Certificate Translation for Specification Preserving Advices

Gilles Barthe and César Kunz
INRIA Sophia Antipolis - Méditerranée

FOAL 2008

MOTIVATION

SPECIFICATION PRESERVING ADVICES

PROVING SPECIFICATION PRESERVING ADVICES

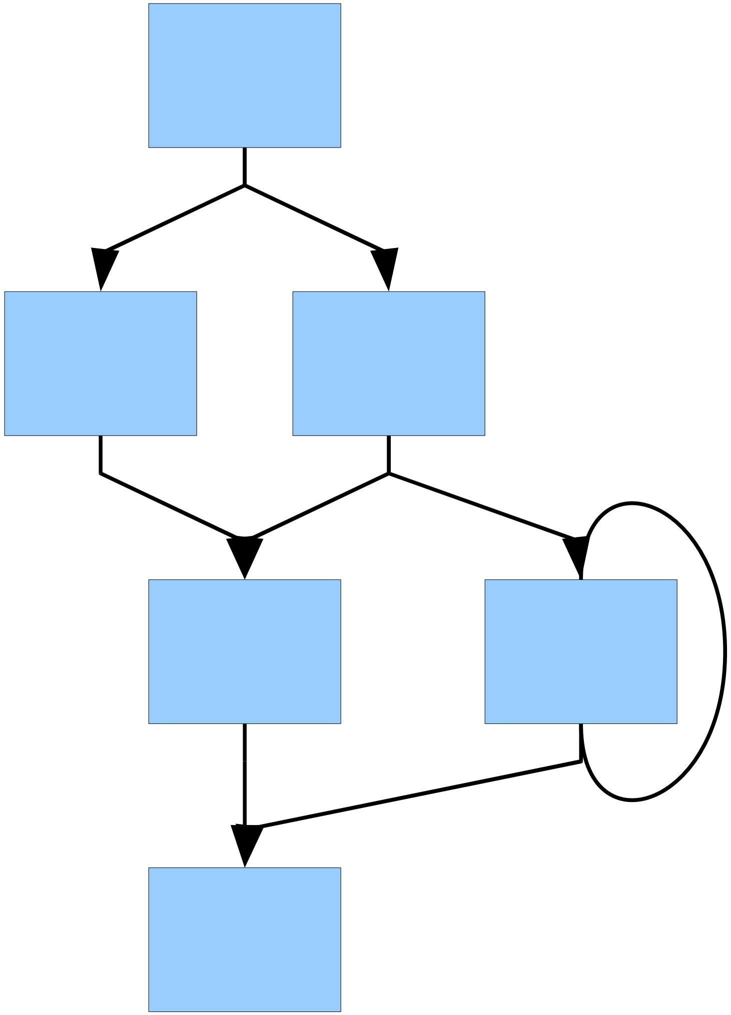
REDUCING PROOF OBLIGATIONS

IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

Local reasoning on:

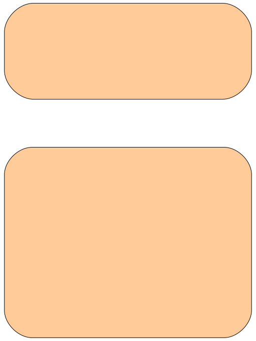
- Baseline Code (to understand main functionality)



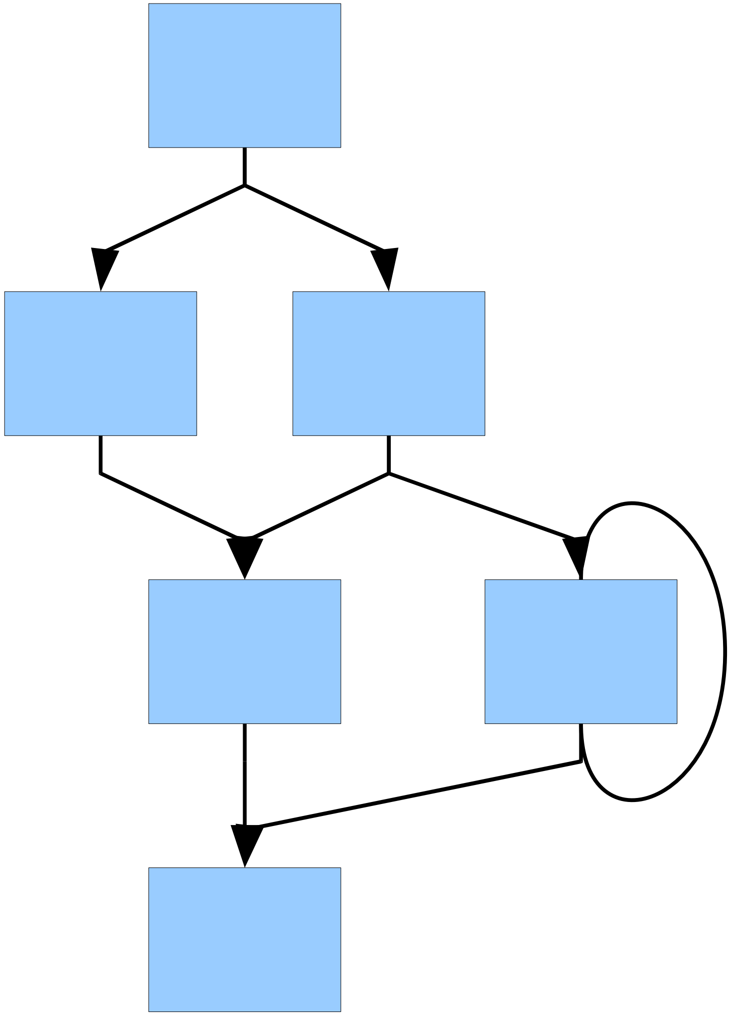
Local reasoning on:

- Baseline Code (to understand main functionality)

- Advice Code
(to understand the implemented aspect



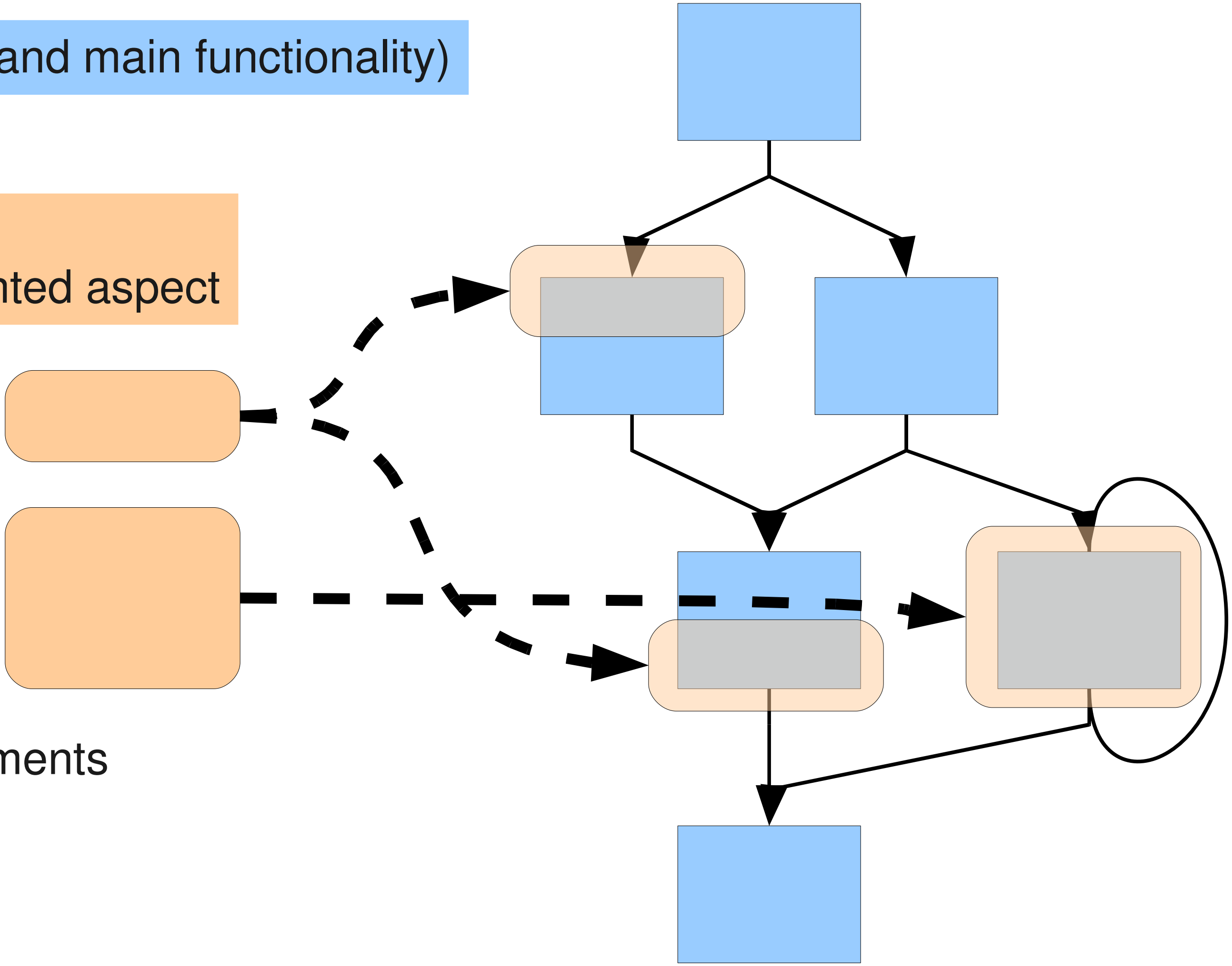
Incremental concerns:
- Contract enforcement
- Logging / Profiling
- Evolving Security Requirements



Local reasoning on:

- Baseline Code (to understand main functionality)

- Advice Code
(to understand the implemented aspect



Incremental concerns:
- Contract enforcement
- Logging / Profiling
- Evolving Security Requirements

Global analysis of pointcuts to understand interaction of aspects

Producer vs Consumer Perspective

Obliviousness -> Local Reasoning?

Syntactic Obliviousness
is not enough

Syntactic Obliviousness
vs.
Semantic Obliviousness

Dantas & Walker [POPL06]:

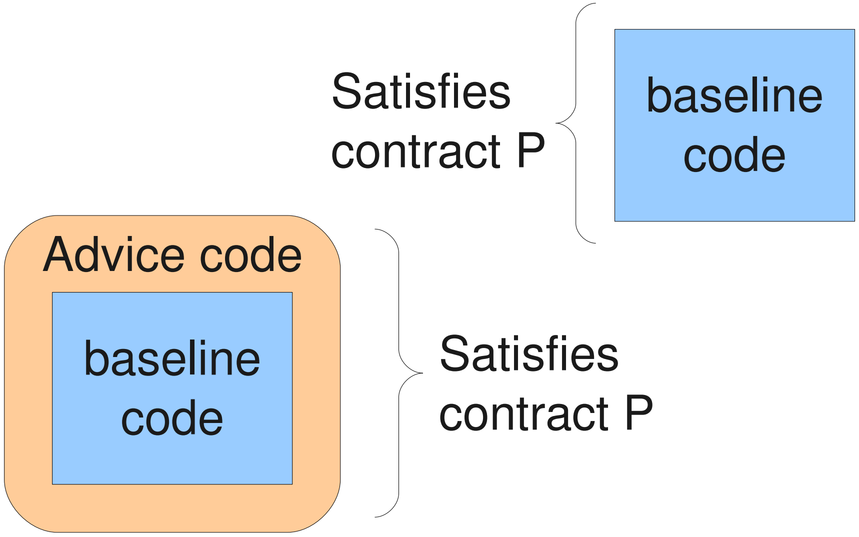
- characterize *Harmless Advices* that allow local reasoning
- information flow analysis to check advice non-interference.

PCC setting: contract enforcement

- functional properties (logic formulae)
- Absence of null pointer access
- Type Safety, etc.

Contract preserv. vs semantic preserv

weaker requirement



Satisfies
contract P

baseline
code

Advice code

baseline
code

Satisfies
contract P

MOTIVATION

SPECIFICATION PRESERVING ADVICES

PROVING SPECIFICATION PRESERVING ADVICES

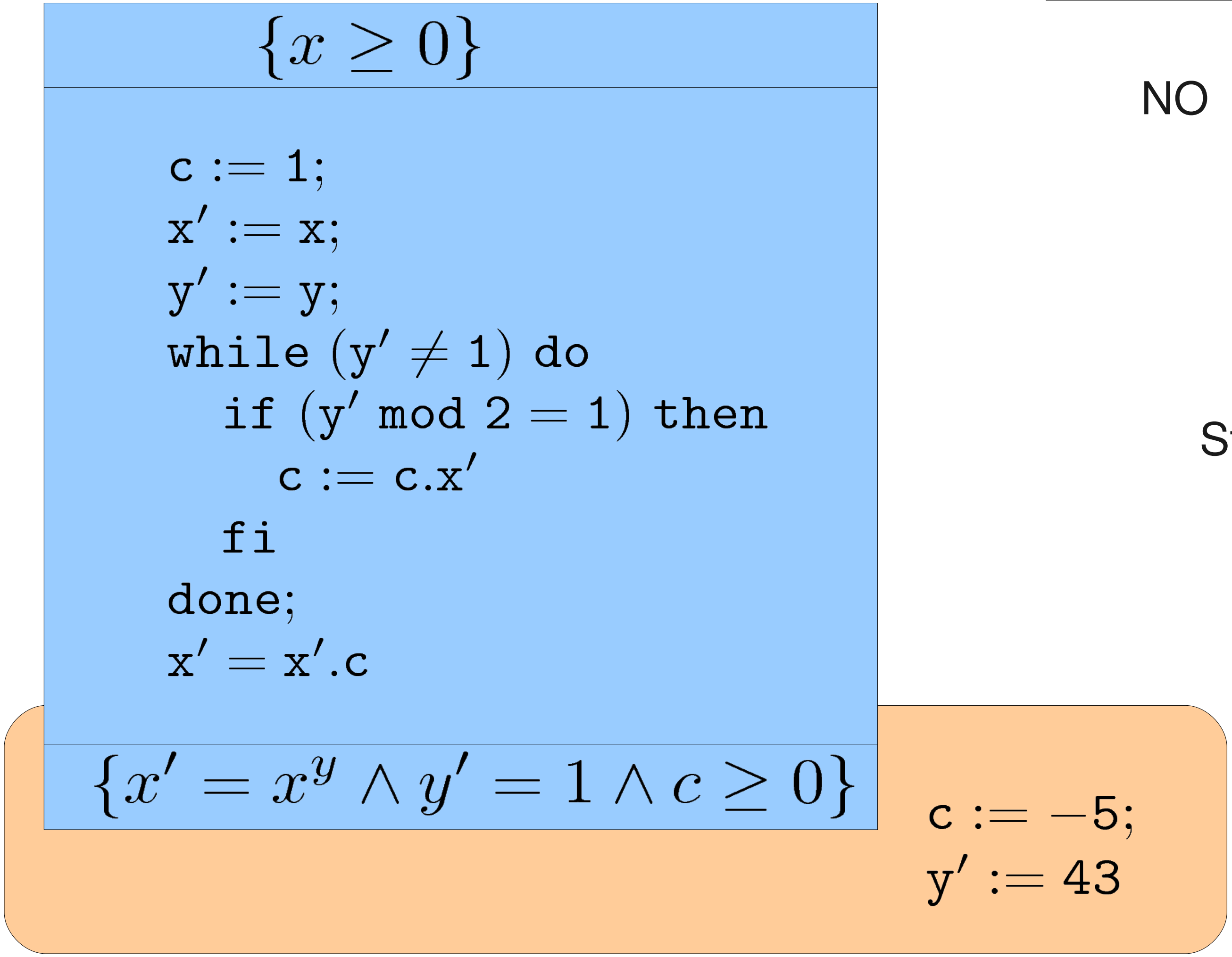
REDUCING PROOF OBLIGATIONS

IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

Specification Preserving Advices

Harmless	Spec. preserving
NO	NO



```
$\{x \geq 0\}$

c := 1;
x' := x;
y' := y;
while (y' ≠ 1) do
   if (y' mod 2 = 1) then
      c := c.x'
   fi
done;
x' = x'.c

$\{x' = x^y \wedge y' = 1 \wedge c \geq 0\}$
```

```
c := −5;
y' := 43
```

