# An explanation of forwards/backwards simulation

*Maarten Fokkinga, Version of October 24, 2005*

Forwards and backwards simulation are techniques to prove that a concrete program satisfies the specification that has been formulated in terms of an abstract program. There are cases where forwards simulation can be applied and not backwards simulation; the converse holds as well.

All this is probably folklore, but I've never seen so clear a presentation as the one that I now present myself…

**Simulations.** We consider two sets, $I$ and $F$ (initial and final) and we are interested in programs that go from $I$ to $F$. The programs will be *specified* "at the abstract level" by means of a set $A$, an initialisation $ai : I \to A$, various operations $a, a', \ldots : A \to A$, and a finalization $af : A \to F$. Actually the programs are *implemented* "at a concrete level", working on a set $C$, and they are composed of an initialization $ci : I \to C$, various operations $c, c', \ldots : C \to C$ (where $c$ corresponds to $a$, and $c'$ to $a'$, etc), and a finalization $cf : C \to F$:

$$\begin{tikzcd}
 & A \arrow[r, "a"] & A \arrow[r, dashed] & A \arrow[r, "a''"] & A \arrow[dr, "af"] & \\
I \arrow[ur, "ai"] \arrow[dr, "ci"'] & & & & & F \\
 & C \arrow[r, "c"'] & C \arrow[r, dashed] & C \arrow[r, "c''"'] & C \arrow[ur, "cf"'] &
\end{tikzcd}$$

In view of the abstract level as the specification, and the concrete level as an implementation, we require that each input-output pair of the concrete program can also be yielded by the abstract one:

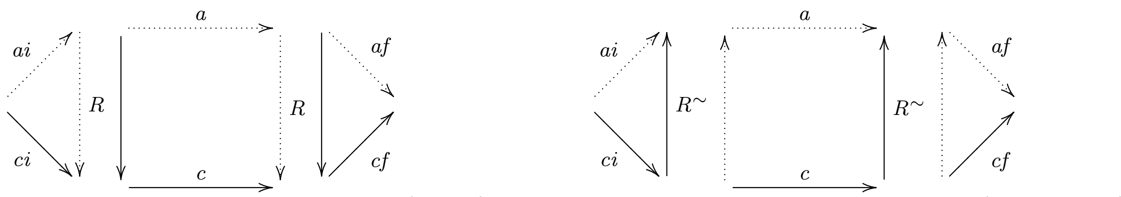
$$ci \mathbin{;} c \mathbin{;} c' \mathbin{;} \cdots \mathbin{;} c'' \mathbin{;} cf \quad \subseteq \quad ai \mathbin{;} a \mathbin{;} a' \mathbin{;} \cdots \mathbin{;} a'' \mathbin{;} af$$

This inclusion is phrased "the abstract program *simulates* the concrete one".

**Proving simulation.** We wish to have a way to prove simulation by "step-by-step simulation". In order to be able to do so, we must be able to relate intermediate results at the abstract and concrete level to each other. To this end we consider a relation $R : A \,—\, C$:

$$\begin{tikzcd}
 & A \arrow[r, "a"] \arrow[dd, dash, dotted, "R"] & A \arrow[r, dashed] \arrow[dd, dash, dotted, "R"] & A \arrow[r, "a''"] \arrow[dd, dash, dotted, "R"] & A \arrow[dr, "af"] \arrow[dd, dash, dotted, "R"] & \\
I \arrow[ur, "ai"] \arrow[dr, "ci"'] & & & & & F \\
 & C \arrow[r, "c"'] & C \arrow[r, dashed] & C \arrow[r, "c''"'] & C \arrow[ur, "cf"'] &
\end{tikzcd}$$

We expect that if all triangles and squares commute, then so do the entire $a$ and $c$ paths. There are however two obvious ways in which the squares can commute (and the triangles analoguously):



$ci \subseteq ai \fatsemi R$ $\quad R \fatsemi c \subseteq a \fatsemi R$ $\quad R \fatsemi cf \subseteq af$ $\quad ci \fatsemi R^{\sim} \subseteq ai$ $\quad c \fatsemi R^{\sim} \subseteq R^{\sim} \fatsemi a$ $\quad cf \subseteq R^{\sim} \fatsemi af$

This leads to the following definitions:

- $R$ is a *forwards* simulation if:
  $ci \subseteq ai \fatsemi R$ and $R \fatsemi cf \subseteq af$ and for all corresponding $a$ and $c$, we have $R \fatsemi c \subseteq a \fatsemi R$.

