

# System Modelling and Design

An Introduction to the B Method

Proof: Discharging Proof Obligations

Revision: 1.1, November 16, 2005

Ken Robinson

April 3, 2006

©Ken Robinson 2005

mailto::k.robinson@unsw.edu.au

## Contents

### 1 Objectives of this Lecture

1

### 1 Objectives of this Lecture

To understand the basic ideas of discharging proof obligations.

To understand the use of inference rules, for forward or backward inference.

To gain a basic understand the use of the BToolkit *BToolProver* for discharging proof obligations.

A proof obligation has the form:

$$\begin{array}{c} \textit{hypotheses} \\ \Rightarrow \\ \textit{goal} \end{array}$$

The hypotheses are coded as:

<i>cst</i> (Machine-name)
<i>ctx</i> (Machine-name)
<i>inv</i> (Machine-name)
<i>asn</i> (Machine-name)
<i>pre</i> (Operation-name)

where

<i>cst</i>	stands for	<i>constraints</i>
<i>ctx</i>	stands for	<i>properties</i>
<i>inv</i>	stands for	<i>invariant</i>
<i>asn</i>	stands for	<i>assertion</i>
<i>pre</i>	stands for	<i>precondition</i>

Note: the anomalous *ctx* used to

stand for *Context*.

There are four types of proof obligations:

**Initialisation** This the proof that the initialisation substitution establishes the invariant, hence the hypotheses consist only of *cst* and *ctx*.

**Operation** This is the proof that the operation restores the invariant, and the hypotheses consist of *cst*, *ctx*, *inv*, *asn* and *pre*.

**Context** This is the proof that there exist sets and constants that satisfy the properties constraints. The hypotheses consist only of *cst*.

**Assertion** The proof that the assertions are a consequence of the invariant, properties and constraints. The hypotheses consist of *cst*, *ctx* and *inv*.

Instead of using the notation  $hypotheses \Rightarrow goal$  we will use  $hypotheses \vdash goal$ , which denotes a *lemma* not a predicate.

The strategy for discharging (*proving*) a proof obligation is to apply *inference rules* to the proof obligation until it is reduced to either of

$hypotheses \vdash true$

or

$false \vdash goal$

An inference theory has the same form as a proof obligation, but we will use *antecedent* in place of *hypotheses* and *consequent* in place of *goal*. Thus the form of an inference rule is:

$\begin{array}{c} antecedent \\ \Rightarrow \\ consequent \end{array}$
--

One significant difference between an inference rule and a proof obligation is that the antecedent and consequent generally contain *jokers*, where the hypotheses and goal contain expressions consisting of variable names and other symbols.

The reason is that a single rule will apply to an infinite number of lemmas, but only if the antecedent *matches* the hypotheses, or the consequent *matches* the goal. The matching will instantiate the jokers to actual expressions.

Jokers are identifiers consisting of a single letter.

Instantiation is a form of substitution. Consider an expression  $E$  containing jokers  $j_1, \dots, j_n$ . Then

$[j_1, \dots, j_n := e_1, \dots, e_n]E$

represents the instantiation of  $E$  in which the jokers  $j_1, \dots, j_n$  are replaced by the expressions  $e_1, \dots, e_n$ .

Consider the inference rule (expressed in ASCII)

$$\begin{array}{l} a <: c \ \& \\ b <: c \\ \Rightarrow \\ a \ \backslash / \ b <: c \end{array}$$

and suppose the consequent is matched against the goal of

$hypotheses \vdash accounts \ \backslash / \ \{acc\} <: ACCOUNT$

then the antecedent will be instantiated to

$accounts <: ACCOUNT \ \& \ \{acc\} <: ACCOUNT$

Forward inference matches the antecedents of an inference rule against the hypotheses of the lemma and *merges* the subsequently instantiated consequent with the hypotheses.

Forward inference attempts to expand the hypotheses in an attempt to generate a hypothesis that will match the goal or a conjunct of the goal.

Example:  $f : X \dashrightarrow Y \Rightarrow f : X \dashrightarrow Y$

Backward inference matches the goal of a lemma against the consequent of an inference rule and *replaces* the goal by the subsequently instantiated antecedents of the inference rule.

Backward inference attempts to transform the current goal into one or more, hopefully, easier to prove goals.

Note:  $H \vdash G_1 \wedge G_2$  can be replaced by  $H \vdash G_1$  and  $H \vdash G_2$ .

Inference rules can be used as either forward or backward, but most are intended to be used in a particular direction.

Example:  $f : X \dashrightarrow Y \ \& \ x : \text{dom}(f) \Rightarrow f(x) : Y$

A tactic is a sequence of inference rules, possibly with repetition, to be applied to the proof of a particular lemma.

We will not be dealing with tactics here, other than to observe and explain the tactics that appear in the proof theory files and menus.

