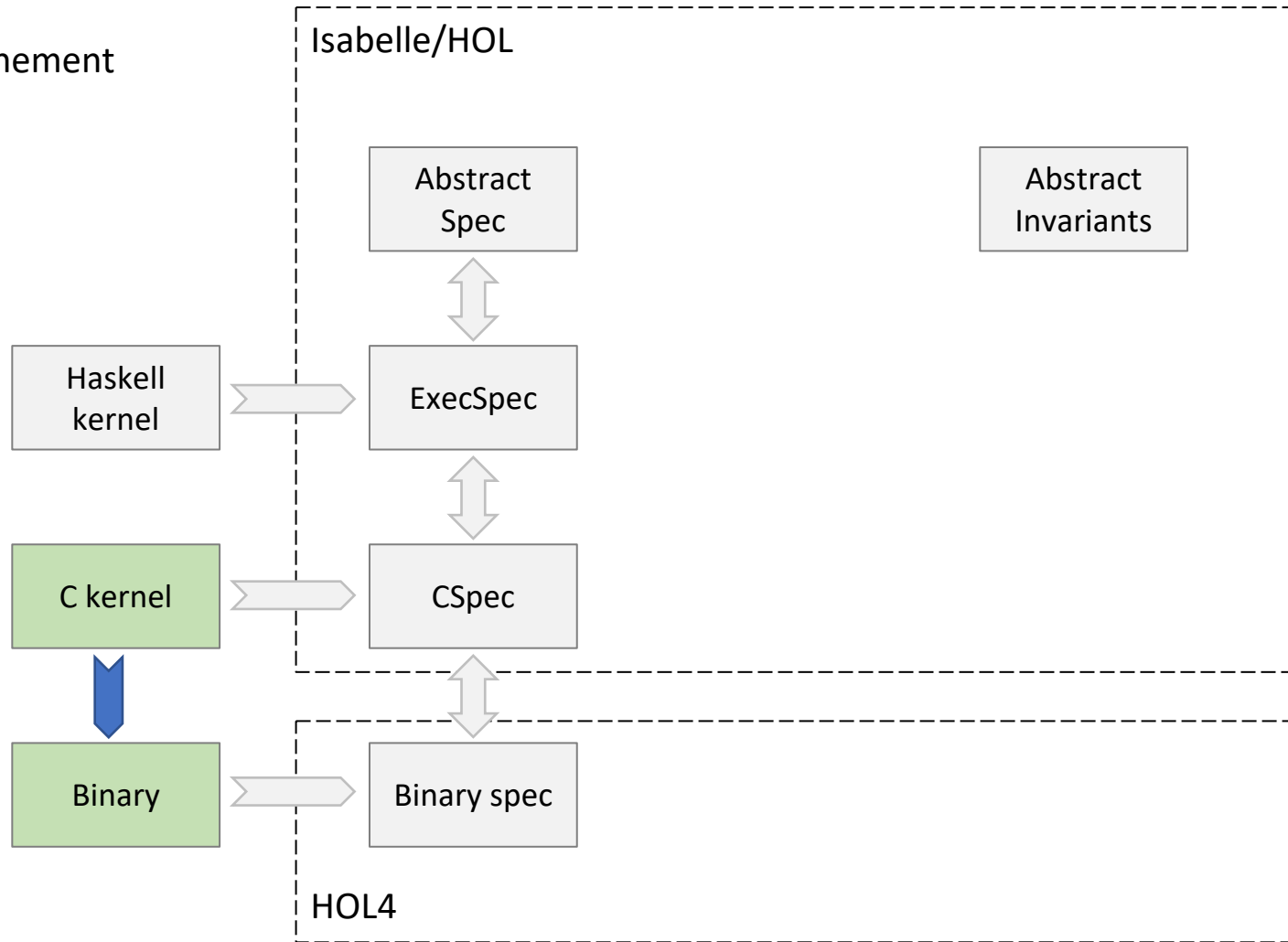
 = translation


 = refinement




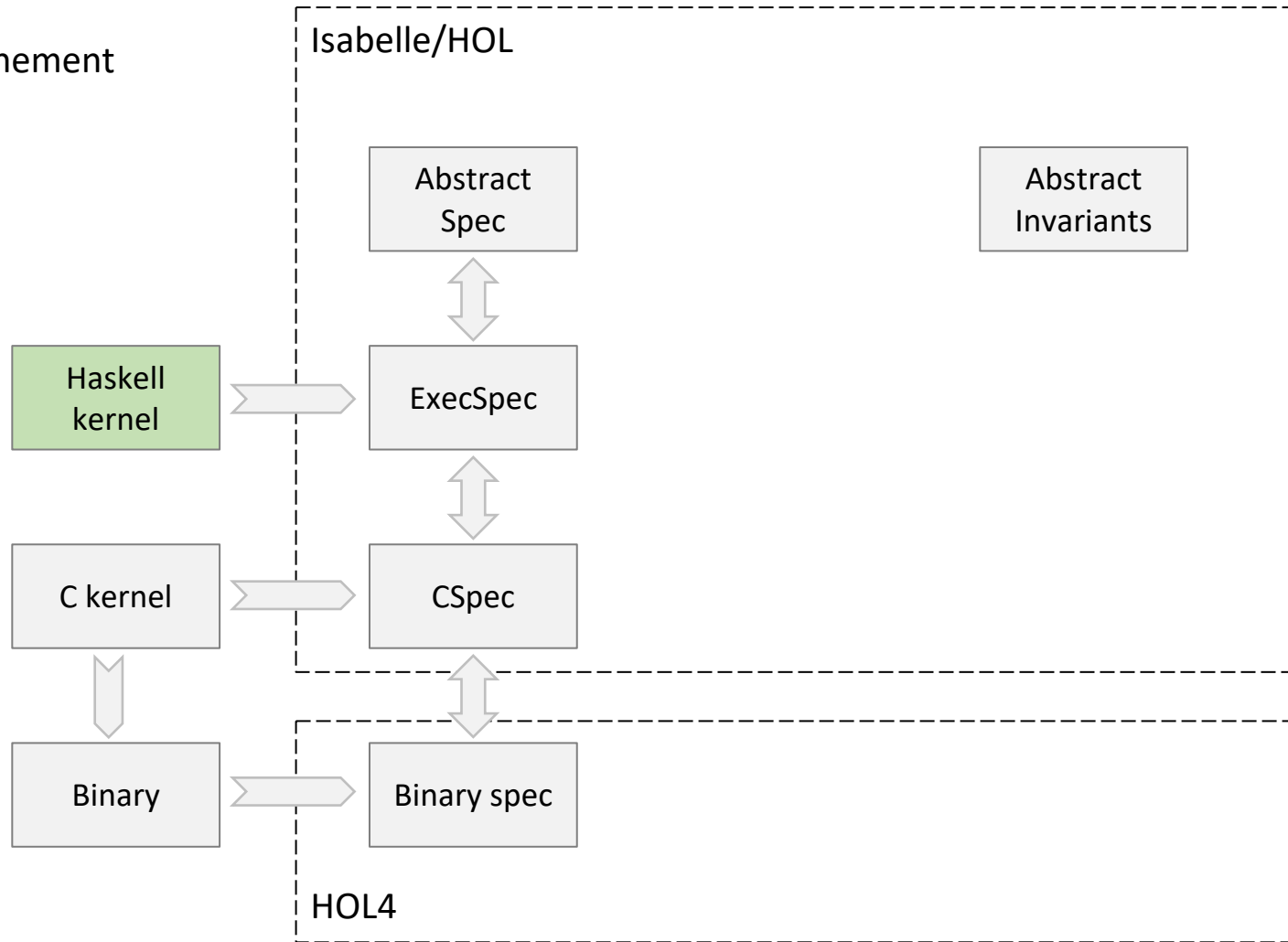
 = translation

 = refinement




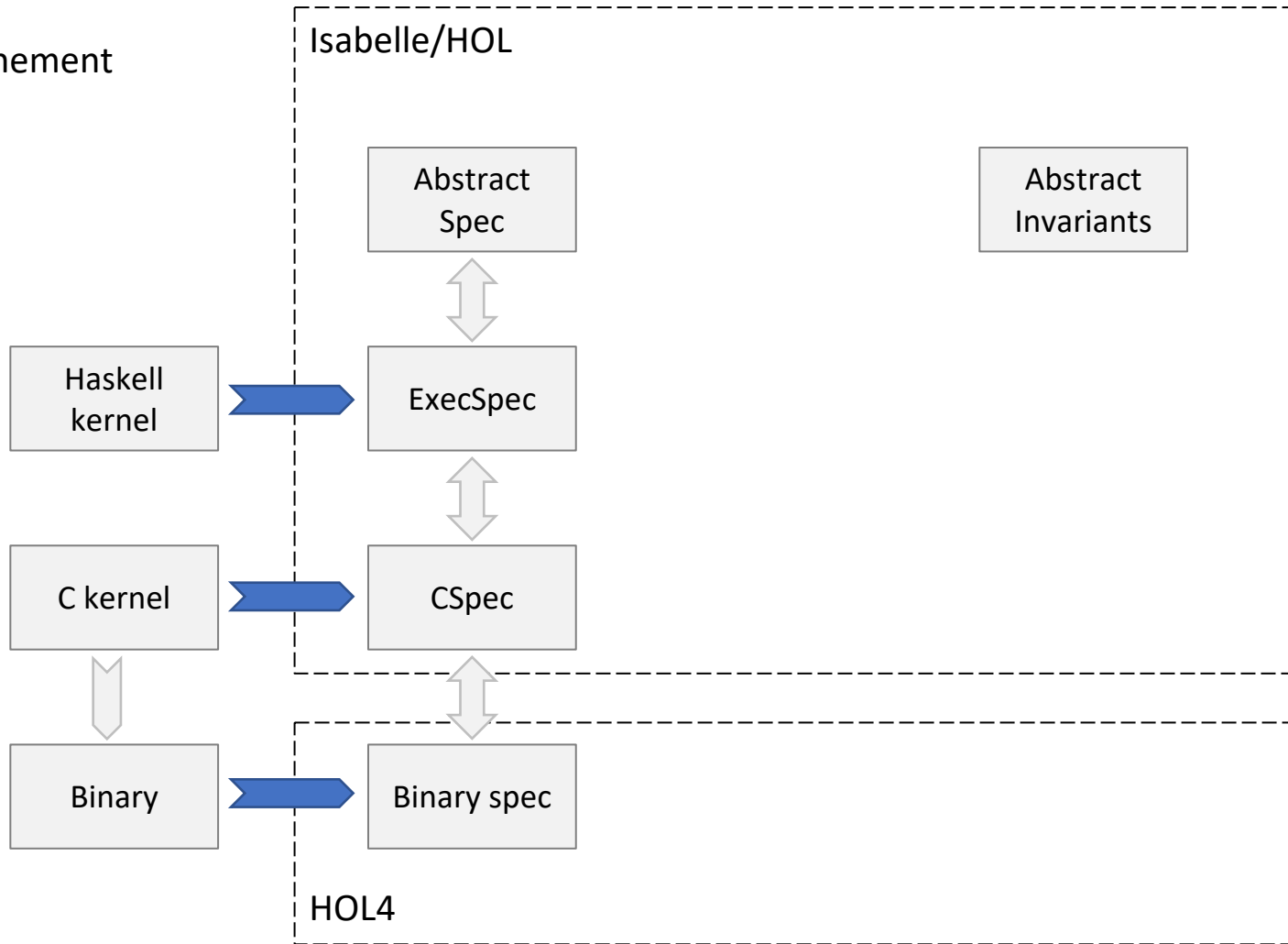
 = translation

 = refinement




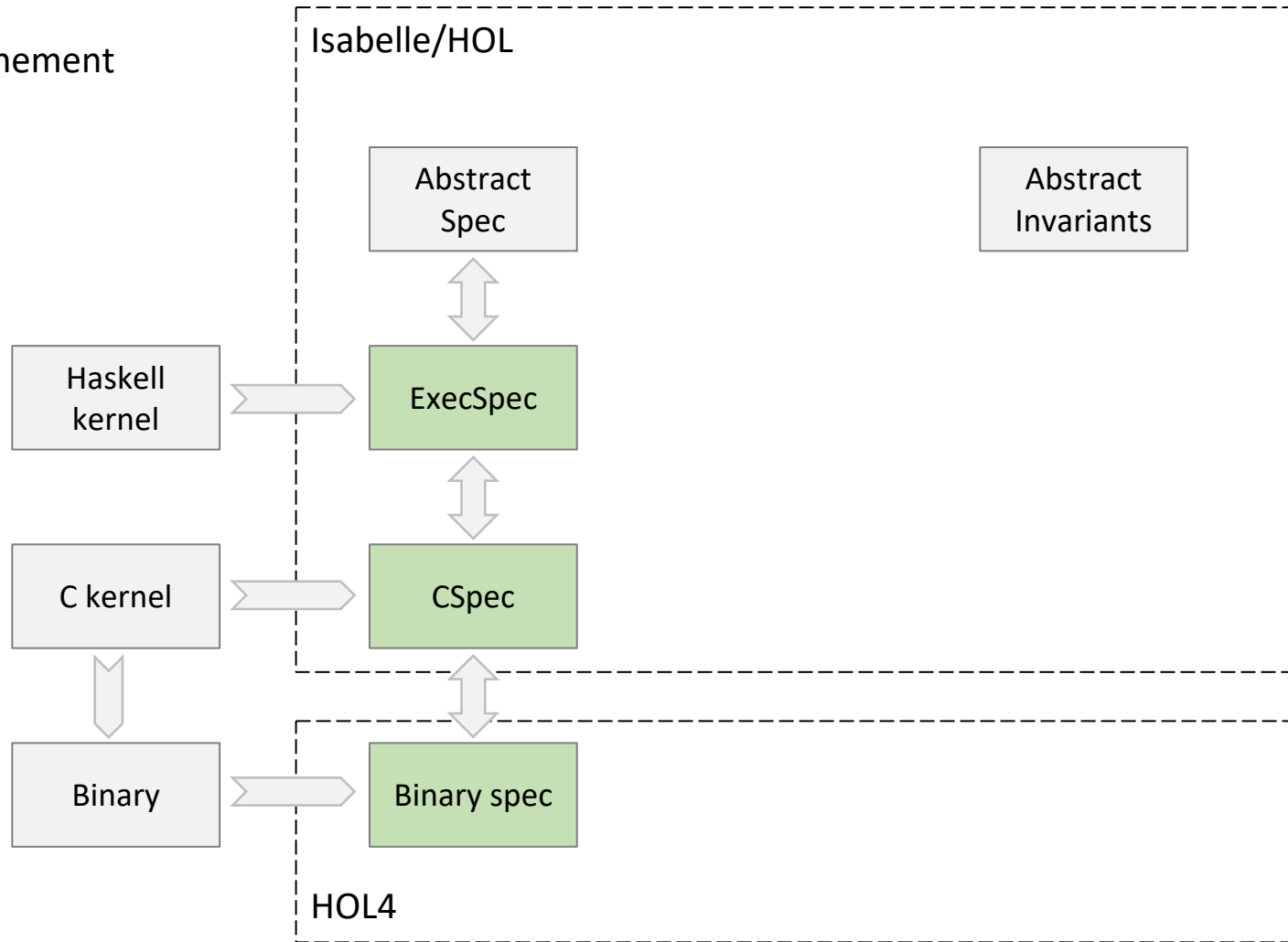
 = translation

 = refinement



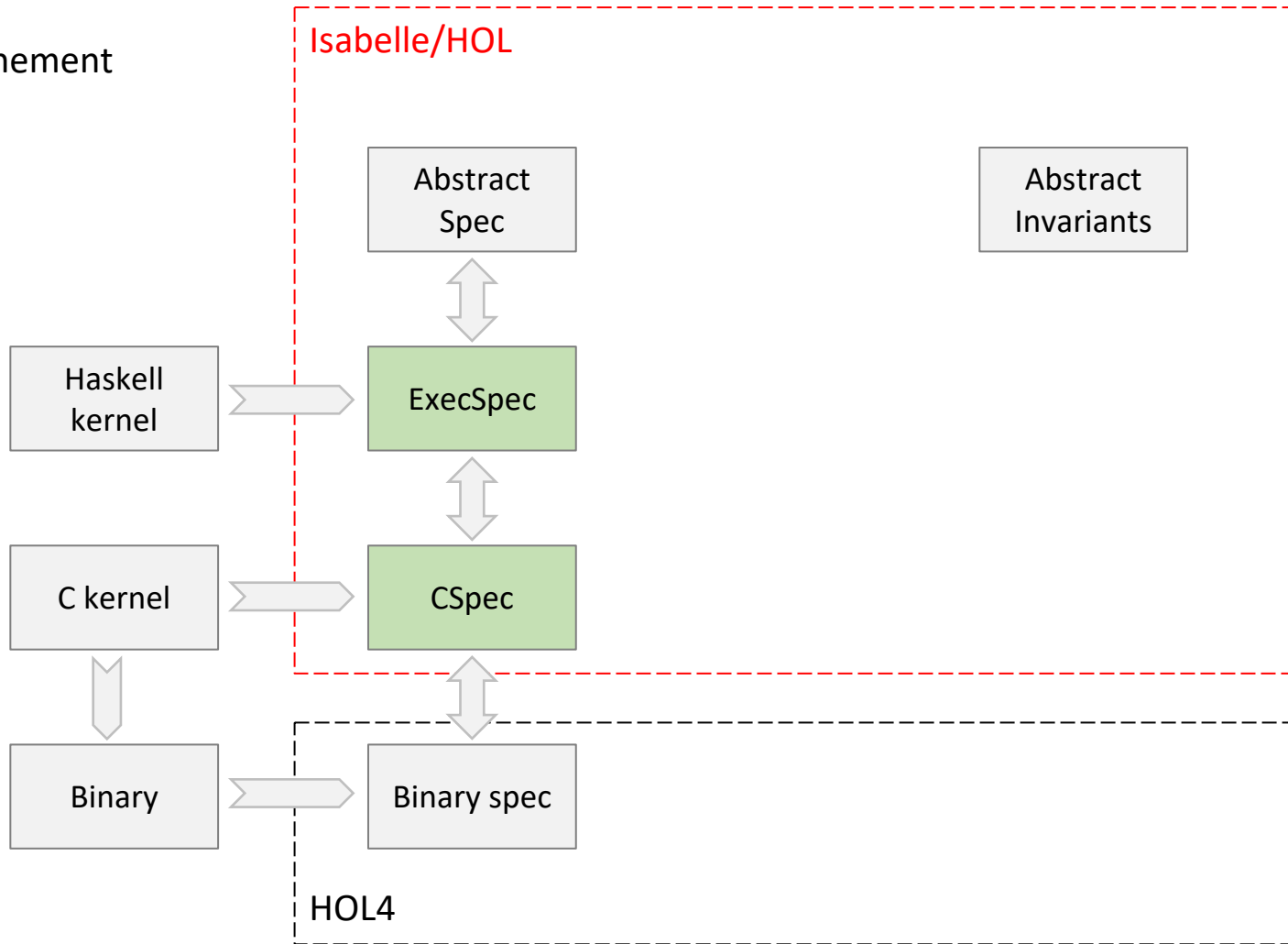