- $R$ is an *backwards* simulation if:
  $ci \fatsemi R^{\sim} \subseteq ai$ and $cf \subseteq R^{\sim} \fatsemi af$ and for all corresponding $a$ and $c$, we have $c \fatsemi R^{\sim} \subseteq R^{\sim} \fatsemi a$.

(The name *forwards* and *backwards* will be clarified below.) We are now ready to fulfill our wish.

**Theorem.** If $R$ is a forwards or backwards simulation, then the abstract program simulates the concrete one.
Proof.

In case $R$ is a forwards simulation:

$$
\begin{array}{ll}
 & ci \fatsemi c \fatsemi c' \fatsemi \cdots \fatsemi c'' \fatsemi cf \\
\subseteq & \\
 & ai \fatsemi R \fatsemi c \fatsemi c' \fatsemi \cdots \fatsemi c'' \fatsemi cf \\
\subseteq & \\
 & ai \fatsemi a \fatsemi R \fatsemi c' \fatsemi \cdots \fatsemi c'' \fatsemi cf \\
\subseteq & \\
 & ai \fatsemi a \fatsemi a' \fatsemi R \fatsemi \cdots \fatsemi c'' \fatsemi cf \\
\vdots & \\
 & ai \fatsemi a \fatsemi a' \fatsemi \cdots \fatsemi R \fatsemi c'' \fatsemi cf \\
\subseteq & \\
 & ai \fatsemi a \fatsemi a' \fatsemi \cdots \fatsemi a'' \fatsemi R \fatsemi cf \\
\subseteq & \\
 & ai \fatsemi a \fatsemi a' \fatsemi \cdots \fatsemi a'' \fatsemi af
\end{array}
$$

In case $R$ is a backwards simulation:

$$
\begin{array}{ll}
 & ci \fatsemi c \fatsemi c' \fatsemi \cdots \fatsemi c'' \fatsemi cf \\
\subseteq & \\
 & ci \fatsemi c \fatsemi c' \fatsemi \cdots \fatsemi c'' \fatsemi R^{\sim} \fatsemi af \\
\subseteq & \\
 & ci \fatsemi c \fatsemi c' \fatsemi \cdots \fatsemi R^{\sim} \fatsemi a'' \fatsemi af \\
\vdots & \\
 & ci \fatsemi c \fatsemi c' \fatsemi R^{\sim} \fatsemi \cdots \fatsemi a'' \fatsemi af \\
\subseteq & \\
 & ci \fatsemi c \fatsemi R^{\sim} \fatsemi a' \fatsemi \cdots \fatsemi a'' \fatsemi af \\
\subseteq & \\
 & ci \fatsemi R^{\sim} \fatsemi a \fatsemi a' \fatsemi \cdots \fatsemi a'' \fatsemi af \\
\subseteq & \\
 & ai \fatsemi a \fatsemi a' \fatsemi \cdots \fatsemi a'' \fatsemi af
\end{array}
$$

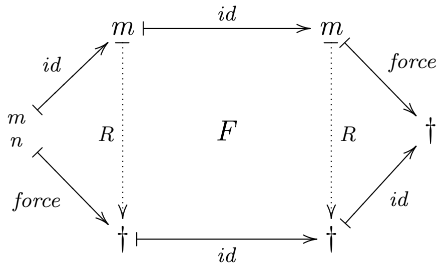
Notice how in the proof in the forwards case, a longer and longer initial part of the concrete program (growing to the tail) is replaced by the "simulating" program. In the backwards case, a longer and longer final part (growing backwards to the front) is replaced by the simulating program. This explains the naming *forwards* and *backwards*. It could already have been seen in the explanatory diagram for the definitions of forwards and backwards simulation: the thick $R$ arrow indicates the position up to which commutativity should already hold, and the dotted $R$ arrow then indicates the position to which commutativity then extends. In the forwards case, the postion moves forwards, whereas in the backwards case the position moves backwards.