Strong specification

Specification Preserving Advices

Harmless	Spec. preserving
NO	YES

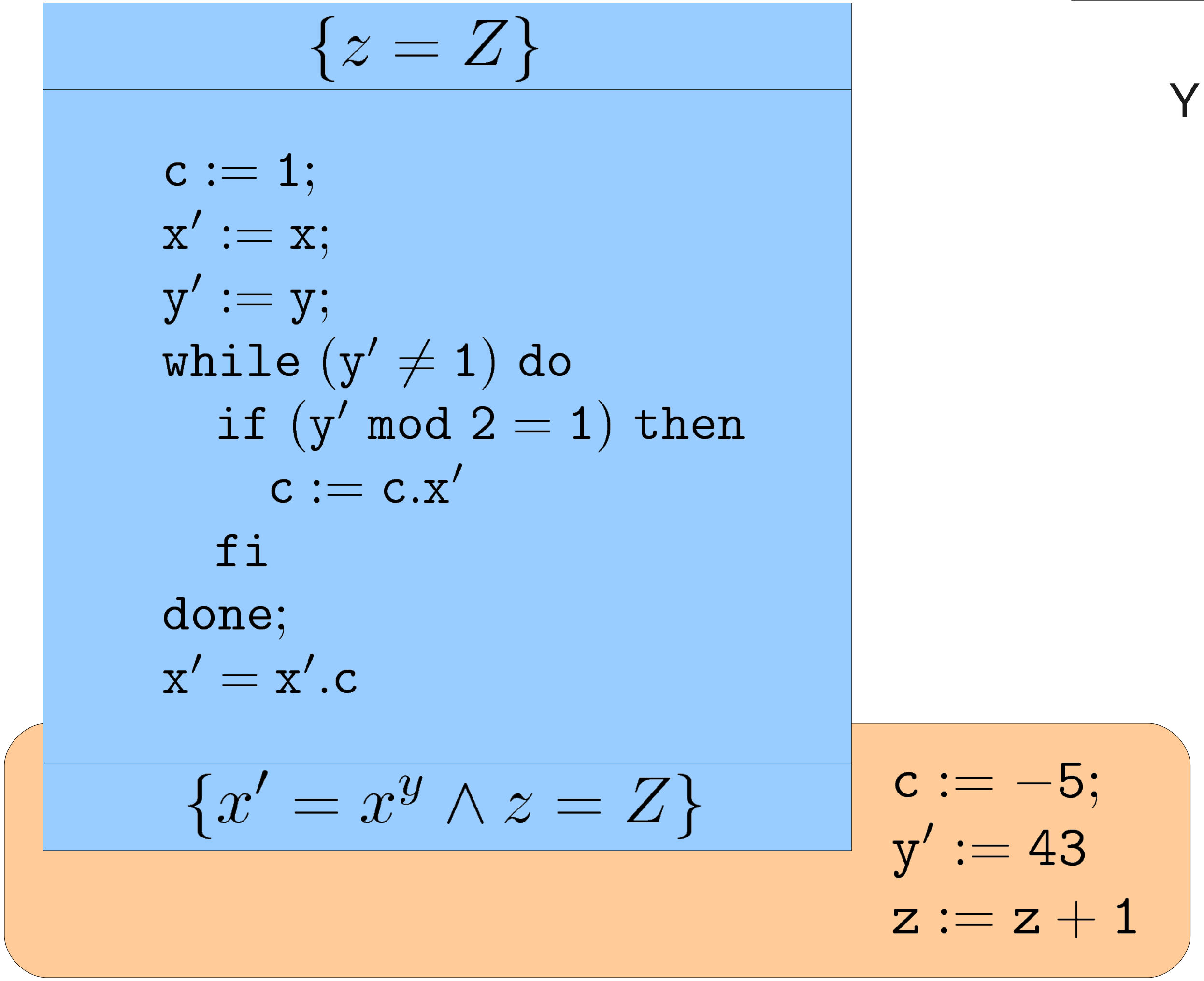
```
{True}

c := 1;
x' := x;
y' := y;
while (y' ≠ 1) do
   if (y' mod 2 = 1) then
      c := c.x'
   fi
done;
x' = x'.c

{$x' = x^y$}
```

```
c := −5;
y' := 43
```

Specification Preserving Advices



$\{z = Z\}$

```
c := 1;
x' := x;
y' := y;
while (y' ≠ 1) do
   if (y' mod 2 = 1) then
      c := c.x'
   fi
done;
x' = x'.c
```

$\{x' = x^y \wedge z = Z\}$

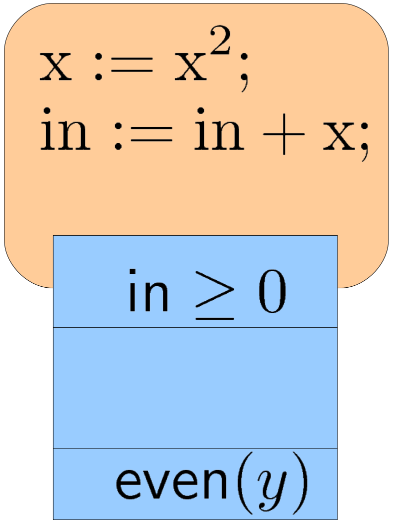
```
c := −5;
y' := 43
z := z + 1
```

Harmless	Spec. preserving
YES	NO

Specification Preserving Advices

A specification preserving advice may modify variables in the specification.

Harmless	Spec. preserving
NO	YES



$\mathrm{x} := \mathrm{x}^2;$
$\mathrm{in} := \mathrm{in} + \mathrm{x};$

in ≥ 0

even(y)

- Output value may differ
- in ≥ 0 is not invalidated.
- even(y) is ensured.

MOTIVATION

SPECIFICATION PRESERVING ADVICES

PROVING SPECIFICATION PRESERVING ADVICES

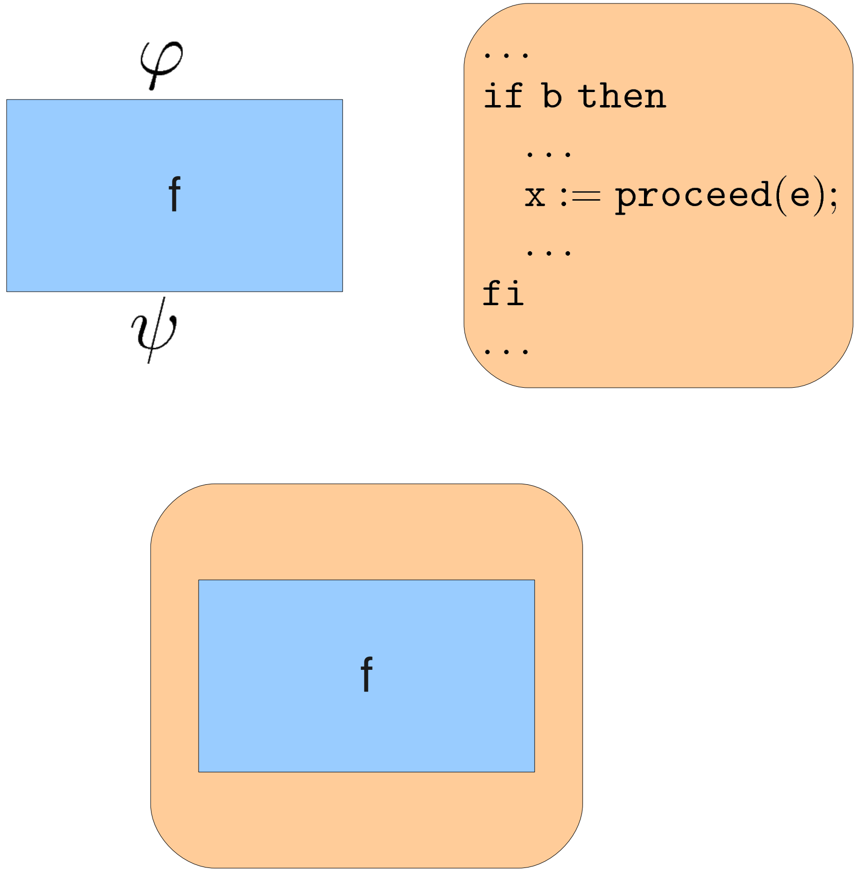
REDUCING PROOF OBLIGATIONS

IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

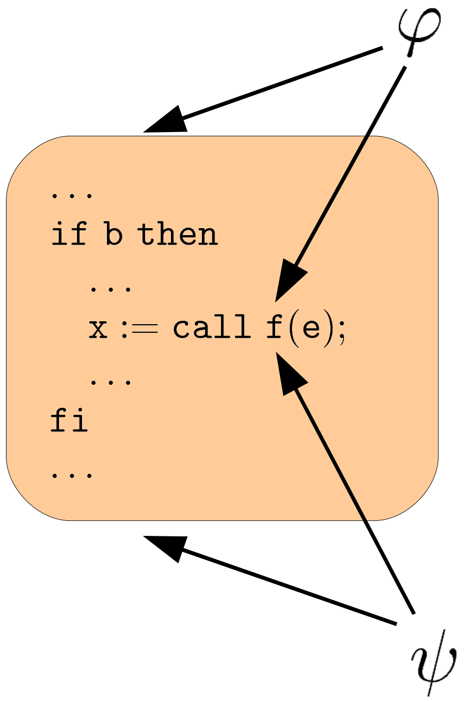
Proving spec-preservation

Baseline Code Verification: wp-based Vcgen



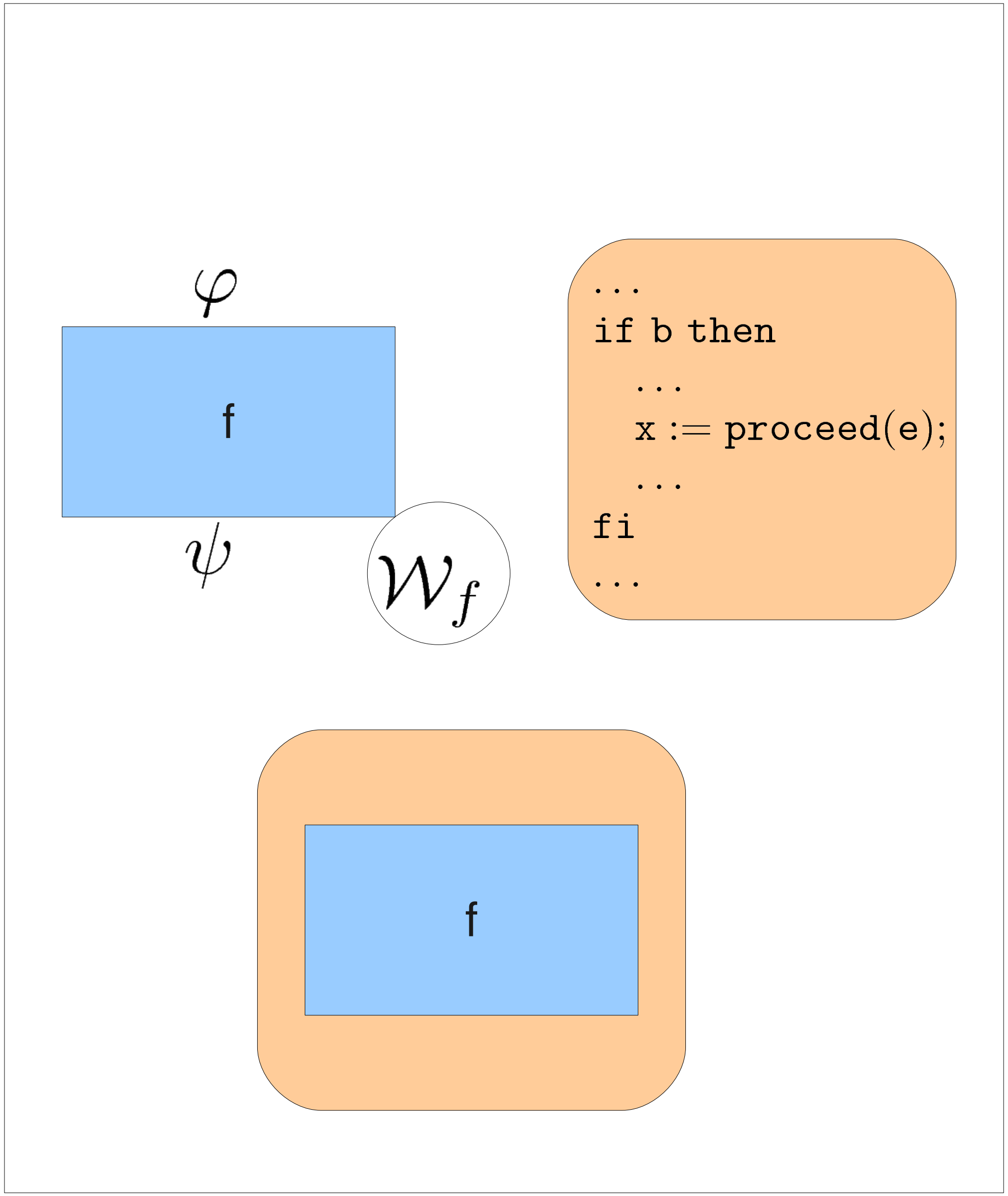
```
...
if b then
  ...
  x := proceed(e);
  ...
fi
...
```

Verification of spec. preservation: wp-based Vcgen over modified advice code.