 = translation

 = refinement




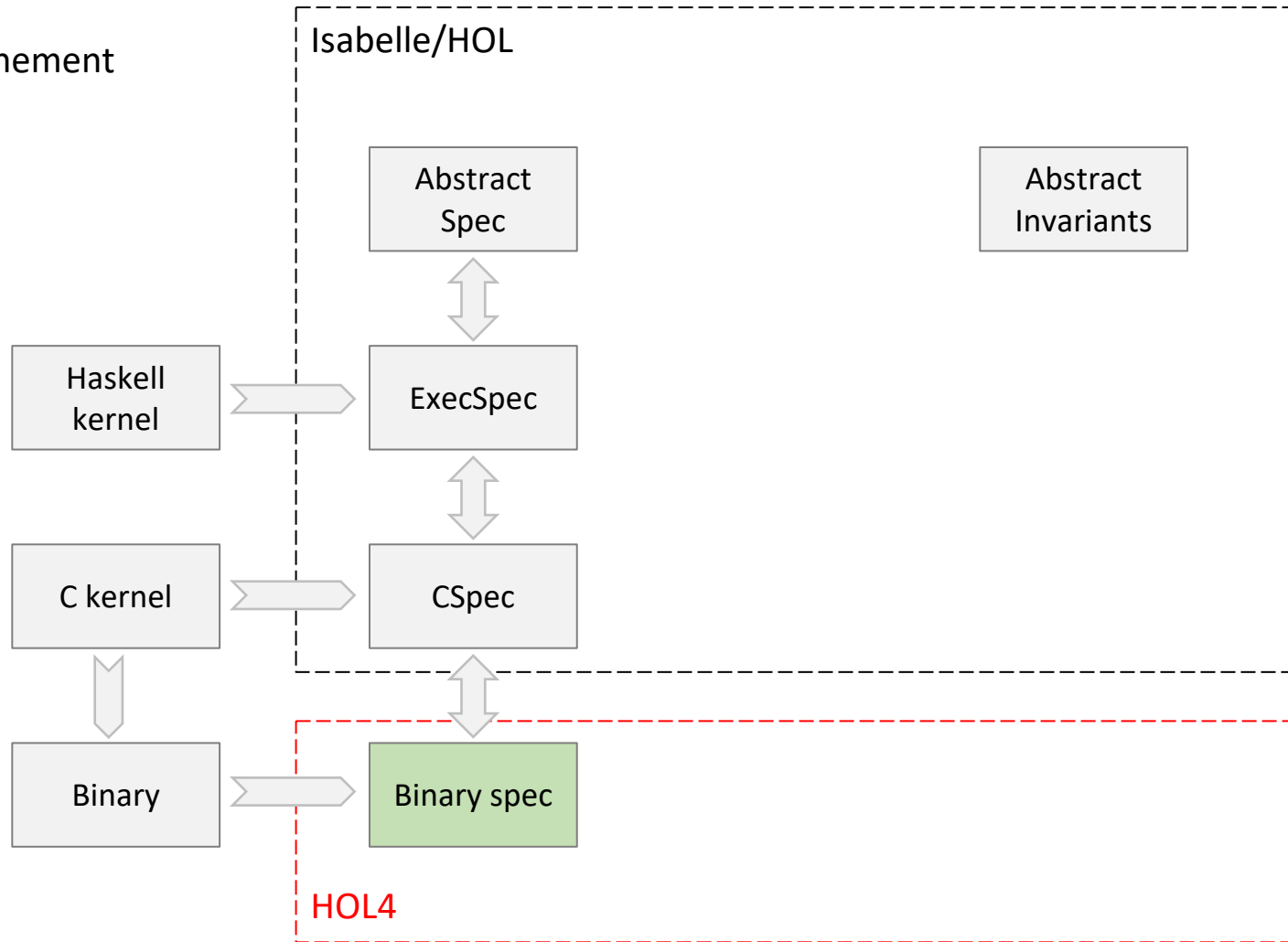
 = translation

 = refinement




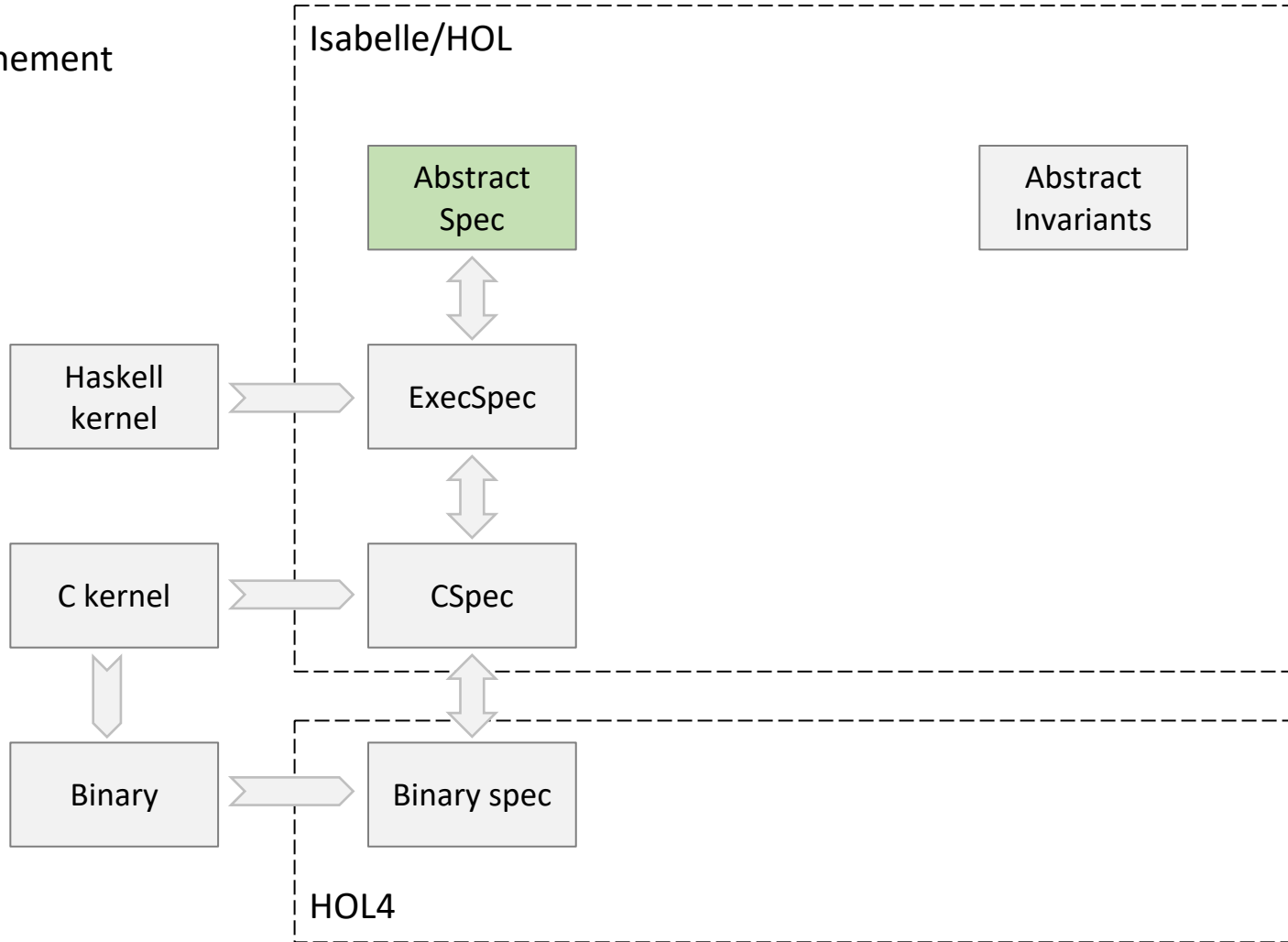
 = translation


 = refinement




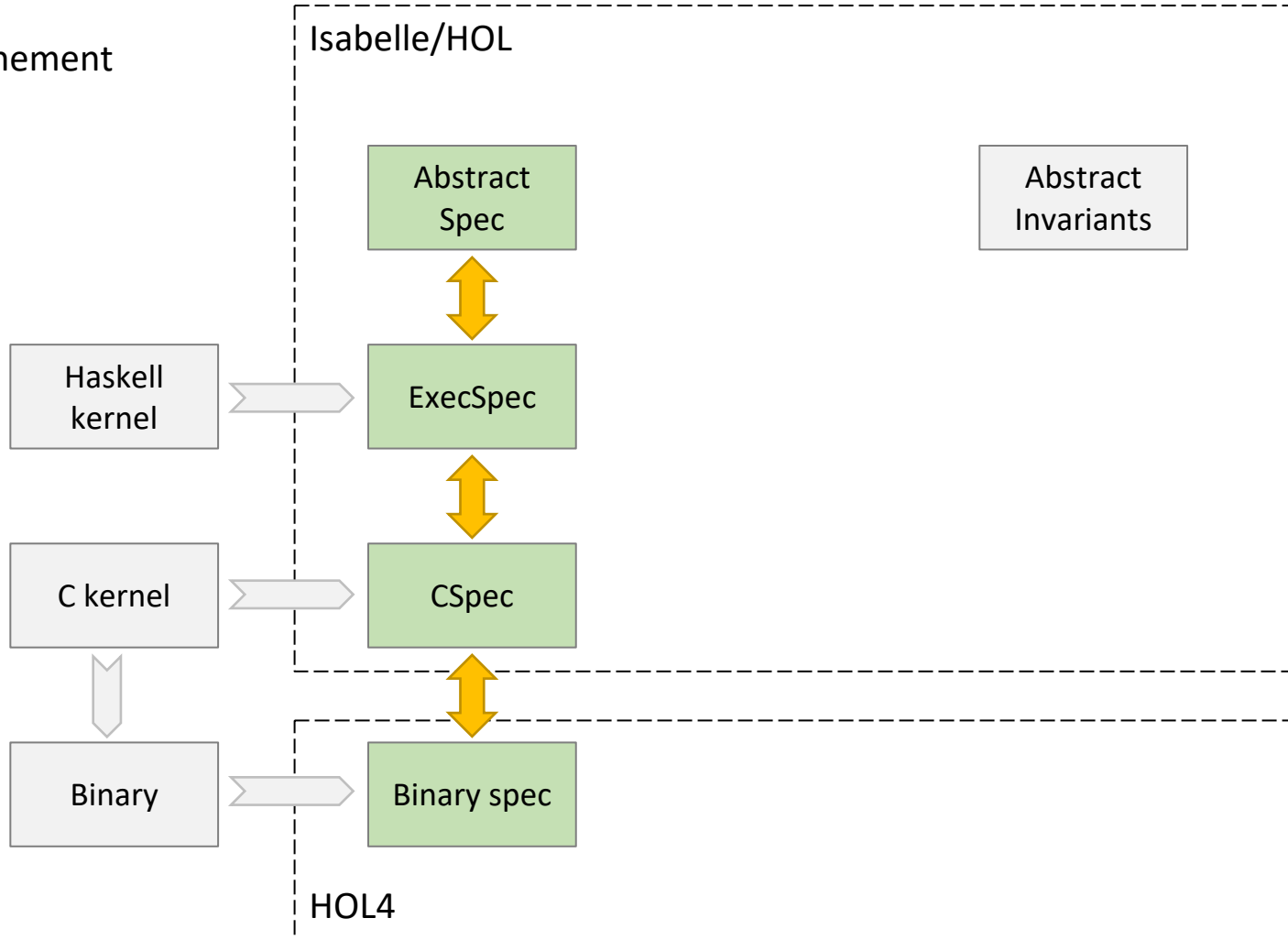
 = translation


 = refinement




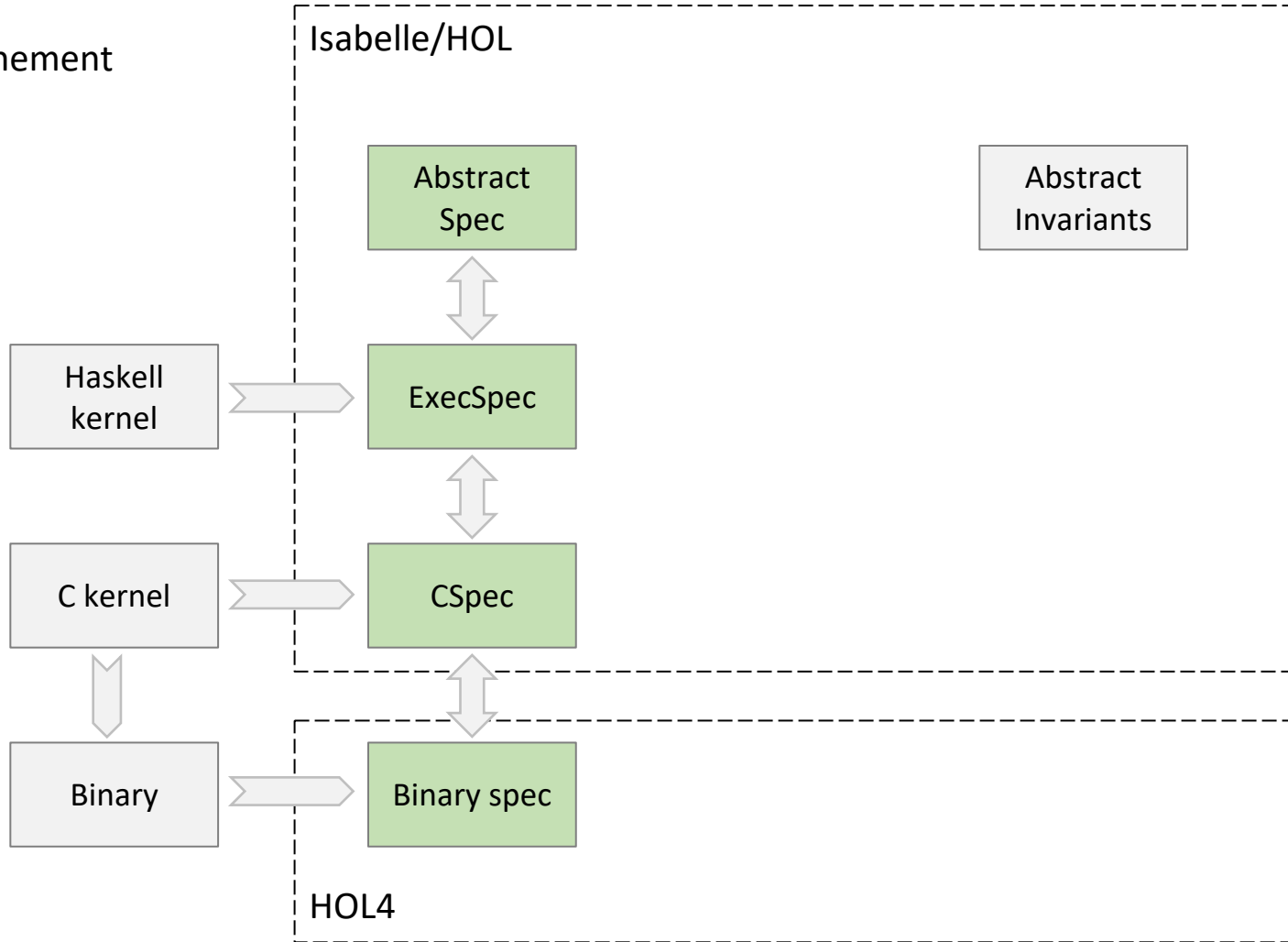
 = translation


