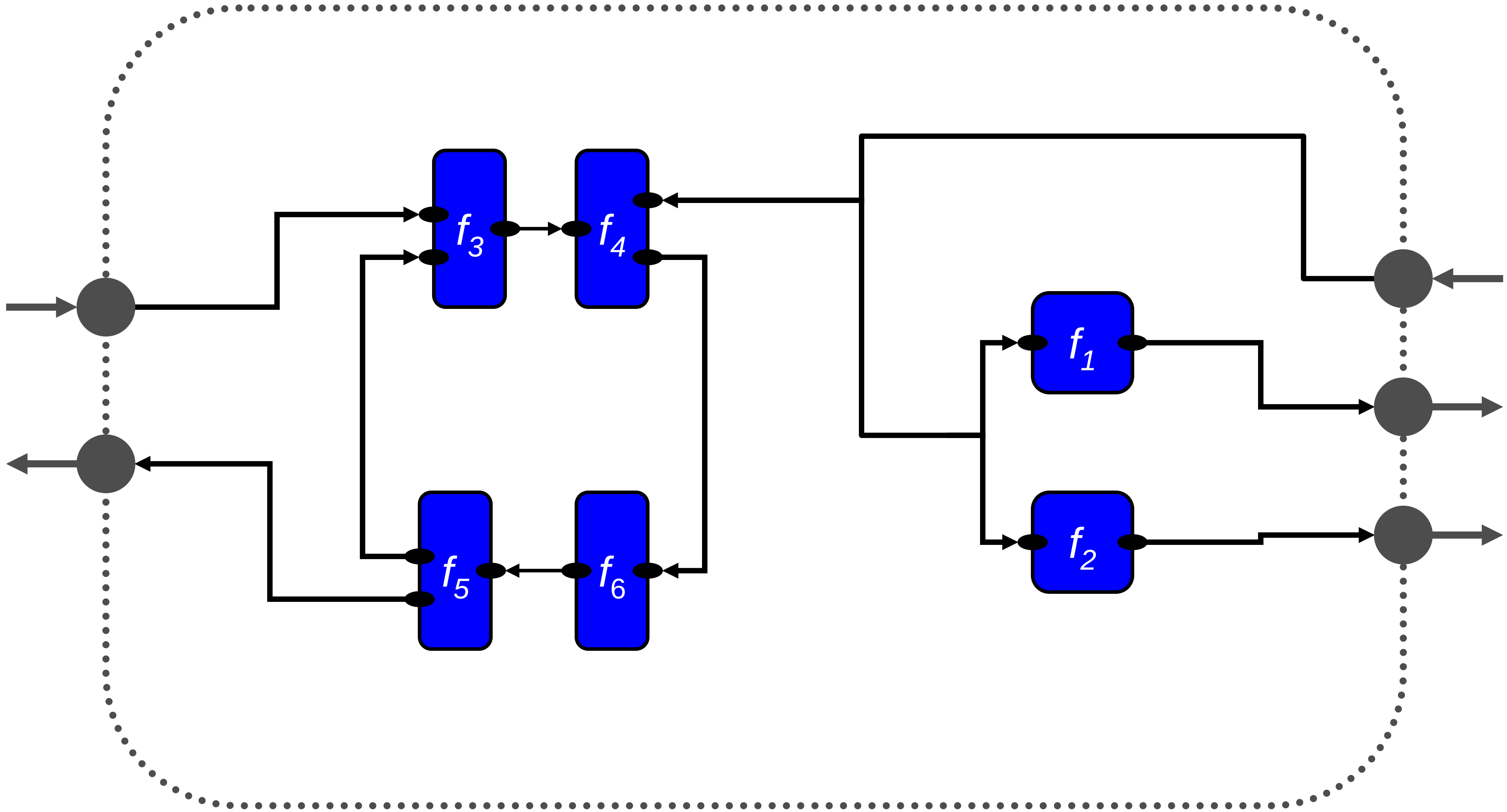
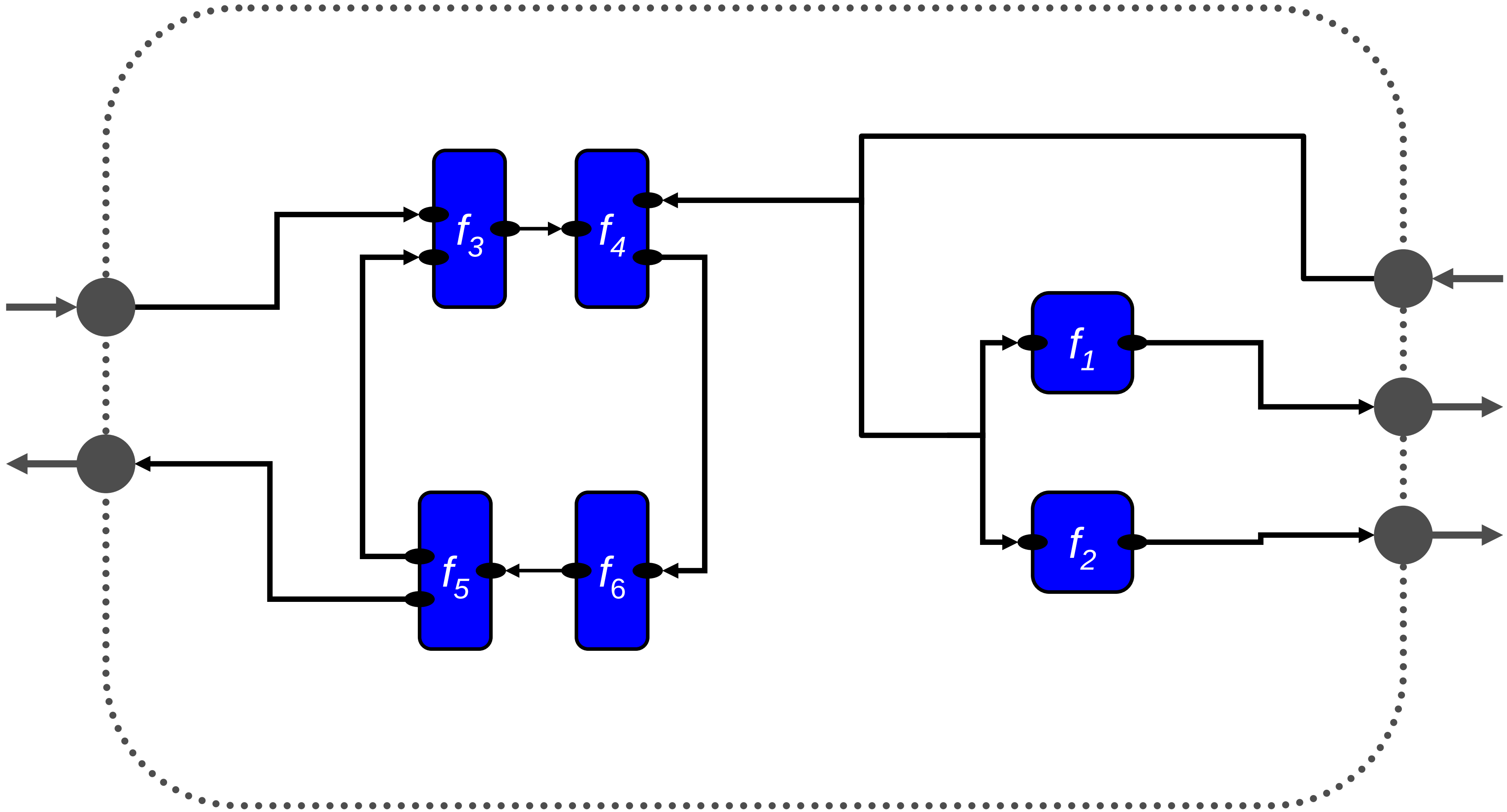
Interface-based Design 1

Tom Henzinger
EPFL and UC Berkeley

A Complex System is built from Components.

