Refining Rely/Guarantee: a (more) algebraic presentation

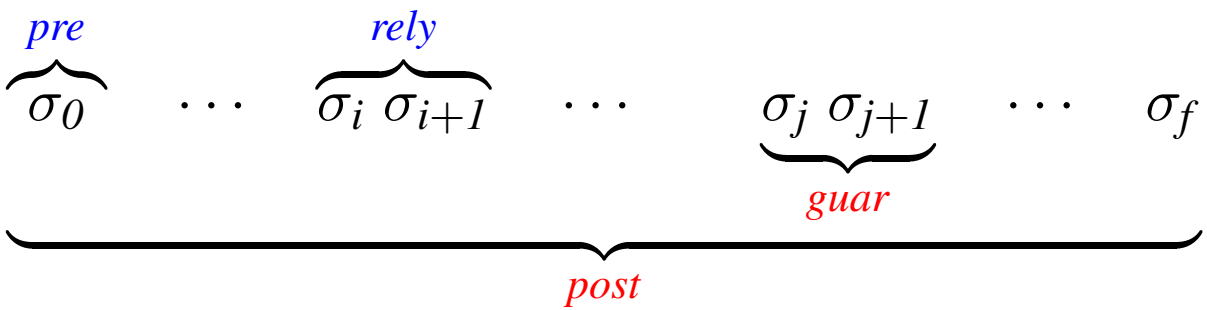
Cliff Jones

Newcastle University
Joint work with Ian Hayes and Rob Colvin (Queensland)

WG 1.9
Vienna
2014-07-14

Basic Rely/Guarantee (R/G) idea (presupposed)

face interference (in specifications and design process)



- assumptions $pre/rely$
- commitments $guar/post$
- can debate specific form of R/G conditions
- many variants/applications — cf. HJJ [HJJ03, JHJ07]

Our aim is to pull apart R/G (and maybe SL)

looking at the issues they cover (rather than the notation *per se*)

- instead of a fixed 5-tuple: $\{P, R\}\ S\ \{G, Q\}$
- separate the concepts
- one presentation in "refinement calculus" style $[P, Q]$
- also allow "framing" as in s: $[s' = s - C]$
- but reservation on refinement calculus presentation below
- allow **guar** $G \bullet c$, **rely** $R \bullet c$
- see [HJC14] (replaces [HJC13])
- preliminary work on SL [JHC14] (replaces [Jon12])

Examples

$$\textbf{guar}\, x < x' \bullet [x' = x + 1] \sqsubseteq x := x + 1$$

$$\textbf{guar}\, x < x' \bullet [x' = x + 2] \sqsubseteq x := x + 1; x := x + 1$$

$$(\textbf{rely}\, x = x' \bullet [x' = x + 1])$$

$$(\textbf{rely}\, x < x' \bullet [x + 1 \leq x'])$$

$$[q_0 \wedge q_1] \sqsubseteq (\textbf{guar}\, g_0 \bullet (\textbf{rely}\, g_1 \bullet [q_0])) \parallel (\textbf{guar}\, g_1 \bullet (\textbf{rely}\, g_0 \bullet [q_1]))$$

Some intuitive Laws

$$(\textbf{guar}\ \textbf{true} \bullet c) = c$$

Nested-G: $$(\textbf{guar}\ g_1 \bullet (\textbf{guar}\ g_2 \bullet c)) = (\textbf{guar}\ g_1 \wedge g_2 \bullet c)$$

Intro-G: $$c \sqsubseteq (\textbf{guar}\ g \bullet c)$$

Trading-G-Q: $$(\textbf{guar}\ g \bullet [g^* \wedge q]) = (\textbf{guar}\ g \bullet [q])$$

$$\textbf{guar}\ g \bullet (c; d) = (\textbf{guar}\ g \bullet c); (\textbf{guar}\ g \bullet d)$$

$$\textbf{guar}\ g \bullet (c \parallel d) = (\textbf{guar}\ g \bullet c) \parallel (\textbf{guar}\ g \bullet d)$$