 = refinement




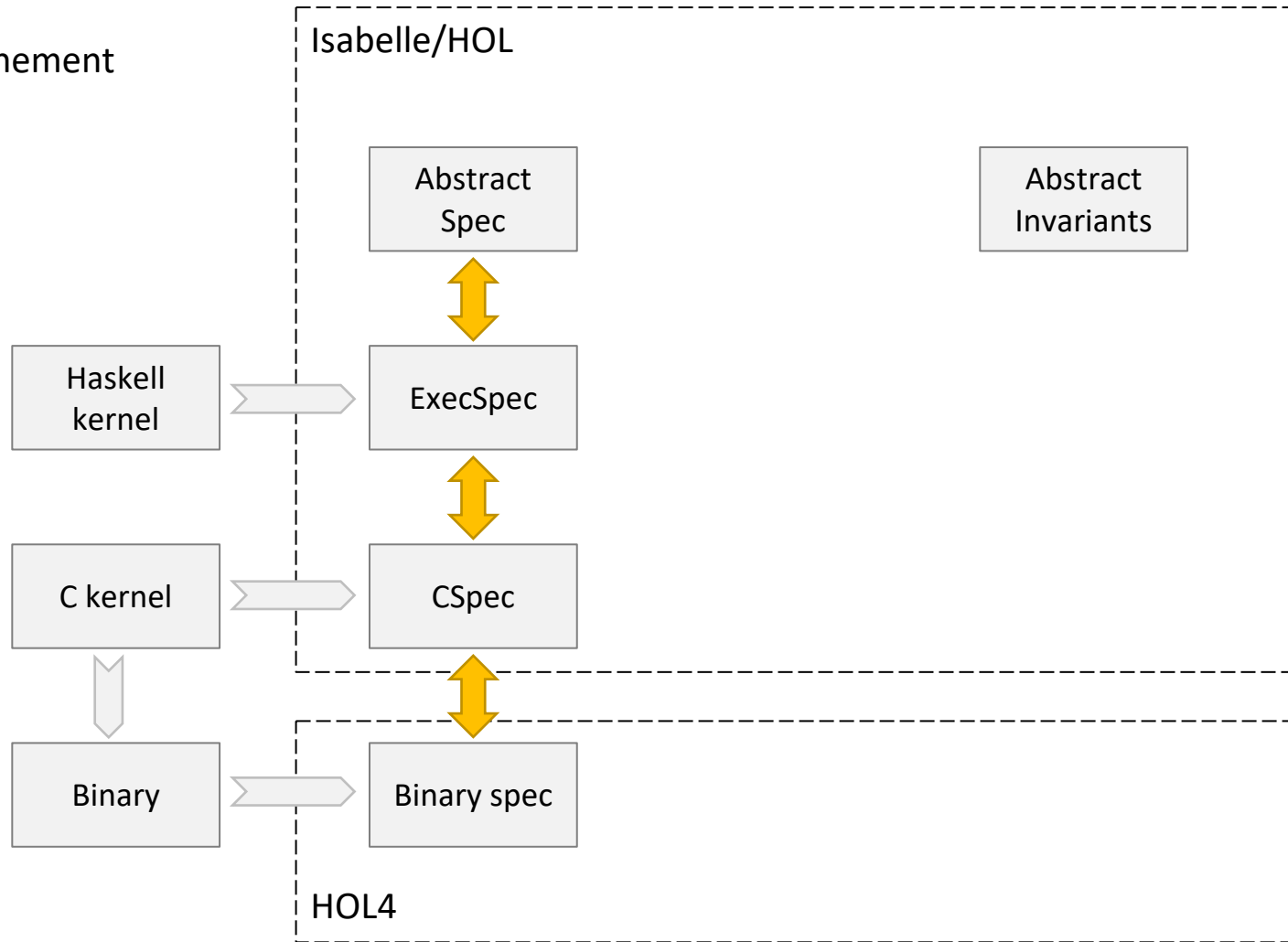
 = translation

 = refinement



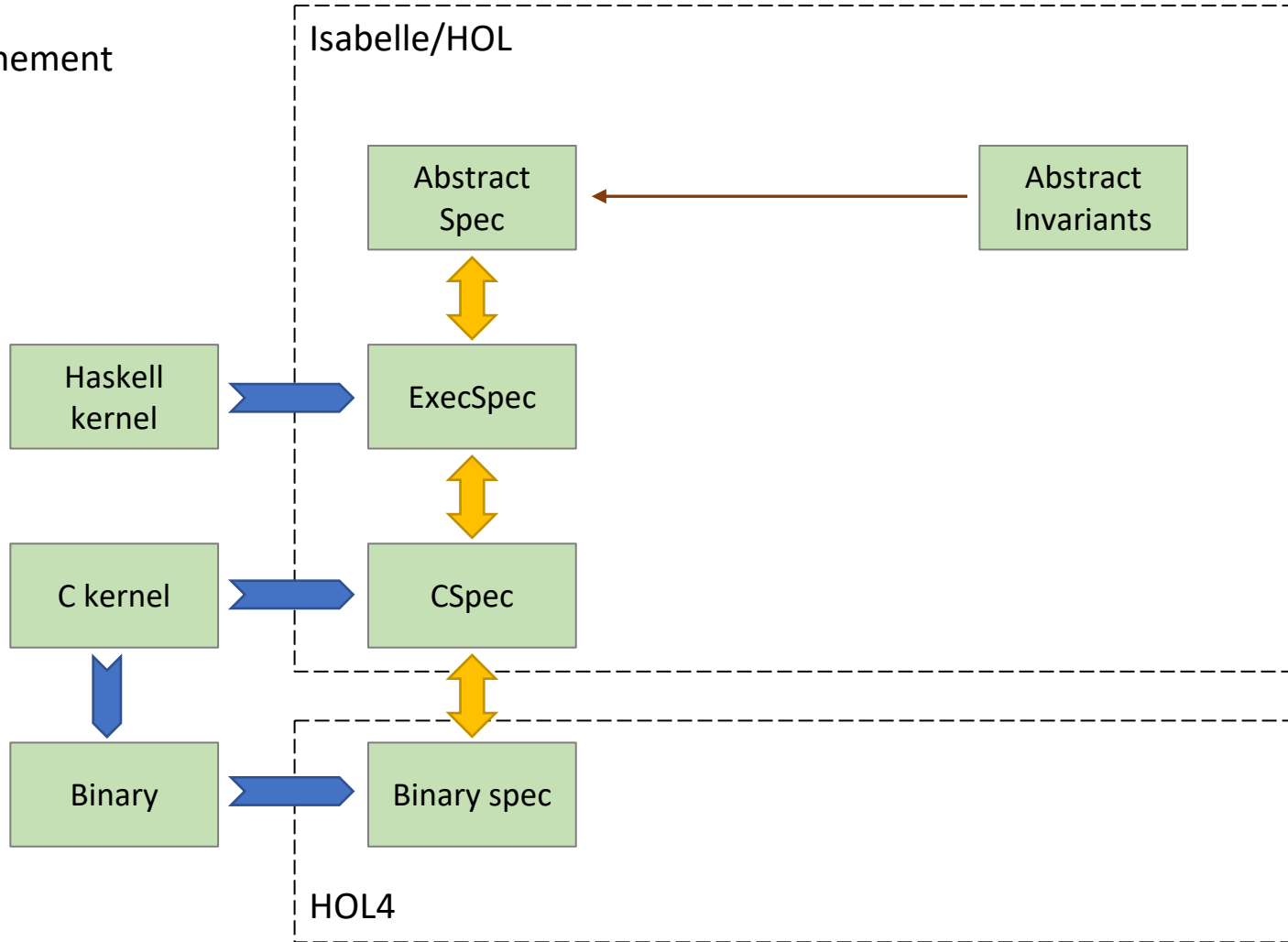
 = translation

 = refinement



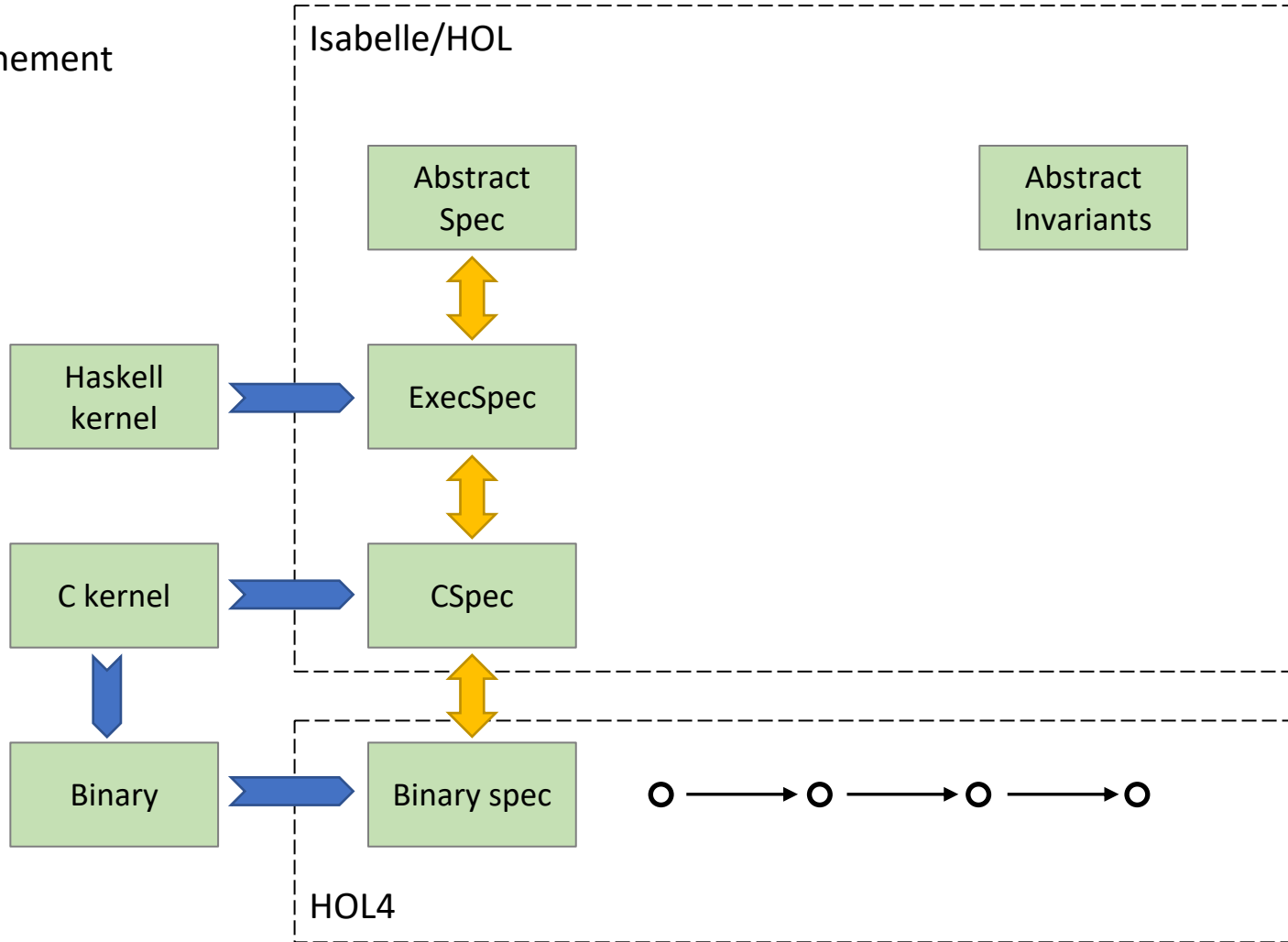
➡ = translation

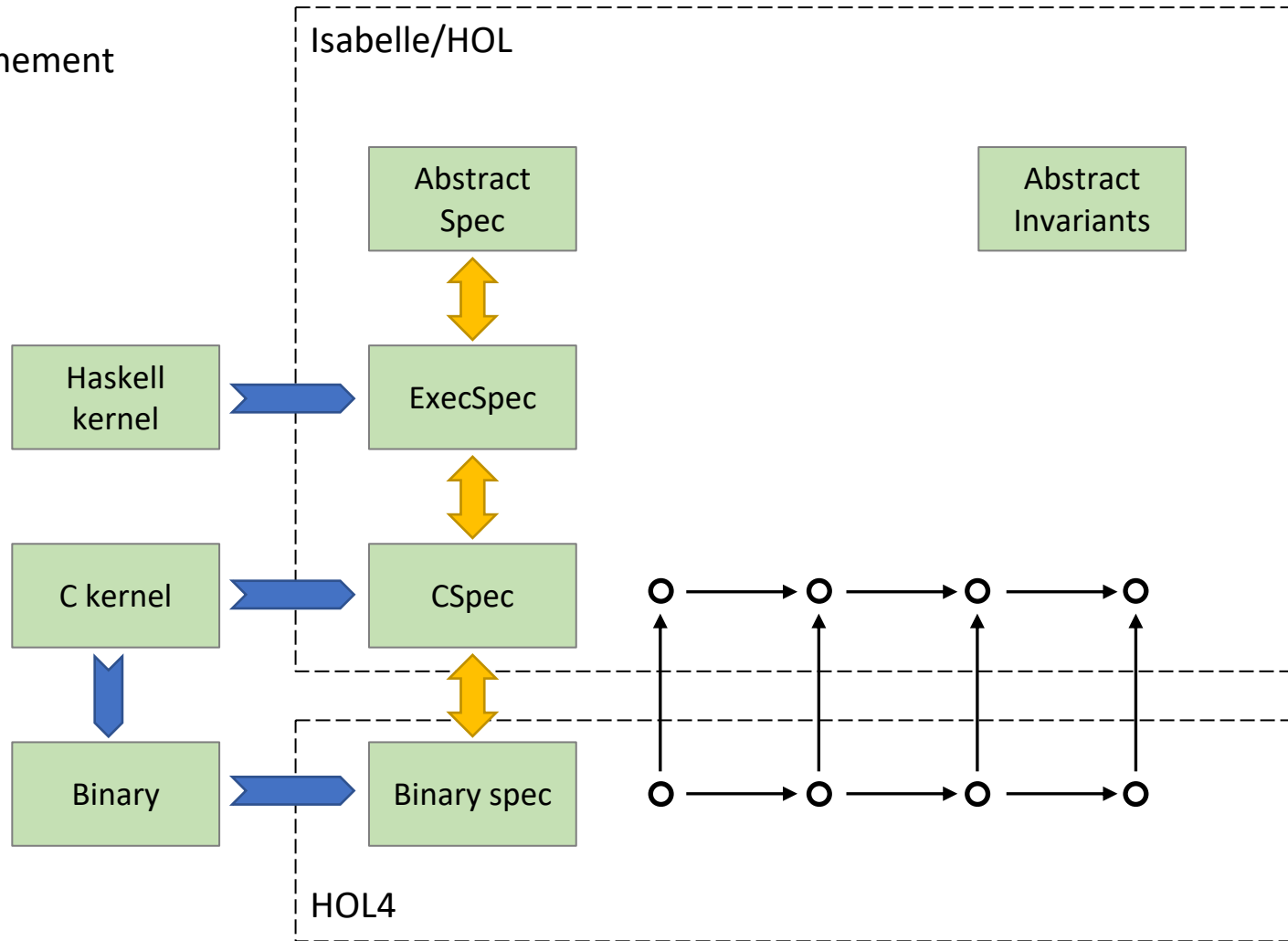
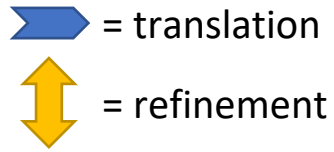
↕ = refinement