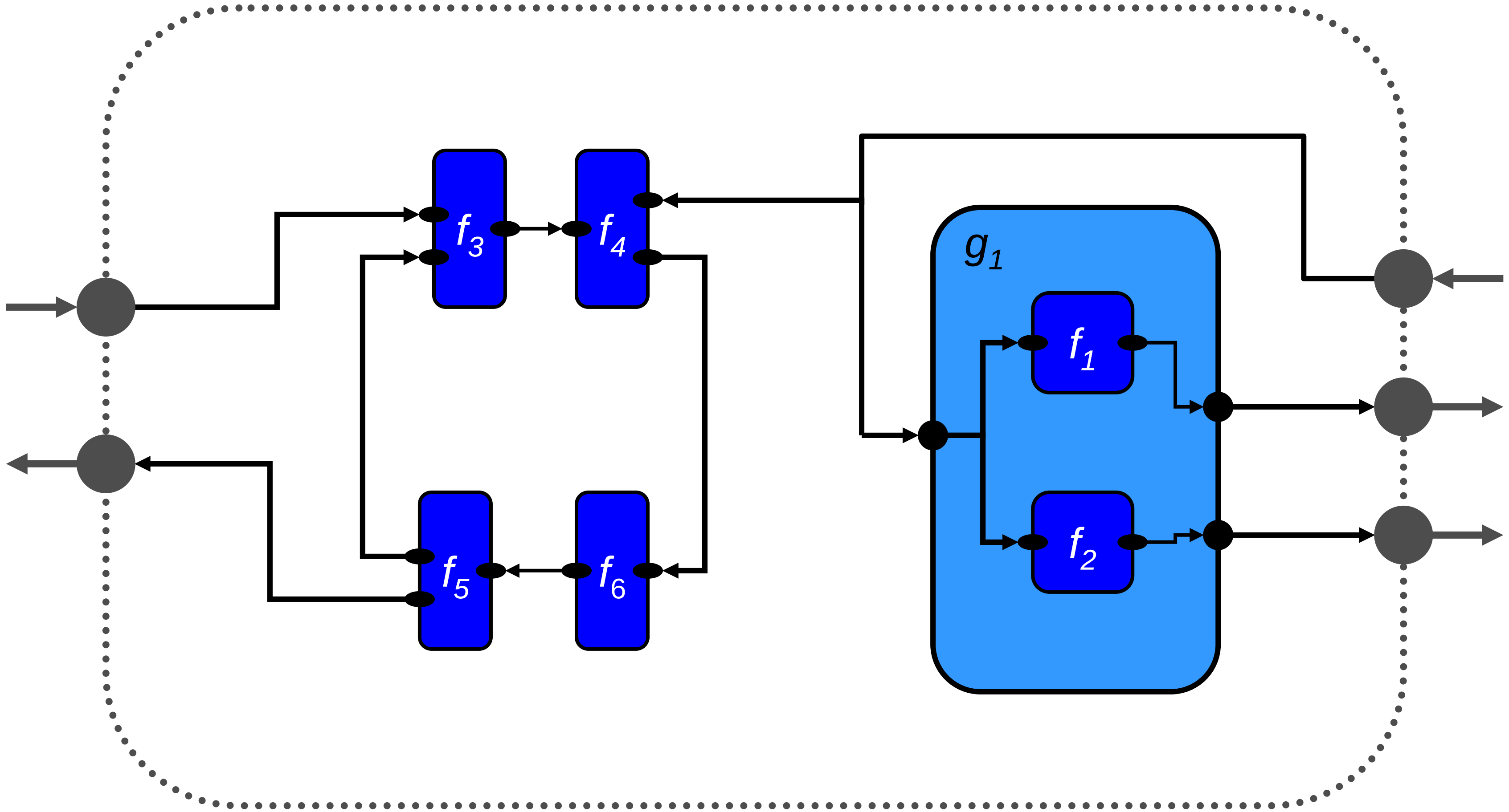


What does “Compositionality” mean?



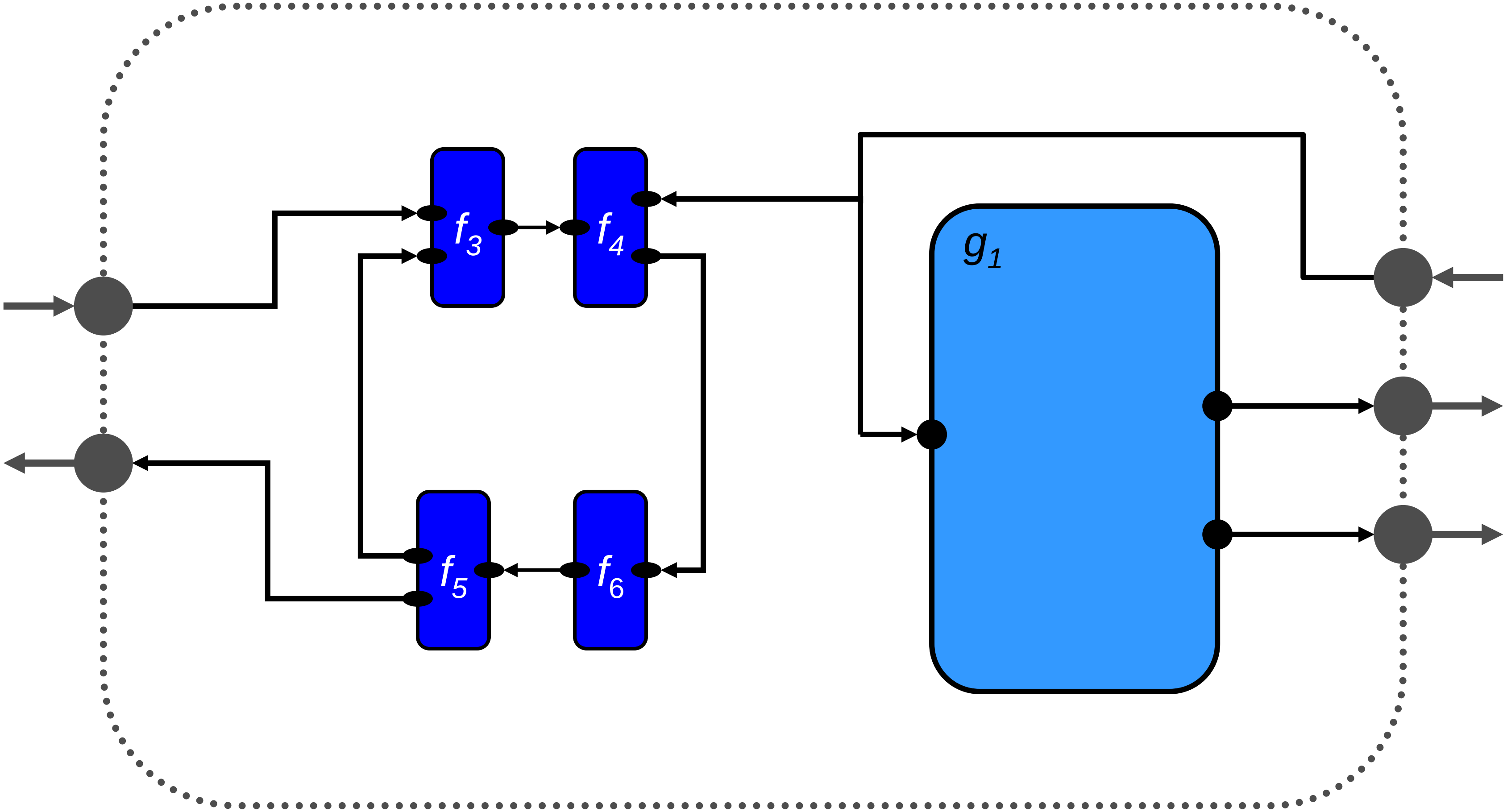
Bottom-up Compositionality:

A collection of components is again a component.



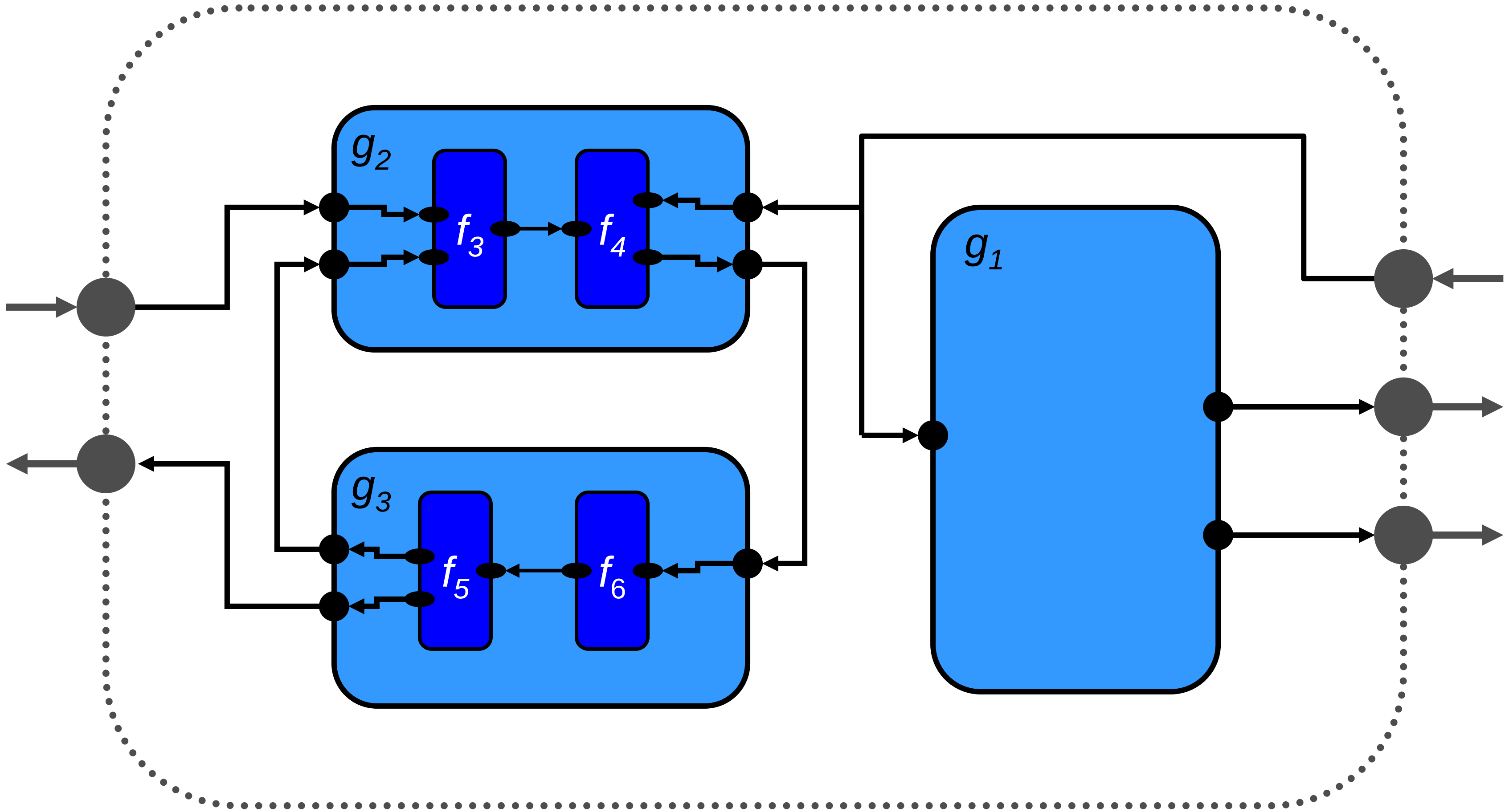
Bottom-up Compositionality:

Supports some form of abstraction (hiding).



