Counterpoint

A UNITY proof with pure nondeterminism

Rajeev Joshi [1]

23 March, 1998

0 Introduction

This note attempts to distill out the essence of a UNITY proof for a problem posed by J Moore (who in turn got it from Bobby Blumofe). The proof is of interest because it uses the UNITY proof rules for pure nondeterminism, which are not that widely known.

Consider a system G with N processes ($N > 0$) denoted F_0 through F_{N-1} , with process F_j ($0 \leq j < N$) executing the single statement α_j as shown below.

program F_j
shared ctr : **integer**
local old_j, new_j : **integer**
local b_j : **boolean**
assign

$$\begin{array}{lrll} \{\ \alpha_j\ \} & ctr\ ,\ b_j := new_j\ ,\ false & \textbf{if} & b_j \ \wedge\ ctr = old_j \\ \sim & old_j\ ,\ b_j := ctr\ ,\ false & \textbf{if} & b_j \ \wedge\ ctr \neq old_j \\ \sim & new_j\ ,\ b_j := 1 + old_j\ ,\ true & \textbf{if} & \neg b_j \end{array}$$

end

Assuming pure nondeterminism [2] (i.e., some statement is eventually executed) we are asked to show the following property:

the value of ctr eventually increases ,

which is expressed in UNITY logic as

$$(\forall\ C\ ::\ ctr = C \quad \mapsto \quad ctr > C \quad \textbf{in}\ G) \qquad (\boldsymbol{ctr\ inc})$$

The rest of this note deals with the proof of this property. Section 1 describes the proof rules for UNITY logic under pure nondeterminism. Section 2 proves a weaker result than (***ctr inc***), requiring an assumption on initial states of the program. Section 3 shows how to strengthen the result by removing the assumption of the previous section; this is achieved by adding auxiliary variables to the program.

Notation Throughout this note, identifiers P, Q, R range over predicates, j ranges over $0 \ldots (N-1)$, and C ranges over the integers.

[1] Dept. of Computer Sciences, The University of Texas at Austin, Email: joshi@cs.utexas.edu

[2] Pure nondeterminism is a special case of *minimal progress* [JM 93-4] obtained when all guards are *true* .

1 UNITY Logic

The main operators of UNITY logic [JM 93-4] are **initially** , **co** , **stable** , **invariant** , **transient** , **ensures** and $\mapsto$. The definitions of the first four are the same as for the standard UNITY model (with unconditional fairness) and are given in the appendix. The definitions of **transient** and **ensures** are different for pure nondeterminism and are given by the equations below, where γ ranges over the statements of a program G , and the restricted everywhere brackets $[\![\cdot]\!]$ denote quantification over the reachable state space of G . [3]

$$\begin{array}{rcl} \textbf{transient} \quad Q \quad \textbf{in } G & = & (\forall\, \gamma \,::\, [\![\, Q \Rightarrow wp.\gamma.(\neg Q) \,]\!]) \\ Q \quad \textbf{ensures} \quad R \quad \textbf{in } G & = & (\forall\, \gamma \,::\, [\![\, Q \wedge \neg R \Rightarrow wp.\gamma.R \,]\!]) \end{array}$$

The operator $\mapsto$ is defined in terms of **ensures** in the usual way [CM 88] .

Most of the theorems about $\mapsto$ in the standard logic carry over to the logic for pure nondeterminism (e.g., PSP, impossibility, lhs-strengthening, rhs-weakening, cancellation and induction rule). Since the proofs presented in this note use the induction rule, it is reproduced below.

Induction Rule for $\mapsto$ Let M be a total function from states of program G to a set W. Let $(W, \prec)$ be well-founded, let w range over W and let P, Q be predicates on the states of G . Then,

$$(\forall\, w \,::\, \quad P \;\wedge\; M = w \quad \mapsto \quad Q \;\vee\; (P \;\wedge\; M \prec w) \;\; \textbf{in } G) \quad \Rightarrow \quad P \quad \mapsto \quad Q \;\; \textbf{in } G$$

Interestingly, the operators **transient** and **ensures** , as defined for pure nondeterminism, satisfy some properties that do not hold in the standard model; some of these are listed in appendix A .