➡ = translation

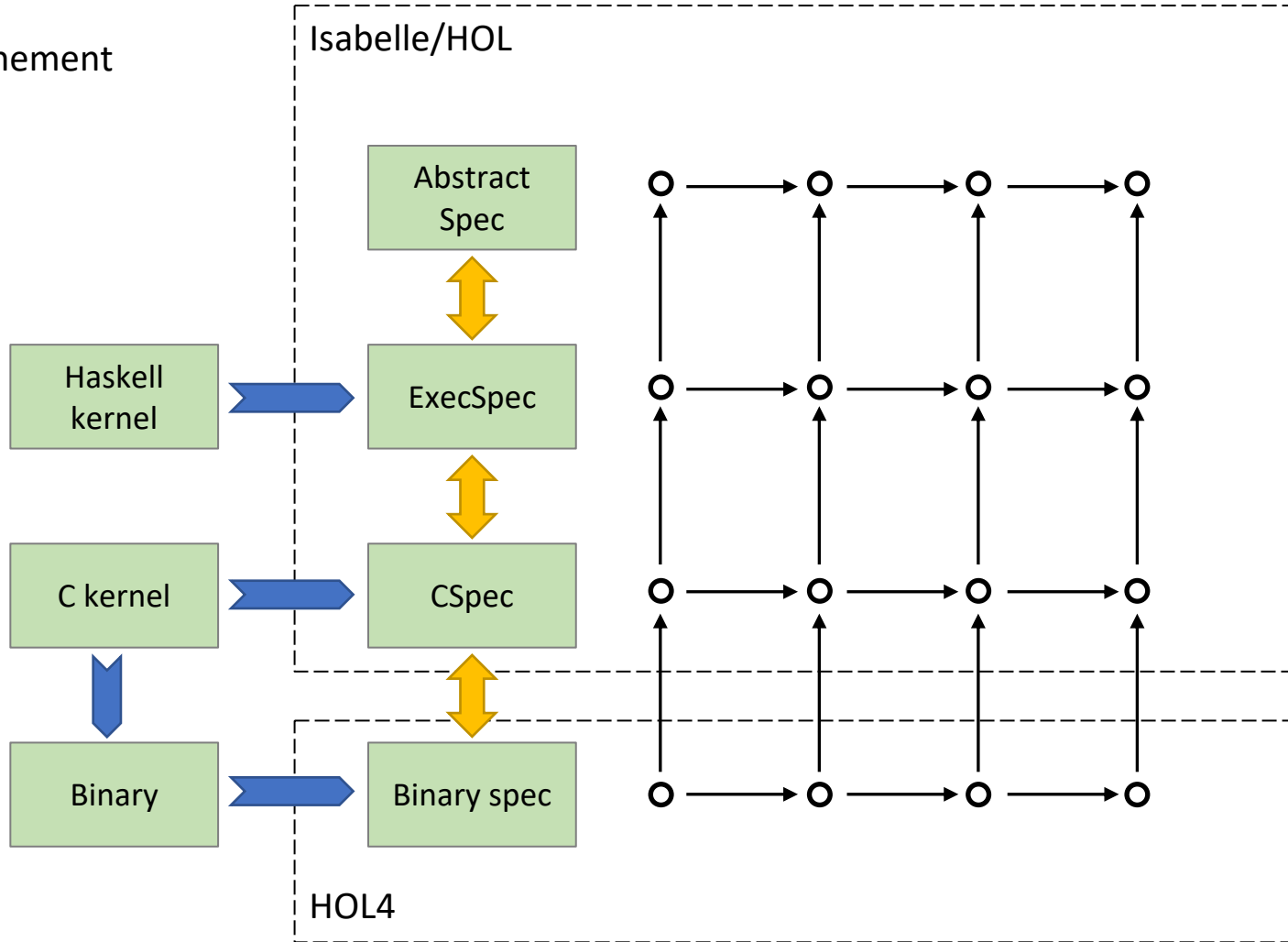
↕ = refinement





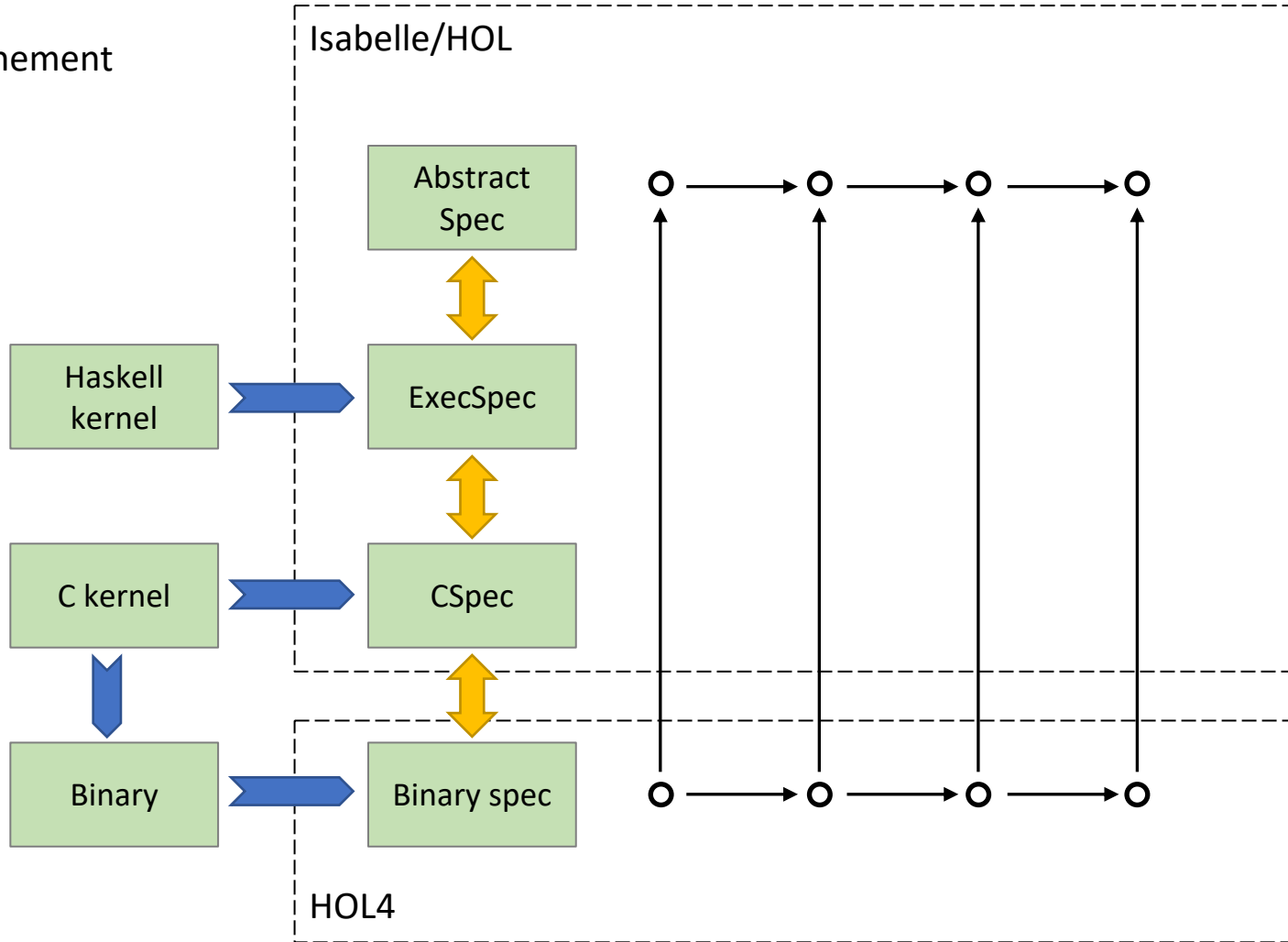
➡ = translation


↕ = refinement




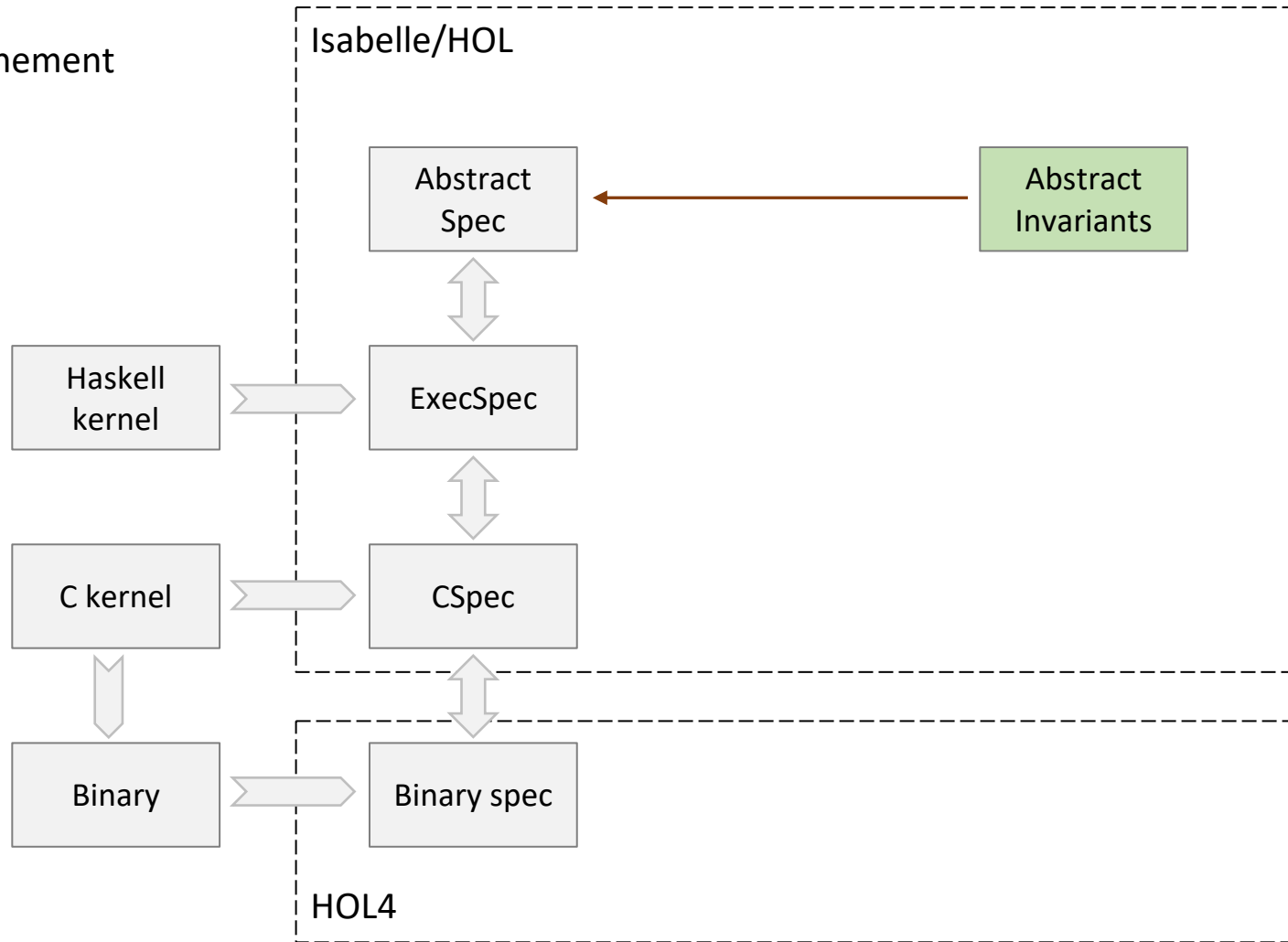
➡ = translation

↕ = refinement



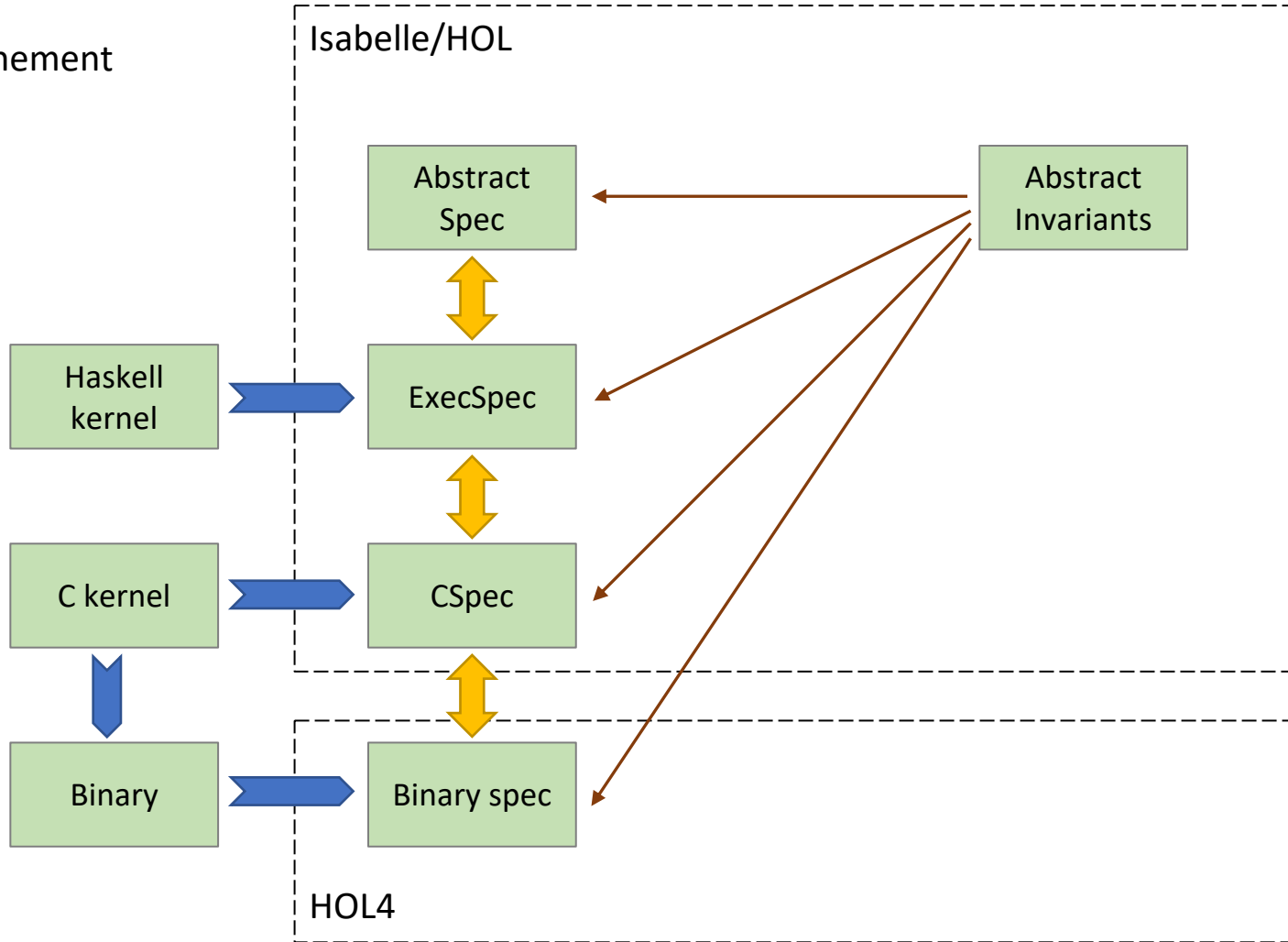
 = translation

 = refinement



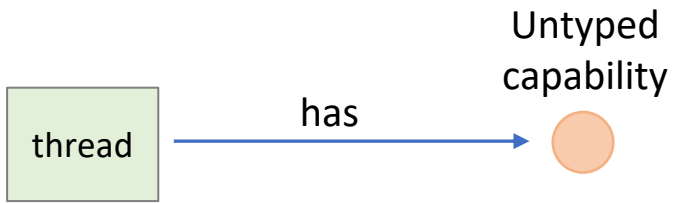