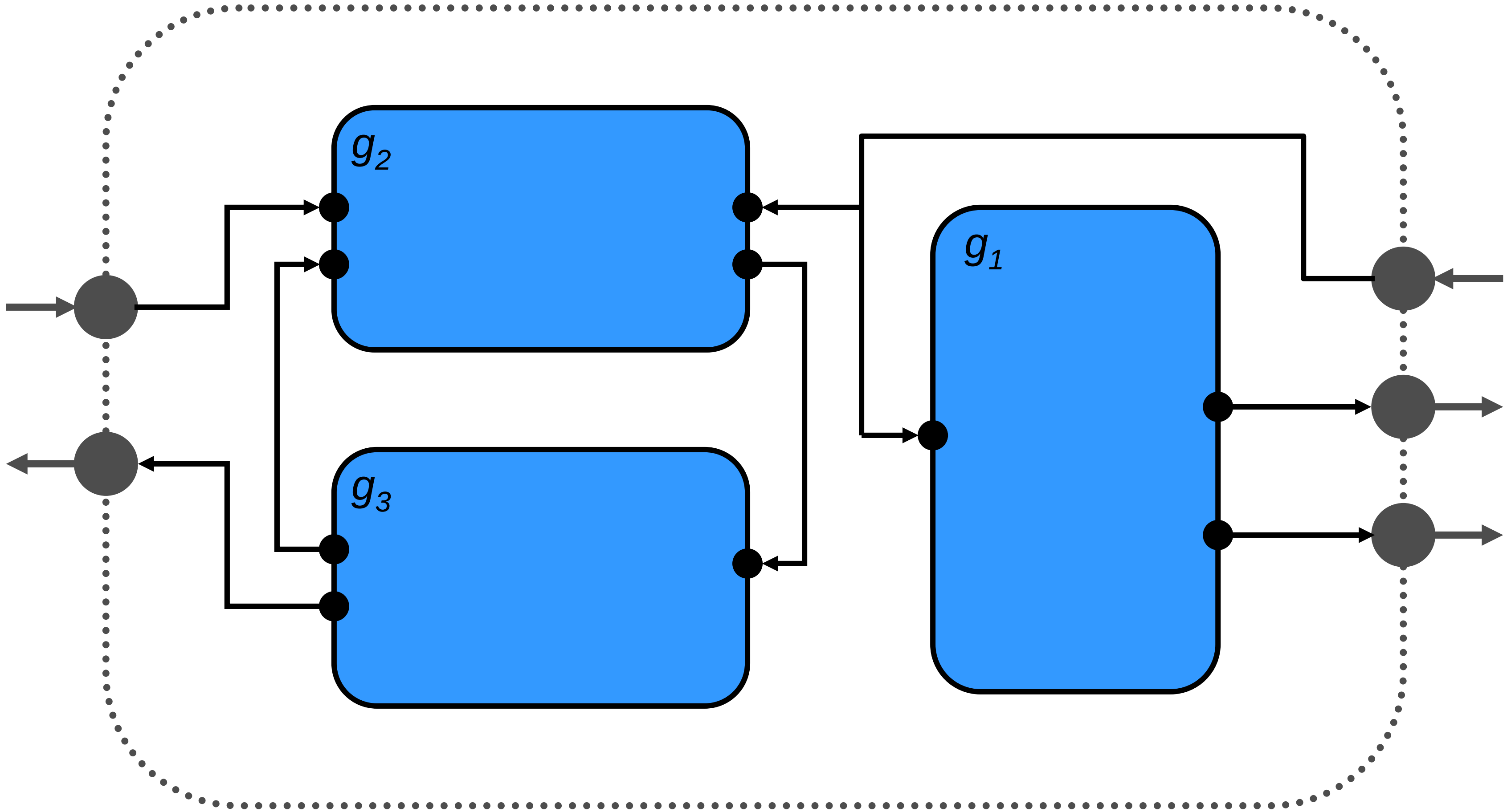
Bottom-up Compositionality:

"Properties" are also abstractions.



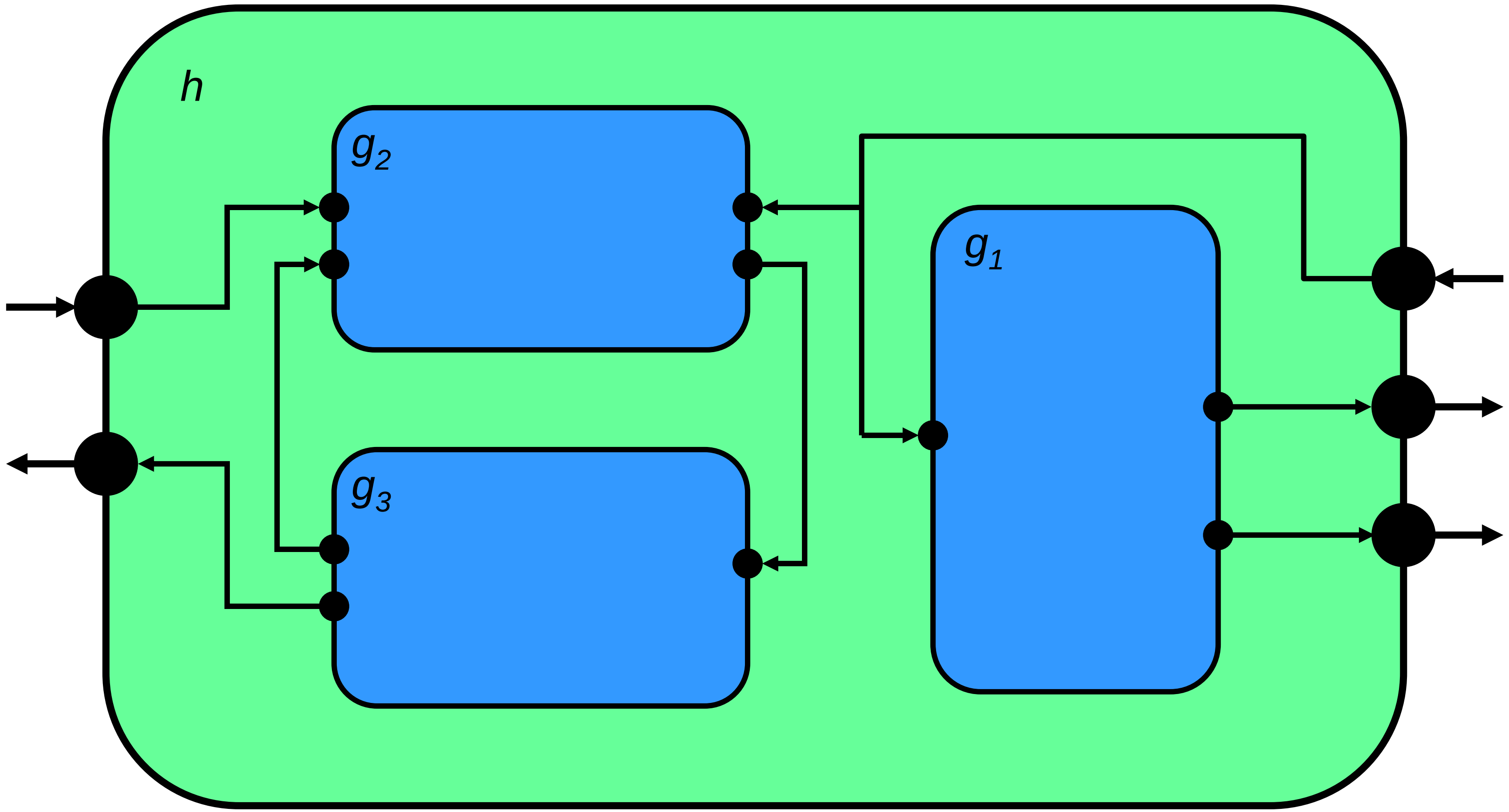
Bottom-up Compositionality:

Abstractions/properties can again be composed.