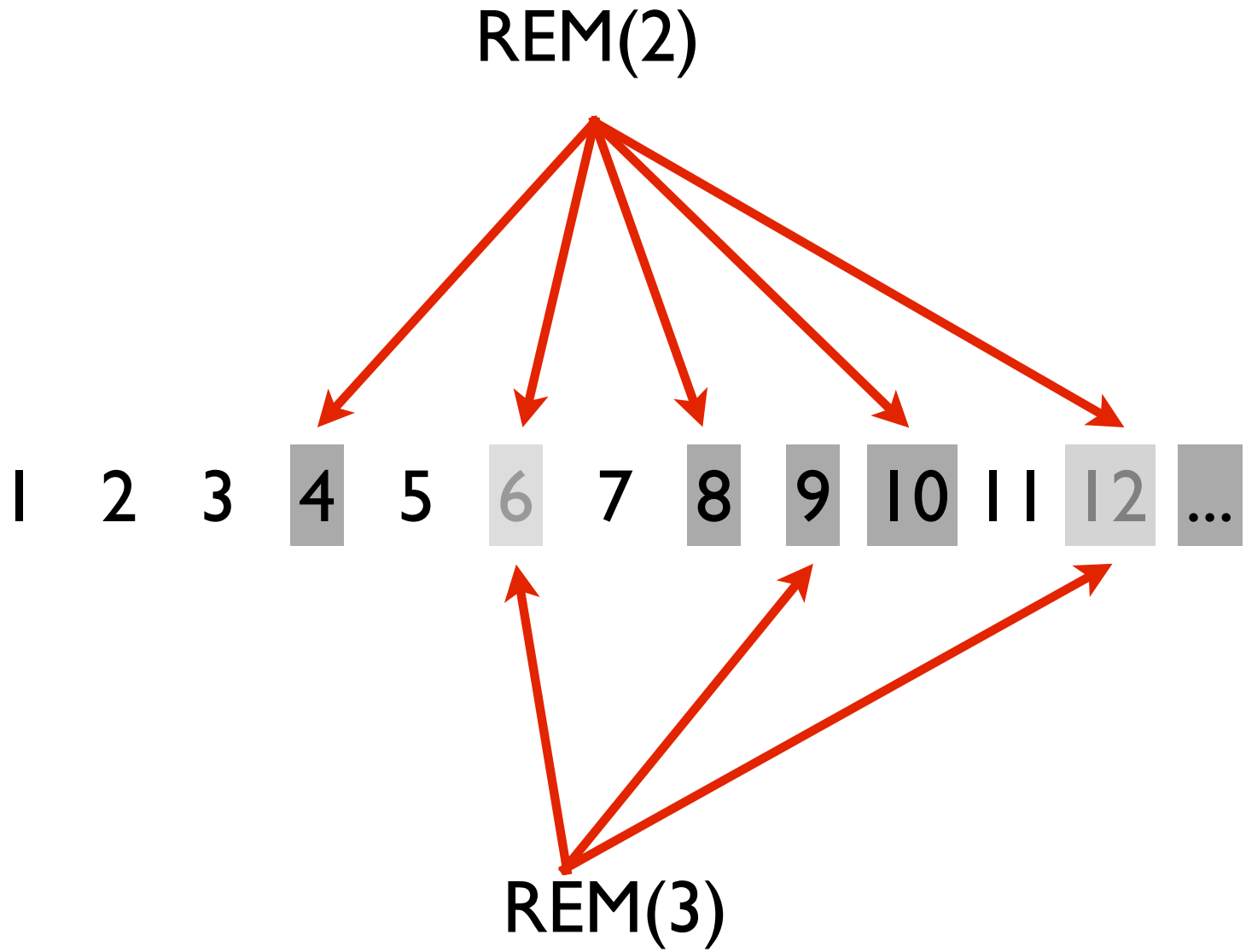
The (actually "a") key Law

Intro-multi-Par: $\wedge_i[q_i] \quad \sqsubseteq \quad \|_i\,(\textbf{guar}\,gr \bullet (\textbf{rely}\,gr \bullet [q_i]))$

This is symmetric (in gr) — many cases are not

Other variants include rules for two operands to $||$

Example: Prime sieve



Example: Prime sieve

illustrates pattern of splitting Q to (weaker) Q and R

SIEVE
wr $s: X$-**set**
post $s' = s - C$

$C = \bigcup\{c_i \mid 2 \leq i \leq \lfloor\sqrt{n}\rfloor\}$
$c_i = \{i * j \mid 2 \leq j \wedge (i * j) \leq n\}$