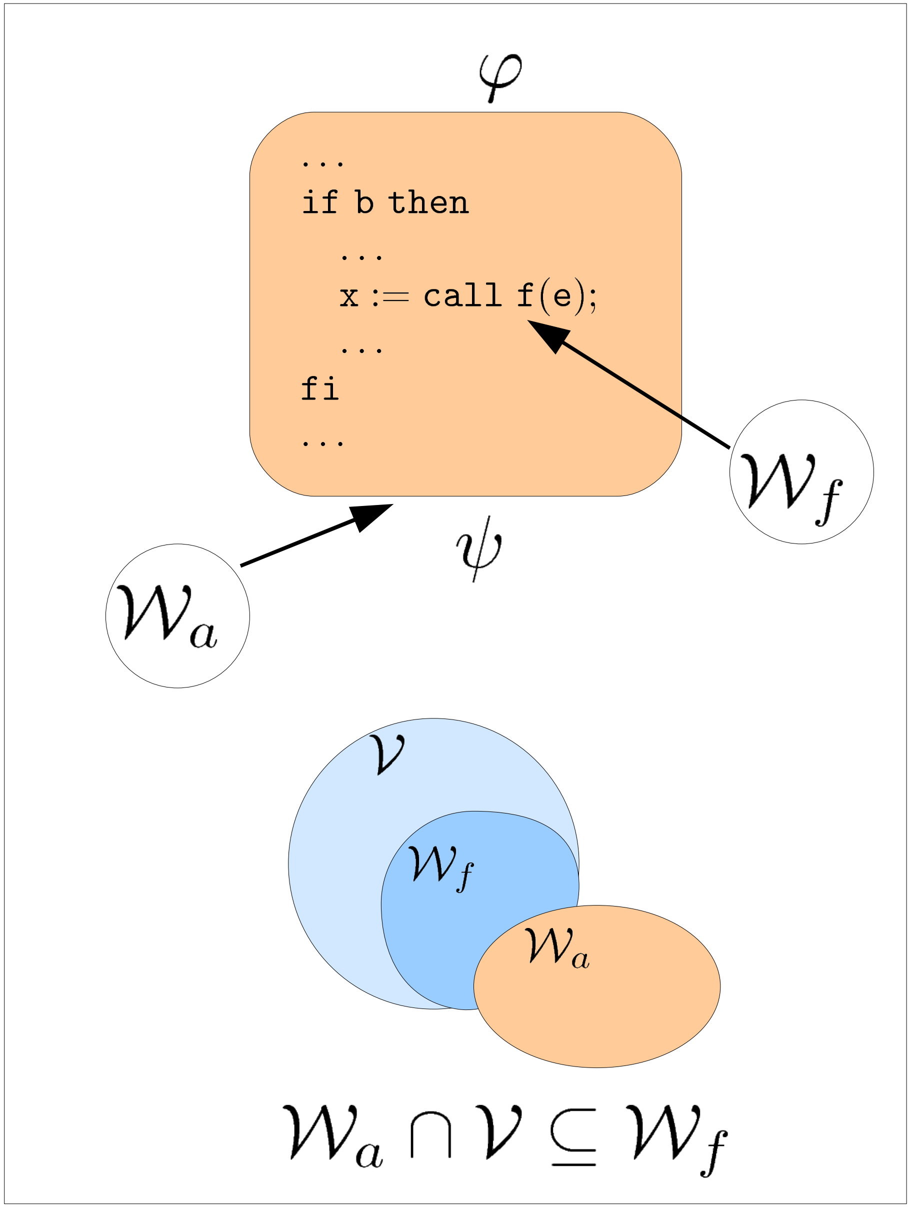


```
...
if b then
  ...
  x := call f(e);
  ...
fi
...
```

Proving spec-preservation



```
...
if b then
  ...
  x := proceed(e);
  ...
fi
...
```



```
...
if b then
  ...
  x := call f(e);
  ...
fi
...
```

$$\mathcal{W}_a \cap \mathcal{V} \subseteq \mathcal{W}_f$$

MOTIVATION

SPECIFICATION PRESERVING ADVICES

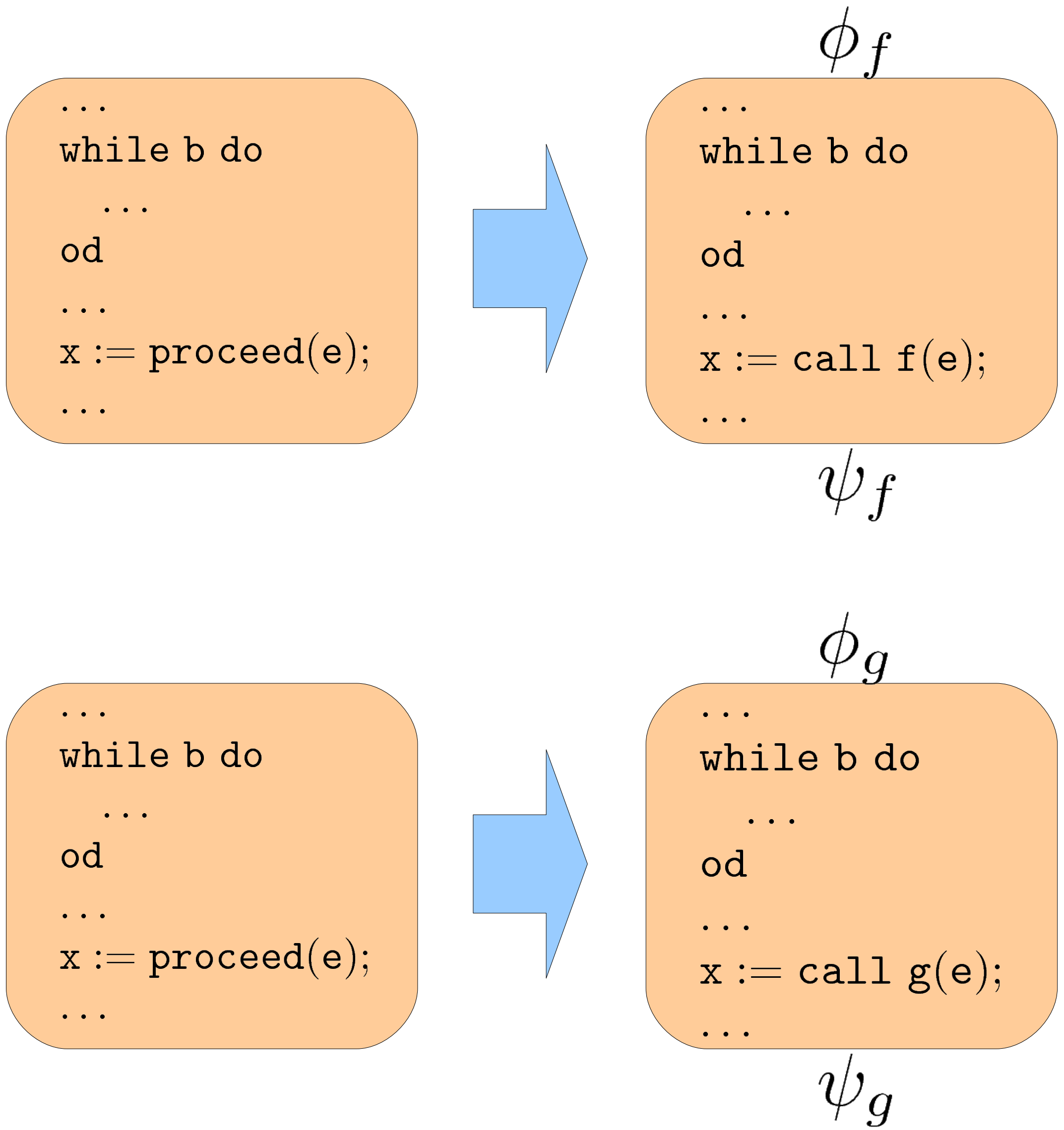
PROVING SPECIFICATION PRESERVING ADVICES

REDUCING PROOF OBLIGATIONS

IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

Specification Harmless Advices



$\{\phi_f\}\ f\ \{\psi_f\}$

```
...
while b do
  ...
od
...
x := proceed(e);
...
```

ϕ_f

```
...
while b do
  ...
od
...
x := call f(e);
...
```

ψ_f

$\{\phi_g\}\ g\ \{\psi_g\}$

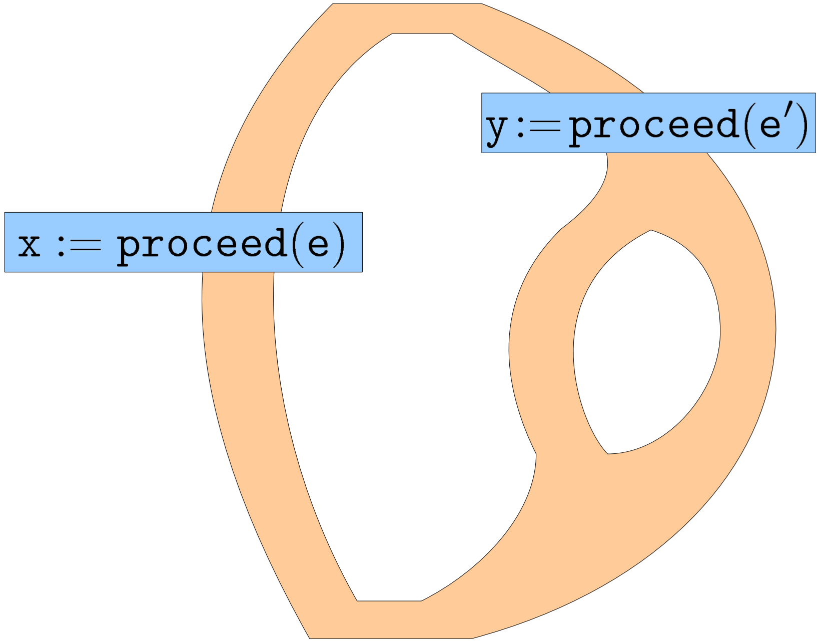
```
...
while b do
  ...
od
...
x := proceed(e);
...
```

ϕ_g

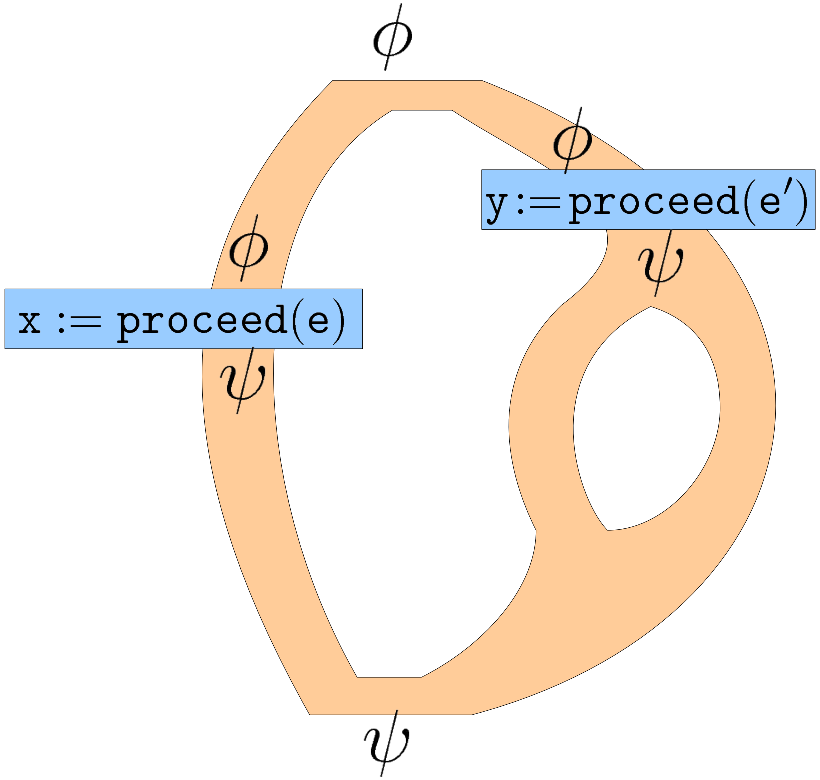
```
...
while b do
  ...
od
...
x := call g(e);
...
```