In general tactics will involve the application of both forward and backward inference rules, although most use will be made of backward inference rules.

Forward inference rules are “blind”: they expand the hypotheses without reference to the goal, whereas backward inference rules are driven directly from the goal.

For a simple example we will take the *Bank* machine. All of the proof obligations, except Context, are discharged by the AutoProver, but we will look at doing the proofs interactively.

To follow this example you should follow using the BToolkit.

Apply the toolkit to the Bank development.

Move to the *Provers* environment.

If any of the proof obligations of the *Bank* machine have been discharged, select the *rpl* (Reset Proof Level) button and reset to level 0. There should be 7 proof obligations and 7 remaining to be proved.

Select the *prv* (Provers) button and choose the *BToolProver*.

On the *Theories* menu, select *Initialisation* line, which shows a total of 2 proof obligations and 2 proof obligations remaining to be proved.

On the *Theory Menu* that is displayed you might want to change *Display all* to *Display unproved only*.

Select *Initialisation.1* by left-mouse clicking anywhere within the proof obligation for *Initialisation.1*.

Now click again anywhere in the body of the proof obligation, or select *Prove Rule* in the *Rule Menu*.

The proof obligation is revealed inside a window as:

$\begin{array}{l} \text{cst}(\text{Bank}) \ \& \ \text{ctx}(\text{Bank}) \\ \Rightarrow \\ \{ \} : \text{ACCOUNT} \end{array}$
--

This window is interactive; try clicking in it.

The *DED* (deduction) button will be highlighted in the *Proof Menu*. Select it.

The deduction rule changes

$$H \vdash P \Rightarrow Q$$

to

$$H \wedge P \vdash Q$$

If DED is offering you should always select it.

You will observe that the current goal window now contains

$$H \vdash \{\} < : \text{ACCOUNT}$$

Try clicking on the  $H$ . You will get a window showing all the hypotheses.

Notice that when you list the hypotheses, all the hypotheses hidden behind the abbreviations  $cst$ ,  $ctx$ ,  $inv$ , etc are made visible.

Also, notice that any applicable forward rules are applied when DED is selected.

In the current case, notice that forward rule FwdInNat1X.19 has been selected. You can find that rule by selecting *Browse Library* looking for the 19th rule in FwdInNat1X. This rule reads

$$\begin{array}{l} n : \text{NAT1} \\ \Rightarrow \\ n : \text{NAT} \end{array}$$

In the *Proof Menu*, the *Show all applicable rules* will be highlighted.

Select it to see what is offering.

There is one rule

$$\begin{array}{l} \textit{Inclusion X.55} \\ \{\} < : a \end{array}$$

This says that *the empty set is a subset of anything*.

Notice that the validity of this depends on all expressions being well-typed.

Select the rule by clicking on it.

The proof completes.

Inspect the proof tree.

Select *Quit to Proof Menu*.

Select *Initialisation.2*

$$\begin{array}{l} cst(\text{Bank}) \ \& \ ctx(\text{Bank}) \\ \Rightarrow \\ \{\} : \{\} \dashrightarrow \text{ACCOUNT} \end{array}$$

and take the proof through its steps

Select *Quit to Proof Menu*.

No more proof obligations left on this menu.

Select *Quit to Main Menu*

Select *NewAccount* and then select *NewAccount.1*.

Take the proof forward. Note two instances of DED.

Keep going to be offered *InclusionX.38* and *InclusionX.2*. Take the former, and complete the proof.

Proceed to

$$H \vdash \text{balance} \triangleleft \{acc \mapsto 0\} \in \text{accounts} \cup \{acc\} \rightarrow \mathbb{N}$$

and be offered:

<p style="text-align: center;"><i>InTotalFunctionX.11</i></p> $\begin{aligned} & \text{binhyp}(f \in u \rightarrow t) \wedge \\ & u \cup \{a\} = s \wedge \\ & b \in t \\ \Rightarrow \\ & f \triangleleft \{a \mapsto b\} : s \rightarrow t \end{aligned}$
---

To prove that  $f \triangleleft g : s \rightarrow t$ , it is sufficient to prove that  $f$  and  $g$  are total functions, that  $\text{dom}(f) \cup \text{dom}(g) = s$ , and that  $\text{ran}(f) \subseteq t$  and  $\text{ran}(g) \subseteq t$ .

