Methods and Tools for the Verification of Critical Properties

Roger Bishop Jones
International Computers Limited, Eskdale Road,
Winnersh, Wokingham, Berks RG11 5TT

tel: 0734 693131 x6536, fax: 0734 697636
email: uucp: rbj@win0109.uucp

TOPICS

methods for verification of critical systems

fully worked micro-examples

illustration of use of Z proof tool

comparisons between proof in Z and in HOL

SPECIFICATION

=?

MODEL

REFINEMENT

SPECIFICATION

=

statement of requirement

=

property of models

MODEL

=

detailed model of behaviour of implementation

may be used for simulation

may be used for reasoning about whether implementation meets requirements

CONFORMANCE CLAIM

MODEL

$\in$

SPECIFICATION

THINGS THAT CAN'T BE SAID USING A MODEL AS A SPECIFICATION

can't specify the domain

can't constrain behaviour off the domain

can't specify that the system is secure

uncountably many other things

HIGH LEVEL DESIGN STEP

INPUT:

system specification

OUTPUTS:

set of subsystems

specification of each subsystem

construction of system from subsystems

correctness proof

LOW LEVEL DESIGN STEP

INPUT:

subsystem specification

OUTPUTS:

model of subsystem

correctness proof

SML

new_theory"***example1***";

Z

$[\boldsymbol{DATA}]$

Z

STATE

$classified\_data : \mathbb{N} \pfun DATA$

Z

ΔSTATE

$STATE;\ STATE'$

Z

OPERATION

$\Delta STATE;$

$clear? : \mathbb{N}$

Z

READ

$$
\begin{array}{|l}
\hline
OPERATION; \\
class?:\mathbb{N}; \\
data! : DATA \\
\hline
class? \in dom\ classified\_data \\
\land\ class? \leq clear? \\
\land\ data! = classified\_data\ class? \\
\land\ classified\_data' = classified\_data \\
\hline
\end{array}
$$

Z

COMPUTE

$$OPERATION;$$

$$class?:\mathbb{N};$$

$$computation? : (\mathbb{N} \pfun DATA) \to DATA$$

$$class? \in dom\ classified\_data$$

$$\land\ class? \geq clear?$$

$$\land\ classified\_data' = classified\_data \oplus \{class? \mapsto computation?\ (0\ ..\ clear? \lhd classified\_data)\}$$

PROBLEMS WITH "NAIVE" SPECIFICATION

- specifies *a* secure system (at best) not secure systems in general

- a real system specification will be very large even if critical property remains simple

- there is no formal basis for judging whether a large specification of this kind is a specification of a *secure* system

- rules of refinement make the *meaning* of such specifications obscure

FORMALISATION OF CRITICAL REQUIREMENT

- establish "type" of system
- formally define which systems meet the requirement
- no need to specify in detail what the systems do

SML

new_theory"**example2**";

Z

[***IN***,***OUT***,***DATA***]

$\boldsymbol{STATE2} == \quad \mathbb{N} \pfun DATA$

$\boldsymbol{SYSTEM} \quad == \quad (\mathbb{N} \times IN \times STATE2)$
$\qquad \fun (STATE2 \times OUT)$

Z

$\boldsymbol{out\_secure} : \mathbb{P}\ SYSTEM$

$\forall sys{:}SYSTEM \bullet\ sys \in out\_secure \Leftrightarrow$

$(\forall\ clear{:}\mathbb{N};\ inp{:}IN;\ s,s'{:}STATE2 \mid$
$((0 .. clear \ndres s) = (0 .. clear \ndres s') \bullet$
$\quad second\ (sys\ (clear,\ inp,\ s))$
$\quad = second\ (sys\ (clear,\ inp,\ s'))))$

