Data Transformation

user's variables u

implementer's variables v

Data Transformation

user's variables u

implementer's variables v

new implementer's variables w

Data Transformation

user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w \cdot \exists v \cdot D$

Data Transformation

user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \land S$

Data Transformation

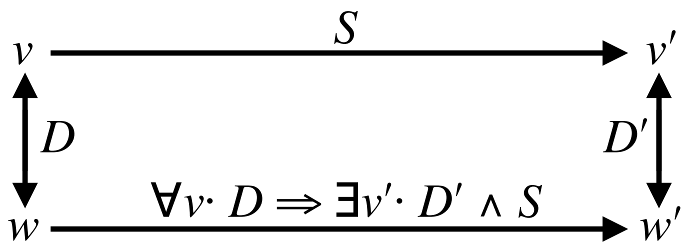
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w \cdot \exists v \cdot D$

specification S is transformed to $\forall v \cdot D \Rightarrow \exists v' \cdot D' \wedge S$



Data Transformation

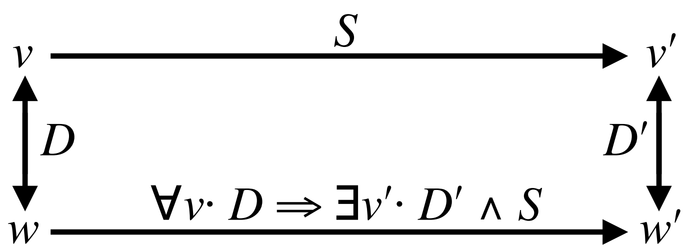
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w \cdot \exists v \cdot D$

specification S is transformed to $\forall v \cdot D \Rightarrow \exists v' \cdot D' \land S$



Data Transformation

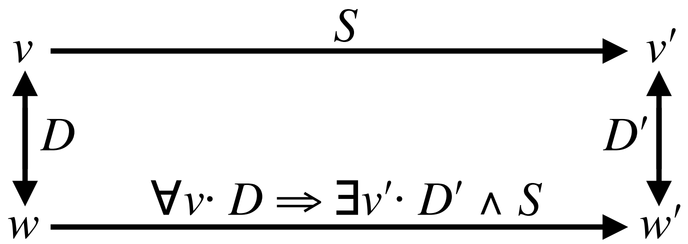
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \land S$



Data Transformation

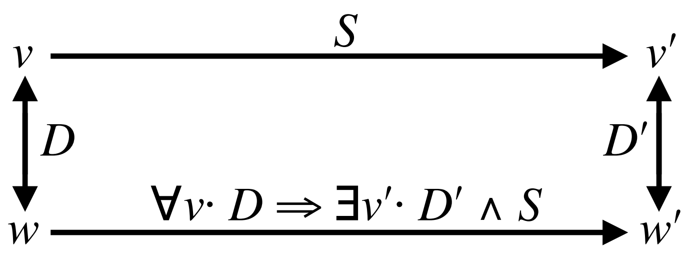
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \wedge S$ ←



Data Transformation

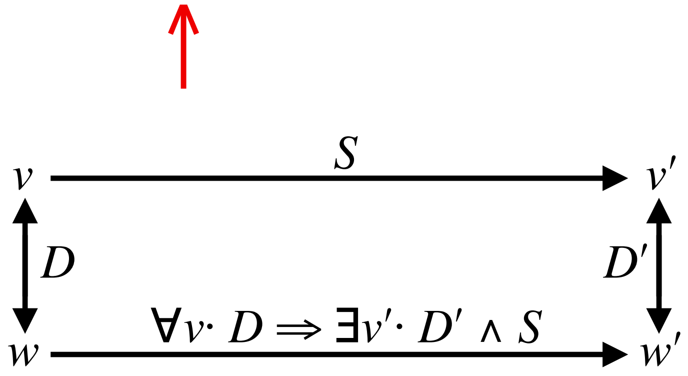
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \wedge S$