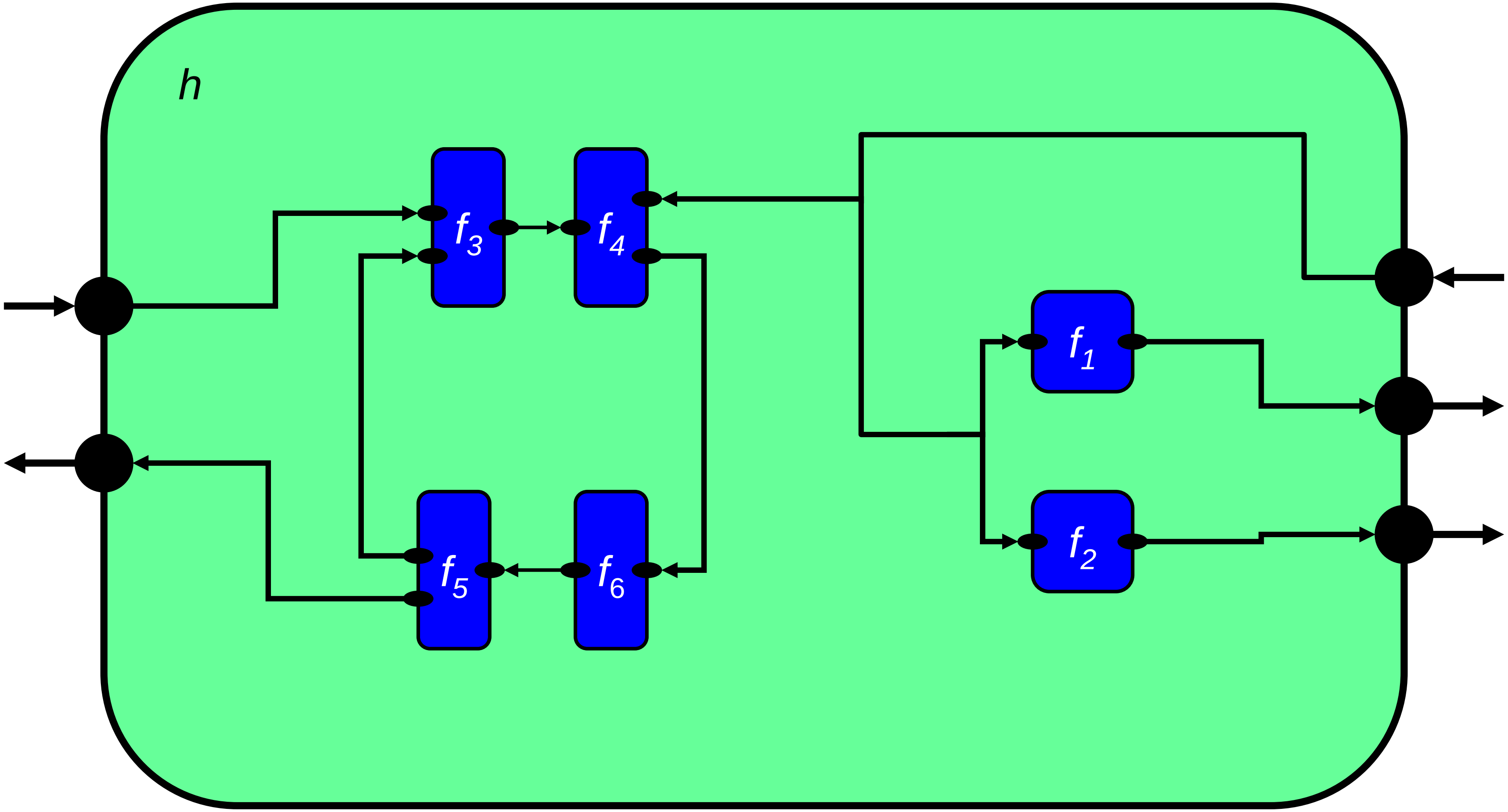
Bottom-up Compositionality:

Stepwise composition and abstraction (hierarchy).



Bottom-up Compositionality:

If $f \cdot g$ and $f' \cdot g'$, then $f||f' \cdot g||g'$.



Bottom-up Compositionality:
Supports Compositional Verification.

