
MODULE *Increment*

This file contains the TLA+ version of the abstract program *Increment* in Figure 3.5 of the book “A Science of Concurrent Programs”. It also contains a proof of the invariance of *Inv* that was checked by the *TLAPS* theorem prover. It first gives the the part of the proof contained in Appendix B.1 of the book. It then gives the complete proof.

EXTENDS *Integers*, *FiniteSets*, *TLAPS*

The module *FiniteSets* defines the operator *Cardinality*, which is written # in the book.

CONSTANT *Procs*

$N \triangleq \text{Cardinality}(\text{Procs})$

VARIABLES x, pc, t

$\text{Init} \triangleq \wedge x = 0$
 $\wedge t = [p \in \text{Procs} \mapsto 0]$
 $\wedge pc = [p \in \text{Procs} \mapsto \text{“a”}]$

$\text{aStep}(p) \triangleq \wedge pc[p] = \text{“a”}$
 $\wedge x' = x$
 $\wedge t' = [t \text{ EXCEPT } ![p] = x]$
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“b”}]$

$\text{bStep}(p) \triangleq \wedge pc[p] = \text{“b”}$
 $\wedge x' = t[p] + 1$
 $\wedge t' = t$
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“done”}]$

$\text{PgmStep} \triangleq \exists p \in \text{Procs} : \text{aStep}(p) \vee \text{bStep}(p)$

$\text{Stutter} \triangleq \wedge \forall p \in \text{Procs} : pc[p] = \text{“done”}$
 $\wedge \text{UNCHANGED } \langle x, t, pc \rangle$

UNCHANGED exp is defined to equals exp' , so

UNCHANGED $\langle x, t, pc \rangle$

is equivalent to

$\langle x', t', pc' \rangle = \langle x, t, pc \rangle$.

However, when evaluating a next-state action, the first time *TLC* encounters a primed variable like x' must normally be in an expression of the form $x' = \dots$. So, we would have to write it as

$(x' = x) \wedge (t' = t) \wedge (pc' = pc)$

But since this is a common idiom, *TLC* will accept the version written with UNCHANGED .

$\text{Next} \triangleq \text{PgmStep} \vee \text{Stutter}$

The definitions of *TypeOK*, *NumberDone*, and *Inv* are the same as in the book.

$\text{TypeOK} \triangleq \wedge x \in 0 \dots N$

$$\begin{aligned} &\wedge t \in [Procs \rightarrow 0 \dots N] \\ &\wedge pc \in [Procs \rightarrow \{\text{"a"}, \text{"b"}, \text{"done"}\}] \end{aligned}$$

$$NumberDone \triangleq Cardinality(\{i \in Procs : pc[i] = \text{"done"}\})$$

$$\begin{aligned} Inv &\triangleq \wedge TypeOK \\ &\wedge \forall i \in Procs : (pc[i] = \text{"b"}) \Rightarrow (t[i] \leq NumberDone) \\ &\wedge x \leq NumberDone \end{aligned}$$

Below is the *TLAPS* proof of all the steps proved in Appendix B.1 of the book.

THEOREM $Next \wedge Inv \Rightarrow Inv'$

$\langle 1 \rangle 1.$ ASSUME NEW $p \in Procs$, $aStep(p)$, Inv
 PROVE Inv'

The following proof decomposition is unnecessary because, as shown below, *TLAPS* can prove the step with

BY $\langle 1 \rangle 1$ DEF $aStep$, Inv , $TypeOK$, $NumberDone$

$\langle 2 \rangle 1.$ $TypeOK'$

BY $\langle 1 \rangle 1$ DEF $aStep$, Inv , $TypeOK$

$\langle 2 \rangle 2.$ $\forall i \in Procs : (pc'[i] = \text{"b"}) \Rightarrow (t'[i] \leq NumberDone')$

$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Procs$, $pc'[i] = \text{"b"}$
 PROVE $t'[i] \leq NumberDone'$

OBVIOUS

$\langle 3 \rangle 2.$ $NumberDone' = NumberDone$

$\langle 3 \rangle 3.$ CASE $i = p$

$\langle 3 \rangle 4.$ CASE $i \neq p$

$\langle 4 \rangle 1.$ $(t'[i] = t[i] \wedge pc'[i] = pc[i])$

BY $\langle 1 \rangle 1$, $\langle 3 \rangle 4$ DEF $aStep$, Inv , $TypeOK$

$\langle 4 \rangle 2.$ $pc[i] = \text{"b"}$

BY $\langle 4 \rangle 1$, $\langle 3 \rangle 1$

$\langle 4 \rangle 3.$ $t[i] \leq NumberDone$

BY $\langle 4 \rangle 2$, $\langle 1 \rangle 1$ DEF Inv

$\langle 4 \rangle 4.$ QED

BY $\langle 4 \rangle 1$, $\langle 4 \rangle 3$, $\langle 3 \rangle 2$

$\langle 3 \rangle 5.$ QED

BY $\langle 3 \rangle 3$, $\langle 3 \rangle 4$

$\langle 2 \rangle 3.$ $x' \leq NumberDone'$

$\langle 2 \rangle 4.$ QED

BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$ DEF Inv

$\langle 1 \rangle 2.$ ASSUME NEW $p \in Procs$, $bStep(p)$, Inv

PROVE Inv'

$\langle 1 \rangle 3.$ ASSUME $Stutter$, Inv

PROVE Inv'

BY $\langle 1 \rangle 3$ DEF $Stutter$, Inv , $TypeOK$, $NumberDone$

$\langle 1 \rangle 4.$ QED

BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$ DEF $Next$, $PgmStep$

Here is a complete *TLAPS* checked proof of the invariance of *Inv*. The first thing we need to do is assert the assumption that *Procs* is a finite non-empty set. An **AXIOM** is a **THEOREM** that has no proof.