Data Transformation

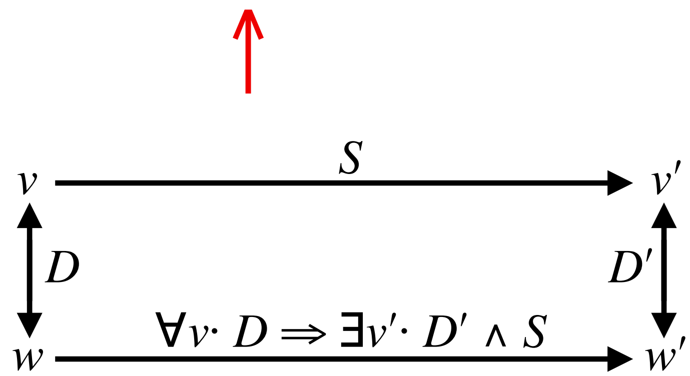
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \land S$



Data Transformation

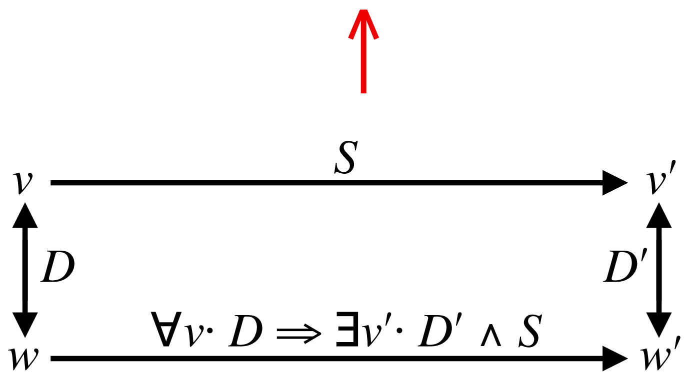
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w \cdot \exists v \cdot D$

specification S is transformed to $\forall v \cdot D \Rightarrow \exists v' \cdot D' \wedge S$



Data Transformation

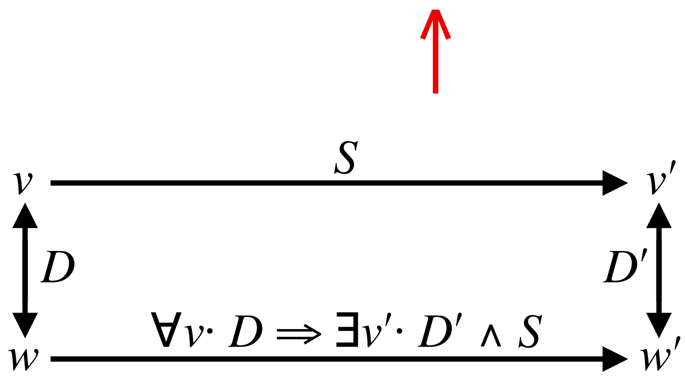
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \land S$



Data Transformation

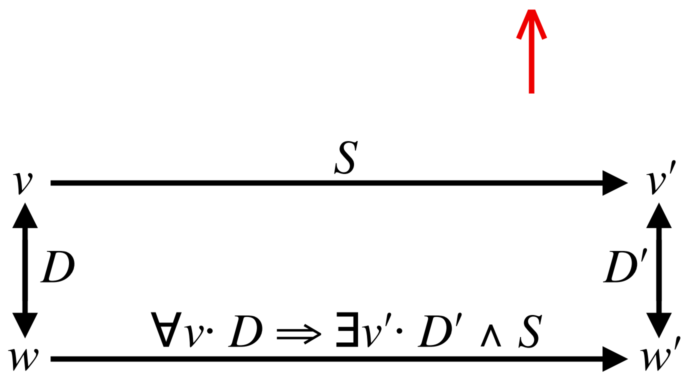
user's variables u

implementer's variables v

new implementer's variables w

data transformer D relates v and w such that $\forall w\cdot \exists v\cdot D$

specification S is transformed to $\forall v\cdot D \Rightarrow \exists v'\cdot D' \wedge S$



Data Transformation

example

user's variable *u*: *bin*

implementer's variable *v*: *nat*

Data Transformation

example

user's variable u: *bin*

implementer's variable v: *nat*

operations

$$zero \;=\; v := 0$$

$$increase \;=\; v := v+1$$

$$inquire \;=\; u := even\ v$$

Data Transformation

example

user's variable u: *bin*

implementer's variable v: *nat*

operations

$$zero \;=\; v:= 0$$

$$increase \;=\; v:= v+1$$

$$inquire \;=\; u:= even\ v$$

new implementer's variable w: *bin*

data transformer $w = even\ v$

Data Transformation

$\forall v \cdot D \;\Rightarrow\; \exists v' \cdot D' \;\;\land\;\; zero$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\wedge\; zero$$

$$=\quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\wedge\; (v:=0)$$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \land\ zero$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land\ (v{:=}\ 0)$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land\ u'{=}u\ \land\ v'{=}0$$