**Examples.** The funny thing is: sometimes there exists a forwards and no backwards simulatiom, sometimes the converse holds.

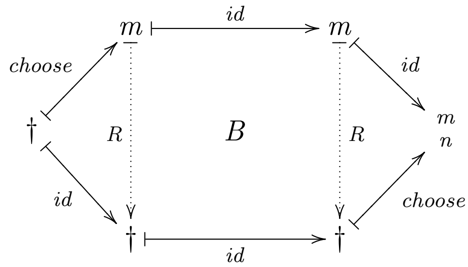
Let *One* $= \{\dagger\}$ be a one-point set, whose only element is denoted $\dagger$. We define two operations that are each others dual:

$$
\begin{array}{lll}
\mathit{force} = \{n : \mathbb{N} \bullet n \mapsto \dagger\} & & \text{non-injectivity is essential} \\
\mathit{choose} = \{n : \mathbb{N} \bullet \dagger \mapsto n\} & & \text{non-functionality is essential}
\end{array}
$$

Now consider the following two cases (diagram F at the left, diagram B at the right):




$\mathit{force} \fatsemi \mathit{id} \fatsemi \mathit{id} \subseteq \mathit{id} \fatsemi \mathit{id} \fatsemi \mathit{force}$




$\mathit{id} \fatsemi \mathit{id} \fatsemi \mathit{choose} \subseteq \mathit{choose} \fatsemi \mathit{id} \fatsemi \mathit{id}$

In both cases the abstract program equals the concrete one, so it certainly simulates the concrete one. The simulation in the F case can be proved by forwards simulation and not by backwards simulation; whereas the converse holds for the B case:

**Case F:** Taking $R = \mathbb{N} \times \mathit{One}$, the conditions for forwards simulation are readily verified:

$$\mathit{force} \subseteq \mathit{id} \fatsemi R, \quad R \fatsemi \mathit{id} \subseteq \mathit{id} \fatsemi R, \quad R \fatsemi \mathit{id} \subseteq \mathit{force}.$$

Regarding backwards simulation, the condition '$\mathit{ci} \fatsemi R^{\sim} \subseteq \mathit{ai}$' will be false for all $R \neq \varnothing$:

$$\mathit{force} \fatsemi R^{\sim} \not\subseteq \mathit{id}.$$

**Case B:** Taking $R = \mathbb{N} \times \mathit{One}$, the conditions for backwards simulation are readily verified:

$$\mathit{id} \fatsemi R^{\sim} \subseteq \mathit{choose}, \quad \mathit{id} \fatsemi R^{\sim} \subseteq R^{\sim} \fatsemi \mathit{id}, \quad \mathit{choose} \subseteq R^{\sim} \fatsemi \mathit{id}.$$

Regarding forwards simulation, the condition '$R \fatsemi \mathit{cf} \subseteq \mathit{af}$' will be false for all $R \neq \varnothing$:

$$R \fatsemi \mathit{choose} \not\subseteq \mathit{id}.$$

In less formal and more intuitive terms, the example reads "*initialize* $\fatsemi$ *compute* $\fatsemi$ *finalize*" where:

F:

| operation name | specification | implementation |
|---|---|---|
| *initialize* | *do nothing* | *print "accepted!"* |
| *compute* | *do nothing* | *do nothing* |
| *finalize* | *print "accepted!"* | *do nothing* |

B:

| operation name | specification | implementation |
|---|---|---|
| *initialize* | *pick a random number* | *do nothing* |
| *compute* | *do nothing* | *do nothing* |
| *finalize* | *print that number* | *print a random number* |