$$f_1||f_2 \cdot g_1$$

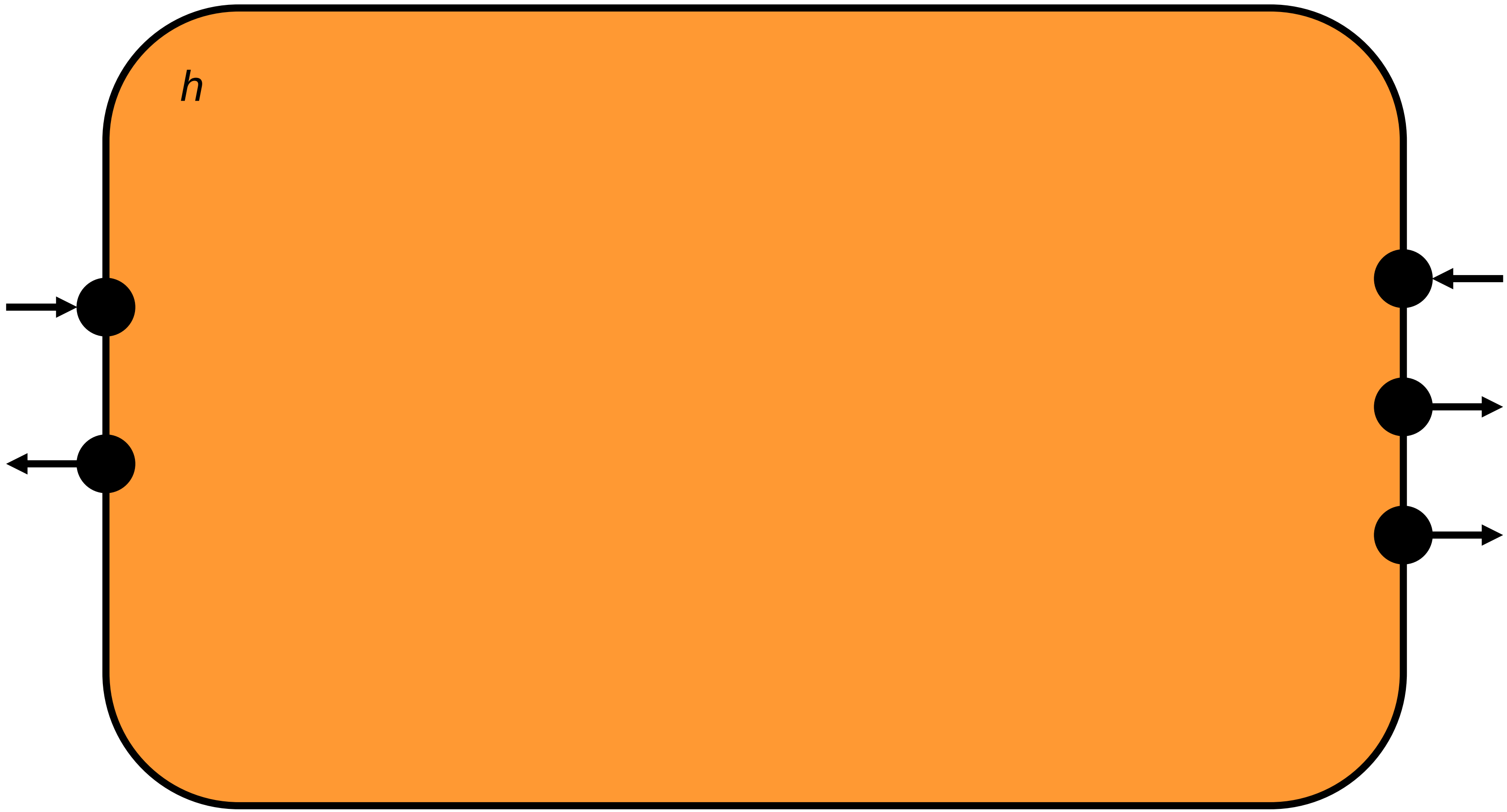
$$f_3||f_4 \cdot g_2$$

$$f_5||f_6 \cdot g_3$$

$$g_1||g_2||g_3 \cdot h$$

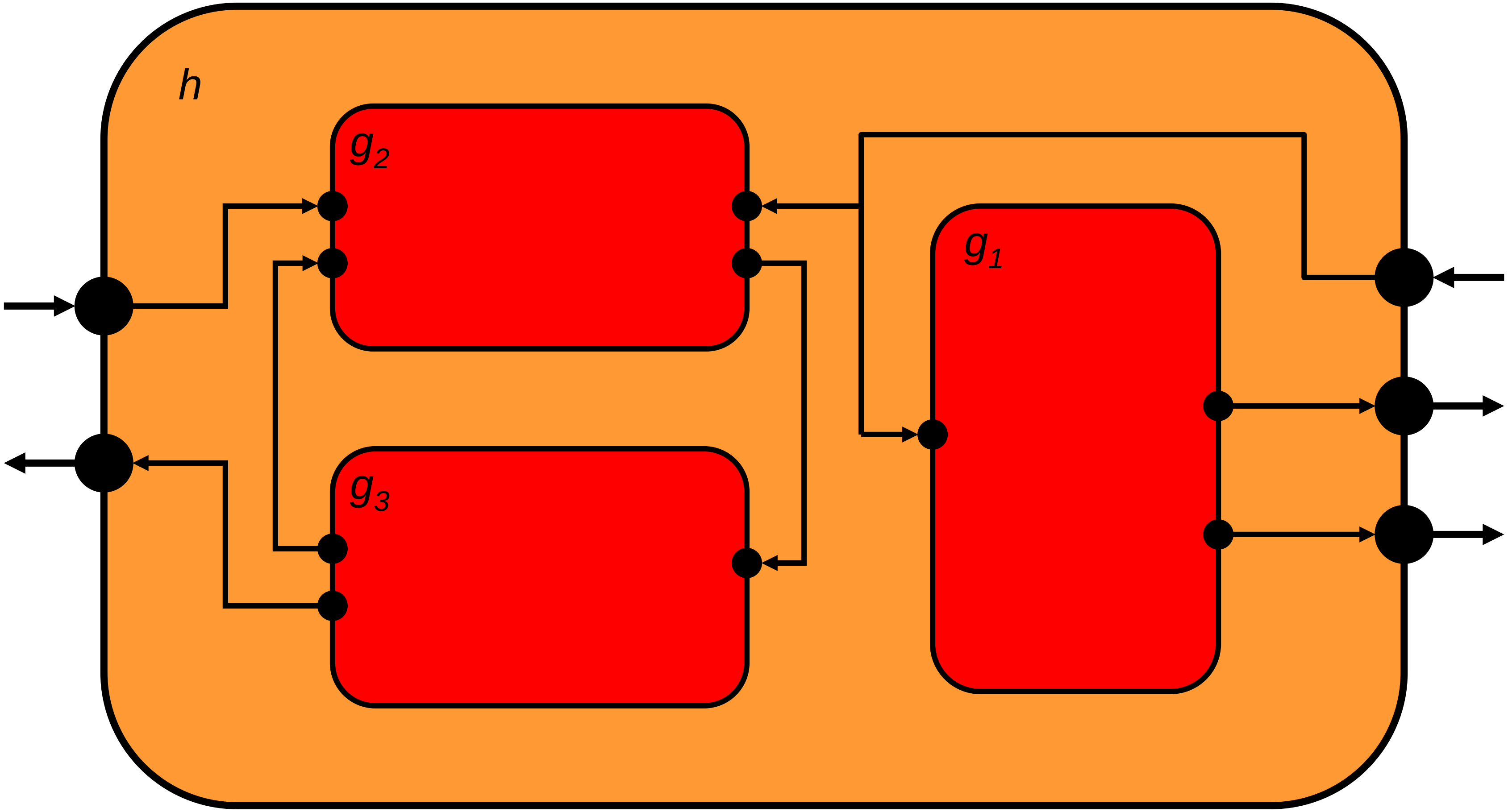
$$f_1||f_2||f_3||f_4||f_5||f_6 \cdot h$$

The Reverse Process

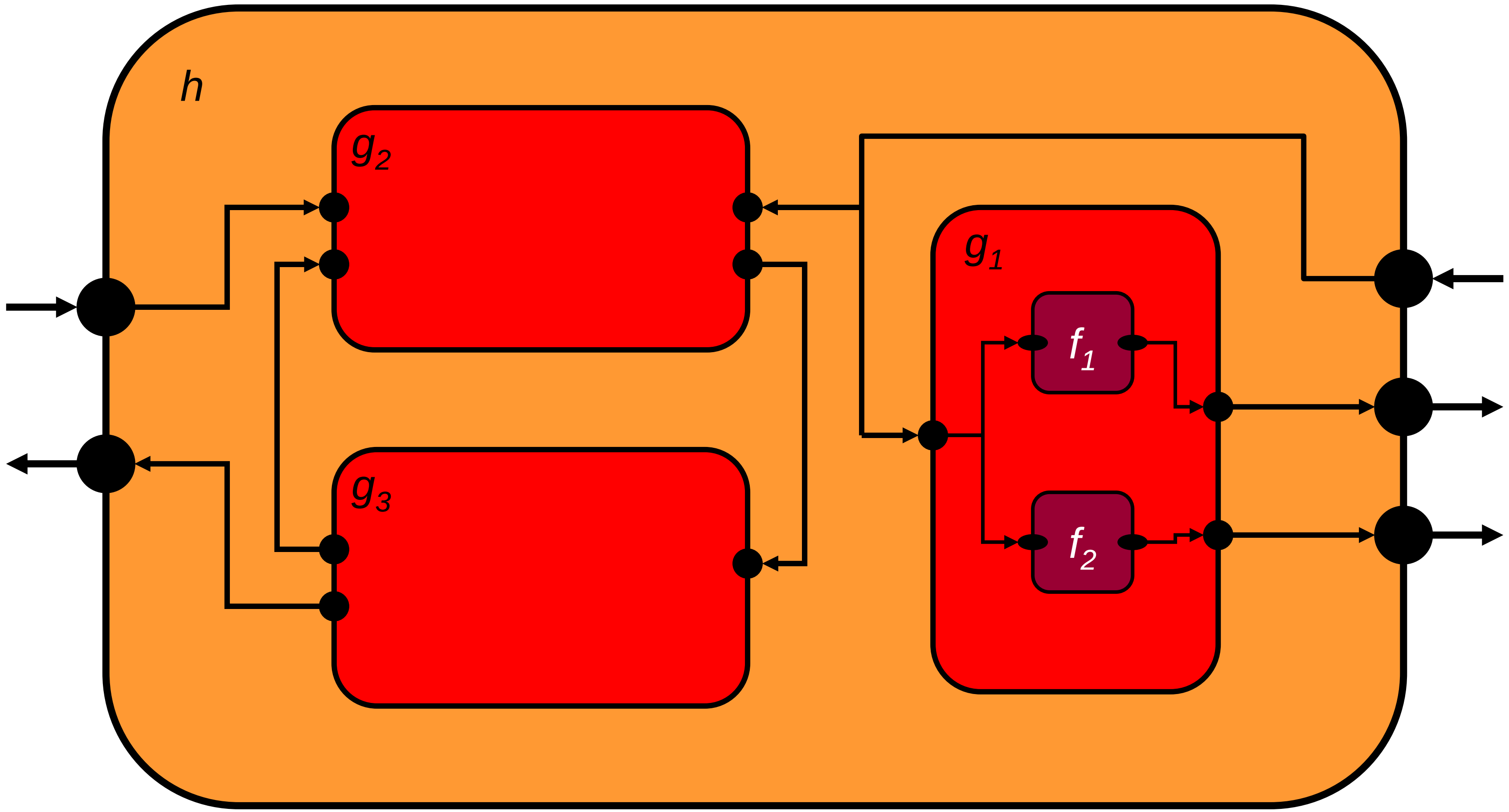


The Reverse Process:

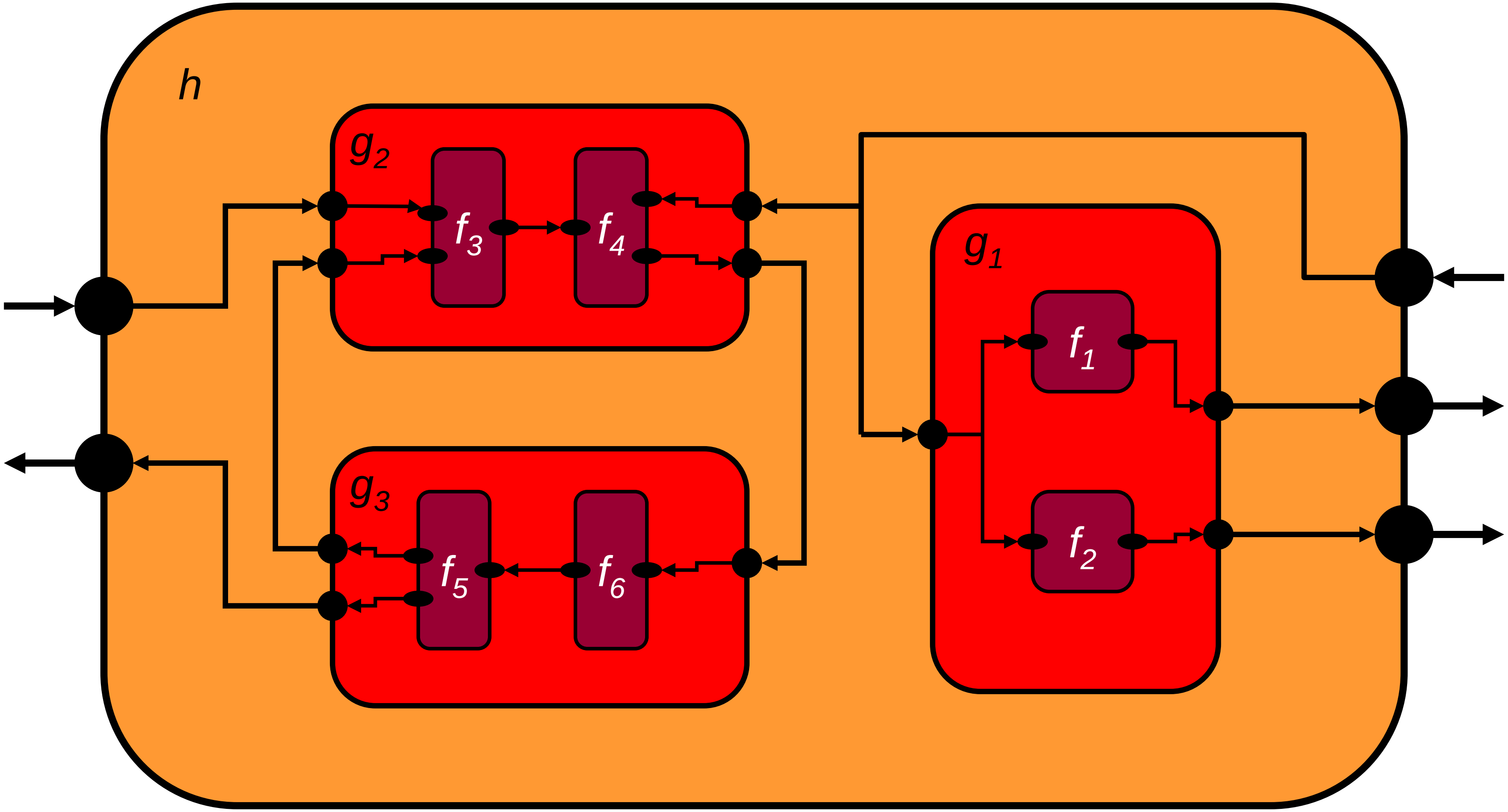
Stepwise decomposition and refinement.