Data Transformation

$$\forall v \cdot D \;\Rightarrow\; \exists v' \cdot D' \;\wedge\; zero$$

$$= \quad \forall v \cdot w = even\ v \;\Rightarrow\; \exists v' \cdot w' = even\ v' \;\wedge\; (v := 0)$$

$$= \quad \forall v \cdot w = even\ v \;\Rightarrow\; \exists v' \cdot w' = even\ v' \;\wedge\; u' = u \;\wedge\; v' = 0 \qquad \text{1-pt}$$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \land\ zero$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ \exists v'\cdot\ w' = even\ v' \ \land\ (v := 0)$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ \exists v'\cdot\ w' = even\ v' \ \land\ u'=u \ \land\ v'=0 \qquad \text{1-pt}$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ w' = even\ 0 \ \land\ u'=u$$

Data Transformation

$$\forall v\cdot D \Rightarrow \exists v'\cdot D' \wedge zero$$

$$= \quad \forall v\cdot w = even\ v \Rightarrow \exists v'\cdot w' = even\ v' \wedge (v{:=}\,0)$$

$$= \quad \forall v\cdot w = even\ v \Rightarrow \exists v'\cdot w' = even\ v' \wedge u'{=}u \wedge v'{=}0 \qquad \text{1-pt}$$

$$= \quad \forall v\cdot w = even\ v \Rightarrow w' = even\ 0 \wedge u'{=}u \qquad \text{change variable}$$

$$\forall v{:}\ domain\cdot (\text{substitute } fv \text{ for } r \text{ in } b)$$

$$= \quad \forall r{:}\ f\ domain\cdot b$$

Data Transformation

$$\forall v \cdot D \;\Rightarrow\; \exists v' \cdot D' \;\wedge\; zero$$

$$= \quad \forall v \cdot w = even\ v \;\Rightarrow\; \exists v' \cdot w' = even\ v' \;\wedge\; (v := 0)$$

$$= \quad \forall v \cdot w = even\ v \;\Rightarrow\; \exists v' \cdot w' = even\ v' \;\wedge\; u'=u \;\wedge\; v'=0 \qquad \text{1-pt}$$

$$= \quad \forall v \cdot w = even\ v \;\Rightarrow\; w' = even\ 0 \;\wedge\; u'=u \qquad \text{change variable}$$

$$= \quad \forall r{:}\ even\ nat \cdot w=r \;\Rightarrow\; w'=\top \wedge u'=u$$

$$\forall v{:}\ domain \cdot (\text{substitute } fv \text{ for } r \text{ in } b)$$

$$= \quad \forall r{:}\ f\ domain \cdot b$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\land\; zero$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\land\; (v{:=}\,0)$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\land\; u'{=}u \;\land\; v'{=}0 \qquad \text{1-pt}$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; w' = even\ 0 \;\land\; u'{=}u \qquad \text{change variable}$$

$$= \quad \forall r\colon even\ nat\cdot w{=}r \;\Rightarrow\; w'{=}\top \land u'{=}u$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\land\; zero$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\land\; (v{:=}\,0)$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\land\; u'{=}u \;\land\; v'{=}0 \qquad \text{1-pt}$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; w' = even\ 0 \;\land\; u'{=}u \qquad \text{change variable}$$

$$= \quad \forall r\colon even\ nat\cdot w{=}r \;\Rightarrow\; w'{=}\top \land u'{=}u \qquad \text{1-pt}$$

$$= \quad w'{=}\top \;\land\; u'{=}u$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\wedge\; zero$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\wedge\; (v{:=}\ 0)$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; \exists v'\cdot w' = even\ v' \;\wedge\; u'{=}u \;\wedge\; v'{=}0 \qquad \text{1-pt}$$

$$= \quad \forall v\cdot w = even\ v \;\Rightarrow\; w' = even\ 0 \;\wedge\; u'{=}u \qquad \text{change variable}$$

$$= \quad \forall r{:}\ even\ nat\cdot w{=}r \;\Rightarrow\; w'{=}\top \wedge u'{=}u \qquad \text{1-pt}$$