2 First Solution

To begin with, we simplify the problem by adding the following initial condition to each process F_j

$$\textbf{initially} \quad b_j \;\; \Rightarrow \;\; new_j = 1 + old_j \;\; \textbf{in } F_j$$

As a result, we can prove the following invariant properties: for each j ,

$$\textbf{invariant} \quad b_j \;\; \Rightarrow \;\; new_j = 1 + old_j \;\; \textbf{in } G$$

Next, we define the following functions on the state space of program G :

$$\begin{array}{l} nq = (\#\, j \,::\, old_j \neq ctr) \\ nb = (\#\, j \,::\, \neg b_j) \end{array}$$

where $(\#\, j \,::\, P_j)$ denotes the number of j for which P_j holds. Note that nq and nb both lie in the range $0 \ldots N$.

The proof of the desired property is now as follows. Let $\prec$ denote the following lexicographic ordering on pairs of naturals –

$$(m0, n0) \prec (m1, n1) \;\; \equiv \;\; (m0 < m1) \;\; \vee \;\; (m0 = m1 \wedge n0 < n1) \;\; ;$$

[3] See the appendix for a different way of defining these operators.

let C range over the integers, let NQ, NB range over $0 \ldots N$, and define P, Q as follows.

$$\begin{array}{lcl} [P & \equiv & ctr = C \ \wedge \ (nq, nb) = (NQ, NB)] \\ [Q & \equiv & ctr > C \ \vee \ (ctr = C \ \wedge \ (nq, nb) \prec (NQ, NB))] \end{array}$$

We observe that the following Hoare triples are true:

$$\begin{array}{r} \{P \wedge b_j \wedge ctr = old_j\} \ \ ctr \ , \ b_j := new_j \ , \ false \ \ \{Q\} \\ \{P \wedge b_j \wedge ctr \neq old_j\} \ \ old_j \ , \ b_j := ctr \ , \ false \ \ \{Q\} \\ \{P \wedge \neg b_j\} \ \ new_j \ , \ b_j := 1 + old_j \ , \ true \ \ \{Q\} \end{array}$$

Thus we have

$$\begin{array}{ll} & true \\ \Rightarrow & \quad \{ \text{ Hoare triples above } \} \\ & (\forall \, j \ :: \ [P \Rightarrow wp.\alpha_j.Q]) \\ \Rightarrow & \quad \{ \ [R] \Rightarrow [\![\, R \,]\!] \text{ , definition of } \textbf{ensures} \ \} \\ & P \ \textbf{ensures} \ \ Q \ \ \textbf{in} \ G \\ \Rightarrow & \quad \{ \text{ Basis for } \mapsto \} \\ & P \ \ \mapsto \ \ Q \ \ \textbf{in} \ G \\ \equiv & \quad \{ \text{ Definition of } P, Q \ \} \\ & ctr = C \wedge (nq, nb) = (NQ, NB) \ \ \mapsto \ \ ctr > C \ \ \vee \ \ (ctr = C \wedge (nq, nb) \prec (NQ, NB)) \ \ \textbf{in} \ G \\ \Rightarrow & \quad \{ \mapsto \text{ induction, using the fact that } (nq, nb) \text{ is bounded from below by } (0, 0) \ \} \\ & ctr = C \ \ \mapsto \ \ ctr > C \ \ \textbf{in} \ G \end{array}$$

3 Second Solution

Informally speaking, we needed the initial conditions in the previous proof in order to ensure that the counter value is nondecreasing. Without these conditions, we can construct executions in which this property is not true – for instance, consider an execution starting with $ctr, old_0, new_0, b_0 = 1, 1, 0, true$ which starts by executing statement α_0 . An examination of the program text shows that in fact the only step of process F_j that might reduce the value of ctr during an execution is its first execution of the first alternative of α_j ; subsequent steps of F_j do not reduce ctr . It turns out, then, that property (***ctr inc***) is still true, and the initial conditions we added are not really needed. In this section, we show how to prove the result without these conditions, by introducing additional auxiliary variables t_j , as shown below.