ψ_g

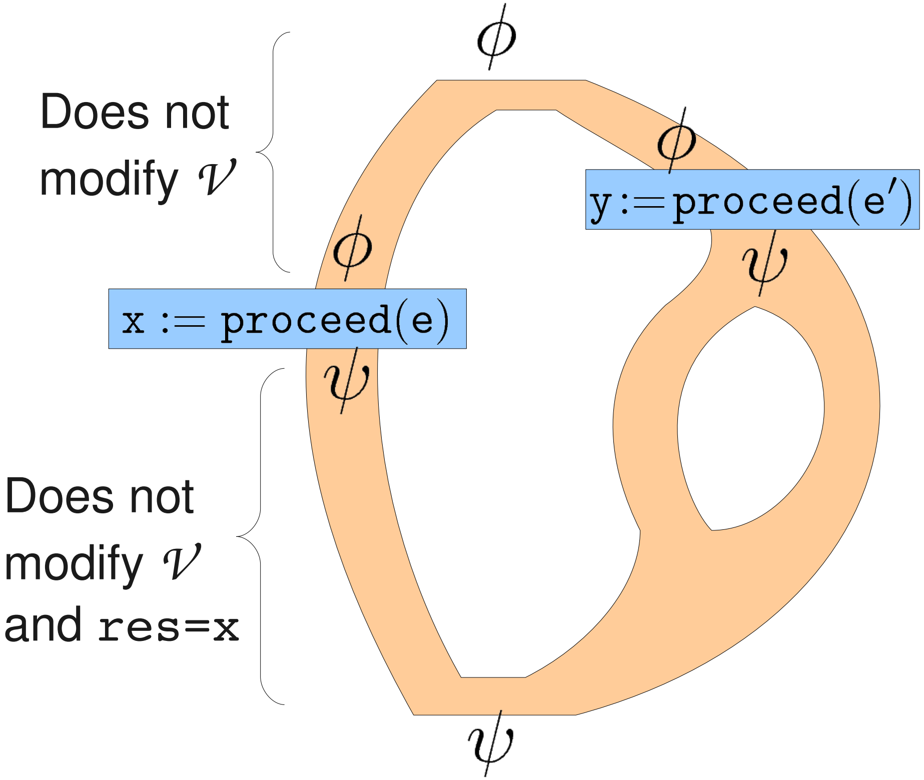
Specification Harmless Advices



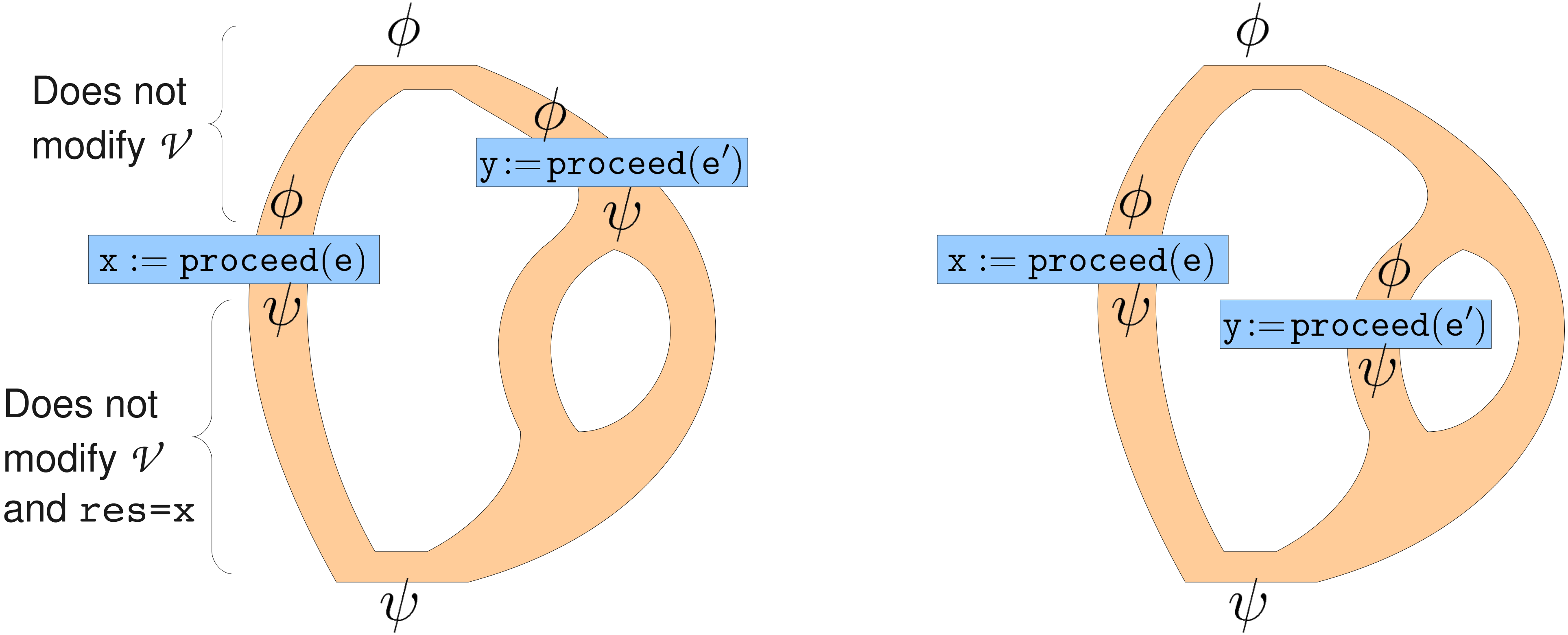
Specification Harmless Advices



Specification Harmless Advices


ϕ
Does not modify $\mathcal{V}$
ϕ
$\mathtt{y{:=}proceed(e')}$
ψ
ϕ
$\mathtt{x := proceed(e)}$
ψ
Does not modify $\mathcal{V}$ and $\mathtt{res{=}x}$
ψ

Specification Harmless Advices



Does not modify $\mathcal{V}$

Does not modify $\mathcal{V}$ and `res=x`

ϕ

ϕ

`x := proceed(e)`

ψ

ϕ

`y:=proceed(e')`

ψ

ψ

ϕ

ϕ

`x := proceed(e)`

ψ

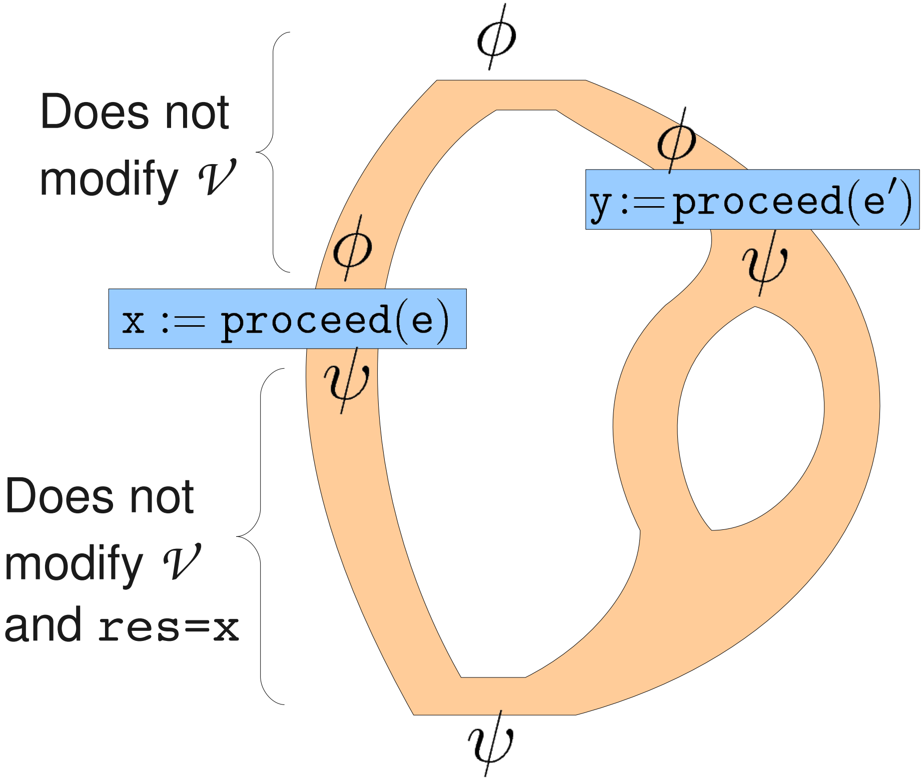
ϕ

`y:=proceed(e')`

ψ

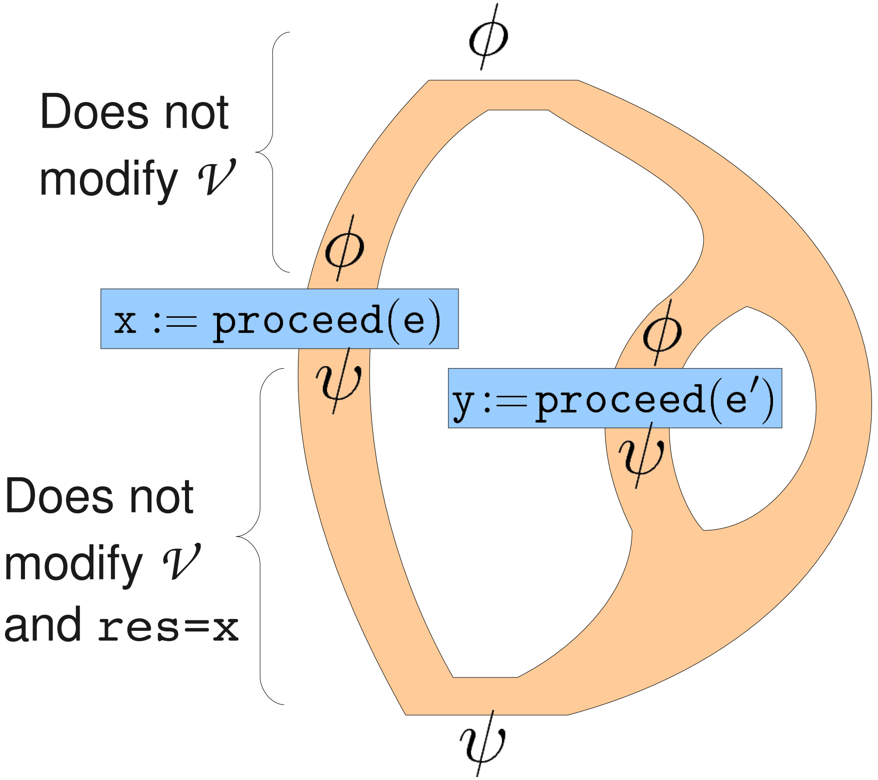
ψ

Specification Harmless Advices



Does not modify $\mathcal{V}$

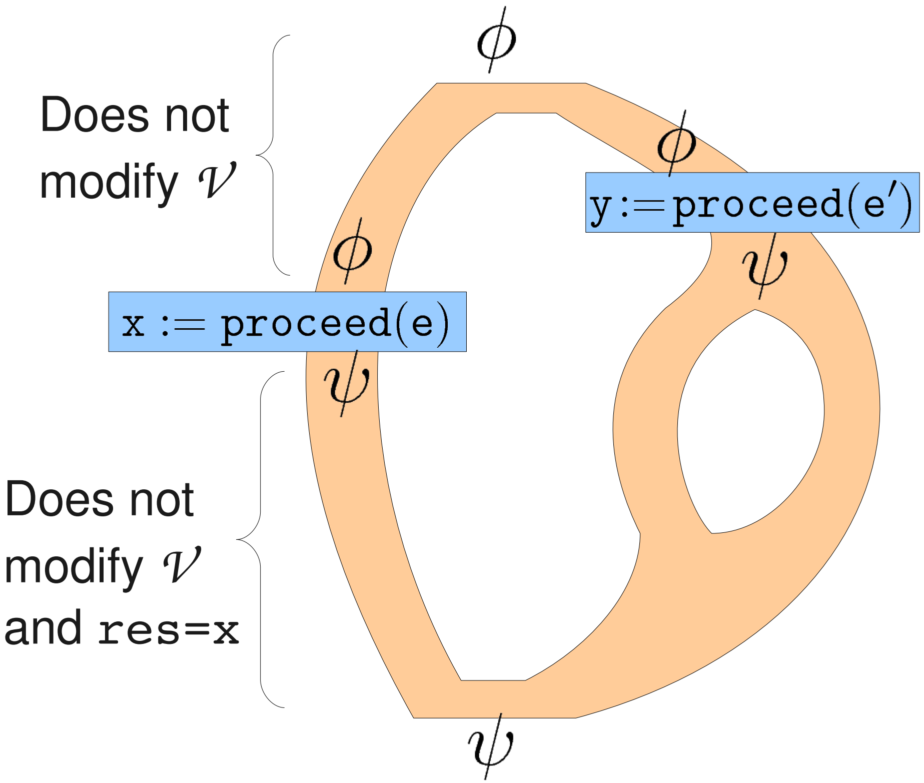
Does not modify $\mathcal{V}$ and `res=x`



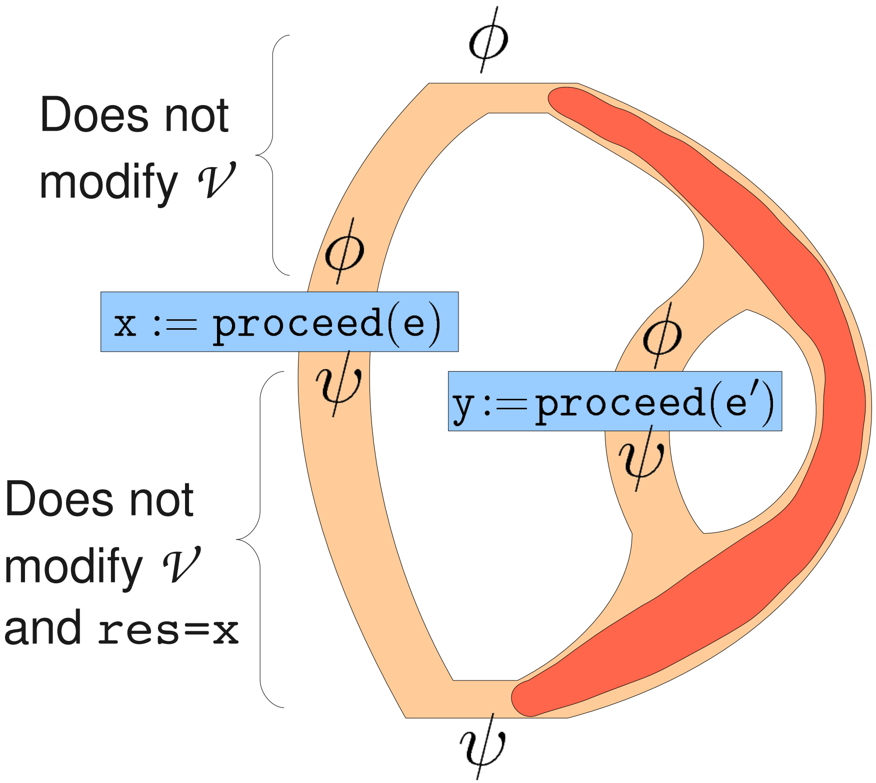
Does not modify $\mathcal{V}$

Does not modify $\mathcal{V}$ and `res=x`

Specification Harmless Advices


Does not modify $\mathcal{V}$

Does not modify $\mathcal{V}$ and `res=x`


Does not modify $\mathcal{V}$

Does not modify $\mathcal{V}$ and `res=x`

MOTIVATION

SPECIFICATION PRESERVING ADVICES

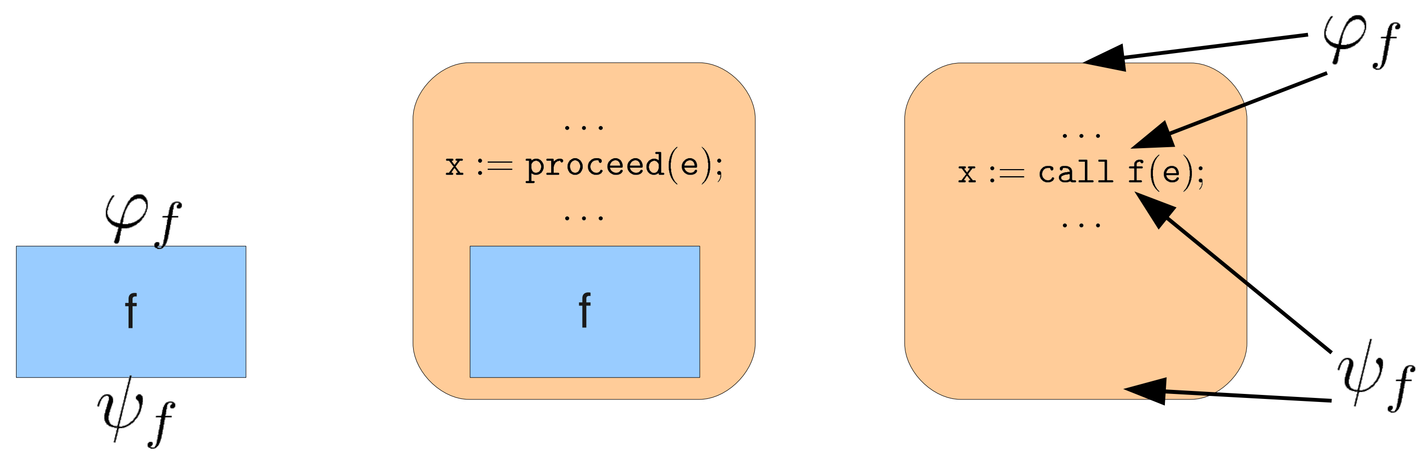
PROVING SPECIFICATION PRESERVING ADVICES