Thus, we can prove that  $f \triangleleft \{a \mapsto b\} \in s \rightarrow t$  if we can prove:

1.  $f$  is a total function, say  $f \in u \rightarrow t$ ;
2.  $u \cup \{a\} = s$ ;
3.  $b \in t$

$\begin{aligned} & f \in u \rightarrow t \wedge \\ & u \cup \{a\} = s \wedge \\ & b \in t \\ \Rightarrow \\ & f \triangleleft \{a \mapsto b\} \in s \rightarrow t \end{aligned}$
--

If the rule in the box in the previous frame is matched against the goal

$$\text{balance} \triangleleft \{\text{acc} \mapsto 0\} \in \text{accounts} \cup \{\text{acc}\} \rightarrow \mathbb{N}$$

then  $f$ ,  $b$ ,  $a$ ,  $s$  and  $t$  will be instantiated, but  $u$  will not be instantiated.

In *InTotalFunctionX.11*, the antecedent  $f \in u \rightarrow t$  is contained in the special guard *binhyp*.

Rather than simply becoming a sub-goal, a guard is evaluated *before* the rule is applied. *binhyp* has the first letter *b*, which identifies this as a guard, followed by *inhyp*.

*binhyp(expression)* will attempt to match *expression* in the current hypotheses and the rule will not be applied unless the match succeeds. As a side effect of the match any uninstantiated jokers will be instantiated.

Thus, in the current example,  $u$  will be instantiated to *accounts*.

Having selected *InTotalFunctionX.11* there are three goals. One is discharged because it is in the hypotheses (INHYP), and the other is an equality and is discharged by equality (EQL). The remaining goal is

$$0 \in \mathbb{N}$$

for which we are offered rule *InNat.24*:

$\begin{aligned} & \text{bnum}(n) \\ \Rightarrow \\ & n \in \mathbb{N} \end{aligned}$
---

*bnum* is a guard: *bnum(n)* is *true* if  $n$  is a natural number.

Proceed to

$H \vdash \text{balance} \triangleleft \{\text{account} \mapsto \text{balance}(\text{account}) + \text{amount}\} \in \text{accounts} \rightarrow \mathbb{N}$
--

and take *InTotalFunctionX.12* as before.

We get three goals two of which are discharged as they are in the hypotheses (INHYP), and we have one goal

$$\text{balance}(\text{account}) + \text{amount} \in \mathbb{N}$$

There is one rule on offer, *InNatX.23*,

$$n \in \mathbb{N} \wedge p \in \mathbb{N} \Rightarrow n + p \in \mathbb{N}$$

Take it.

We now have two goals of which the first is

$$balance(account) \in \mathbb{N}$$

For which we will take the rule *InSetX.9*

$\begin{aligned} & binhyp(f \in s \leftrightarrow t) \wedge \\ & a \in \text{dom}(f) \\ \Rightarrow & \\ & f(a) \in t \end{aligned}$
--

Notice the use of *binhyp* again.

When this rule is chosen the proof completes.

Follow the proof through to

$$balance(account) - amount \in \mathbb{N}$$

for which we are offered *InNat.19*

$\begin{aligned} & \textit{InNat.19} \\ & n \in \mathbb{N} \\ & p \in \mathbb{N} \\ & p \leq n \\ \Rightarrow & \\ & n - p \in \mathbb{N} \end{aligned}$
--

This gives three sub-goals:

1.  $balance(account) \in \mathbb{N}$
2.  $amount \in \mathbb{N}$
3.  $amount \leq balance(account)$

The first is proved as in frame 26, the second and third are in the hypotheses; the third from the precondition.

The only remaining undischarged proof obligation is a context proof that would not be discharged by the AutoProver.

Take this proof through to

$H \vdash \exists ACCOUNT.(card(ACCOUNT) = maxaccount \wedge card(ACCOUNT) \in \mathbb{N}_1)$
---

This obligation comes from our properties constraints and is requiring proof that such a set does exist.

At this stage the BToolProver is not offering any applicable proof rules.

Clicking on H to display the hypotheses shows that  $maxaccount \in \mathbb{N}_1$ . It is clear that such a set exists.

*Why no proof rules?*

If we browse the BTool theory library we encounter

$\begin{aligned} & \textit{Exist0X.17} \\ & n \in \mathbb{N}_1 \\ \Rightarrow & \\ & \exists A.(card(A) \in \mathbb{N}_1 \wedge card(A) = n) \end{aligned}$
---

The consequent is equivalent to our goal, but the conjuncts are swapped. If we can swap them around we should be offered this rule. We can do this with a *rewrite* rule.

Select *Edit BTool Library* and select *BToolUsersTheory*, which is for our own backward inference rules. Select *Add New Rule* and add:

$$(P \ \& \ Q) \ == \ (Q \ \& \ P)$$

Go back to the prover. The *Show all applicable rules* should be highlighted. If it isn't you have entered the rule incorrectly.

Select the rule, and then select *ExistOX.17* when it is offered.

The BTool prover accepts a number of guards, two of which we've met above. Guards are intended for use in the antecedents of backward inference rules, and in all cases the guard will be used to determine the applicability of a rule. A short summary of guards is given below.