Z

$$\boldsymbol{state\_secure} : \mathbb{P}\ SYSTEM$$

$$\forall sys{:}SYSTEM \bullet\ sys \in state\_secure \Leftrightarrow$$

$$(\forall class,\ clear{:}\mathbb{N};\ inp{:}IN;\ s,s'{:}STATE2 \mid$$
$$((0\mathinner{..}\ class \lhd s) = (0\mathinner{..}\ class \lhd s')\bullet$$

$$(0\mathinner{..}\ class \lhd (first\ (sys\ (clear,\ inp,\ s))))$$
$$= (0\mathinner{..}\ class \lhd (first\ (sys\ (clear,\ inp,\ s'))))))$$

Z

$$\boldsymbol{secure} : \mathbb{P}\ SYSTEM$$

$$\forall sys{:}SYSTEM \bullet\ sys \in secure \Leftrightarrow$$
$$sys \in state\_secure \land sys \in out\_secure$$

"ARCHITECTURAL DESIGN"

- formal model of the top level structure of the system
- separate out critical from non-critical functions
- identify top level subsystems
- specify how the system is constructed from the subsystems
- specify the critical requirements on the subsystems

z

$$\boldsymbol{APPLICATION} == (IN \times STATE2) \rightarrow (STATE2 \times OUT)$$

$$\boldsymbol{KERNEL} == APPLICATION \rightarrow SYSTEM$$

z

$$\boldsymbol{construction} : APPLICATION \times KERNEL \rightarrow SYSTEM$$

$$\forall appl{:}APPLICATION;\ kernel{:}KERNEL \bullet construction\ (appl,\ kernel) = kernel\ appl$$

z

$$\boldsymbol{secure\_kernel} : \mathbb{P}\ KERNEL$$

$$\forall\ kernel{:}KERNEL \bullet\ kernel \in secure\_kernel \Leftrightarrow$$

$$(\forall appl{:}APPLICATION \bullet$$
$$(construction\ (appl,\ kernel)) \in secure)$$

PROOF OF CORRECTNESS of ARCHITECTURAL DESIGN

Prove that IF the subsystems satisfy their critical requirements, AND they are fitted together in the prescribed way, THEN the resulting system will satisfy the system critical requirements.

Z PROOF STYLE

- special Z goal package implemented in ICL HOL prototype

- proofs like HOL proofs except that assumptions and conclusions are Z predicates

- parser and pretty printer expect and deliver Z not HOL (with escape into HOL if necessary)

- implementation on "product" HOL will be much better integrated

SML

```
zset_goal([],⌜
 ∀kernel:KERNEL;appl:APPLICATION•
  kernel ∈ secure_kernel
  ⇒ secure (construction (appl,kernel))⌝);
```

HOL output

```
(∀kernel: KERNEL; appl: APPLICATION | true •
  ((kernel ∈ secure_kernel)
  ⇒ ((construction (appl, kernel)) ∈ secure)))
```

REWRITING IN Z

- most axiomatic descriptions give rise (at best) to *conditional* rewriting rules

- definitions of predicates can be transformed to unconditional rewriting rules (often)

- definitions of functions cannot (in general) be made unconditional

- present proof system requires either explicit instantiation of conditional rewriting rules or instantiation by resolution

- ideal (target for product) is powerful conditional context sensitive rewriting

SML

```
val secure_kernel_sim = iff_simp
  (z_specification "—" "secure_kernel");
```

HOL output

```
val secure_kernel_sim = ⊢
  ((kernel ∈ secure_kernel) ⇔
   ((kernel ∈ KERNEL) ∧
    (∀appl: APPLICATION | true •
    ((construction (appl, kernel)) ∈ secure)))) : thm
```

SML

```
ze (Zrewrite_tac[secure_kernel_sim]);
```

HOL output

```
(∀kernel: KERNEL; appl: APPLICATION | true •
  (((kernel ∈ KERNEL) ∧
  (∀appl: APPLICATION | true •
    ((construction (appl, kernel)) ∈ secure)))
  ⇒ ((construction (appl, kernel)) ∈ secure)))
```