$$= \quad w'{=}\top \;\wedge\; u'{=}u$$

$$= \quad w{:=}\ \top$$

Data Transformation

$\forall v \cdot D \Rightarrow \exists v' \cdot D' \wedge increase$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \land\ increase$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ \exists v'\cdot\ w' = even\ v' \ \land\ (v{:=}\ v+1)$$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \wedge\ increase$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ (v{:=}\ v{+}1)$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ u'{=}u\ \wedge\ v'{=}v{+}1$$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \wedge\ increase$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ (v{:=}\ v{+}1)$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ u'{=}u\ \wedge\ v'{=}v{+}1 \qquad \text{1-pt}$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ w' = even\ (v{+}1)\ \wedge\ u'{=}u$$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \wedge\ increase$$

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ (v{:=}\ v{+}1)$

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \wedge\ u'{=}u\ \wedge\ v'{=}v{+}1$ 1-pt

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ w' = even\ (v{+}1)\ \wedge\ u'{=}u$ change var

$= \quad \forall r{:}\ even\ nat\cdot\ w{=}r\ \Rightarrow\ w' = \neg r\ \wedge\ u'{=}u$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \land\ increase$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land\ (v{:=}\ v{+}1)$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land\ u'{=}u\ \land\ v'{=}v{+}1 \qquad \text{1-pt}$$

$$=\quad \forall v\cdot\ w = even\ v\ \Rightarrow\ w' = even\ (v{+}1)\ \land\ u'{=}u \qquad \text{change var}$$

$$=\quad \forall r{:}\ even\ nat\cdot\ w{=}r\ \Rightarrow\ w' = \neg r\ \land\ u'{=}u \qquad \text{1-pt}$$

$$=\quad w' = \neg w\ \land\ u'{=}u$$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \wedge\ increase$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ \exists v'\cdot\ w' = even\ v' \ \wedge\ (v:= v+1)$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ \exists v'\cdot\ w' = even\ v' \ \wedge\ u'=u \ \wedge\ v'=v+1 \qquad \text{1-pt}$$

$$=\quad \forall v\cdot\ w = even\ v \ \Rightarrow\ w' = even\ (v+1) \ \wedge\ u'=u \qquad \text{change var}$$

$$=\quad \forall r{:}\ even\ nat\cdot\ w=r \ \Rightarrow\ w' = \neg r \ \wedge\ u'=u \qquad \text{1-pt}$$

$$=\quad w' = \neg w \ \wedge\ u'=u$$

$$=\quad w:= \neg w$$

Data Transformation

$\forall v \cdot D \;\Rightarrow\; \exists v' \cdot D' \;\land\; inquire$

Data Transformation

$$\forall v\cdot\ D\ \Rightarrow\ \exists v'\cdot\ D'\ \land\ inquire$$

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land\ (u{:=}\ even\ v)$

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ \exists v'\cdot\ w' = even\ v'\ \land u' = even\ v \land v'{=}v$ 1-pt

$= \quad \forall v\cdot\ w = even\ v\ \Rightarrow\ w' = even\ v\ \land\ u' = even\ v$ change var

$= \quad \forall r{:}\ even\ nat\cdot\ w{=}r\ \Rightarrow\ w'{=}r\ \land\ u'{=}r$ 1-pt

$= \quad w'{=}w\ \land\ u'{=}w$

$= \quad u{:=}\ w$

Data Transformation

example

user's variable u: *bin*

implementer's variable v: *bin*

operations

$$set \;=\; v:= \top$$

$$flip \;=\; v:= \neg v$$

$$ask \;=\; u:= v$$

new implementer's variable w: *nat*

data transformer $v = even\ w$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \wedge\ set$$

$$=\quad \forall v\cdot\ v = even\ w \ \Rightarrow\ \exists v'\cdot\ v' = even\ w' \ \wedge\ (v{:=}\ \top)$$

$$=\quad even\ w' \ \wedge\ u'{=}u$$

$$\Leftarrow\quad w{:=}\ 0$$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \land\ set$$

$$=\quad \forall v\cdot\ v = even\ w \ \Rightarrow\ \exists v'\cdot\ v' = even\ w' \ \land\ (v{:=}\ \top)$$