program F_j
shared ctr : **integer**
local old_j, new_j : **integer**
local b_j : **boolean**
local t_j : **boolean**
initially $t_j = false$
assign
$ctr\ ,\ b_j\ ,\ t_j := new_j\ ,\ false\ ,\ true$ **if** $b_j \wedge ctr = old_j$
$\sim\quad old_j\ ,\ b_j := ctr\ ,\ false$ **if** $b_j \wedge ctr \neq old_j$
$\sim\quad new_j\ ,\ b_j := 1 + old_j\ ,\ true$ **if** $\neg b_j$
end

Informally speaking, the auxiliary variable t_j acts as a "trigger", which is set upon the first execution of the first alternative of α_j . Once t_j is set, process F_j does not decrease the value of ctr . Formally, instead of the invariants used in the previous solution, we now have, for each j

$$\textbf{invariant}\quad t_j \wedge b_j \quad \Rightarrow \quad new_j = 1 + old_j \quad \textbf{in}\ G$$

We define nq and nb as before; in addition, we define

$$nt = (\#\, j\ ::\ \neg t_j)$$

Note that, like nq and nb , the quantity nt lies in the range $0 \ldots N$.

The rest of the proof is similar, except that the lexicographic ordering $\prec$ is now applied to triples. For any integer C and any NQ, NB, NT in the range $0 \ldots N$, define P, Q as

$$[P \quad \equiv \quad ctr = C \quad \wedge \quad (nt, nq, nb) = (NT, NQ, NB)]$$
$$[Q \quad \equiv \quad ctr > C \quad \vee \quad (ctr = C \quad \wedge \quad (nt, nq, nb) \prec (NT, NQ, NB))]$$

Then, we have

$true$
$\Rightarrow$ { Calculating Hoare triples as before }
$(\forall\, j\ ::\ [P \Rightarrow wp.\alpha_j.Q])$
$\Rightarrow$ { $[R] \Rightarrow [\![\ R\ ]\!]$, definition of **ensures** }
P **ensures** Q **in** G
$\Rightarrow$ { Basis for $\mapsto$ }
$P \quad \mapsto \quad Q$ **in** G
$\Rightarrow$ { $\mapsto$ induction, using the fact that (nt, nq, nb) is bounded from below by $(0, 0, 0)$ }
$ctr = C \quad \mapsto \quad ctr > C$ **in** G

A UNITY for Pure Nondeterminism

I shall define primed and unprimed versions of all UNITY operators except **initially** ; the primed versions are introduced in order to keep the treatment of the Substitution Axiom simple. The following notation is used: the initial condition of program G is denoted $G.IC$ and the set of statements of G is denoted $G.Act$. As before, γ ranges over $G.Act$.

$$
\begin{array}{rrcl}
\textbf{initially} & Q & \textbf{in } G & = [G.IC \Rightarrow Q] \\
Q & \textbf{co}' \ R & \textbf{in } G & = [Q \Rightarrow R] \ \wedge \ (\forall\, \gamma :: [Q \Rightarrow wp.\gamma.R]) \\
\textbf{stable}' & Q & \textbf{in } G & = Q \ \textbf{co}' \ Q \ \textbf{in } G \\
\textbf{invariant}' & Q & \textbf{in } G & = \textbf{initially } Q \ \textbf{in } G \ \wedge \ \textbf{stable}' \ Q \ \textbf{in } G \\
\textbf{transient}' & Q & \textbf{in } G & = (\forall\, \gamma :: [Q \Rightarrow wp.\gamma.(\neg Q)]) \\
Q & \textbf{ensures}' \ R & \textbf{in } G & = (Q \wedge \neg R) \ \textbf{co}' \ (Q \vee R) \ \textbf{in } G \ \wedge \ \textbf{transient}' \ (Q \wedge \neg R) \ \textbf{in } G
\end{array}
$$

The operator $\mapsto'$ is defined as the transitive, disjunctive closure of **ensures** $'$ in the usual way. Finally, we add the following postulate, which gives the relationship between unprimed operators and their primed counterparts.