REDUCING PROOF OBLIGATIONS

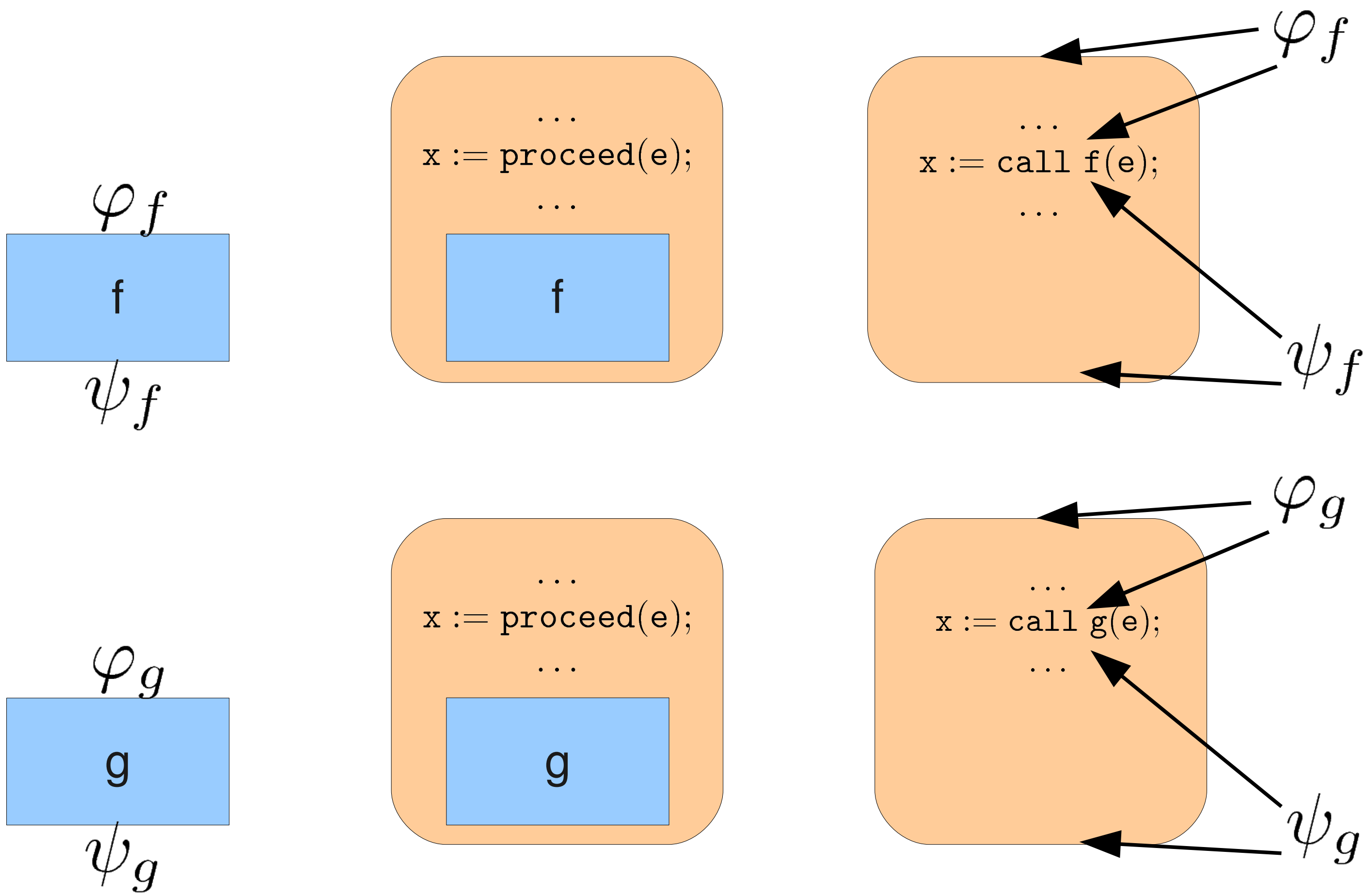
IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

IMPROVING THE VERIFICATION POWER



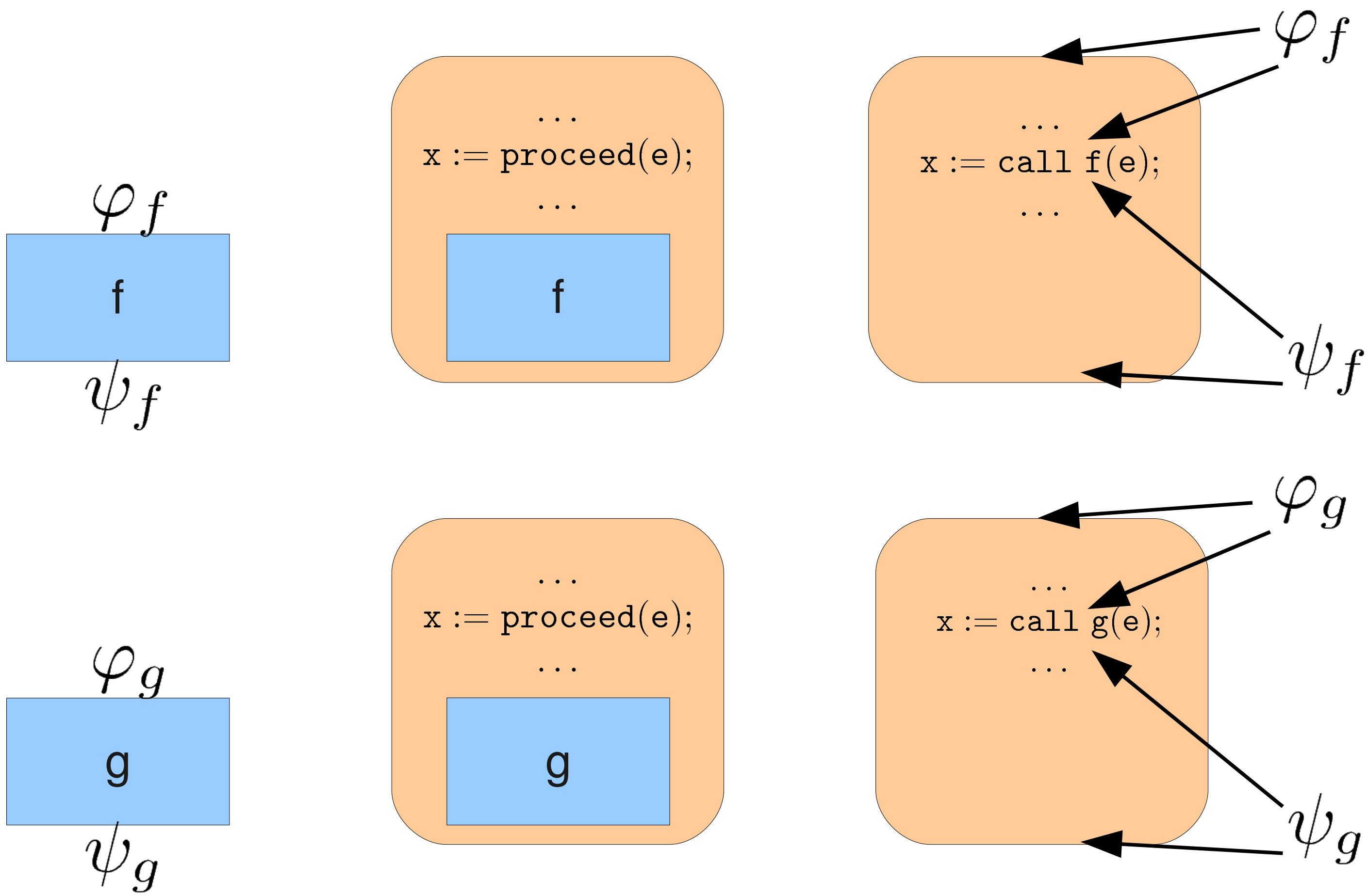
IMPROVING THE VERIFICATION POWER



IMPROVING THE VERIFICATION POWER

Drawback

Multiple advised procedures = multiple verification invariants.



```
...
x := proceed(e);
...
```

```
...
x := call f(e);
...
```

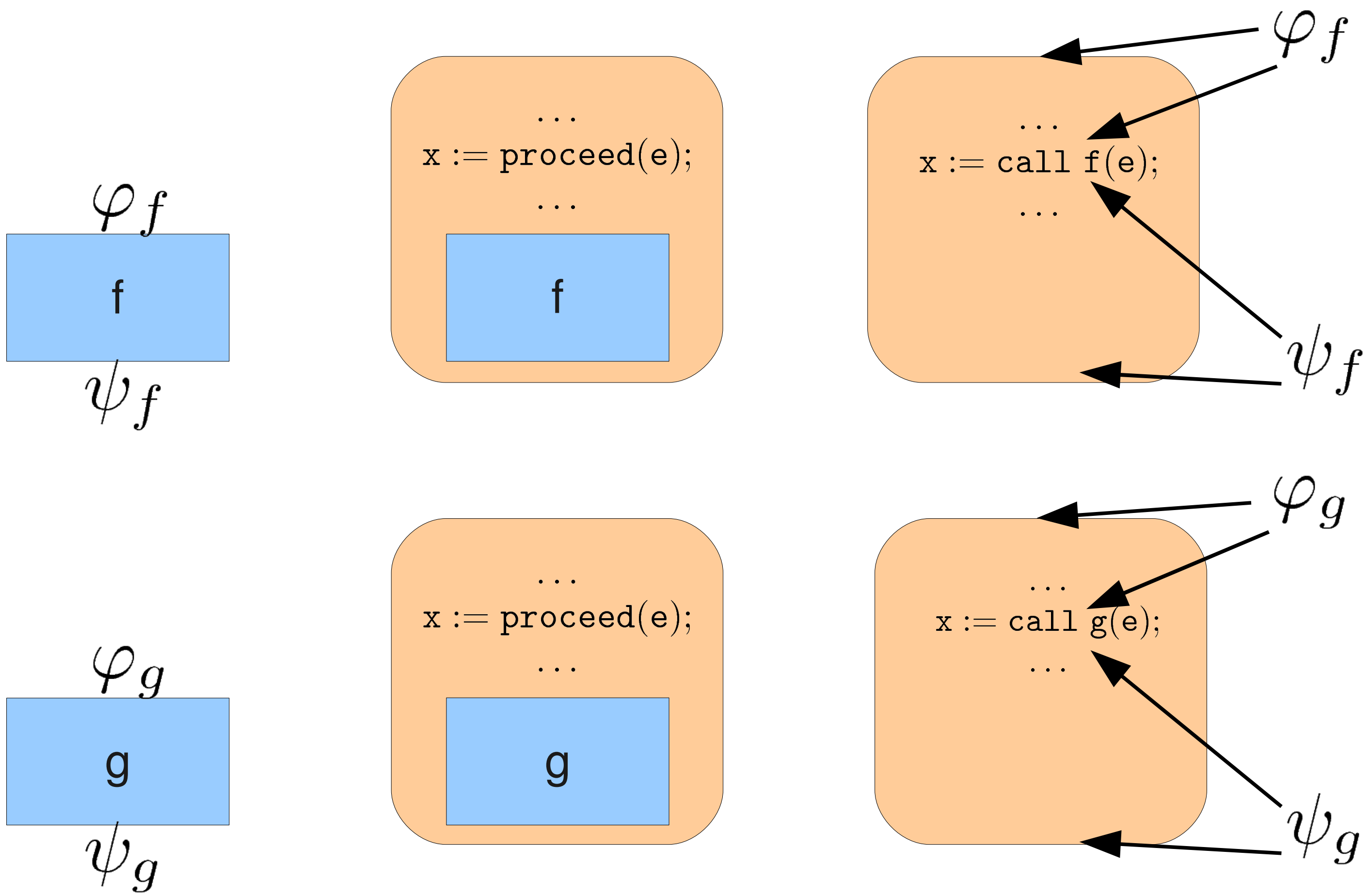
```
...
x := proceed(e);
...
```

```
...
x := call g(e);
...
```

IMPROVING THE VERIFICATION POWER

Drawback

Multiple advised procedures = multiple verification invariants.

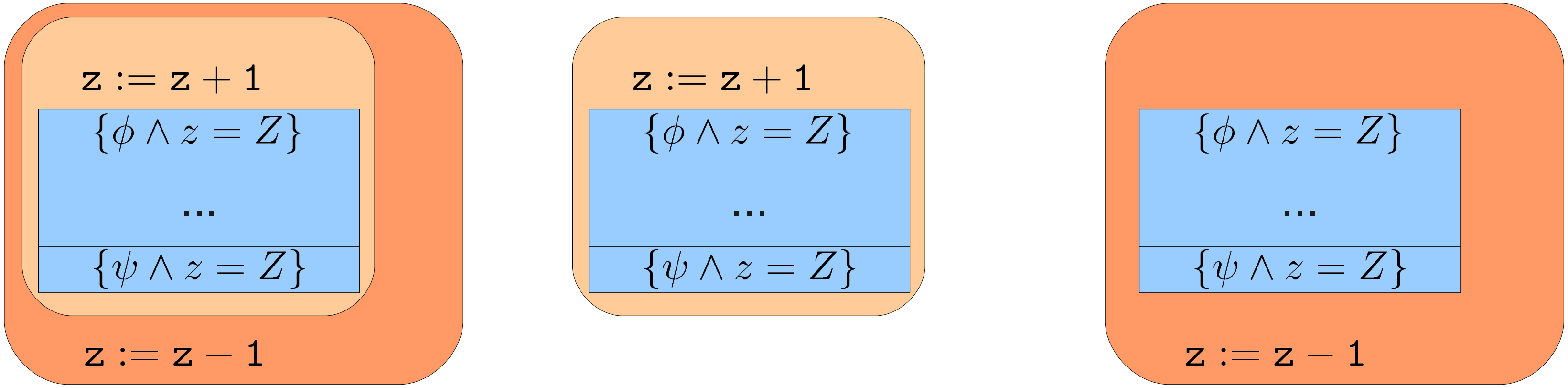


(specification of proceed improves modularity)

IMPROVING THE VERIFICATION POWER

Interference is not always a bad thing.

Some advices are be spec-preserving when combined but not when analyzed in isolation



IMPROVING THE VERIFICATION POWER

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

$(\Phi_1, \Psi_1, \mathcal{W}_1)$

a_1

$(\Phi_1^p, \Psi_1^p, \mathcal{W}_1^p)$

$(\Phi_2, \Psi_2, \mathcal{W}_2)$

a_2

$(\Phi_2^p, \Psi_2^p, \mathcal{W}_2^p)$

...

$(\Phi_n, \Psi_n, \mathcal{W}_n)$

a_n

$(\Phi_n^p, \Psi_n^p, \mathcal{W}_n^p)$

IMPROVING THE VERIFICATION POWER

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

$(\Phi_1, \Psi_1, \mathcal{W}_1)$

a_1

$(\Phi_1^p, \Psi_1^p, \mathcal{W}_1^p)$

$(\Phi_2, \Psi_2, \mathcal{W}_2)$

a_2

$(\Phi_2^p, \Psi_2^p, \mathcal{W}_2^p)$

...