SML

$ze\ (REPEAT\ \ Zstrip\_tac);$

HOL output

$1\ \ subgoal$

$\quad [\ (kernel \in KERNEL)\ ]$

$\quad [\ (appl \in APPLICATION)\ ]$

$\quad [\ (\forall appl\colon APPLICATION \mid true \bullet$

$\qquad ((construction\ (appl,\ kernel)) \in secure))\ ]$

$((construction\ (appl,\ kernel)) \in secure)$

SML

$ze\ \ Zres\_tac;$

HOL output

$subgoal\ \ proved$

Z

$$\boldsymbol{kernel\_implementation} : KERNEL$$

$$\forall\ clear{:}\mathbb{N};\ inp{:}IN;\ state{:}STATE2;$$
$$appl{:}APPLICATION \bullet$$

$$kernel\_implementation\ appl\ (clear,\ inp,\ state) =$$

$$((state \oplus ((0\ ..\ (clear-1)) \lhd$$
$$(first\ (appl\ (inp,\ (0\ ..\ clear) \lhd state)))),$$
$$second\ (appl\ (inp,\ (0\ ..\ clear) \lhd state))))$$

Set Theoretic Lemmas

SML

$$[Zset\_eq\_thm,\ \bigcap\_thm,\ \lhd\_thm1,$$
$$\lhd\!\!\!\!-\_thm4,\ \oplus\_thm,\ first\_thm];$$

HOL output

$$val\ it =$$
$$[\vdash ((x = y) =$$
$$(\forall z\colon U \mid true\bullet ((z \in x) \Leftrightarrow (z \in y)))),$$
$$\vdash ((P \cap Q) \Leftrightarrow$$
$$(\forall z\colon U \mid true\bullet ((z \in P) \Rightarrow (z \in Q)))),$$
$$\vdash ((x \in (S \lhd R)) \Leftrightarrow$$
$$((x \in R) \wedge ((first\ x) \in S))),$$
$$\vdash ((x \in (S \lhd\!\!\!\!- R)) \Leftrightarrow$$
$$((x \in R) \wedge \neg((first\ x) \in S))),$$
$$\vdash ((x \in (A \oplus B)) \Leftrightarrow ((x \in B) \vee$$
$$((x \in A) \wedge \neg(\exists y\colon U \mid true\ \bullet$$
$$(((first\ x),\ y) \in B))))),$$
$$\vdash ((first\ (x,\ y)) = x)] : thm\ list$$

SML

```
val SET_TAC = EVERY
  [Zrewrite_tac[Zset_eq_thm, ⋂_thm,
    ◁_thm1, ⩤_thm4, ⊕_thm, first_thm],
  REPEAT Zstrip_tac,
  Zres_tac, Zres_tac,
  Zasm_rewrite_tac[]];
```

HOL output

```
val SET_TAC = fn : tactic
```

SML

```
fun SET_RULE t =
    TAC_PROOF(([],t),SET_TAC);
```

HOL output

```
val SET_RULE = fn : TERM -> thm
```

Arithmetic Lemmas Required for Proof

SML

$$val\ le..lemma1 = new\_axiom\ \text{"}le..lemma1\text{"}$$
$$\quad \llbracket x \leq y \Rightarrow (0\ ..\ x) \cap (0\ ..\ y) \rrbracket;$$

$$val\ le..lemma2 = new\_axiom\ \text{"}le..lemma2\text{"}$$
$$\quad \llbracket \neg\ x \leq y \Rightarrow (0\ ..\ y) \cap (0\ ..\ (x - 1)) \rrbracket;$$

SML

$$zset\_goal([],\llbracket$$
$$\quad kernel\_implementation \in secure\_kernel$$
$$\rrbracket);$$