➡ = translation

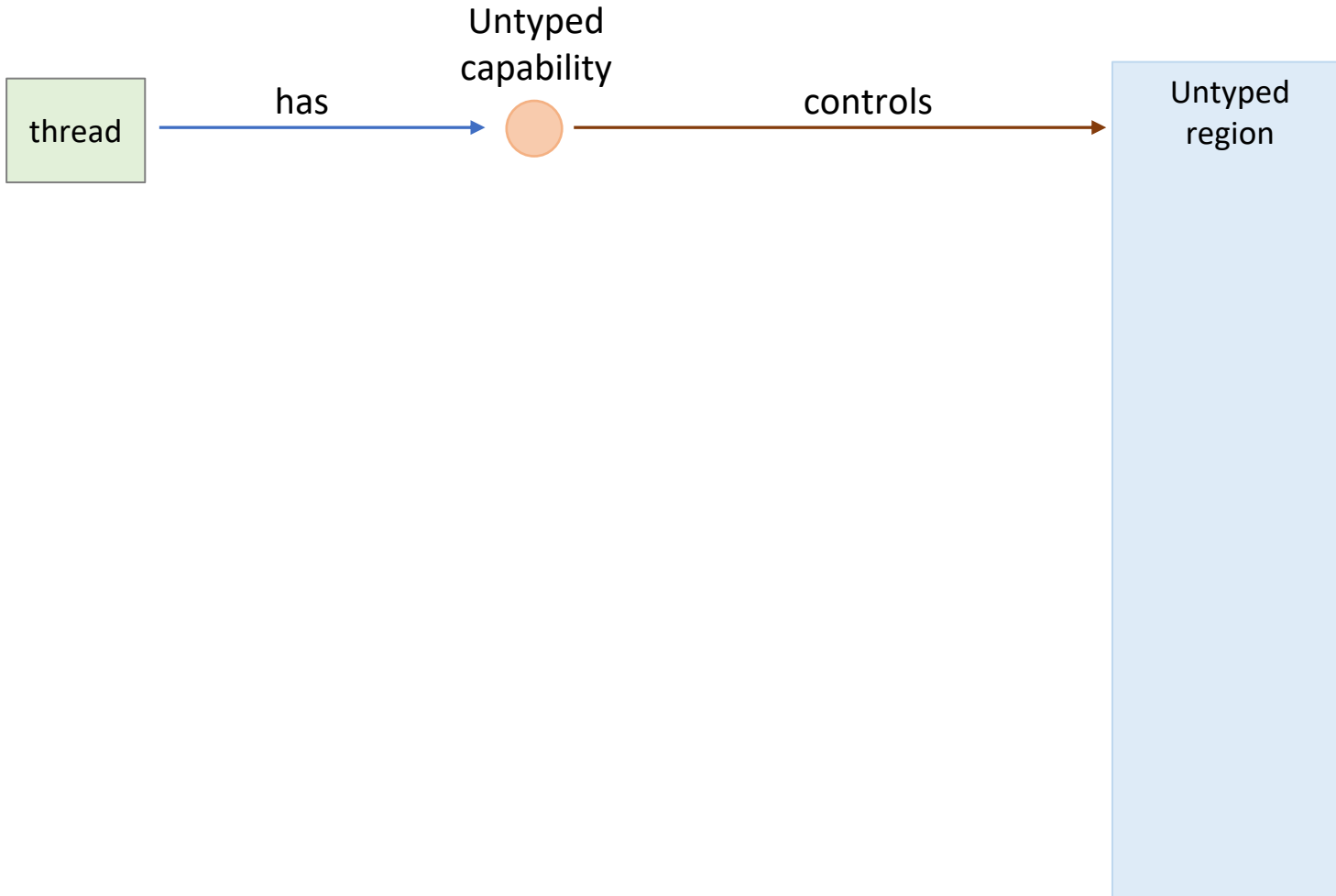
↕ = refinement

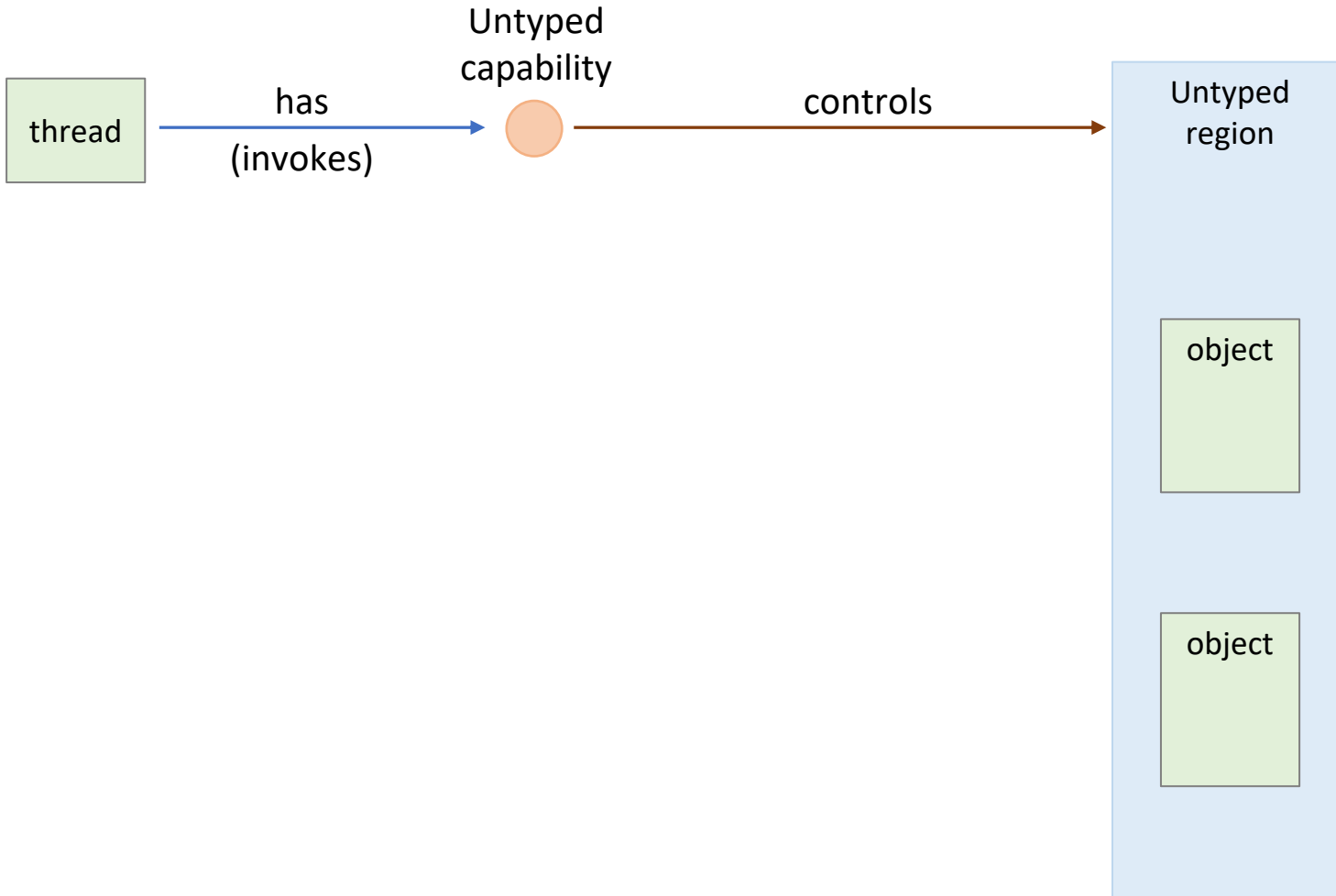


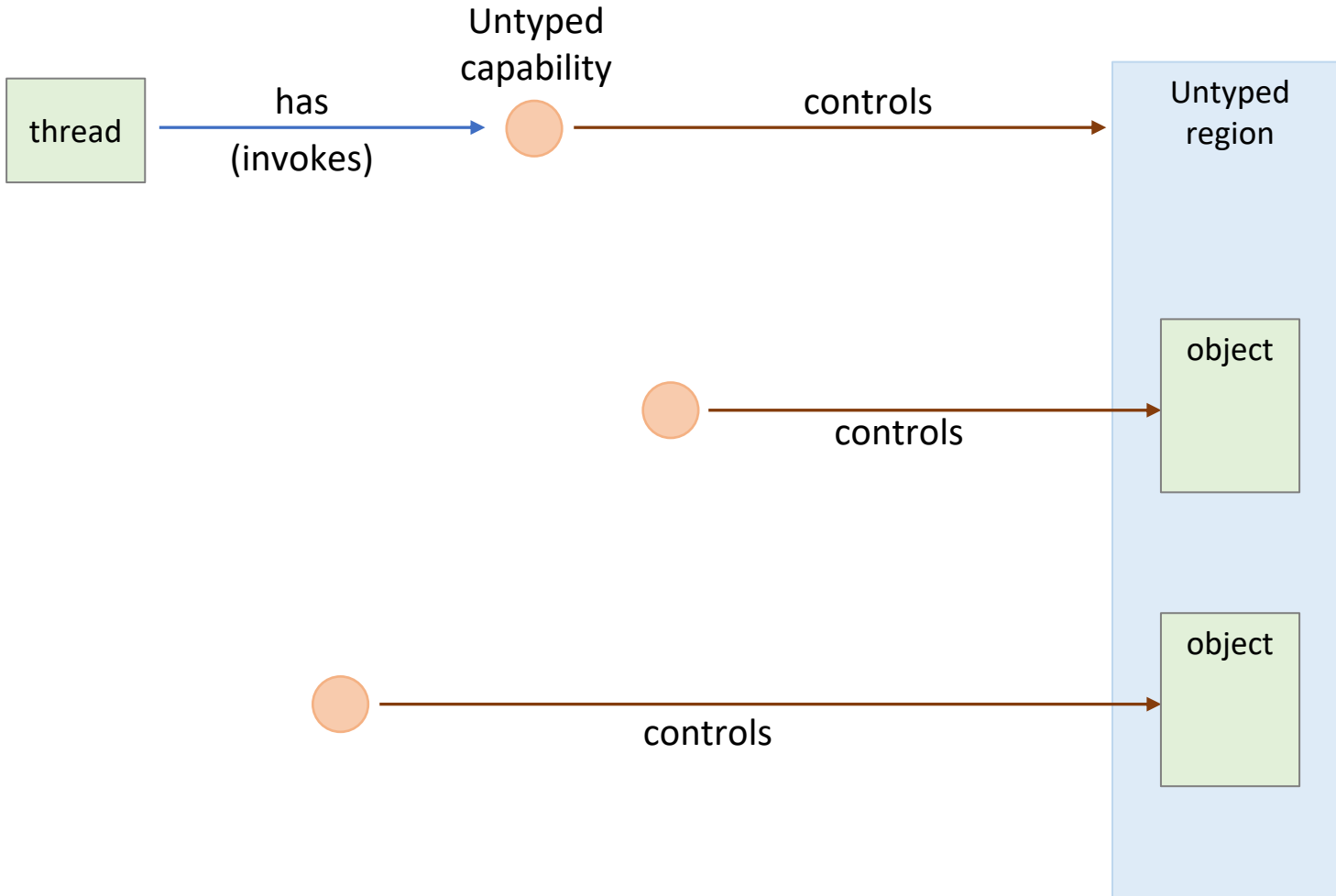
The tour:

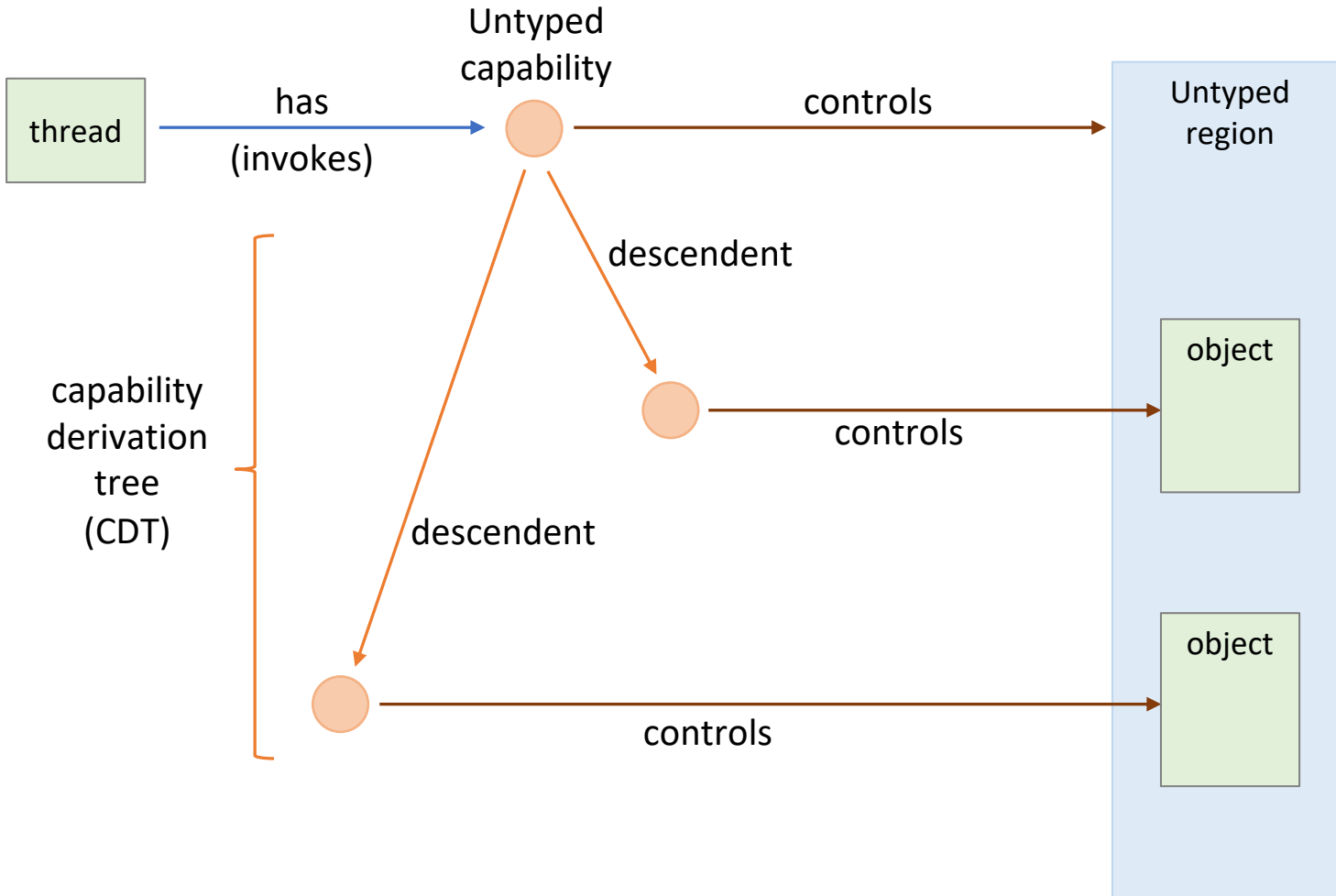
- Introduction to Isabelle/HOL
- Introduction to specifications and Hoare triples
- Abstract specification
- Invariants