AXIOM *ProcsFinite* \triangleq *IsFiniteSet(Procs)* \wedge (*Procs* \neq {})

We will need some simple facts about finite sets that are in the library module *FiniteSetTheorems*, which we now import.

INSTANCE *FiniteSetTheorems*

We will need these theorems from the *FiniteSetTheorems* module. Note that theorems can be given a name so they can be applied in later proofs.

THEOREM *FS_Subset* \triangleq
 ASSUME NEW *S*, *IsFiniteSet(S)*, NEW *T* \in SUBSET *S*
 PROVE \wedge *IsFiniteSet(T)*
 \wedge *Cardinality(T)* \leq *Cardinality(S)*
 \wedge *Cardinality(S)* = *Cardinality(T)* \Rightarrow *S* = *T*

THEOREM *FS_CardinalityType* \triangleq
 ASSUME NEW *S*, *IsFiniteSet(S)*
 PROVE \wedge *Cardinality(S)* \in *Nat*
 \wedge *ExistsBijection*(1 .. *Cardinality(S)*, *S*)

THEOREM *FS_AddElement* \triangleq
 ASSUME NEW *S*, NEW *x*, *IsFiniteSet(S)*
 PROVE \wedge *IsFiniteSet(S* \cup {*x*}
 \wedge *Cardinality(S* \cup {*x*}) =
 IF *x* \in *S* THEN *Cardinality(S)* ELSE *Cardinality(S)* + 1

We use these theorems to prove the two simple lemmas about finite sets that we will need.

THEOREM *Lemma1* \triangleq ASSUME NEW *S*, *IsFiniteSet(S)*,
 NEW *T*, *T* \subseteq *S*
 PROVE \wedge *Cardinality(S)* \in *Nat*
 \wedge *IsFiniteSet(T)*
 \wedge *Cardinality(T)* \in 0 .. *Cardinality(S)*
 \wedge (*T* \neq *S*) \Rightarrow (*Cardinality(T)* < *Cardinality(S)*)
 BY *FS_Subset*, *FS_CardinalityType*

THEOREM *Lemma2* \triangleq ASSUME NEW *S*, *IsFiniteSet(S)*,
 NEW *e*, *e* \notin *S*
 PROVE *Cardinality(S* \cup {*e*}) = *Cardinality(S)* + 1
 BY *FS_AddElement*

Here is the complete proof of that *Inv* is an invariant of program *Increment*. The machine-checked proof is as simple as we might expect for proving an invariant of such a simple program. However, it was simple only because the *FiniteSetTheorems* was available to prove the two lemmas stating the facts about finite sets that were needed. Fortunately, reasoning about most programs only requires facts about a few simple data types such as sets, functions, and sequences for which we have library modules.

THEOREM *Next* \wedge *Inv* \Rightarrow *Inv'*

```

<1>1. ASSUME NEW  $p \in Procs$ ,  $aStep(p)$ ,  $Inv$ 
    PROVE  $Inv'$ 
    BY <1>1 DEF  $aStep$ ,  $Inv$ ,  $TypeOK$ ,  $NumberDone$ 
<1>2. ASSUME NEW  $p \in Procs$ ,  $bStep(p)$ ,  $Inv$ 
    PROVE  $Inv'$ 
    <2>1.  $(N \in Nat) \wedge (NumberDone \in 0 \dots N) \wedge (NumberDone < N)$ 
    BY  $ProcsFinite$ ,  $Lemma1$ , <1>2 DEF  $NumberDone$ ,  $N$ ,  $bStep$ 
    <2>2.  $NumberDone' = NumberDone + 1$ 
    BY <1>2,  $Inv$ ,  $ProcsFinite$ ,  $Lemma1$ ,  $Lemma2$ 
    DEF  $bStep$ ,  $Inv$ ,  $TypeOK$ ,  $NumberDone$ 
    <2>3. QED
    BY <1>2, <2>1, <2>2 DEF  $Inv$ ,  $bStep$ ,  $TypeOK$ 
<1>3. ASSUME  $Stutter$ ,  $Inv$ 
    PROVE  $Inv'$ 
    BY <1>3 DEF  $Stutter$ ,  $Inv$ ,  $TypeOK$ ,  $NumberDone$  ,  $vars$ 
<1>4. QED
    BY <1>1, <1>2, <1>3 DEF  $Next$ ,  $PgmStep$ 

```

```

\ * Modification History
\ * Last modified Wed Oct 23 11:58:59 CEST 2024 by lamport
\ * Created Thu May 12 09:39:37 PDT 2022 by lamport

```