**binhyp** *binhyp*(*P*) succeeds only if *P* matches a current hypotheses.

**bnum** *bnum*(*n*) succeeds only if *n* is a natural number.

**bident** *bident*(*a*) succeeds only if *a* is a valid variable identifier.

**btest** *btest*(*x rel y*) succeeds only if *x* and *y* are valid numbers and *x rel y* is *true*, where *rel* is one of *<*, *<=*, *=*, */=*, *>=*, *>*.

**bstring** *bstring*(*s*) succeeds only if *s* is a valid quoted string.

**bsearch** see BToolkit documentation.

**breade** see BToolkit documentation.

Rewrite rules have the form

$$\text{antecedent} \Rightarrow \text{expression-1} == \text{expression-2}$$

This has the understanding that if the antecedent holds then any sub-expression of the current goal matching *expression-1* may be rewritten as *expression-2*.

There is limited backtracking through rewrites so in general the antecedents should be guards.

There may be more than one applicable proof rule.

A proof rule *antecedent*  $\Rightarrow$  *consequent* means, "if you want to prove *consequent* then it is *sufficient* to prove *antecedent*".

Thus it may not be *necessary* to prove *antecedent*.

If you choose an inappropriate rule then this may lead to a goal that cannot be proved. In that case, it is necessary to back up to the most recently chosen rule and make another choice, and so on recursively.

The AutoProver tries all applicable rules until either the goal is discharged or all applicable rules are exhausted and the goal has not been discharged.

## 2 Prover Rules

A box of buttons on the top right of the screen represent rules for guiding the prover.

### 2.1 Deduction (DED)

The **DED** button will be highlighted when the current goal contains implication.

The DED button represents the proof strategy:

$$\frac{H \wedge P \vdash Q}{H \vdash P \Rightarrow Q}$$

which can be interpreted as

if the hypotheses are  $H$  and the current goal is  $P \Rightarrow Q$ , then add  $P$  to the hypotheses and make  $Q$  the goal

## 2.2 Hypotheses (HYP)

Some hypotheses are offered via the **HYP** button.

Selecting the HYP button, when it is highlighted, will offer hypotheses interpreted at proof rules, as follows.

### Equality

A hypothesis of the form

$$X = Y$$

can be interpreted as the rewrite proof rule

$$\boxed{X \cong Y}$$

*Note:* this means that equality, which in logic is an equivalence, when interpreted as a proof rule is directional and care must be taken in choosing between  $X = Y$  and  $Y = X$

### Universal Quantification

The universal quantification

$$\forall(z).(P \Rightarrow Q)$$

can be interpreted by the prover as the proof rule

$$\boxed{\begin{array}{c} P \\ \Rightarrow \\ Q \end{array}}$$

with  $z$  as jokers.

If  $Q$  is of the form  $X = Y$ , then the consequent,  $Q$ , is a rewrite rule with antecedents.

## 2.3 Proof of disjunction

Proof of a goal of the form  $P \vee Q$  may be achieved by one of the rules:

$$\boxed{\begin{array}{c} P \\ \Rightarrow \\ P \vee Q \end{array}}, \quad \boxed{\begin{array}{c} Q \\ \Rightarrow \\ P \vee Q \end{array}}$$

but these rules will only succeed when the disjunction is exclusive.

When it is possible for either  $P$  or  $Q$  to be true then the following rule will be found to be more appropriate

$$\boxed{\begin{array}{c} \neg(P) \Rightarrow Q \\ \Rightarrow \\ P \vee Q \end{array}}$$

and the following rewrite rule may be required

$$\neg\neg P \hat{=} P$$

## 2.4 Proof by case analysis

There are many goals, for example those involving overridden functions where case and analysis is required.

Consider the goal  $(f \triangleright \{a \mapsto b\})(x) \leq m$ , using jokers instead of actual variable names.

Clearly, there are two cases:  $x = a$  and  $x \neq a$ , so the following rule is useful

$$\begin{array}{l} (x = a \Rightarrow b \leq m) \wedge \\ (\neg(x = a) \Rightarrow f(x) \leq m) \\ \Rightarrow \\ (f \triangleright \{a \mapsto b\})(x) \leq m \end{array}$$

There are an infinity of variations on the above.

## 3 But the proof rules may be wrong!

Yes, that is correct. What is the consequence?

If the proof rules are incorrect then the discharge of any proof obligations using those rules may be invalid.

Does this invalidate the whole B Method? How do we validate the outcomes? Do we resort to testing?

As stated at the beginning of this course, proof is not to be regarded as a “stamp of approval”. The proofs, including all dependent proof rules, can be marked up into documents by the BToolkit. These can be, and should be, read critically. All dependent proof rules should be very carefully checked.

Whether you test or not is a question that is quite independent of proof.

*Extra testing cannot make up for bad proof rules.*