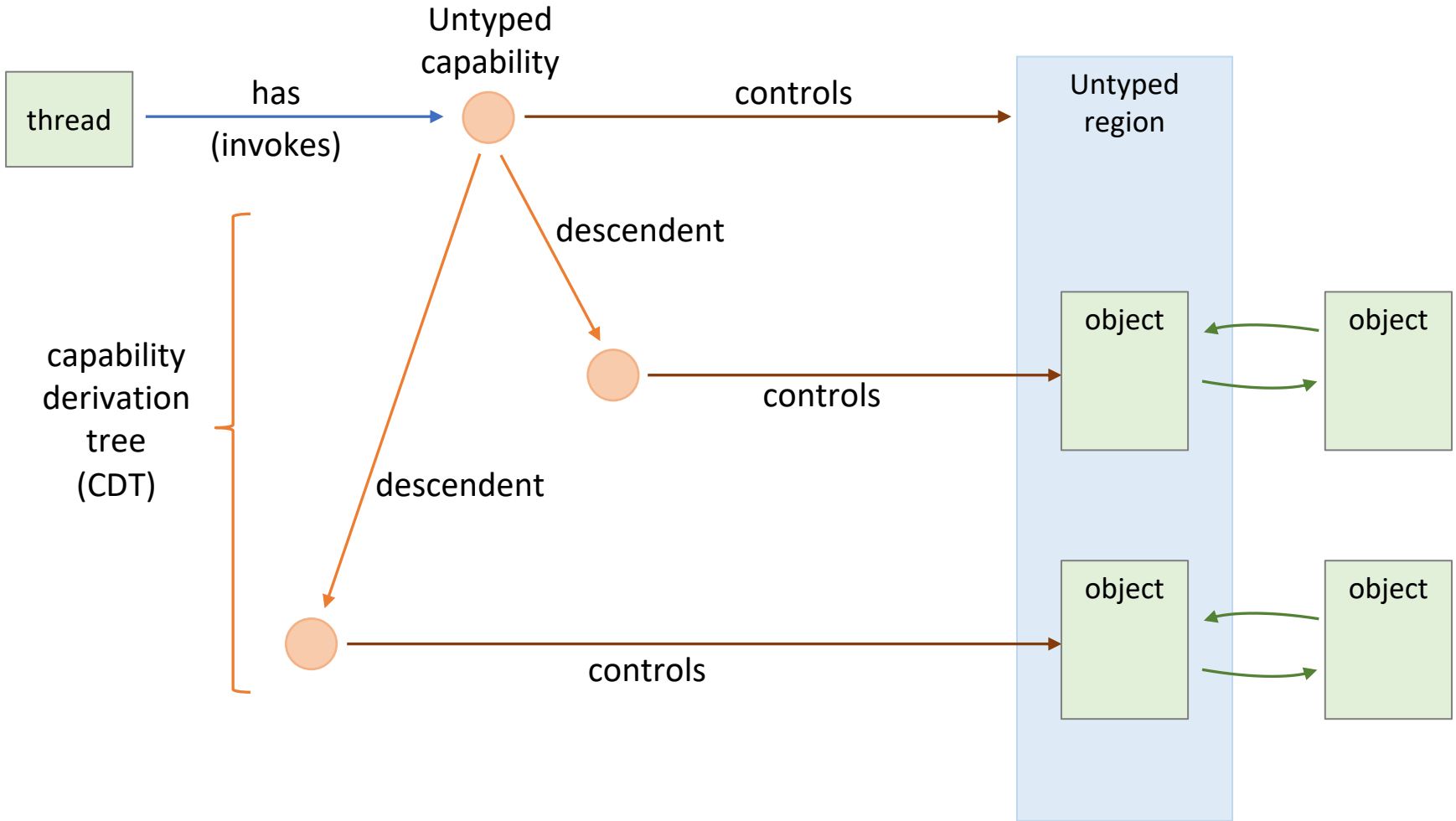


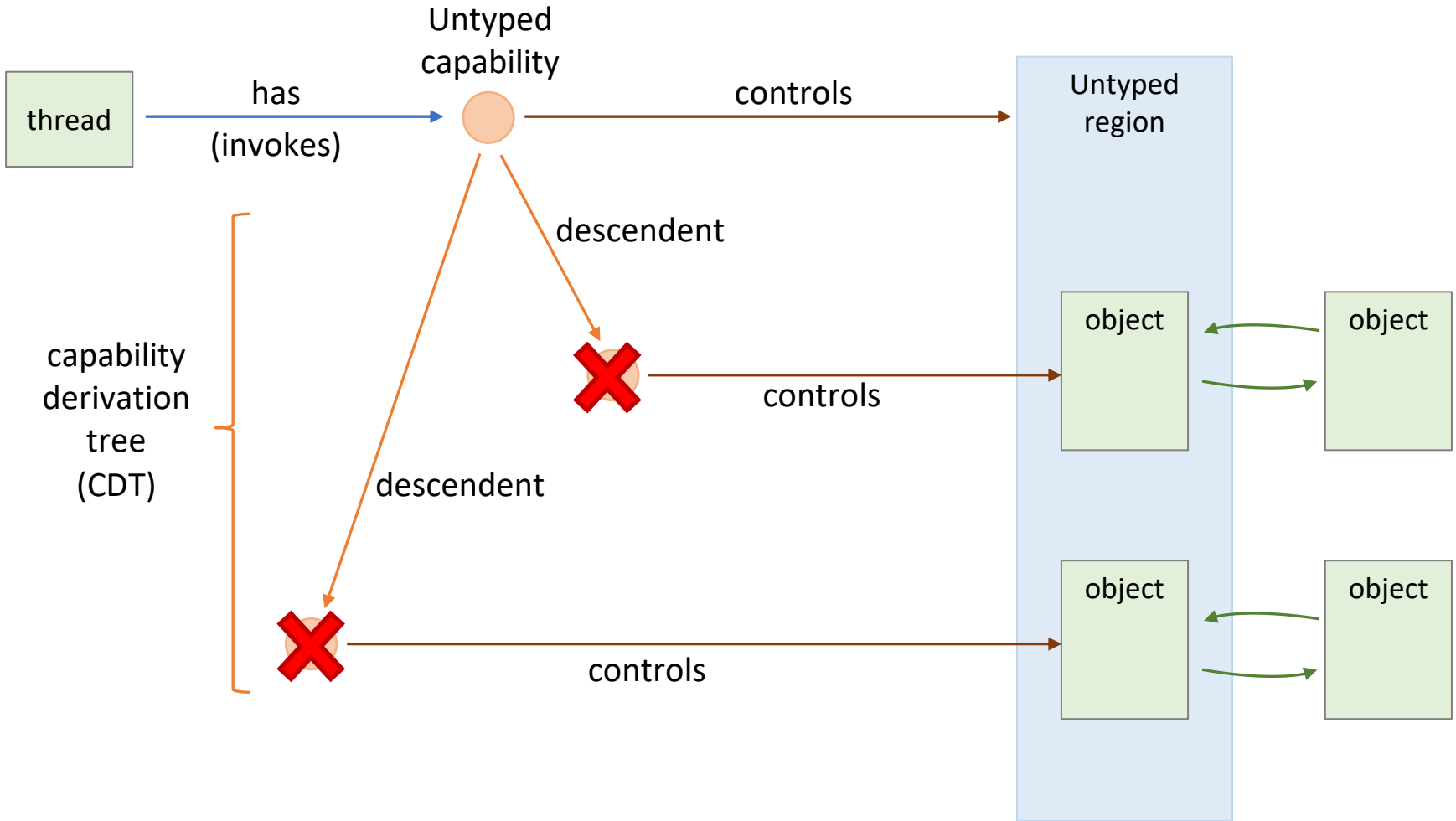


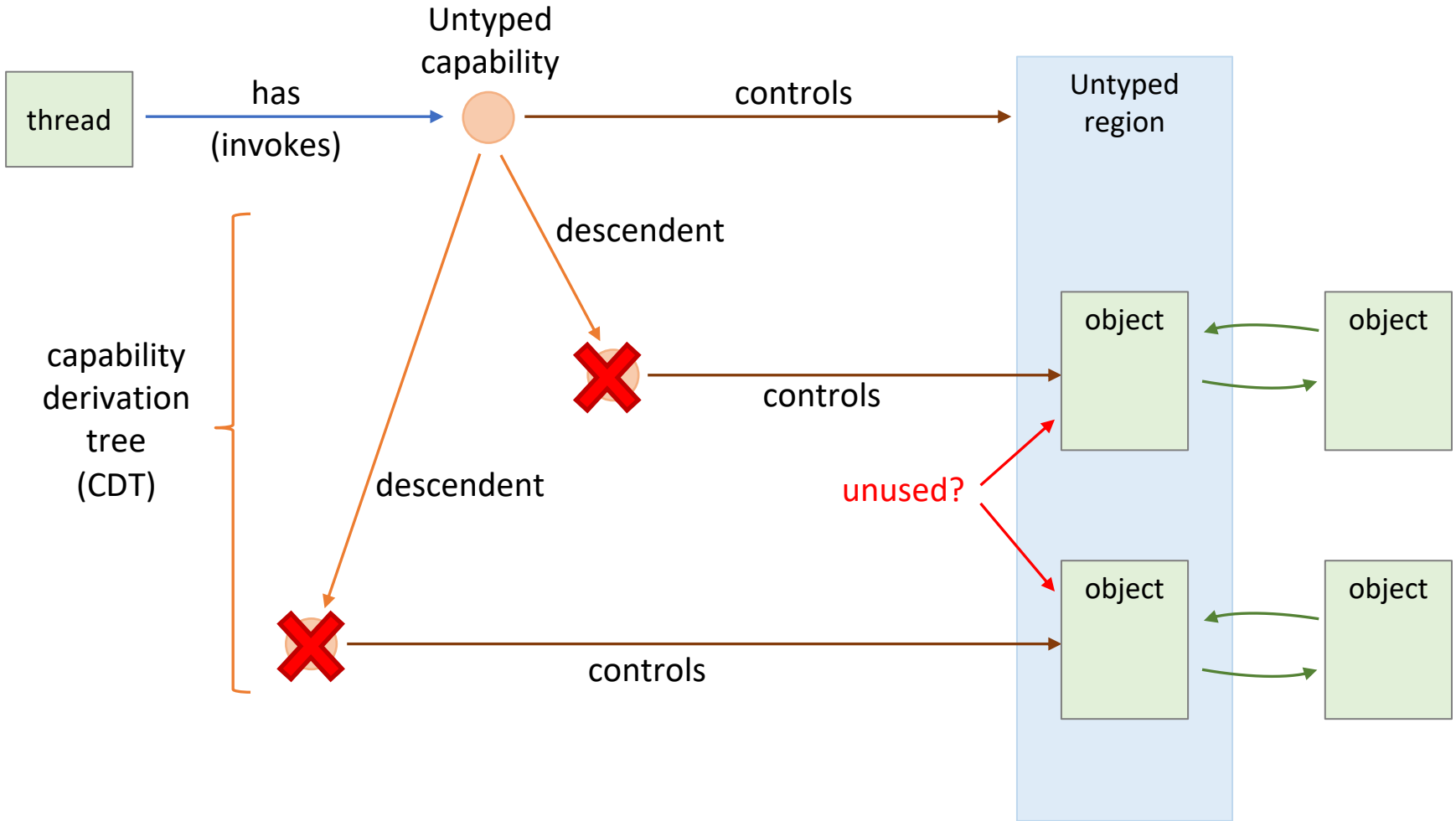


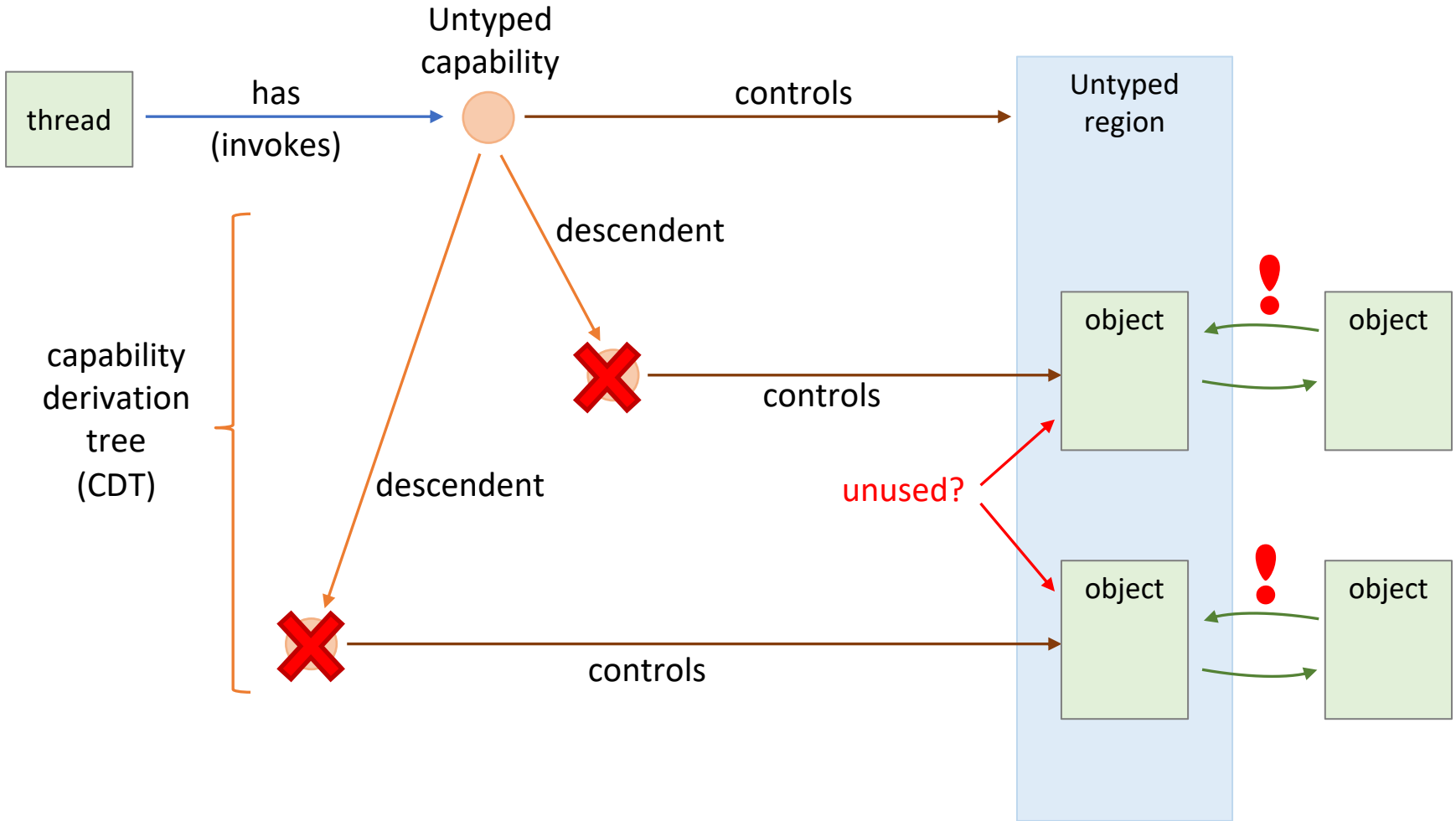


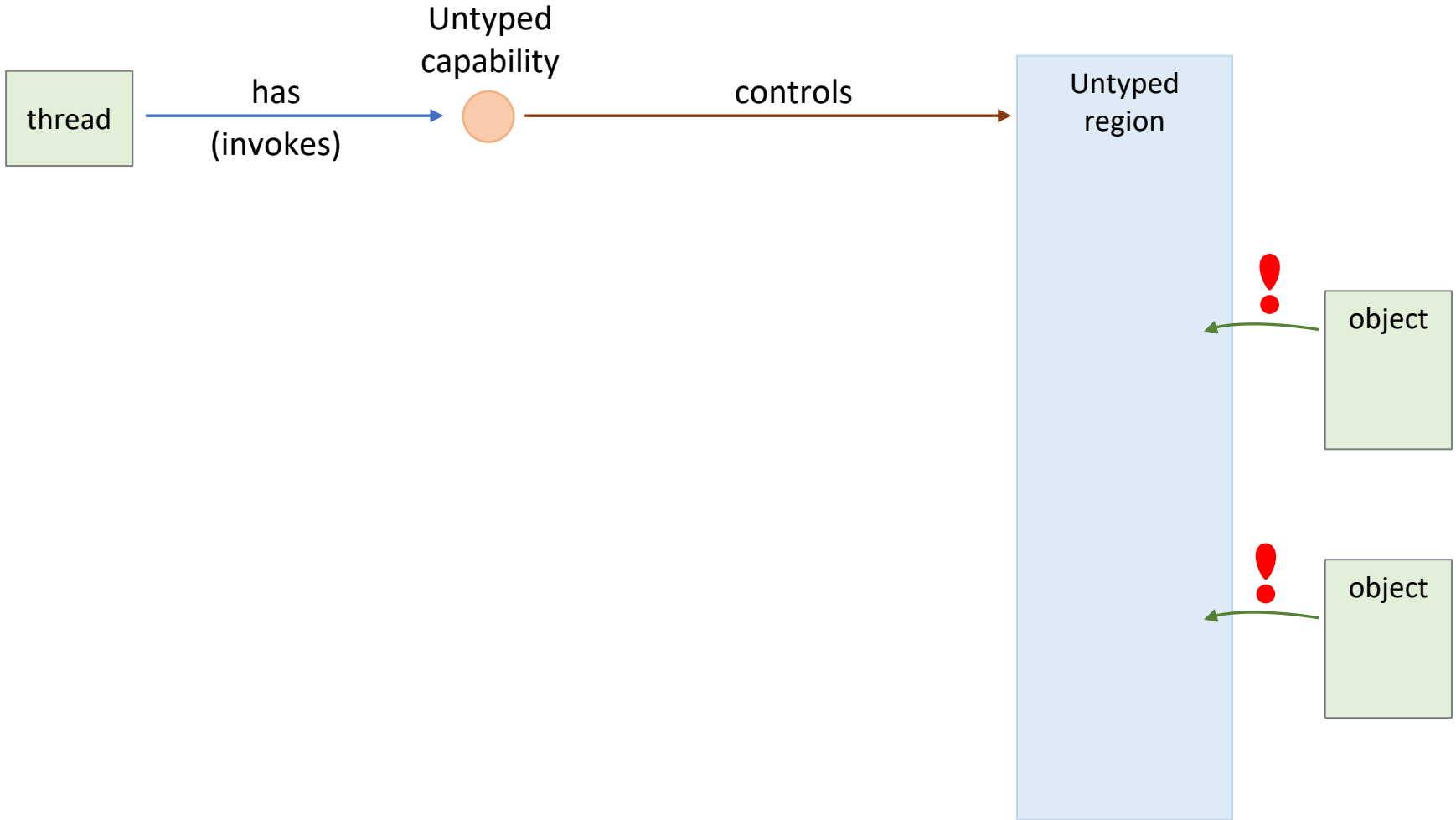


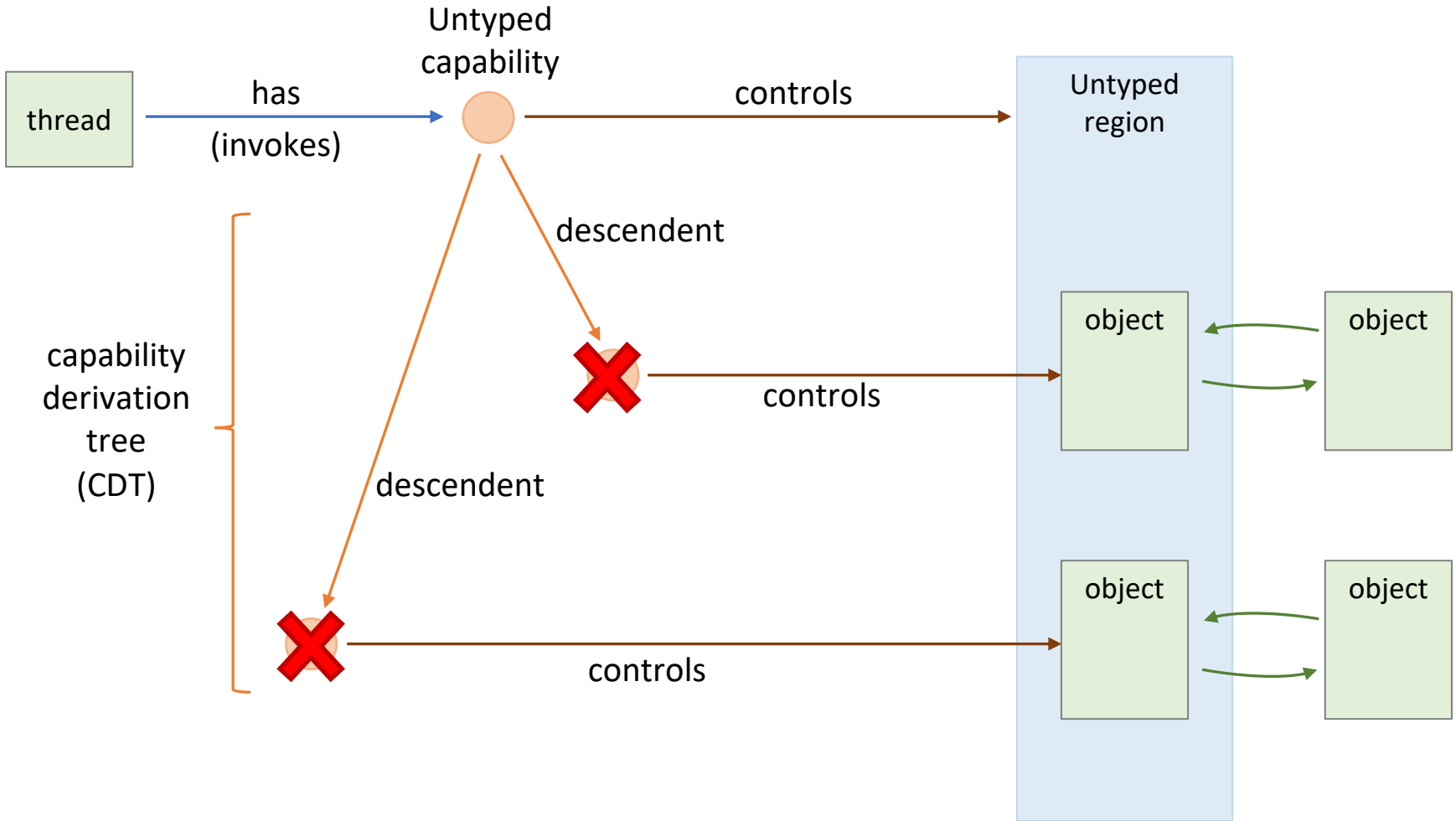


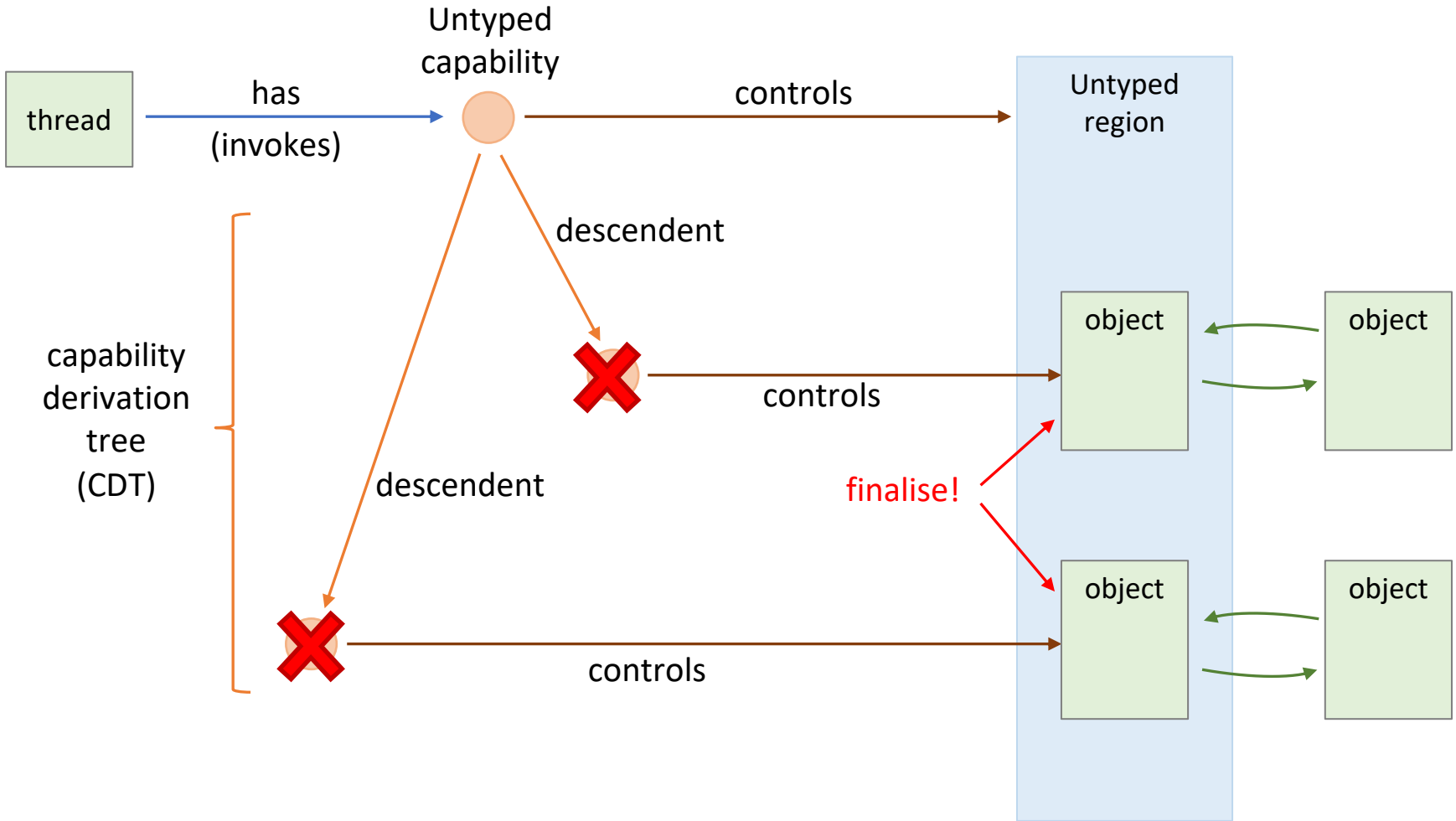


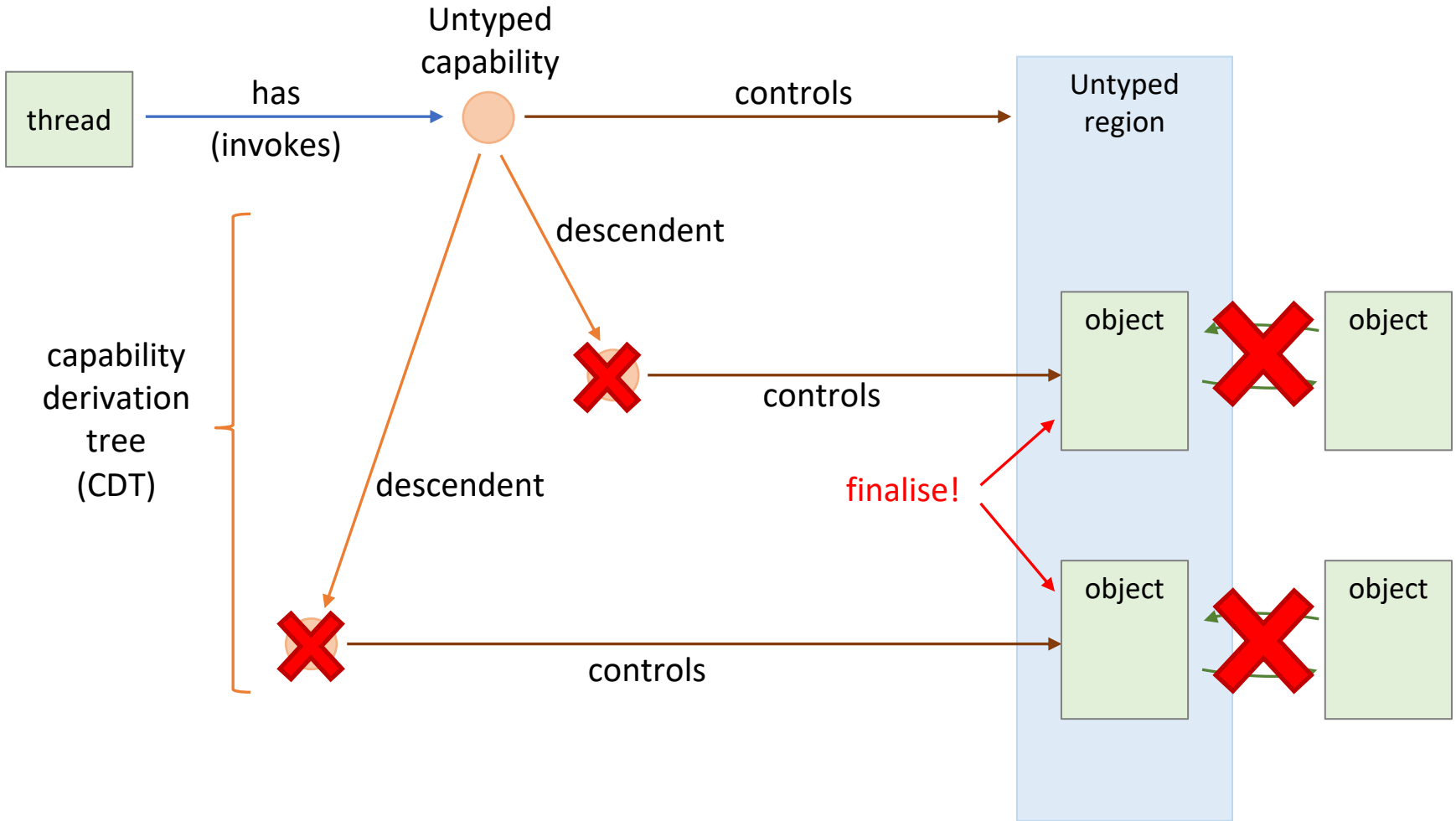