HOL output

$$(kernel\_implementation \in secure\_kernel)$$

SML

$$ze\ (Zrewrite\_tac[kidec,\ secure\_kernel\_sim,$$
$$\quad secure\_sim,\ state\_secure\_sim,\ out\_secure\_sim]$$
$$\quad THEN\ REPEAT\ Zstrip\_tac);$$

HOL output

$$Note\text{: } tactic\ produced\ 2\ duplicated\ subgoals$$
$$3\ subgoals$$
$$\ldots$$
$$\quad [\ (appl \in APPLICATION)\ ]$$
$$((construction\ (appl,\ kernel\_implementation))$$
$$\quad \in SYSTEM)$$

SML

```
[condec, kidec, Z_pair_×_sym, fun_app_thm];
```

HOL output

```
val it =
  [⊢ (construction
      ∈ ((APPLICATION × KERNEL) → SYSTEM)),
   ⊢ (kernel_implementation ∈ KERNEL),
   ⊢ (((x ∈ X) ∧ (y ∈ Y)) ⇔ ((x, y) ∈ (X × Y))),
   ⊢ (∀f: (X → Y); x: X | true • ((f x) ∈ Y))]
 : thm list
```

SML

```
ze (EVERY[
    ASSUME_TAC condec,
    ASSUME_TAC kidec,
    Zimp_res_tac    Z_pair_×_sym,
    Zimp_res_tac    fun_app_thm]);
```

HOL output

$$subgoal\ proved$$

$$2\ subgoals$$

$$\ldots$$

$$[\ (appl \in APPLICATION)\ ]$$

$$[\ (class \in \mathbb{N})\ ]$$

$$[\ (clear \in \mathbb{N})\ ]$$

$$[\ (inp \in IN)\ ]$$

$$[\ (s \in STATE2)\ ]$$

$$[\ (s' \in STATE2)\ ]$$

$$[\ (((0\ ..\ class) \lhd s) = ((0\ ..\ class) \lhd s'))\ ]$$

$$(((0\ ..\ class) \lhd (first\ ((construction\ (appl,\ kernel\_implementation))\ (clear,\ inp,\ s))))$$

$$=$$

$$((0\ ..\ class) \lhd (first\ ((construction\ (appl,\ kernel\_implementation))\ (clear,\ inp,\ s'))))$$

SML

```
ze (EVERY[
    ASSUME_TAC kidec,
    Zimp_res_rewrite_tac conpred,
    Zimp_res_rewrite_tac kipred,
    Zrewrite_tac [first_thm]]);
```

HOL output

```
1 subgoal

...
   [ (((0 .. class) ◁ s) = ((0 .. class) ◁ s')) ]
...

(((0 .. class) ◁ (s ⊕ ((0 .. (clear − 1))
      ⩤ (first (appl (inp, ((0 .. clear) ◁ s))))))))
 =
((0 .. class) ◁ (s' ⊕ ((0 .. (clear − 1))
      ⩤ (first (appl (inp, ((0 .. clear) ◁ s'))))))))
```

SML

$$ze\ (ASM\_CASES\_TAC \lceil\!\lceil clear \leq class \rceil\!\rceil);$$

HOL output

$$2\ subgoals$$

$$\ldots$$

$$[\ (((0\ ..\ class) \lhd s) = ((0\ ..\ class) \lhd s'))\ ]$$

$$\ldots$$

$$[\ (clear \leq class)\ ]$$

$$(((0\ ..\ class) \lhd (s \oplus ((0\ ..\ (clear - 1))$$
$$\lhd (first\ (appl\ (inp, ((0\ ..\ clear) \lhd s)))))))$$
$$=$$
$$((0\ ..\ class) \lhd (s' \oplus ((0\ ..\ (clear - 1))$$
$$\lhd (first\ (appl\ (inp, ((0\ ..\ clear) \lhd s'))))))))$$