The Reverse Process:
Independent Implementability.

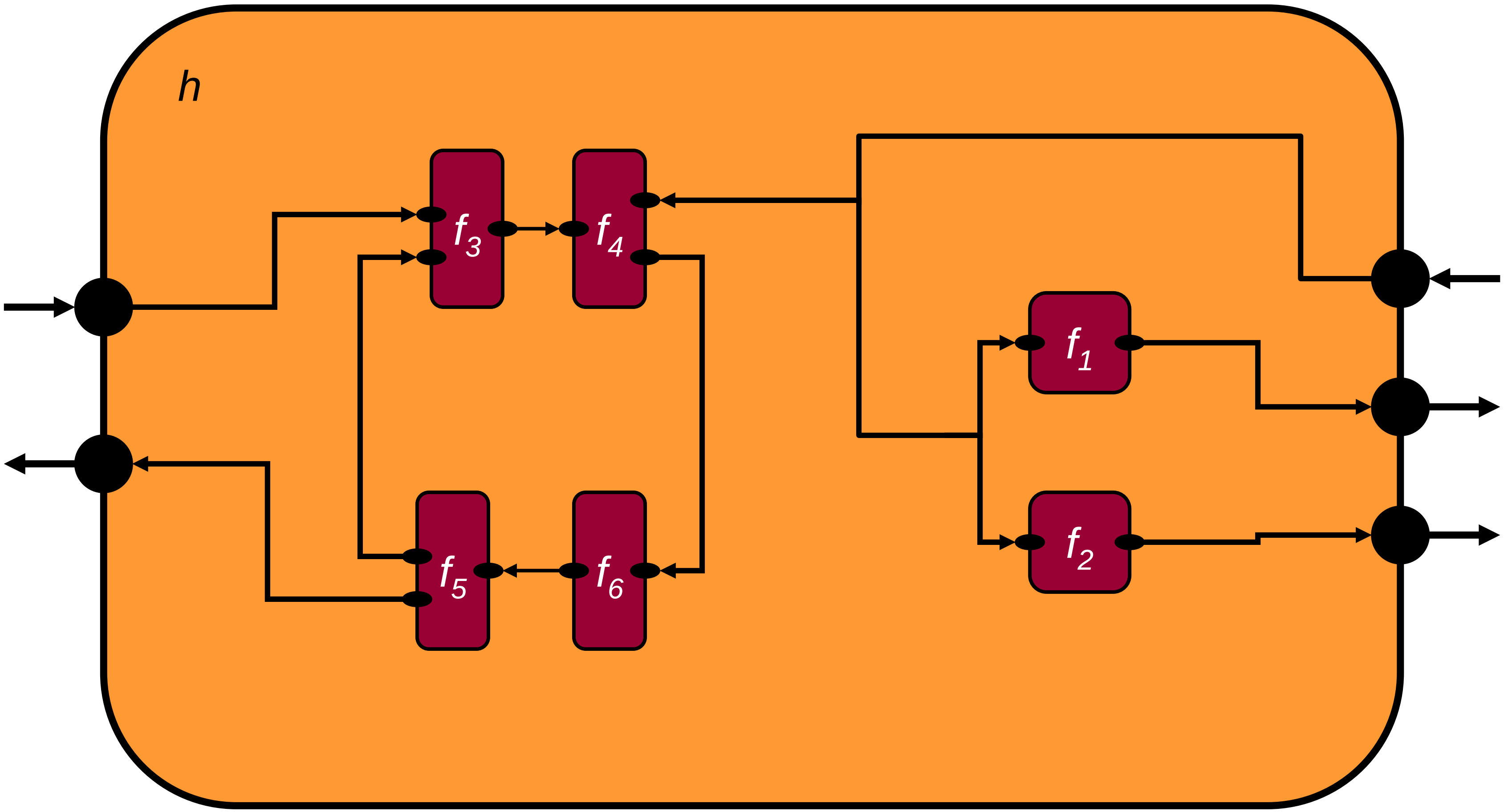


The Reverse Process:
Independent Implementability.



Top-down Compositionality:

If *g* ¸ *f* and *g'* ¸ *f'*, then *g*||*g'* ¸ *f*||*f'*.



Top-down Compositionality:
Supports Compositional Design.

h ¸ $g1||g2||g3$

g_1 ¸ $f_1||f_2$

g_2 ¸ $f_3||f_4$

g_3 ¸ $f_5||f_6$

---

h ¸ $f_1||f_2||f_3||f_4||f_5||f_6$

Bottom-up Compositionality:

If *f* · *g* and *f'* · *g'*, then *f*||*f'* · *g*||*g'*.

Top-down Compositionality:

If *g* ¸ *f* and *g'* ¸ *f'*, then *g*||*g'* ¸ *f*||*f'*.

What's the Difference?

Bottom-up Compositionality:

If *f* · *g* and *f’* · *g’*, then *f*||*f’* · *g*||*g’*.

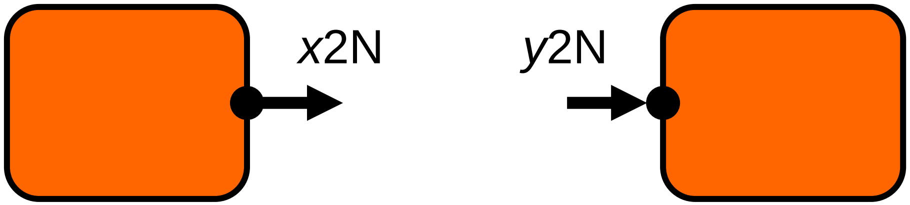
Top-down Compositionality:

If *g* ¸ *f* and *g’* ¸ *f’*, then *g*||*g’* ¸ *f*||*f’*.

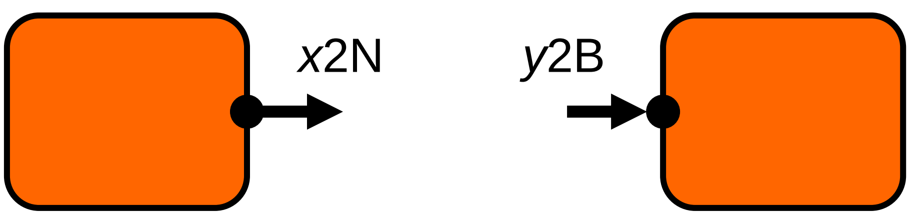
What’s the Difference?

If composition is a total function, then there is none.

However, Composition is often Partial.



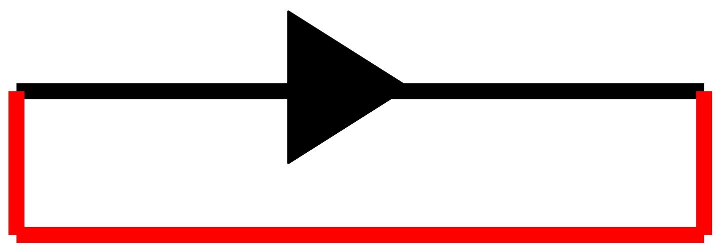
Compatible.



Incompatible.

However, Composition is often Partial.

Compatible.



Incompatible.

Partial Composition

Bottom-up Compositionality:

If $f||f'$ defined and $f \cdot g$ and $f' \cdot g'$,
then $g||g'$ defined and $f||f' \cdot g||g'$.

Top-down Compositionality:

If $g||g'$ defined and g ¸ f and g' ¸ f',
then $f||f'$ defined and $g||g'$ ¸ $f||f'$.

Partial Composition

Bottom-up Compositionality:

If *f*||*f’* defined and *f* · *g* and *f’* · *g’*,
then *g*||*g’* defined and *f*||*f’* · *g*||*g’*.

Processes

Top-down Compositionality:

If *g*||*g’* defined and *g* ¸ *f* and *g’* ¸ *f’*,
then *f*||*f’* defined and *g*||*g’* ¸ *f*||*f’*.

Interfaces

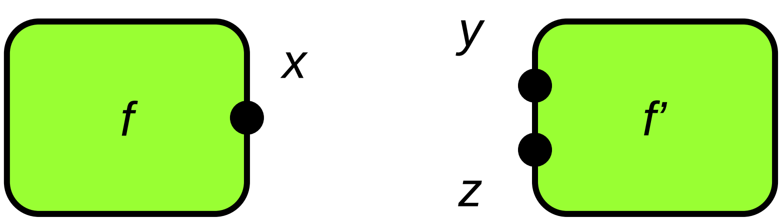
Block Diagram Algebra

-A set *P* of (typed) variables.

-A set *F* of *blocks*.

-For each block *f*2*F*, a set P_fµ*P* of *ports*.

-A partial binary function || on blocks, called *composition*.



Block Diagram Algebra

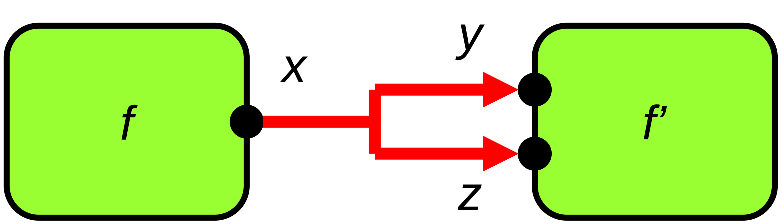
-A set *P* of (typed) variables.

