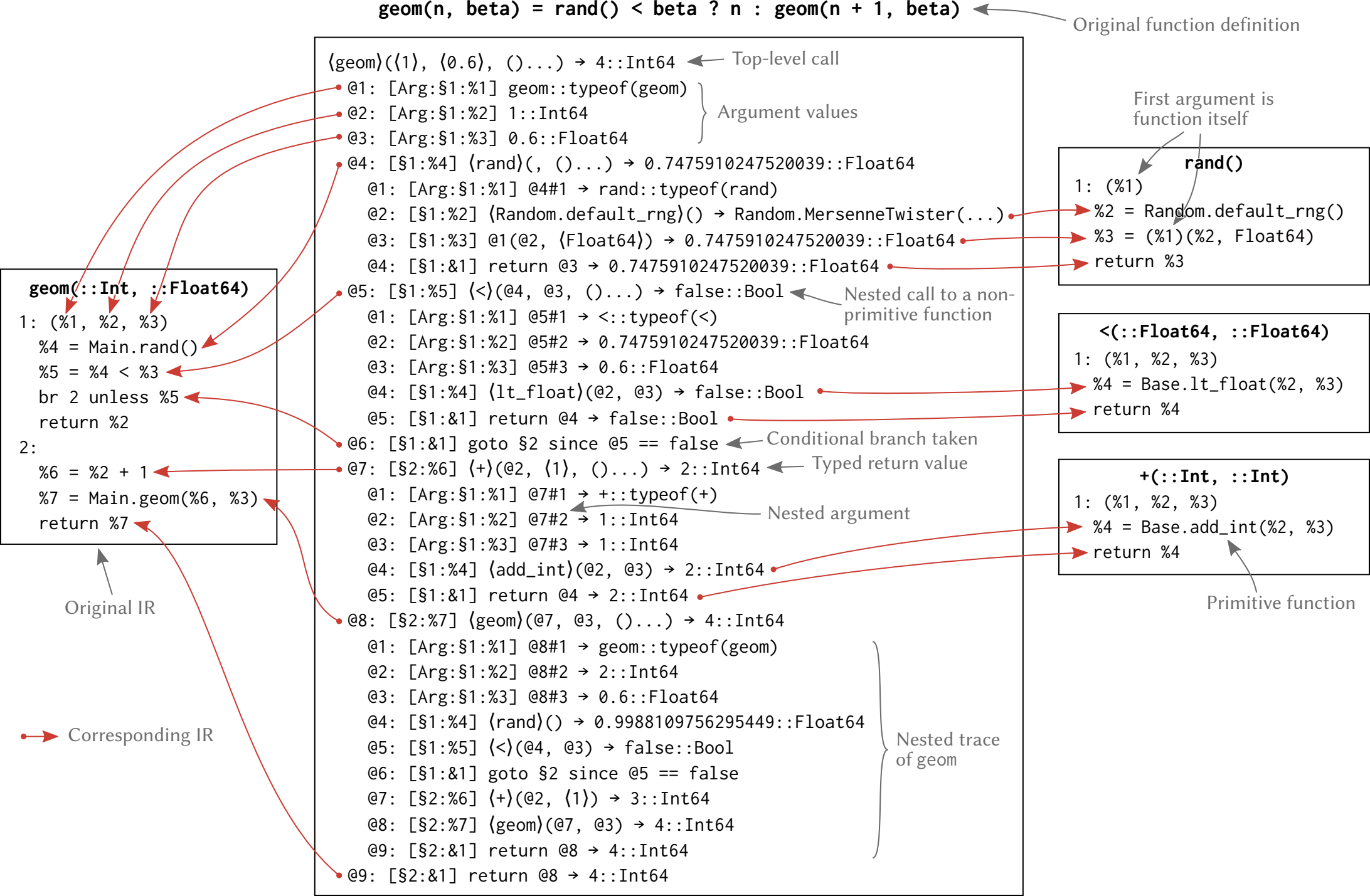
geom(n, beta) = rand() < beta ? n : geom(n + 1, beta)
Original function definition
⟨geom⟩(⟨1⟩, ⟨0.6⟩, ()...) → 4::Int64
Top-level call
@1: [Arg:§1:%1] geom::typeof(geom)
@2: [Arg:§1:%2] 1::Int64
@3: [Arg:§1:%3] 0.6::Float64
Argument values
@4: [§1:%4] ⟨rand⟩(, ()...) → 0.7475910247520039::Float64
@1: [Arg:§1:%1] @4#1 → rand::typeof(rand)
@2: [§1:%2] ⟨Random.default_rng⟩() → Random.MersenneTwister(...)
@3: [§1:%3] @1(@2, ⟨Float64⟩) → 0.7475910247520039::Float64
@4: [§1:&1] return @3 → 0.7475910247520039::Float64
@5: [§1:%5] ⟨<⟩(@4, @3, ()...) → false::Bool
Nested call to a non-primitive function
@1: [Arg:§1:%1] @5#1 → <::typeof(<)
@2: [Arg:§1:%2] @5#2 → 0.7475910247520039::Float64
@3: [Arg:§1:%3] @5#3 → 0.6::Float64
@4: [§1:%4] ⟨lt_float⟩(@2, @3) → false::Bool
@5: [§1:&1] return @4 → false::Bool
@6: [§1:&1] goto §2 since @5 == false
Conditional branch taken
@7: [§2:%6] ⟨+⟩(@2, ⟨1⟩, ()...) → 2::Int64
Typed return value
@1: [Arg:§1:%1] @7#1 → +::typeof(+)
@2: [Arg:§1:%2] @7#2 → 1::Int64
Nested argument
@3: [Arg:§1:%3] @7#3 → 1::Int64
@4: [§1:%4] ⟨add_int⟩(@2, @3) → 2::Int64
@5: [§1:&1] return @4 → 2::Int64
@8: [§2:%7] ⟨geom⟩(@7, @3, ()...) → 4::Int64
@1: [Arg:§1:%1] @8#1 → geom::typeof(geom)
@2: [Arg:§1:%2] @8#2 → 2::Int64
@3: [Arg:§1:%3] @8#3 → 0.6::Float64
@4: [§1:%4] ⟨rand⟩() → 0.9988109756295449::Float64
@5: [§1:%5] ⟨<⟩(@4, @3) → false::Bool
@6: [§1:&1] goto §2 since @5 == false
@7: [§2:%6] ⟨+⟩(@2, ⟨1⟩) → 3::Int64
@8: [§2:%7] ⟨geom⟩(@7, @3) → 4::Int64
@9: [§2:&1] return @8 → 4::Int64
Nested trace of geom
@9: [§2:&1] return @8 → 4::Int64
geom(::Int, ::Float64)
1: (%1, %2, %3)
%4 = Main.rand()
%5 = %4 < %3
br 2 unless %5
return %2
2:
%6 = %2 + 1
%7 = Main.geom(%6, %3)
return %7
Original IR
Corresponding IR
First argument is function itself
rand()
1: (%1)
%2 = Random.default_rng()
%3 = (%1)(%2, Float64)
return %3
<(::Float64, ::Float64)
1: (%1, %2, %3)
%4 = Base.lt_float(%2, %3)
return %4
+(::Int, ::Int)
1: (%1, %2, %3)
%4 = Base.add_int(%2, %3)
return %4
Primitive function