SIEVE is satisfied by
do $i = 2$ to $\lfloor\sqrt{n}\rfloor$ $REM(i)$

$REM(i)$
post $s' = s - c_i$

Example: Concurrent prime sieve

. . . as a conjuring trick (with rabbits)

$REM(i)$
post $s' = s - c_i$

$SIEVE$ is satisfied by $\|_{i=2}^{\lfloor\sqrt{n}\rfloor} REM(i)$

$REM(i)$
rely $s' \subseteq s$
guar $s - s' \subseteq c_i \wedge s' \subseteq s$
post $s' \cap c_i = \{\}$

Refinement calculus style development

Set s initially contains all (?) natural numbers up to some n

$$[s' = s - C] = [s' \subseteq s \wedge s - s' \subseteq C \wedge s' \cap C = \{\}]$$

$\sqsubseteq$ by Intro-G

$$\textbf{guar}\ s' \subseteq s \wedge s - s' \subseteq C \wedge s' \cap C = \{\} \bullet [s' \subseteq s \wedge s - s' \subseteq C \wedge s' \cap C = \{\}]$$

$\sqsubseteq$ by Trading-G-Q

$$\textbf{guar}\ s' \subseteq s \wedge s - s' \subseteq C \bullet [s' \cap C = \{\}]$$

$\sqsubseteq$ by Intro-muti-Par

$$\textbf{guar}\ s' \subseteq s \wedge s - s' \subseteq C \bullet (\|_i\ \textbf{guar}\ s' \subseteq s \bullet \textbf{rely}\ s' \subseteq s \bullet [s' \cap c_i = \{\}])$$

$\sqsubseteq$ Nested-G

$$\textbf{guar}\ s - s' \subseteq C \wedge s' \subseteq s \bullet (\|_i\ \textbf{rely}\ s' \subseteq s \bullet [s' \cap c_i = \{\}])$$

Reservations

RC is very pretty, but industrial specs are not one-liners

Possible values

- aversion to “history” (aka “ghost”) variables [Jon10]
- “possible values” might offer a new concept in specifications
- we needed something like $post: x = y \vee x = y'$
  - . . . but multiple changes to y possible!
- enter $\widehat{y}$

possible values

$\{P\}x \leftarrow y\{x \in \widehat{y}\}$
remember $\widehat{y}$ is a set

The original one (in developing Simpson's 4-slot):

$post\text{-}start\text{-}Read: hold\text{-}r \in \widehat{fresh\text{-}w}$

SIEVE again

- a useful check at the beginning of $REM(i)$ is whether $i \in s$
- but only of use if the "threads" are launched in sequence
- a better check might be to test $i \in s$ frequently
- but the specification here could be delicate
  $rely\text{-}REM \triangleq i \notin s \Rightarrow$ multiples of i will be deleted
- but with posvals:
  $post\text{-}REM \triangleq (\forall pos \in \widehat{s} \cdot i \in pos) \Rightarrow s' \cap c_i = \{\}$
- remember $guar\text{-}REM$

possible values: good uses

$y \leftarrow 1;$
$(y \leftarrow 3) \;||\; x \leftarrow y \;||\; (y \leftarrow 4)$

$x \leftarrow y$ could have a rely

$rely: \widehat{y} \subseteq \{1, 3, 4\}$

$pre: is\text{-}odd(y)$
$rely: y \neq \overleftarrow{y} \;\Rightarrow\; is\text{-}odd(y)$
or:

$rely: \forall v \in \widehat{y} \cdot is\text{-}odd(v)$

$rely: p(\widehat{(y, z)})$

"Towards" reasoning about posvals

$\{\textbf{true}\}$ **while** $y \neq 0$ **do** $x \leftarrow x + 1$ **od** $\{0 \in \widehat{y}\}$
$\{\textbf{true}\}\ l \leftarrow [v] \frown l\ \{\exists s \in \widehat{l} \cdot \textbf{hd}\, s = v\}$

or:
guar $x \neq \overleftarrow{x} \Rightarrow x = y \cdot C$ **satisifies** $[x \in \widehat{y} \vee x = \overleftarrow{x}]$

with x owned:
(**if** $y = 7$ **then** $x \leftarrow$ **false**) **satisfies** $[x = \textbf{true}, 7 \notin \widehat{y} \Rightarrow x = \textbf{true}]$

FINDP

- classic problem from Owicki's thesis
- illustrates preservation of a property (if it holds)
- **guar-inv** $p \bullet c \triangleq (\textbf{guar}(p \Rightarrow p') \bullet c)$
- (in both the sequential and concurrent development)
- repeats experience that data abstraction/reification intimately linked to R/G
- and ...