-A set *F* of *blocks*.

-For each block *f*2*F*, a set P_fµ*P* of *ports*.

-A partial binary function || on blocks, called *composition*.

-A partial function mapping a block *f*2*F* and an interconnect *θ*2*P*£*P* to a block *Pθ*.



$\theta = \{ (x,y), (x,z) \}$

Block Diagram Algebra

-A set *P* of (typed) variables.

-A set *F* of *blocks*.

-For each block *f*2*F*, a set P_fµ*P* of *ports*.

-A partial binary function || on blocks, called *composition*.

-A partial function mapping a block *f*2*F* and an interconnect θ2*P*£*P* to a block *P*θ.

-A binary relation · on blocks, called *hierarchy*.

Side Conditions on Composition:

-if $f||g$ defined, then $g||f$ defined and equal

-if $(f||g)||h$ defined, then $f||(g||h)$ defined and equal

-if $f||g$ defined, then $P_{f||g} = P_f$ [P_g

Side Conditions on Connection:

$I_\theta = \{ x \mid (9\ y)(x,y)2\theta \}$

$O_\theta = \{ y \mid (9\ x)(x,y)2\theta \}$

$p_\theta = Æ_{(x,y)2\theta} (x=y)$

-if θ = ;, then $f\theta$ defined and equal to f

-if $f\theta$ defined, then $P_{f\theta} = P_f\ [\ I_\theta\ [\ O_\theta$

Side Conditions on Hierarchy:

-$f \cdot f$

-if $f \cdot g$ and $g \cdot h$, then $f \cdot h$

A block diagram algebra is an ***interface algebra*** if

1. if $g||h$ defined and g ¸ f, then $f||h$ defined and $g||h$ ¸ $f||h$

2. if $g\theta$ defined and g ¸ f, then $f\theta$ defined and $g\theta$ ¸ $f\theta$

A block diagram algebra is an ***interface algebra*** if

1. if $g||h$ defined and g ¸ f, then $f||h$ defined and $g||h$ ¸ $f||h$

2. if $g\theta$ defined and g ¸ f, then $f\theta$ defined and $g\theta$ ¸ $f\theta$

A block diagram algebra is a ***process algebra*** if

1. if $f||h$ defined and $f \cdot g$, then $g||h$ defined and $f||h \cdot g||h$

2. if $f\theta$ defined and $f \cdot g$, then $g\theta$ defined and $f\theta \cdot g\theta$

Stateless Input/Output Processes

$f = (I_f, O_f, p_f)$

I_f µ P ... *input ports*

O_f µ P \ I_f ... *output ports*

p_f ... *input/output relation:* predicate on I_f [O_f
such that *(8 I_f)(9 O_f) p_f*

Stateless Input/Output Processes

$f = (I_f, O_f, p_f)$ $g = (I_g, O_g, p_g)$

$P_f = I_f$ [O_f

$f||g$ *defined* if P_f Å P_g = ;

Then:

$I_{f||g} = I_f$ [I_g

$O_{f||g} = O_f$ [O_g

$p_{f||g} = p_f$ Æ p_g

$f = (I_f, O_f, p_f)$ θ µ I_θ £ O_θ

$f\theta$ *defined* if

1. I_f Å I_θ = ;
2. O_f Å O_θ = ;
3. (8 $I_{f\theta}$)(9 $O_{f\theta}$) $p_{f\theta}$ (as defined below)

Then:

$I_{f\theta}$ = (I_f [I_θ) \ $O_{f\theta}$

$O_{f\theta}$ = O_f [O_θ

$p_{f\theta}$ = p_f Æ p_θ

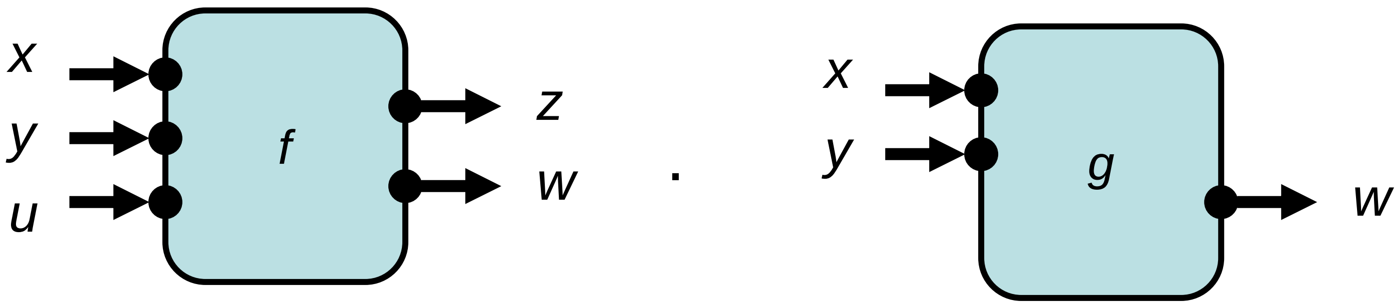
$f = (I_f, O_f, p_f)$ $g = (I_g, O_g, p_g)$

$f \cdot g$ if

1. I_f ¶ I_g
2. O_f ¶ O_g
3. p_f) p_g

abstraction has fewer inputs

and fewer outputs



z = *x* + *y* Æ

(*u*=0) *w*=*x*) Æ

(*u*≠ 0) *w*=*y*)

w=*x* Ç *w*=*y*

Why is this a process algebra?

1. if I_f Å I_θ = ; and I_f ¶ I_g, then I_g Å I_θ = ;
2. if O_f Å O_θ = ; and O_f ¶ O_g, then O_g Å O_θ = ;
3. if *(8* $I_{f\theta}$*)(9* $O_{f\theta}$*)(*p_f Æ p_θ*)* and p_f *)*p_g and $I_{f\theta}$ ¶ $I_{g\theta}$ and $O_{f\theta}$ ¶ $O_{g\theta}$, then *(8* $I_{g\theta}$*)(9* $O_{g\theta}$*)(*p_g Æ p_θ*)*

Stateless Input/Output Interfaces

$f = (I_f, O^+_f, O_f)$

O_f µ O^+_f

I_f Å O^+_f = ;

O^+_f ... set of *available* output port names

$P_f = I_f$ [O_f

$P^+_f = I_f$ [O^+_f

Stateless Input/Output Interfaces

$f = (I_f, O^+_f, O_f)$ $g = (I_g, O^+_g, O_g)$

$f||g$ *defined* if P^+_f Å P^+_g = ;

Then:

$I_{f||g} = I_f$ [I_g

$O^+_{f||g} = O^+_f$ [O^+_g

$O_{f||g} = O_f$ [O_q

$f = (I_f,\ O^+_f,\ O_f)$ θ µ I_θ £ O_θ

$f\theta$ *defined* if

1. I_θ µ O_f
2. O_θ Å O^+_f = ;
3. if $(x,y),(x',y')$2θ and $x \neq x'$, then $y \neq y'$

Then:

$I_{f\theta} = I_f$ \ O_θ

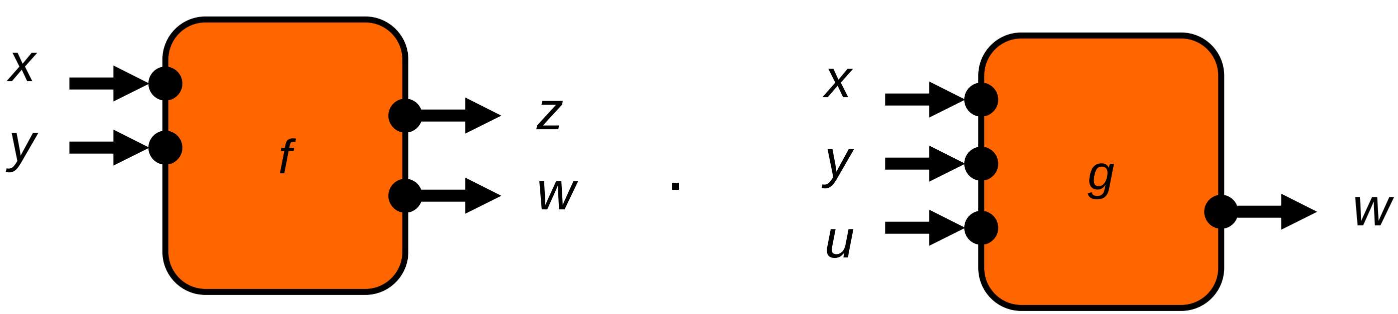
$O^+_{f\theta} = O^+_f$ [O_θ

$O_{f\theta} = O_f$ [O_θ

$f = (I_f, O^+_f, O_f)$ $g = (I_g, O^+_g, O_g)$

f · g if

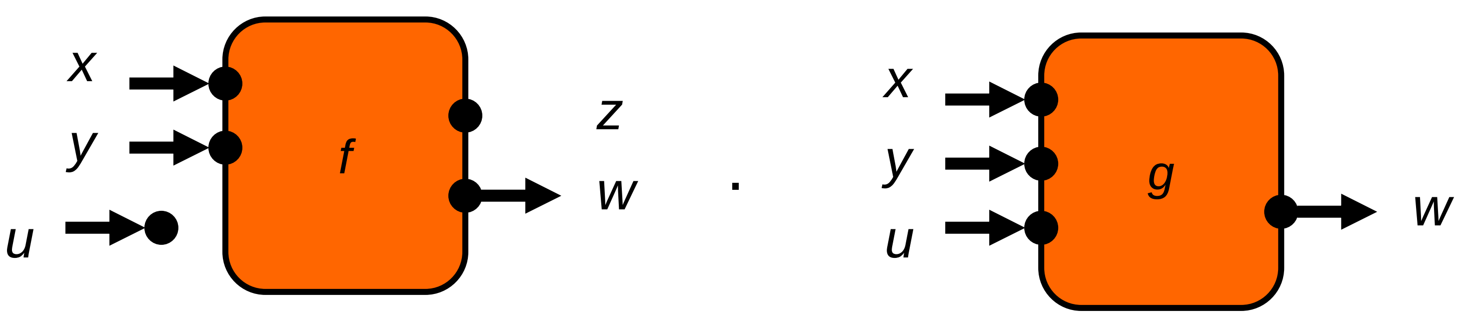
1. I_f µ I_g *refinement has **fewer** inputs*
2. O^+_f µ O^+_g
3. O_f ¶ O_g *and **more** outputs*