$(\Phi_n, \Psi_n, \mathcal{W}_n)$

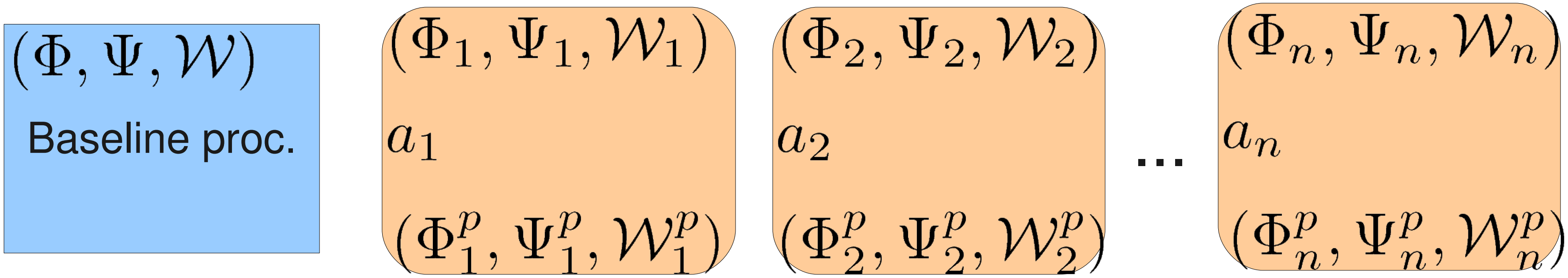
a_n

$(\Phi_n^p, \Psi_n^p, \mathcal{W}_n^p)$

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

IMPROVING THE VERIFICATION POWER



$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

$(\Phi_1, \Psi_1, \mathcal{W}_1)$

a_1

$(\Phi_1^p, \Psi_1^p, \mathcal{W}_1^p)$

$(\Phi_2, \Psi_2, \mathcal{W}_2)$

a_2

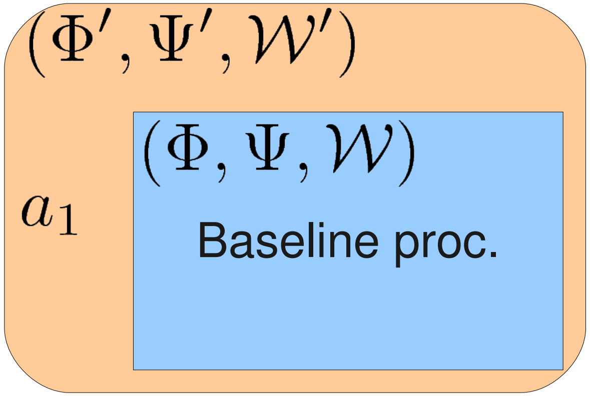
$(\Phi_2^p, \Psi_2^p, \mathcal{W}_2^p)$

...

$(\Phi_n, \Psi_n, \mathcal{W}_n)$

a_n

$(\Phi_n^p, \Psi_n^p, \mathcal{W}_n^p)$



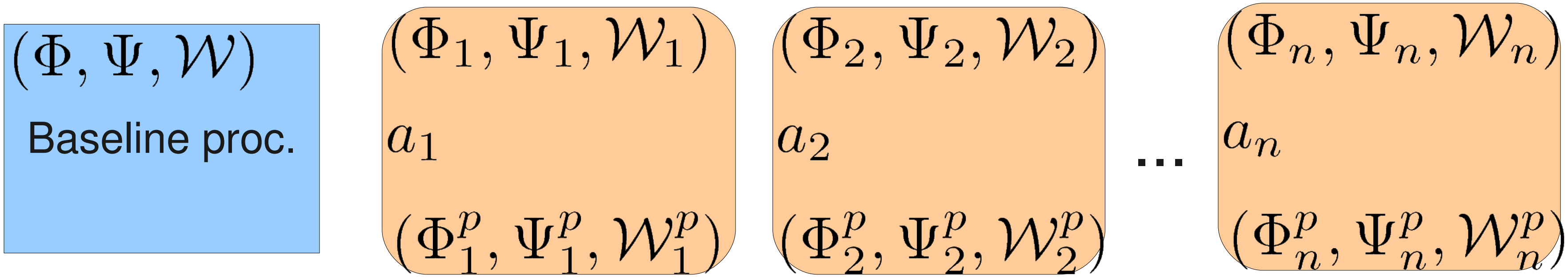
$(\Phi', \Psi', \mathcal{W}')$

a_1

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

IMPROVING THE VERIFICATION POWER



$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

$(\Phi_1, \Psi_1, \mathcal{W}_1)$

a_1

$(\Phi_1^p, \Psi_1^p, \mathcal{W}_1^p)$

$(\Phi_2, \Psi_2, \mathcal{W}_2)$

a_2

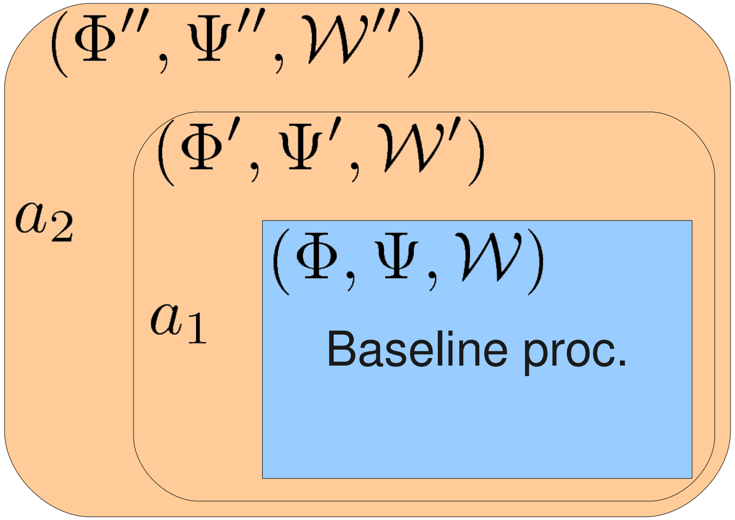
$(\Phi_2^p, \Psi_2^p, \mathcal{W}_2^p)$

...

$(\Phi_n, \Psi_n, \mathcal{W}_n)$

a_n

$(\Phi_n^p, \Psi_n^p, \mathcal{W}_n^p)$



$(\Phi'', \Psi'', \mathcal{W}'')$

a_2

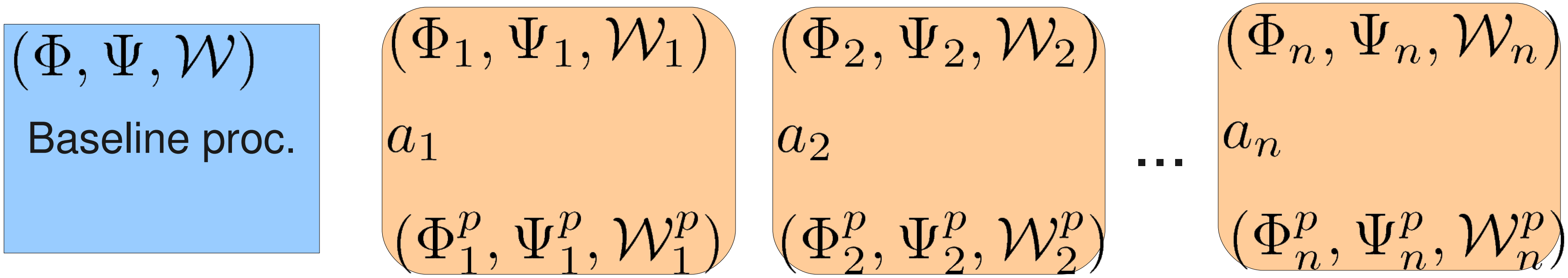
$(\Phi', \Psi', \mathcal{W}')$

a_1

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

IMPROVING THE VERIFICATION POWER



$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

$(\Phi_1, \Psi_1, \mathcal{W}_1)$

a_1

$(\Phi_1^p, \Psi_1^p, \mathcal{W}_1^p)$

$(\Phi_2, \Psi_2, \mathcal{W}_2)$

a_2

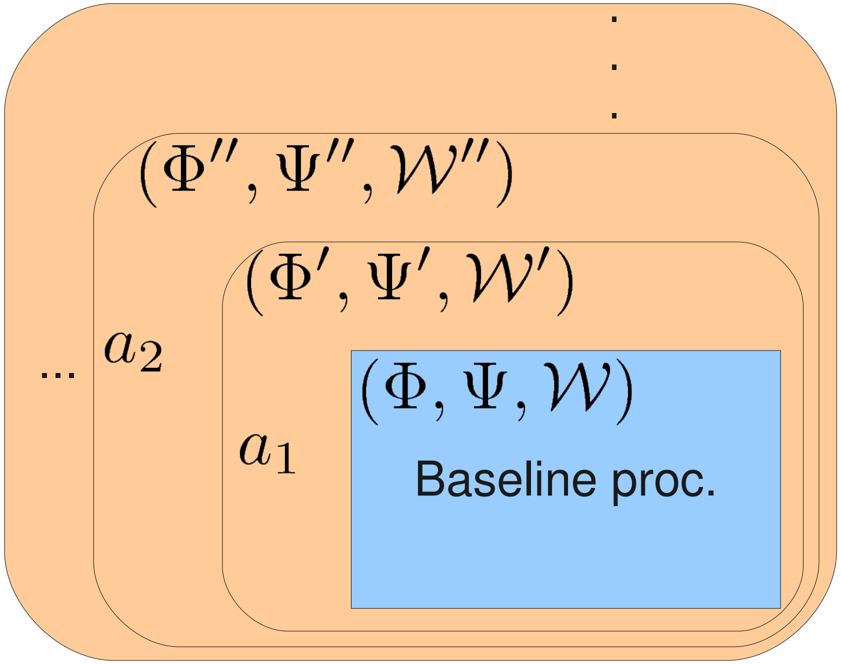
$(\Phi_2^p, \Psi_2^p, \mathcal{W}_2^p)$

...

$(\Phi_n, \Psi_n, \mathcal{W}_n)$

a_n

$(\Phi_n^p, \Psi_n^p, \mathcal{W}_n^p)$



$\vdots$

$(\Phi'', \Psi'', \mathcal{W}'')$

$(\Phi', \Psi', \mathcal{W}')$

...

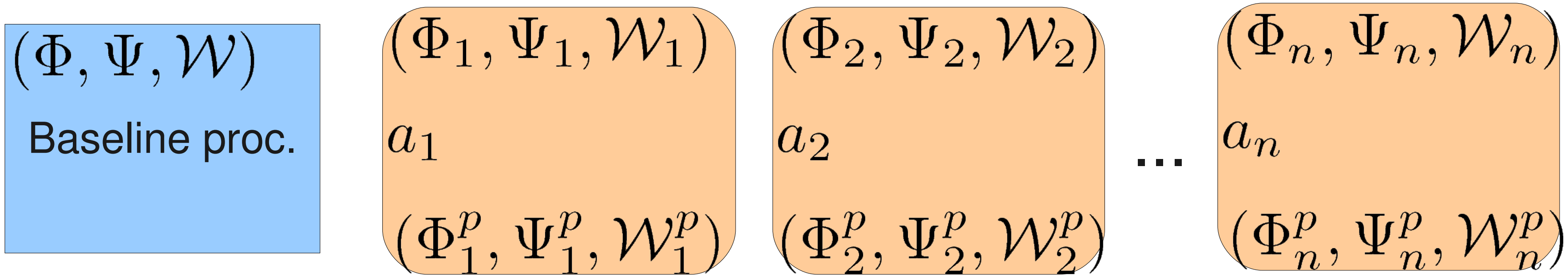
a_2

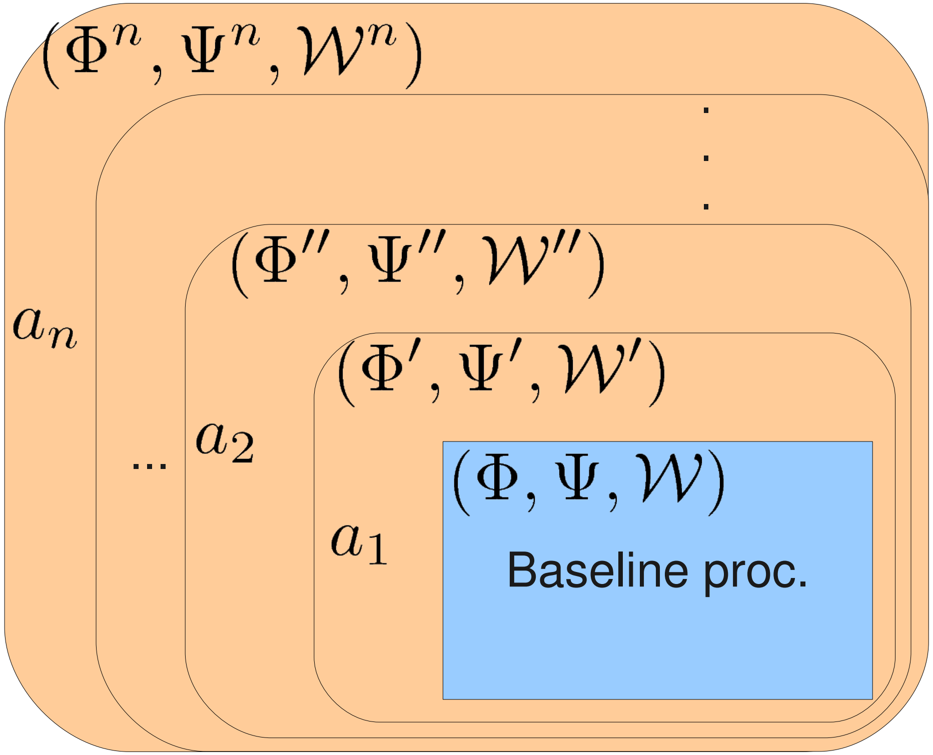
a_1

$(\Phi, \Psi, \mathcal{W})$

Baseline proc.