FINDP: [HJC14] goes through development of

with:

$satp(v,t) \triangleq t \in \mathbf{dom}\,(v) \wedge p(v(t))$

$notp(v,s,t) \triangleq (\forall i \cdot \in s \bullet i < t \Rightarrow \neg p(v(i)))$

t: **rely** $id(\{v,t\}) \bullet [(t' = len(v)+1 \vee satp(v,t')) \wedge notp(v,dom(v),t')]$

$\sqsubseteq$

$$
\begin{array}{l}
\textbf{var}\ ot, et \bullet \\
ot := len(v) + 1\ ; \\
et := len(v) + 1\ ; \\
\left(\begin{array}{l} \textbf{var}\ oc \bullet \\ oc := 1\ ; \\ \textbf{while}\ oc < ot \wedge oc < et\ \textbf{do} \\ \quad \textbf{if}\ p(v(oc))\ \textbf{then}\ ot := oc \\ \quad\quad \textbf{else}\ oc := oc + 2 \end{array} \right) \\
\quad \| \\
\left(\begin{array}{l} \textbf{var}\ ec \bullet \\ ec := 2\ ; \\ \textbf{while}\ ec < ot \wedge ec < et\ \textbf{do} \\ \quad \textbf{if}\ p(v(ec))\ \textbf{then}\ et := ec \\ \quad\quad \textbf{else}\ ec := ec + 2 \end{array} \right)\ ; \\
t := min(ot, et)
\end{array}
$$

NB tests: use shared variables

are not assumed to be executed atomically

Other on-going work

- semantics (difficult)
- even more abstract R/G — invite Ian to describe (cf. CKAs)
- data abstraction/reification is everywhere — working on best style/fit
- review some of the older extensions to R/G
- “separation as an abstraction” — [JHC14]

Where are we heading?

- R/G has spawned a lot of ideas
- 2 new projects (EPSRC, ARC)
- aim: ("pull apart" R/G and SL) start from issues
  - separation
  - ownership
  - interference
  - progress
  - do once (cf. Linearisability (vs. splitting atoms))
- don't take position:
  "my notation (aka "hammer") solves every problem"
- balance expressive strength/weakness against tractability


Ian J. Hayes, Cliff B. Jones, and Robert J. Colvin.
Reasoning about concurrent programs: Refining rely-guarantee thinking.
Technical Report CS-TR-1395, Newcastle University, September 2013.


Ian J. Hayes, Cliff B. Jones, and Robert J. Colvin.
Laws and semantics for rely-guarantee refinement.
Technical Report CS-TR-1425, Newcastle University, July 2014.


Ian Hayes, Michael Jackson, and Cliff Jones.
Determining the specification of a control system from that of its environment.
In Keijiro Araki, Stefani Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 154–169. Springer Verlag, 2003.


Cliff B. Jones, Ian J. Hayes, and Robert J. Colvin.
Balancing expressiveness in formal approaches to concurrency.
Formal Aspects of Computing, (in press), 2014.


Cliff B. Jones, Ian J. Hayes, and Michael A. Jackson.
Deriving specifications for systems that are connected to the physical world.
In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems: Essays in Honour of Dines Bjørner and Zhou Chaochen on the Occasion of Their 70th Birthdays*, volume 4700 of *Lecture Notes in Computer Science*, pages 364–390. Springer Verlag, 2007.


C. B. Jones.
The role of auxiliary variables in the formal development of concurrent programs.
In Cliff B. Jones, A. W. Roscoe, and Kenneth Wood, editors, *Reflections on the work of C.A.R. Hoare*, chapter 8, pages 167–188. Springer, 2010.


Cliff B. Jones.
Abstraction as a unifying link for formal approaches to concurrency.
In G. Eleftherakis, M. Hinchey, and M. Holcombe, editors, *Software Engineering and Formal Methods*, volume 7504 of *Lecture Notes in Computer Science*, pages 1–15, October 2012.