SML

$$ze\ (Zimp\_res\_tac\ le..lemma1);$$
$$ze\ (Zimp\_res\_rewrite\_tac\ (SET\_RULE \lceil\!\lceil (A \cap B) \Rightarrow$$
$$(B \lhd z) = (B \lhd z') \Rightarrow (A \lhd z) = (A \lhd z') \rceil\!\rceil));$$

HOL output

$$
\begin{array}{l}
1\ \ subgoal \\
\ \ \ldots \\
\quad [\ (((0\ ..\ class) \lhd s) = ((0\ ..\ class) \lhd s'))\ ] \\
\ldots \\
\quad [\ (((0\ ..\ clear) \lhd s) = ((0\ ..\ clear) \lhd s'))\ ] \\
\\
\quad (((0\ ..\ class) \lhd (s \oplus ((0\ ..\ (clear - 1)) \\
\qquad \lhdblk (first\ (appl\ (inp,\ ((0\ ..\ clear) \lhd s')))))))) \\
\ = ((0\ ..\ class) \lhd (s' \oplus ((0\ ..\ (clear - 1)) \\
\qquad \lhdblk (first\ (appl\ (inp,\ ((0\ ..\ clear) \lhd s'))))))))
\end{array}
$$

SML

$$
\begin{array}{l}
ze\ (Zimp\_res\_rewrite\_tac(SET\_RULE \\
\lceil \forall x \lhd z = x \lhd z' \Rightarrow x \lhd (z \oplus y) = x \lhd (z' \oplus y) \rceil));
\end{array}
$$

HOL output

$$
\begin{array}{l}
subgoal\ proved \\
1\ \ subgoal \\
... \\
\quad [\ (((0\ ..\ class) \lhd s) = ((0\ ..\ class) \lhd s'))\ ] \\
... \\
\quad [\ \neg(clear \leq class)\ ] \\
\\
\quad (((0\ ..\ class) \lhd (s \oplus ((0\ ..\ (clear - 1)) \\
\qquad \lhd\!\!| (first\ (appl\ (inp, ((0\ ..\ clear) \lhd s))))))) \\
= ((0\ ..\ class) \lhd (s' \oplus ((0\ ..\ (clear - 1)) \\
\qquad \lhd\!\!| (first\ (appl\ (inp, ((0\ ..\ clear) \lhd s'))))))))
\end{array}
$$

SML

$$
\begin{array}{l}
ze\ (EVERY[Zimp\_res\_then\ MP\_TAC\ le..lemma2, UNDIS \\
\ulcorner Z(0\ ..\ class) \lhd (s \overset{\oplus}{\oplus} STATE2) = (0\ ..\ class) \lhd s' \urcorner, \\
Zrewrite\_tac[SET\_RULE \\
\ulcorner Z(A \lhd z) = (A \lhd z') \Rightarrow (A \cap B) \Rightarrow \\
\qquad (A \lhd (z \oplus (B \lhd\!\!| s))) = (A \lhd (z' \oplus (B \lhd\!\!| s')))\urcorner]])
\end{array}
$$

HOL output

```
subgoal proved
1 subgoal
...
   [ (((0 .. clear) ◁ s) = ((0 .. clear) ◁ s')) ]

   ((second ((construction
      (appl, kernel_implementation))(clear, inp, s)))
 = (second ((construction
      (appl, kernel_implementation))(clear, inp, s'))))
```

SML

```
ze (EVERY[
   ASSUME_TAC kidec,
   Zimp_res_rewrite_tac conpred,
   Zimp_res_rewrite_tac kipred,
   Zasm_rewrite_tac [second_thm]]);
```

HOL output

```
subgoal proved

main goal proved
```

THE CORRECTNESS PROPOSITION

Z

$$\forall app{:}APPLICATION \bullet$$
$$construction(app,\ kernel\_implementation)$$
$$\in secure$$

- this is a trivial consequence of the previous results

- in general overall correctness results will follow straightforwardly from design/implementation correctness results