A.0 Substitution Axiom

For **op** ranging over **stable** , **invariant** and **transient**

$$\textbf{op} \ \ Q \ \ \textbf{in } G \ \ = \ \ (\exists\, J : \textbf{invariant}' \ \ J \ \ \textbf{in } G : \textbf{op}' \ \ (Q \wedge J) \ \ \textbf{in } G)$$

For **op** ranging over **co** , **ensures** and $\mapsto$

$$Q \ \textbf{op} \ \ R \ \ \textbf{in } G \ \ = \ \ (\exists\, J : \textbf{invariant}' \ \ J \ \ \textbf{in } G : (Q \wedge J) \ \textbf{op}' \ \ (R \wedge J) \ \ \textbf{in } G)$$

A.1 Program Union

Let $\mathcal{F}$ denote a nonempty set of programs which have the same set of variable declarations. Define the union of the set as a program H , having the same variables as the programs in $\mathcal{F}$, and having the initial condition and set of statements defined below.

$$
\begin{aligned}
&[H.IC = (\forall\, G :\ G \in \mathcal{F} :\ G.IC)] \\
&H.Act = (\cup\, G :\ G \in \mathcal{F} :\ G.Act)
\end{aligned}
$$

Union Theorem
Let H denote the union of a set $\mathcal{F}$ of programs, as defined above. Let P, Q range over predicates on the state space of the programs. Then we have, with G ranging over $\mathcal{F}$,

$$
\begin{aligned}
P \ \textbf{co}' \ \ Q \ \ \textbf{in } H \ &\equiv \ (\forall\, G :: P \ \textbf{co}' \ \ Q \ \ \textbf{in } G) \\
\textbf{transient}' \ \ P \ \ \textbf{in } H \ &\equiv \ (\forall\, G :: \textbf{transient}' \ \ P \ \ \textbf{in } G)
\end{aligned}
$$

A.2 Properties of ensures

Here are some properties of **ensures** for pure nondeterminism that do not hold in the standard model. For any bag X of predicates, any nonempty bag Z of predicates, and any predicates P, Q, R, S ,

$$(\forall\, P :\ P \in X :\ P \ \textbf{ensures}\ Q \ \textbf{in}\ G) \ \equiv\ (\exists\, P :\ P \in X :\ P)\ \textbf{ensures}\ Q \ \textbf{in}\ G$$

$$[P \Rightarrow Q] \ \wedge\ Q \ \textbf{ensures}\ R \ \textbf{in}\ G \ \Rightarrow\ P \ \textbf{ensures}\ R \ \textbf{in}\ G \qquad (\textbf{\textit{lhs-str}})$$

$$R \ \textbf{co}\ S \ \textbf{in}\ G \ \wedge\ P \ \textbf{ensures}\ Q \ \textbf{in}\ G \ \Rightarrow\ P \wedge R \ \textbf{ensures}\ Q \wedge S \ \textbf{in}\ G \qquad (\textbf{\textit{Strong PSP}})$$

Acknowledgments. I am grateful to J Moore, who posed the problem and commented on the original solution, to Jay Misra, who invented the simpler program shown here and encouraged me to write this note, and to Pete Manolios, who pointed out several typographical errors in a previous version. (*End of Acknowledgments.*)

References

[CM 88] K.M. Chandy and J.Misra, *Parallel Program Design: A Foundation*, Addison-Wesley, 1988

[JM 93-4] Misra, J., Notes on the new UNITY logic, unpublished manuscripts, 1993-4
Available from `ftp://ftp.cs.utexas.edu/pub/psp/unity/new_unity`