$f = (I_f, O^+_f, O_f)$ $g = (I_g, O^+_g, O_g)$

f · g if

1. I_f µ I_g *refinement has **fewer** inputs*
2. O^+_f µ O^+_g
3. O_f ¶ O_g *and **more** outputs*



Why is this an interface algebra?

1. if P^{+}_{g} Å P^{+}_{h} = ; and P^{+}_{g} ¶ P^{+}_{f}, then P^{+}_{f} Å P^{+}_{h} = ;
2. if I_{θ}µ O_{g} and O_{g} µ O_{f}, then I_{θ}µ O_{f}
3. if O^{+}_{g} Å O_{θ} = ; and O^{+}_{g} ¶ O^{+}_{f}, then O^{+}_{f} Å O_{θ} = ;

Stateless Assume/Guarantee Interfaces

$f = (I_f, O^+_f, O_f, in_f, out_f)$

in_f ... *input assumption*: satisfiable predicate on I_f

out_f ... *output guarantee*: satisfiable predicate on O_f

Stateless Assume/Guarantee Interfaces

$f = (I_f, O^+_f, O_f, in_f, out_f)$ $g = (I_g, O^+_g, O_g, in_g, out_g)$

$f||g$ *defined* if P^+_f Å P^+_g = ;

Then:

$in_{f||g} = in_f$ Æ in_g

$out_{f||g} = out_f$ Æ out_g

$f = (I_f, O^+_f, O_f, in_f, out_f)$ θ µ I_θ £ O_θ

$f\theta$ *defined* if

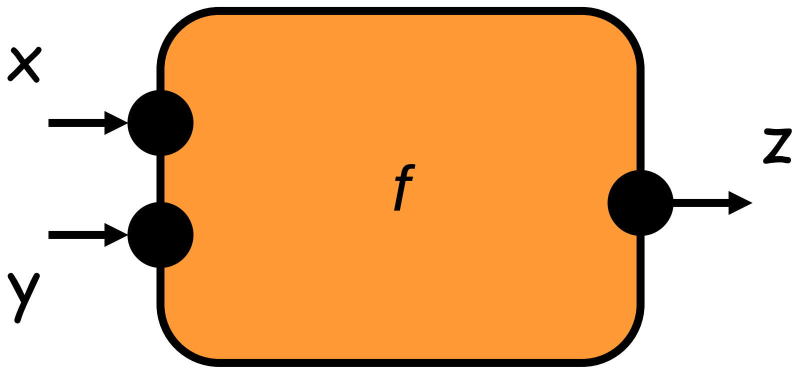
1-3 as before

4. $in_{f\theta}$ satisfiable (as defined below)

Then:

$in_{f\theta} = (8\ O_{f\theta})((out_{f\theta}$ Æ $p_\theta)\ )\ in_f)$

$out_{f\theta} = (9\ I_{f\theta})(out_f$ Æ $p_\theta)$



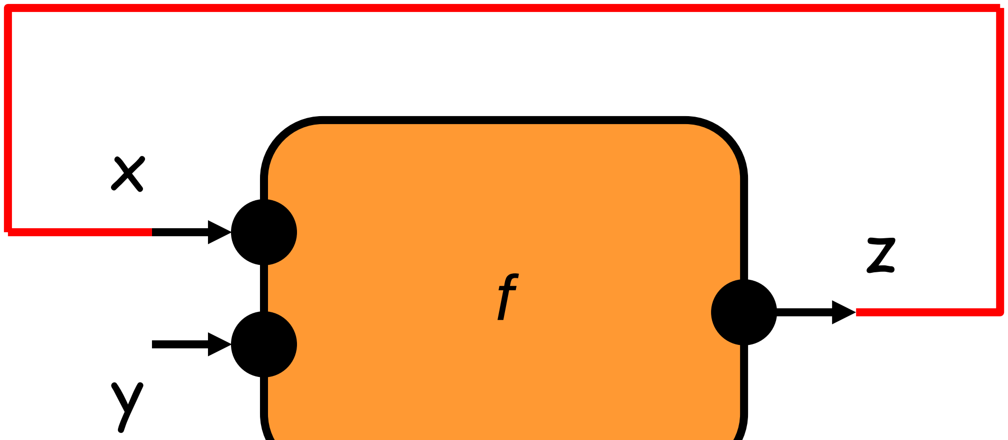
$x=0 \Rightarrow y=0$

true

Input assumption

Output guarantee

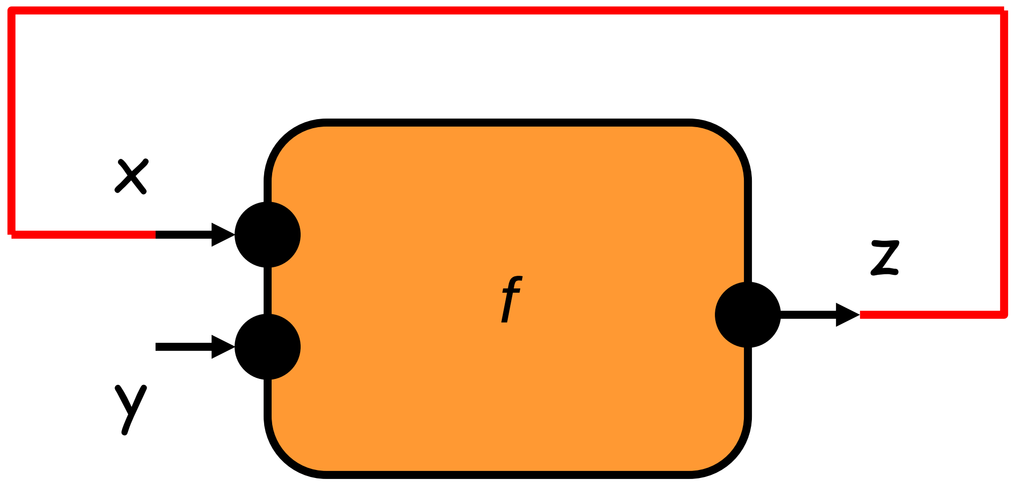
$\theta = \{(z,x)\}$



x=0 $\Rightarrow$ y=0

true

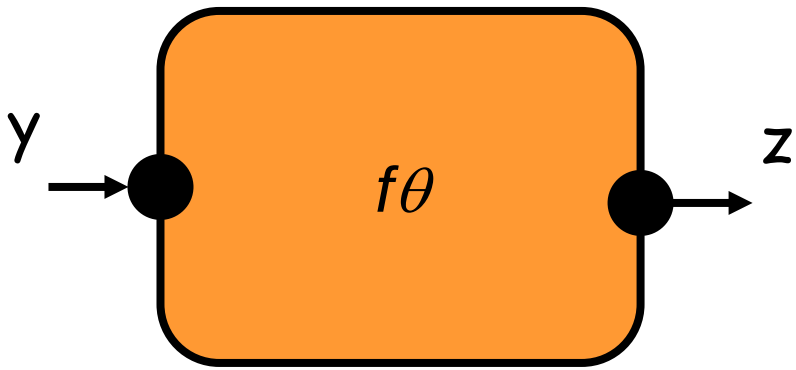
$\theta = \{(z,x)\}$



$x=0 \Rightarrow y=0$ true

$y = 0$

$(\forall x,z)\ ((\ true \wedge x=z\ ) \Rightarrow (\ x=0 \Rightarrow y=0\ ))$



y = 0

true

$f = (I_f, O^+_f, O_f, in_f, out_f)$ $g = (I_g, O^+_g, O_g, in_g, out_g)$

$f \cdot g$ if

1-3 as above

4. in_f (in_g

5. out_f) out_g

Next Lecture:

We will add *state*.

What is the Compositionality of your Favorite Model ?

Bottom-up Compositionality:

If *f*||*f’* defined and *f* · *g* and *f’* · *g’*,
then *g*||*g’* defined and *f*||*f’* · *g*||*g’*.

Processes

Top-down Compositionality:

If *g*||*g’* defined and *g* ¸ *f* and *g’* ¸ *f’*,
then *f*||*f’* defined and *g*||*g’* ¸ *f*||*f’*.

Interfaces

“defined” could be typeable, deadlock-free, etc.

Lesson 1:

Never dogmatically believe in one particular model / formalism / language.

Always evaluate your choices for your given special situation.