$$=\quad even\ w' \ \land\ u'{=}u \quad \longleftarrow$$

$$\Leftarrow\quad w{:=}\ 0$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\land\; set$$

$$= \quad \forall v\cdot v = even\ w \;\Rightarrow\; \exists v'\cdot v' = even\ w' \;\land\; (v{:=}\ \top)$$

$$= \quad even\ w' \;\land\; u'{=}u$$

$$\Leftarrow \quad w{:=}\ 0 \quad \longleftarrow$$

Data Transformation

$$\forall v \cdot D \;\Rightarrow\; \exists v' \cdot D' \;\wedge\; flip$$

$$= \qquad \forall v \cdot v = even\ w \;\Rightarrow\; \exists v' \cdot v' = even\ w' \;\wedge\; (v := \neg v)$$

$$= \qquad even\ w' \ne even\ w \;\wedge\; u'=u$$

$$\Leftarrow \qquad w := w+1$$

Data Transformation

$$\forall v\cdot\ D \ \Rightarrow\ \exists v'\cdot\ D' \ \land\ ask$$

$$=\quad \forall v\cdot\ v = even\ w \ \Rightarrow\ \exists v'\cdot\ v' = even\ w' \ \land\ (u{:=}\ v)$$

$$=\quad even\ w' = even\ w = u'$$

$$\Leftarrow\quad u{:=}\ even\ w$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\wedge\; ask$$

$$=\quad \forall v\cdot v = even\ w \;\Rightarrow\; \exists v'\cdot v' = even\ w' \;\wedge\; (u{:=}\ v)$$

$$=\quad even\ w' = even\ w = u' \qquad \longleftarrow$$

$$\Leftarrow\quad u{:=}\ even\ w$$

Data Transformation

$$\forall v\cdot D \;\Rightarrow\; \exists v'\cdot D' \;\wedge\; ask$$

$$= \qquad \forall v\cdot v = even\ w \;\Rightarrow\; \exists v'\cdot v' = even\ w' \;\wedge\; (u{:=}\ v)$$

$$= \qquad even\ w' = even\ w = u'$$

$$\Leftarrow \qquad u{:=}\ even\ w \qquad \longleftarrow$$

Security Switch

A security switch has three binary user's variables a , b , and c . The users assign values to a and b as input to the switch. The switch's output is assigned to c . The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

Security Switch

A security switch has three binary user's variables a, b, and c. The users assign values to a and b as input to the switch. The switch's output is assigned to c. The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

binary implementer's variables

A records the state of input a at last output change

B records the state of input b at last output change

Security Switch

A security switch has three binary user's variables a , b , and c . The users assign values to a and b as input to the switch. The switch's output is assigned to c . The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

Security Switch

A security switch has three binary user's variables a, b, and c. The users assign values to a and b as input to the switch. The switch's output is assigned to c. The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a:=\neg a.$ **if** $a \neq A \wedge b \neq B$ **then** $c:=\neg c.\ A:=a.\ B:=b$ **else** ok **fi**

Security Switch

A security switch has three binary user's variables a , b , and c . The users assign values to a and b as input to the switch. The switch's output is assigned to c . The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$$a:= \neg a.\ \textbf{if}\ a{\ne}A \wedge b{\ne}B\ \textbf{then}\ c:= \neg c.\ A:= a.\ B:= b\ \textbf{else}\ ok\ \textbf{fi}$$

↑

Security Switch

A security switch has three binary user's variables a , b , and c . The users assign values to a and b as input to the switch. The switch's output is assigned to c . The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a{:=}\ \neg a.$ **if** $a{\neq}A \wedge b{\neq}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

Security Switch

A security switch has three binary user's variables a , b , and c . The users assign values to a and b as input to the switch. The switch's output is assigned to c . The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a{:=}\ \neg a.$ **if** $a{\ne}A \wedge b{\ne}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** *ok* **fi**

Security Switch

A security switch has three binary user's variables a, b, and c. The users assign values to a and b as input to the switch. The switch's output is assigned to c. The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a:= \neg a.$ **if** $a \neq A \land b \neq B$ **then** $c:= \neg c.\ A:= a.\ B:= b$ **else** ok **fi**

Security Switch

A security switch has three binary user's variables a, b, and c. The users assign values to a and b as input to the switch. The switch's output is assigned to c. The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a:= \neg a.$ **if** $a \ne A \wedge b \ne B$ **then** $c:= \neg c.\ A:= a.\ B:= b$ **else** ok **fi**