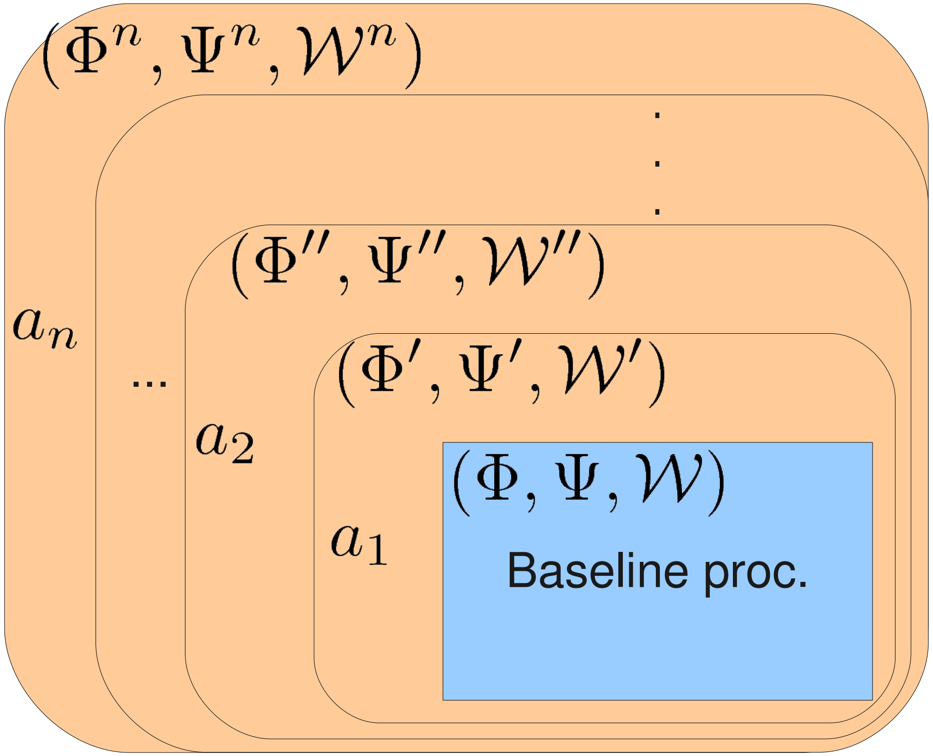
IMPROVING THE VERIFICATION POWER


$(\Phi, \Psi, \mathcal{W})$
Baseline proc.
$(\Phi_1, \Psi_1, \mathcal{W}_1)$
a_1
$(\Phi^p_1, \Psi^p_1, \mathcal{W}^p_1)$
$(\Phi_2, \Psi_2, \mathcal{W}_2)$
a_2
$(\Phi^p_2, \Psi^p_2, \mathcal{W}^p_2)$
...
$(\Phi_n, \Psi_n, \mathcal{W}_n)$
a_n
$(\Phi^p_n, \Psi^p_n, \mathcal{W}^p_n)$


$(\Phi^n, \Psi^n, \mathcal{W}^n)$
$\vdots$
$(\Phi'', \Psi'', \mathcal{W}'')$
$(\Phi', \Psi', \mathcal{W}')$
a_n
...
a_2
a_1
$(\Phi, \Psi, \mathcal{W})$
Baseline proc.

IMPROVING THE VERIFICATION POWER

$$\Phi \Rightarrow \Phi^n \qquad \Psi^n \Rightarrow \Psi \qquad \mathcal{W}^n \cap \mathcal{V} \subseteq \mathcal{W}$$


$(\Phi^n, \Psi^n, \mathcal{W}^n)$
$\vdots$
$(\Phi'', \Psi'', \mathcal{W}'')$
a_n
...
$(\Phi', \Psi', \mathcal{W}')$
a_2
$(\Phi, \Psi, \mathcal{W})$
a_1
Baseline proc.

Specification Refinement instead of Specification Preservation

MOTIVATION

SPECIFICATION PRESERVING ADVICES

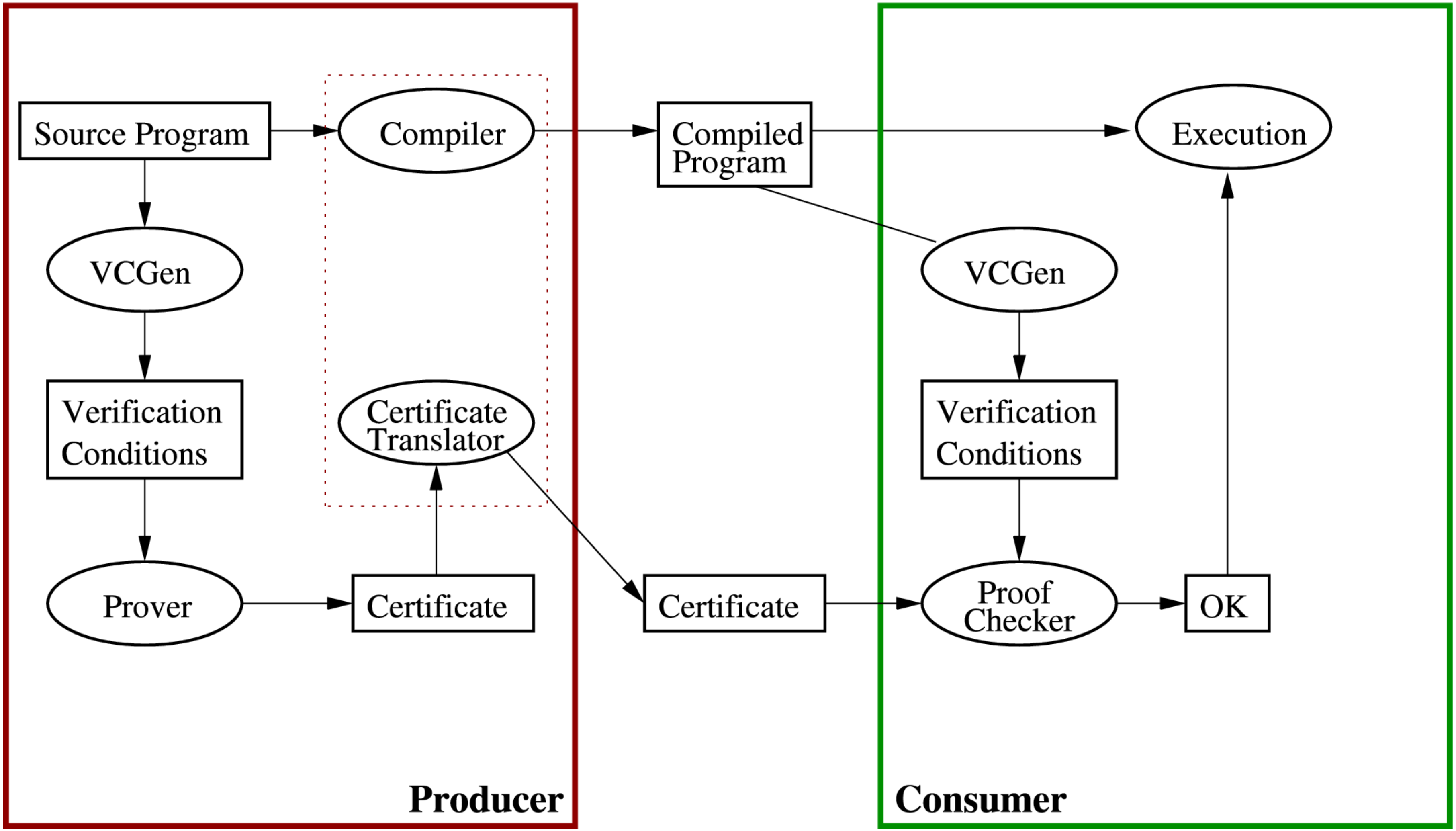
PROVING SPECIFICATION PRESERVING ADVICES

REDUCING PROOF OBLIGATIONS

IMPROVING THE VERIFICATION POWER

CERTIFICATE TRANSLATION

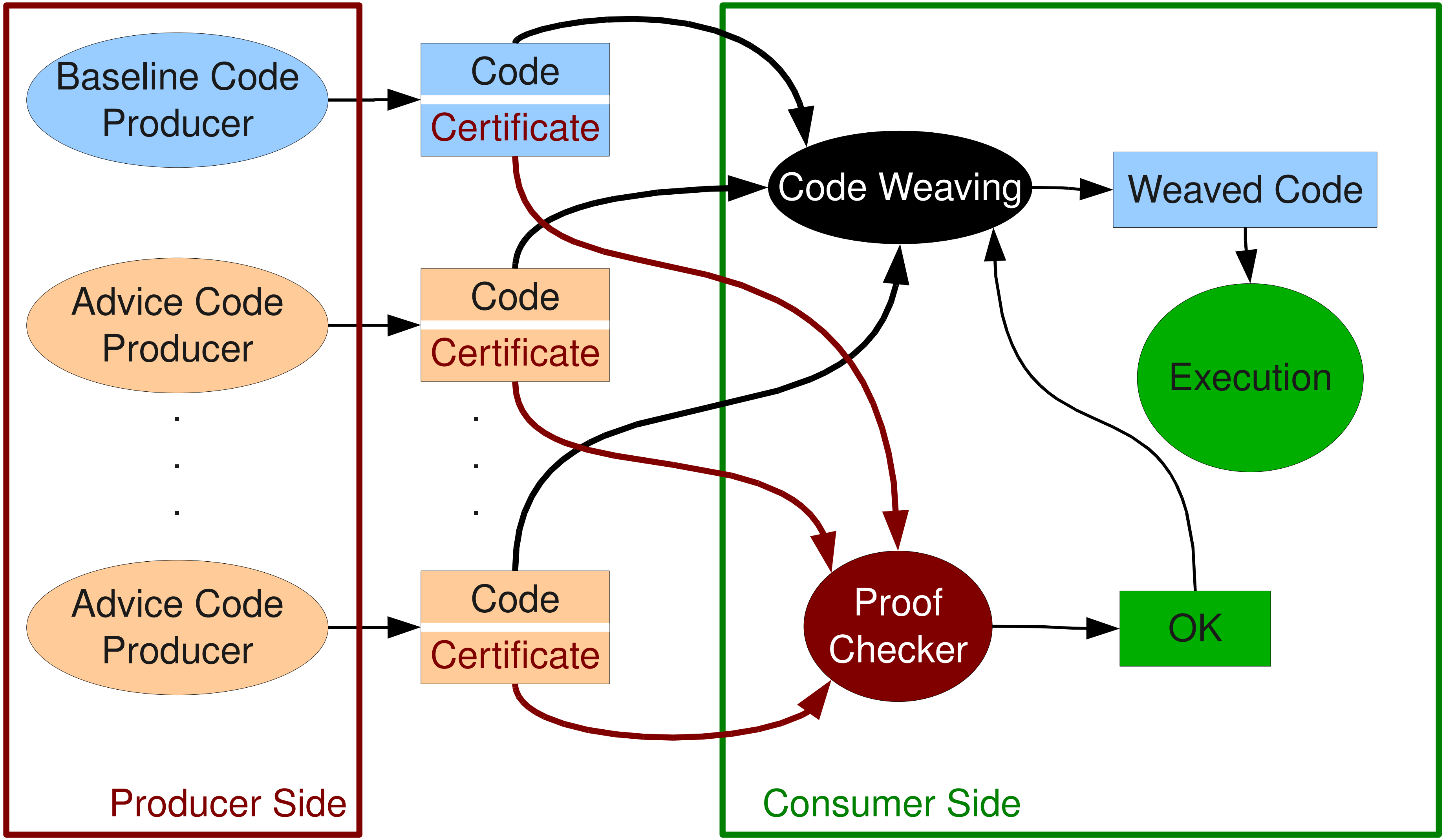
Certificate Translation



Certificate Translation

Consider the situation:

- Client verification and execution environment not AOP-oriented
- Code generated by multiple producers is weaved before execution