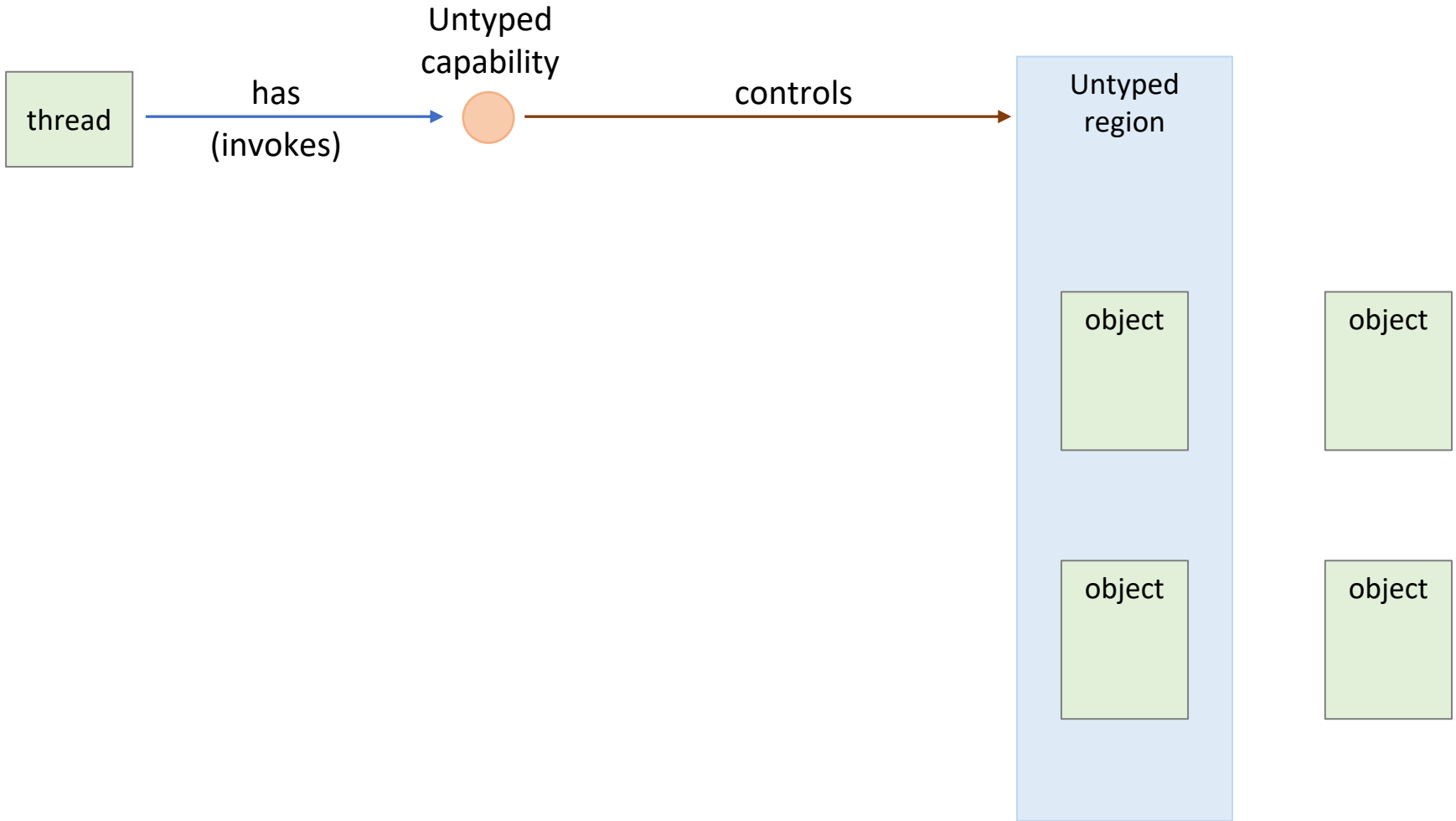


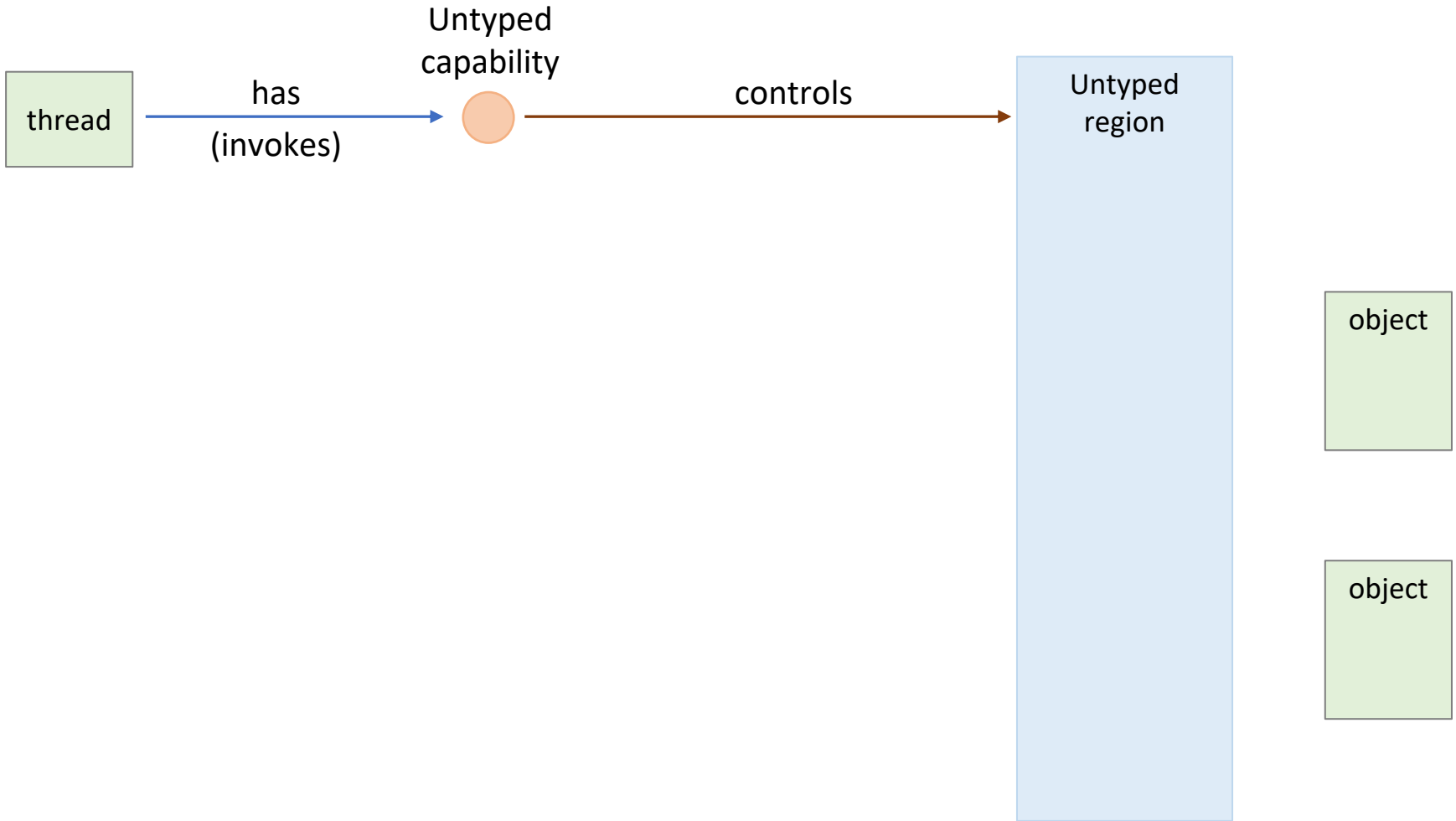


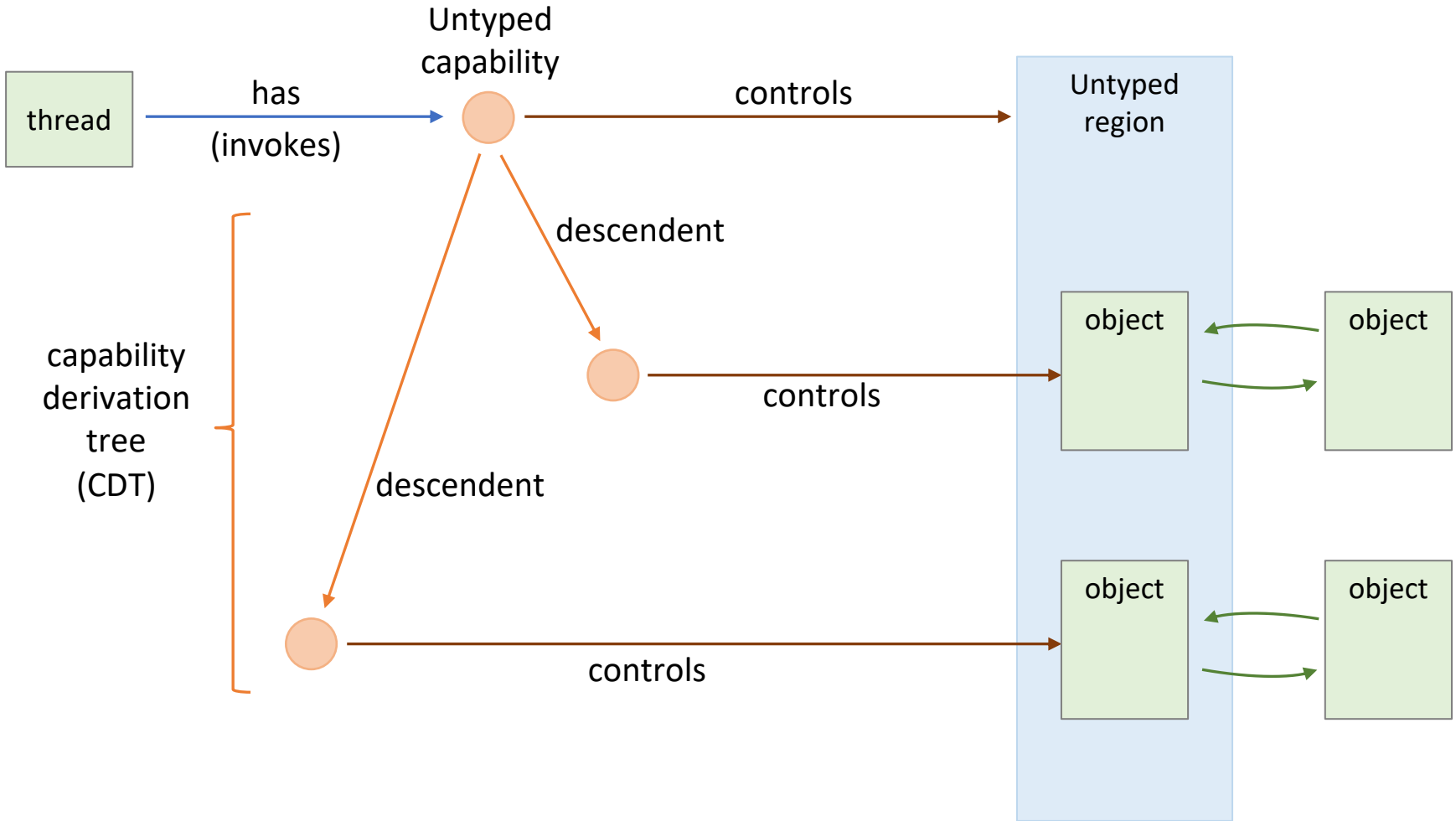


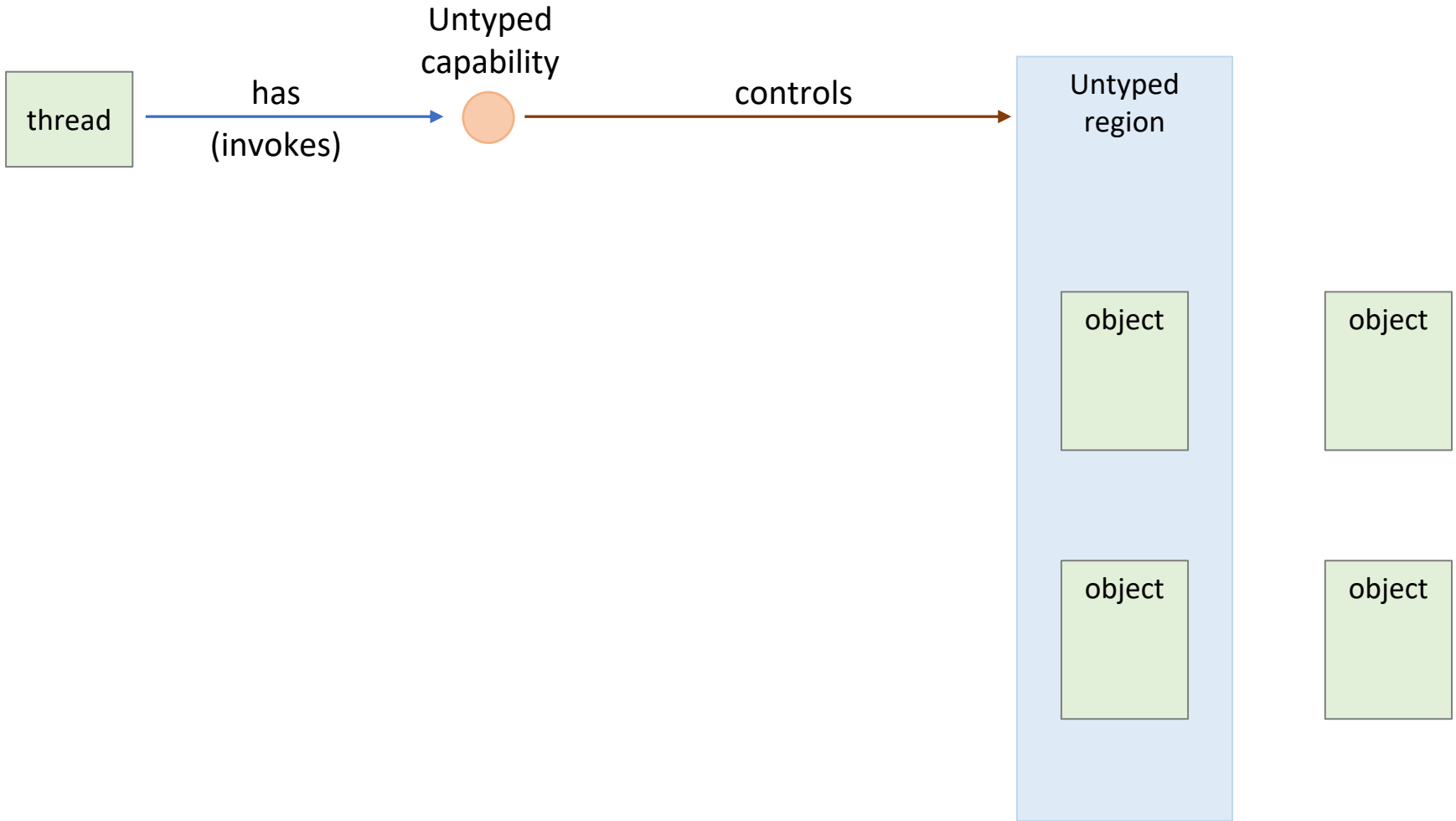


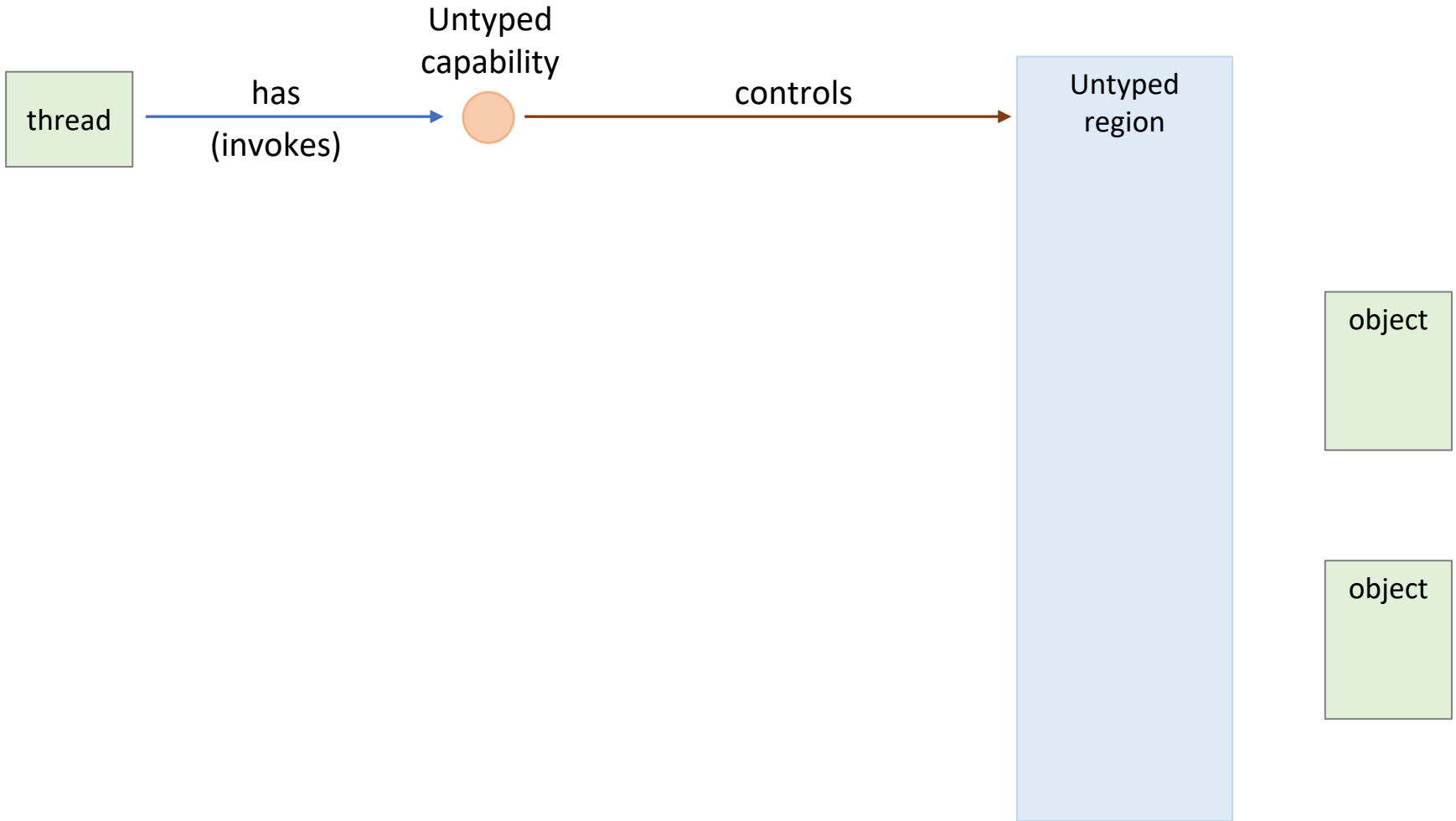


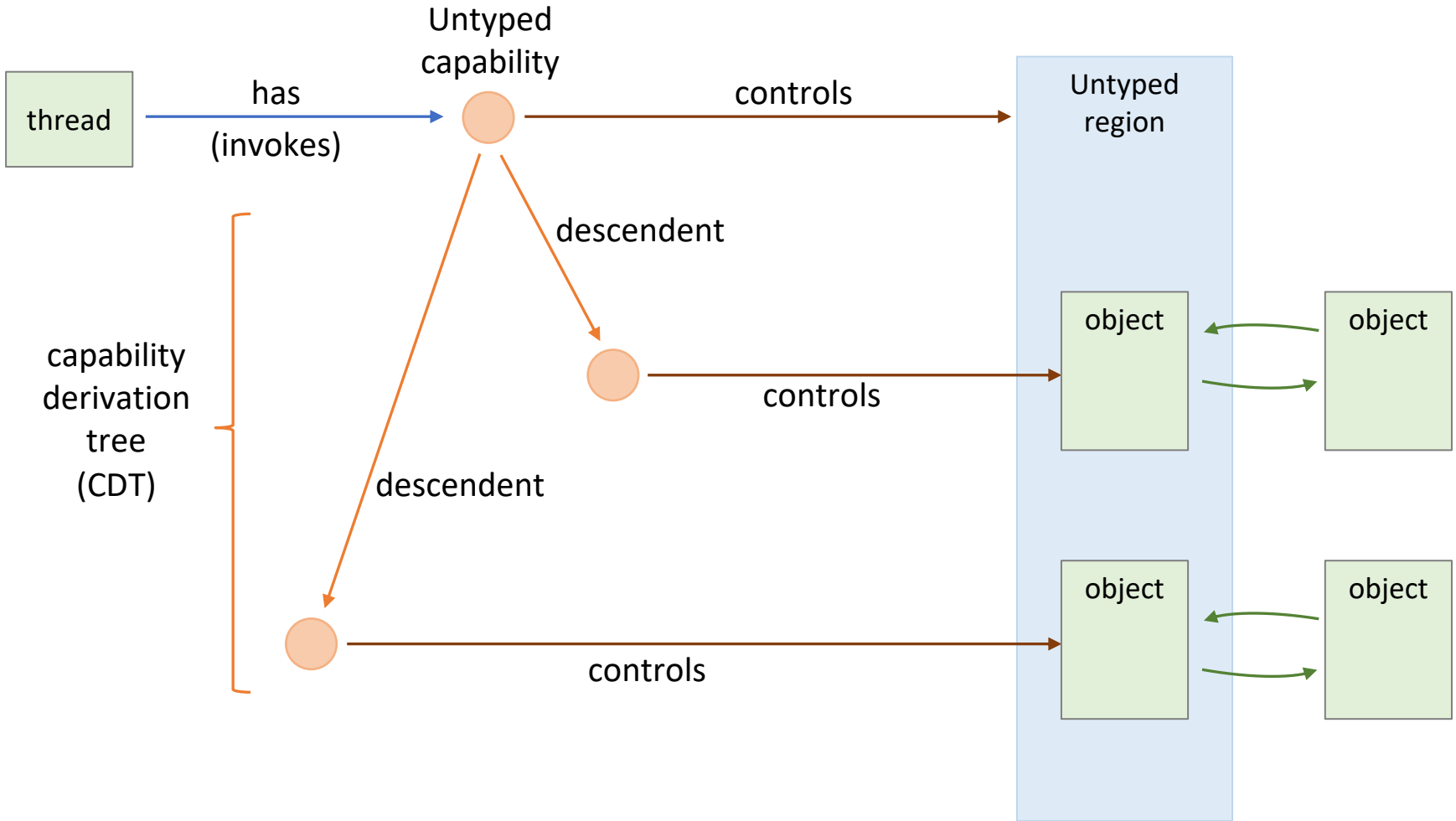












Proving `if_live_then_nonz_cap`:

- Object becomes live
 - need a capability
 - from user via invocation
- Delete a cap
 - need to show not live
 - show finalization achieves this

Summary:

- Specifications are like programs, but:
 - in a logical language with formal semantics
 - optimised for clarity
- Invariants
 - a coherent collection of interdependent properties
 - extremely effective at forcing understanding