↑

Security Switch

A security switch has three binary user's variables a, b, and c. The users assign values to a and b as input to the switch. The switch's output is assigned to c. The output changes when both inputs have changed. More precisely, the output changes when both inputs differ from what they were the previous time the output changed. The idea is that one user might flip their input indicating a desire for the output to change, but the output does not change until the other user flips their input indicating agreement that the output should change. If the first user changes back before the second user changes, the output does not change.

operations

$a{:=}\ \neg a.\ \ \textbf{if}\ a{\neq}A \land b{\neq}B\ \textbf{then}\ c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b\ \textbf{else}\ ok\ \textbf{fi}$

$b{:=}\ \neg b.\ \ \textbf{if}\ a{\neq}A \land b{\neq}B\ \textbf{then}\ c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b\ \textbf{else}\ ok\ \textbf{fi}$

Security Switch

replace old implementer's variables A and B with nothing!

operations

$a{:=}\ \neg a.\ $ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

$b{:=}\ \neg b.\ $ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

Security Switch

replace old implementer's variables A and B with nothing!

data transformer

$A=B=c$

operations

$a:=\neg a.$ **if** $a \neq A \wedge b \neq B$ **then** $c:=\neg c.\ A:=a.\ B:=b$ **else** ok **fi**

$b:=\neg b.$ **if** $a \neq A \wedge b \neq B$ **then** $c:=\neg c.\ A:=a.\ B:=b$ **else** ok **fi**

Security Switch

replace old implementer's variables A and B with nothing!

data transformer

$A=B=c$

proof

$\forall new\cdot\ \exists old\cdot\ transformer$

operations

$a{:=}\ \neg a.\ $ **if** $a{\neq}A\ \wedge\ b{\neq}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$ **else** ok **fi**

$b{:=}\ \neg b.\ $ **if** $a{\neq}A\ \wedge\ b{\neq}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$ **else** ok **fi**

Security Switch

replace old implementer's variables A and B with nothing!

data transformer

$A=B=c$

proof

$\exists A, B\cdot\ A=B=c$

operations

$a:= \neg a.\ $ **if** $a \neq A \wedge b \neq B$ **then** $c:= \neg c.\ A:= a.\ B:= b$ **else** ok **fi**

$b:= \neg b.\ $ **if** $a \neq A \wedge b \neq B$ **then** $c:= \neg c.\ A:= a.\ B:= b$ **else** ok **fi**

Security Switch

replace old implementer's variables A and B with nothing!

data transformer

$A=B=c$

proof

$\exists A, B\cdot\ A=B=c$ generalization, using c for both A and B

$\Leftarrow\quad \top$

operations

$a{:=}\ \neg a.\ $ **if** $a{\ne}A \wedge b{\ne}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

$b{:=}\ \neg b.\ $ **if** $a{\ne}A \wedge b{\ne}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

Security Switch

operations

$a := \neg a.\ $ **if** $a \neq A \wedge b \neq B$ **then** $c := \neg c.\ A := a.\ B := b$ **else** ok **fi**

$b := \neg b.\ $ **if** $a \neq A \wedge b \neq B$ **then** $c := \neg c.\ A := a.\ B := b$ **else** ok **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c'\ \land$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$
else ok **fi**

operations

$a{:=}\ \neg a.$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$ **else** ok **fi**

$b{:=}\ \neg b.$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$ **else** ok **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \ \land$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else *ok* **fi**

Security Switch

$\forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\neg c.\ A{:=}a.\ B{:=}b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

Security Switch

$$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land \quad \textbf{if } a{\ne}A \land b{\ne}B \textbf{ then } c{:=}\neg c.\ A{:=}a.\ B{:=}b \textbf{ else } ok \textbf{ fi}$$

expand assignments, sequential compositions, and *ok*

$$= \quad \forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land \quad \textbf{if } a{\ne}A \land b{\ne}B \textbf{ then } a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b \textbf{ else } a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B \textbf{ fi}$$

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=\quad \forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \wedge$ **if** $a{\ne}A \wedge b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \wedge$ **if** $a{\ne}A \wedge b{\ne}B$

then $a'{=}a \wedge b'{=}b \wedge c'{=}\neg c \wedge A'{=}a \wedge B'{=}b$

else $a'{=}a \wedge b'{=}b \wedge c'{=}c \wedge A'{=}A \wedge B'{=}B$ **fi**

use one-point law for A and B, and for A' and B'