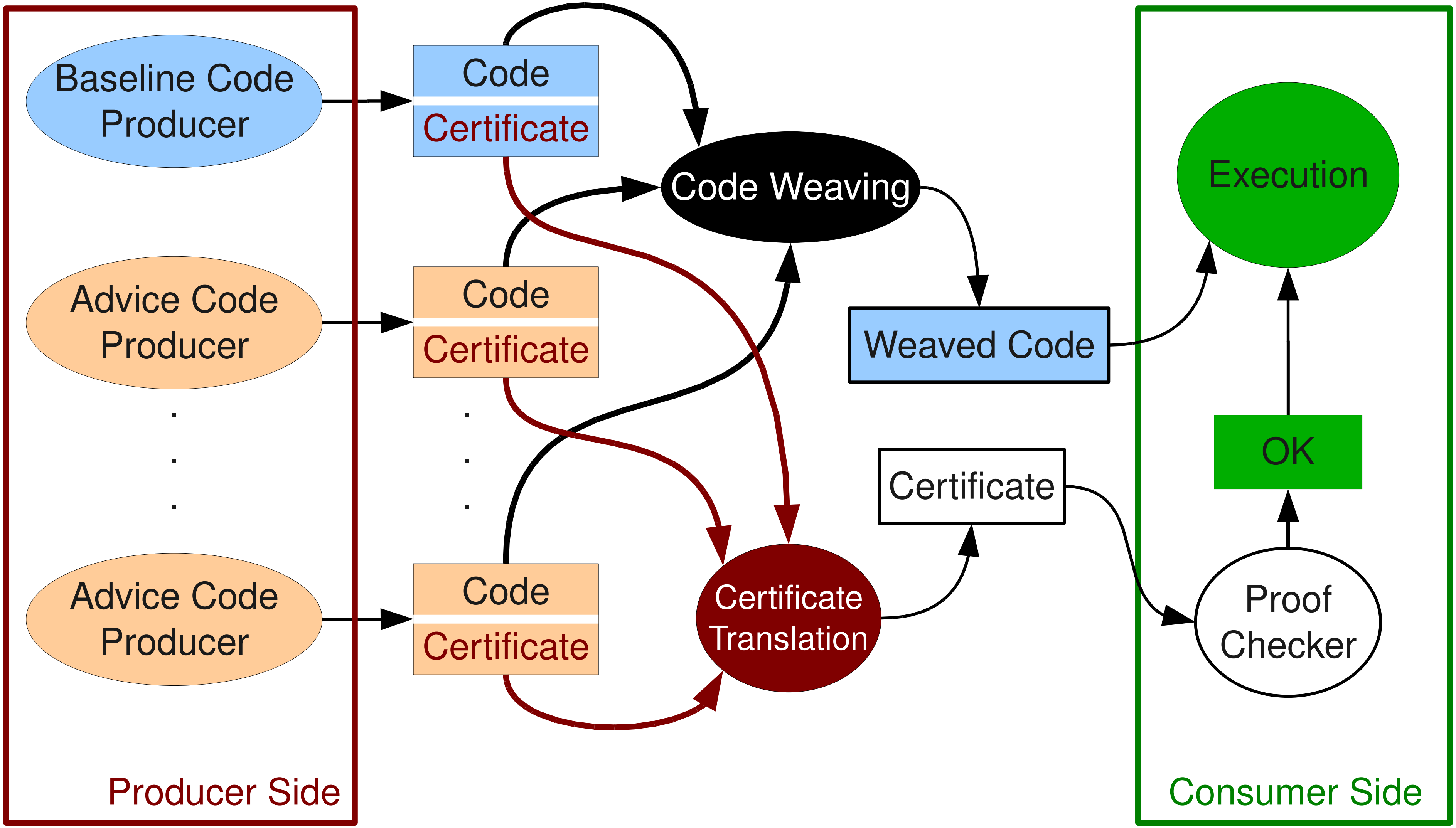


Certificate Translation

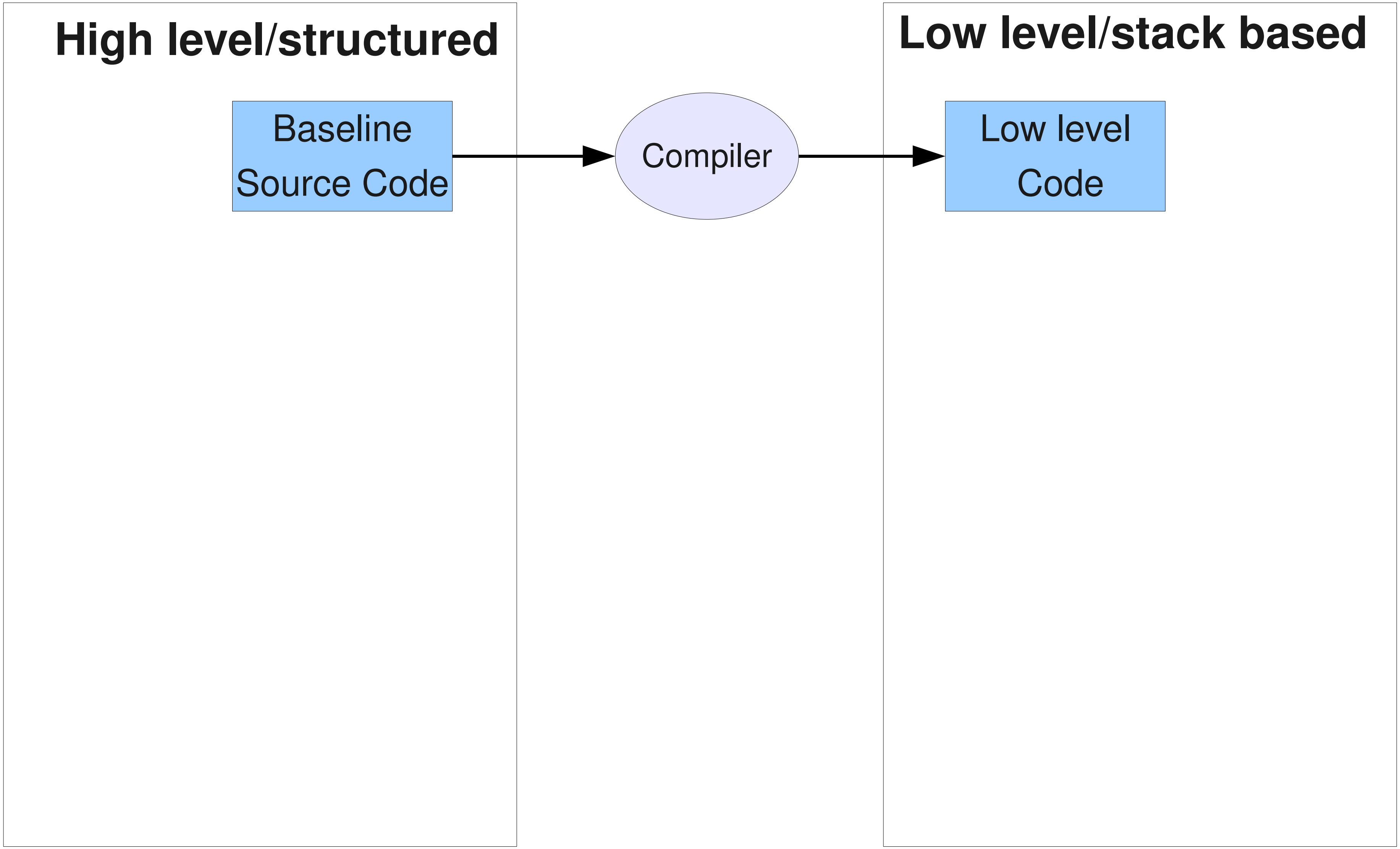
Consider the situation:

•Client verification and execution environment not AOP-oriented

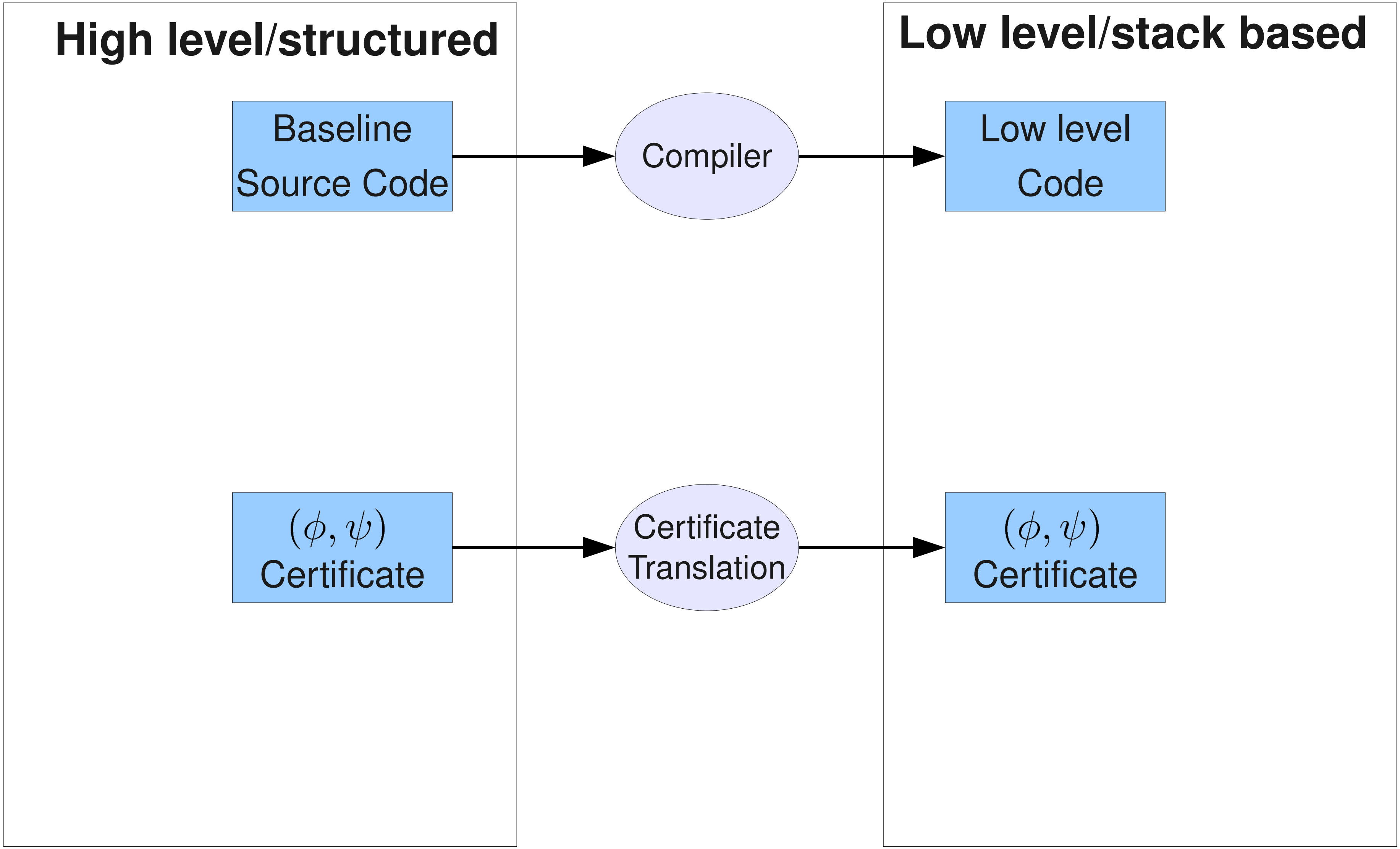
•Code generated by multiple producers is weaved before execution



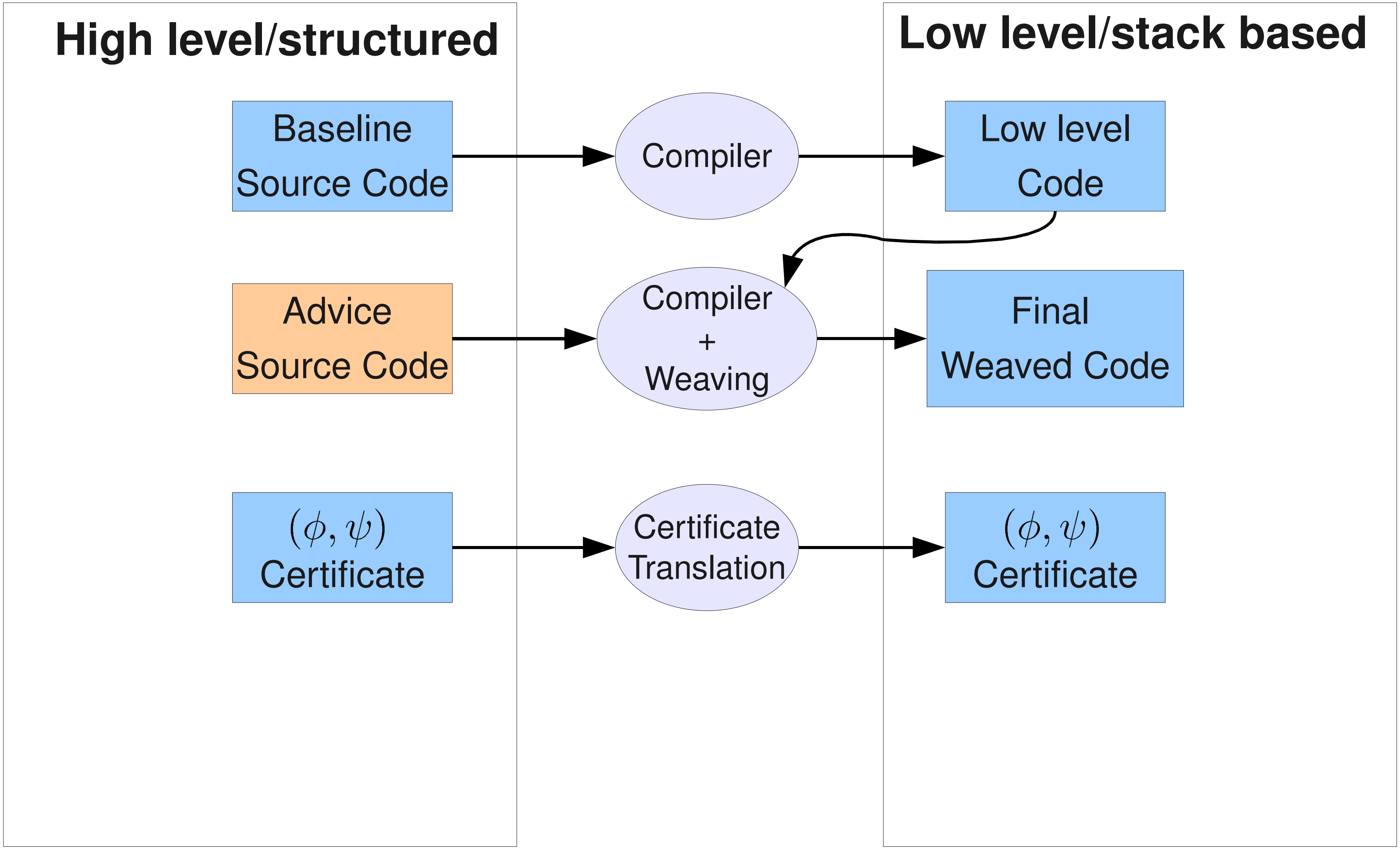
Certificate Translation



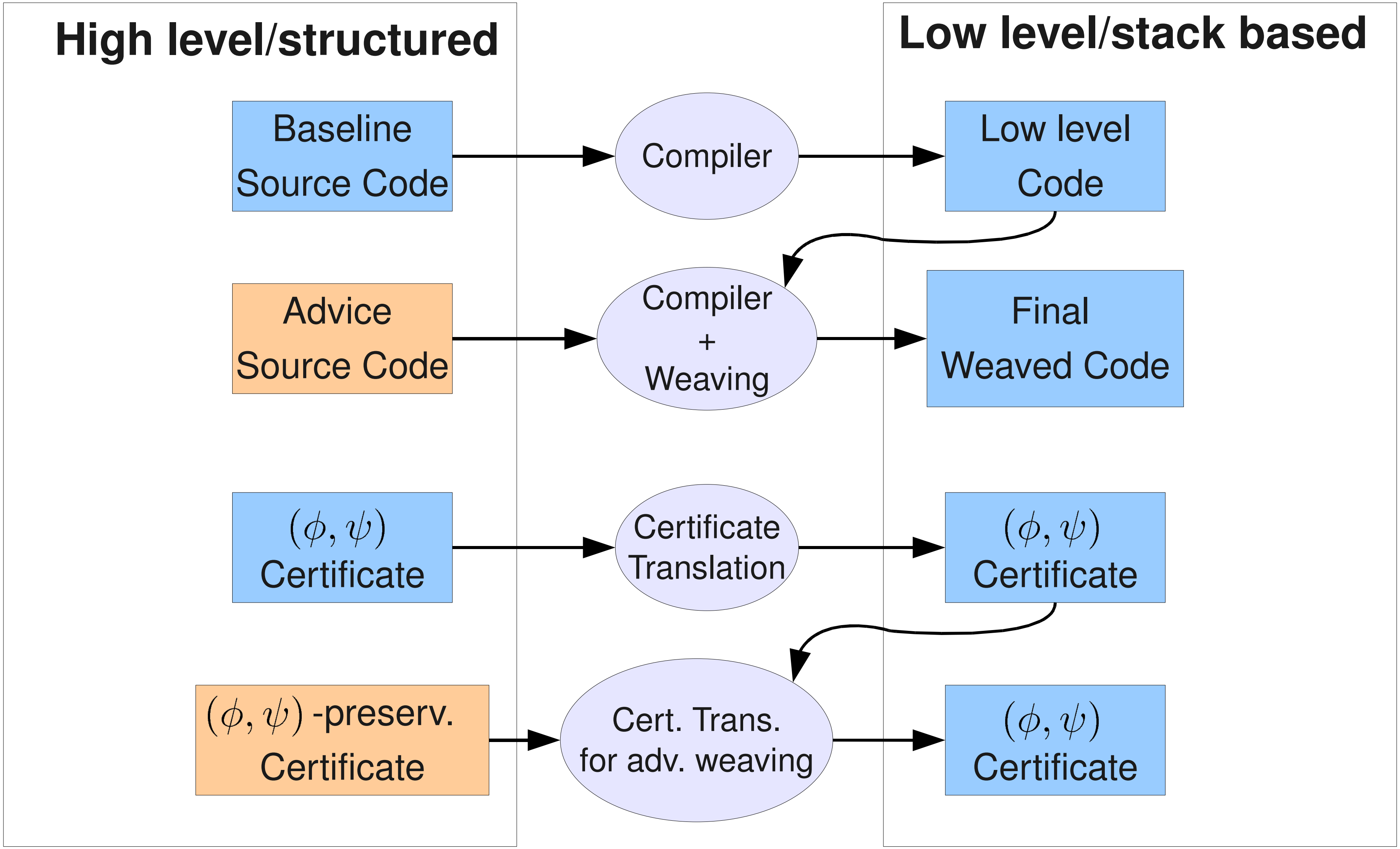
Certificate Translation



Certificate Translation



Certificate Translation



Conclusions

- A more flexible notion of non-interfering advices
- Stronger non-interference analyses reduce proof obligations
- Certificate translation targetting a typical backend