$=$ **if** $a{\ne}c \wedge b{\ne}c$ **then** $a'{=}a \wedge b'{=}b \wedge c'{=}\neg c \wedge c'{=}a \wedge c'{=}b$

else $a'{=}a \wedge b'{=}b \wedge c'{=}c \wedge c'{=}c \wedge c'{=}c$ **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\neg c.\ A{:=}a.\ B{:=}b$
else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$
then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$
else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

$\downarrow$ use one-point law for A and B, and for A' and B'

$=$ **if** $a{\ne}c \land b{\ne}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$
else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else *ok* **fi**

expand assignments, sequential compositions, and *ok*

$=$ $\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \land$ **if** $a{\ne}A \land b{\ne}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

use one-point law for A and B, and for A' and B'

$=$ **if** $a{\ne}c \land b{\ne}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

Security Switch

$\forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\neg c.\ A{:=}a.\ B{:=}b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

↓ use one-point law for A and B, and for A' and B'

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \wedge$ **if** $a{\ne}A \wedge b{\ne}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \wedge$ **if** $a{\ne}A \wedge b{\ne}B$

then $a'{=}a \wedge b'{=}b \wedge c'{=}\neg c \wedge A'{=}a \wedge B'{=}b$

else $a'{=}a \wedge b'{=}b \wedge c'{=}c \wedge A'{=}A \wedge B'{=}B$ **fi**

↓ use one-point law for A and B, and for A' and B'

$=$ **if** $a{\ne}c \wedge b{\ne}c$ **then** $a'{=}a \wedge b'{=}b \wedge c'{=}\neg c \wedge c'{=}a \wedge c'{=}b$

else $a'{=}a \wedge b'{=}b \wedge c'{=}c \wedge c'{=}c \wedge c'{=}c$ **fi** ↑

Security Switch

$\forall A, B\cdot A{=}B{=}c \Rightarrow \exists A', B'\cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\neg c.\ A{:=}a.\ B{:=}b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B\cdot A{=}B{=}c \Rightarrow \exists A', B'\cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

use one-point law for A and B, and for A' and B'

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$ use context

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}\neg c \land c'{=}\neg c$

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

Security Switch

$\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \ \land$ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\ \neg c.\ \ A{:=}\ a.\ \ B{:=}\ b$

else ok **fi**

expand assignments, sequential compositions, and ok

$=$ $\forall A, B\cdot\ A{=}B{=}c \Rightarrow \exists A', B'\cdot\ A'{=}B'{=}c' \ \land$ **if** $a{\neq}A \land b{\neq}B$

then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$

else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

use one-point law for A and B, and for A' and B'

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$ use context

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}\neg c \land c'{=}\neg c$

else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

$=$ **if** $a{\neq}c \land b{\neq}c$ **then** $c{:=}\ \neg c$ **else** ok **fi**

Security Switch

$\forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$ **then** $c{:=}\ \neg c.\ A{:=}\ a.\ B{:=}\ b$ **else** ok **fi**

expand assignments, sequential compositions, and ok

$=\quad \forall A, B \cdot A{=}B{=}c \Rightarrow \exists A', B' \cdot A'{=}B'{=}c' \land$ **if** $a{\neq}A \land b{\neq}B$
then $a'{=}a \land b'{=}b \land c'{=}\neg c \land A'{=}a \land B'{=}b$
else $a'{=}a \land b'{=}b \land c'{=}c \land A'{=}A \land B'{=}B$ **fi**

use one-point law for A and B, and for A' and B'

$=\quad$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}a \land c'{=}b$ — use context
else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

$=\quad$ **if** $a{\neq}c \land b{\neq}c$ **then** $a'{=}a \land b'{=}b \land c'{=}\neg c \land c'{=}\neg c \land c'{=}\neg c$
else $a'{=}a \land b'{=}b \land c'{=}c \land c'{=}c \land c'{=}c$ **fi**

$=\quad$ **if** $a{\neq}c \land b{\neq}c$ **then** $c{:=}\ \neg c$ **else** ok **fi**

$=\quad c{:=}\ (a{\neq}c \land b{\neq}c) \neq c$