

# Intuition before Formalism

Cliff Jones  
Newcastle University

FM 2015 Oslo  
2015-06-25

# Contribution!?

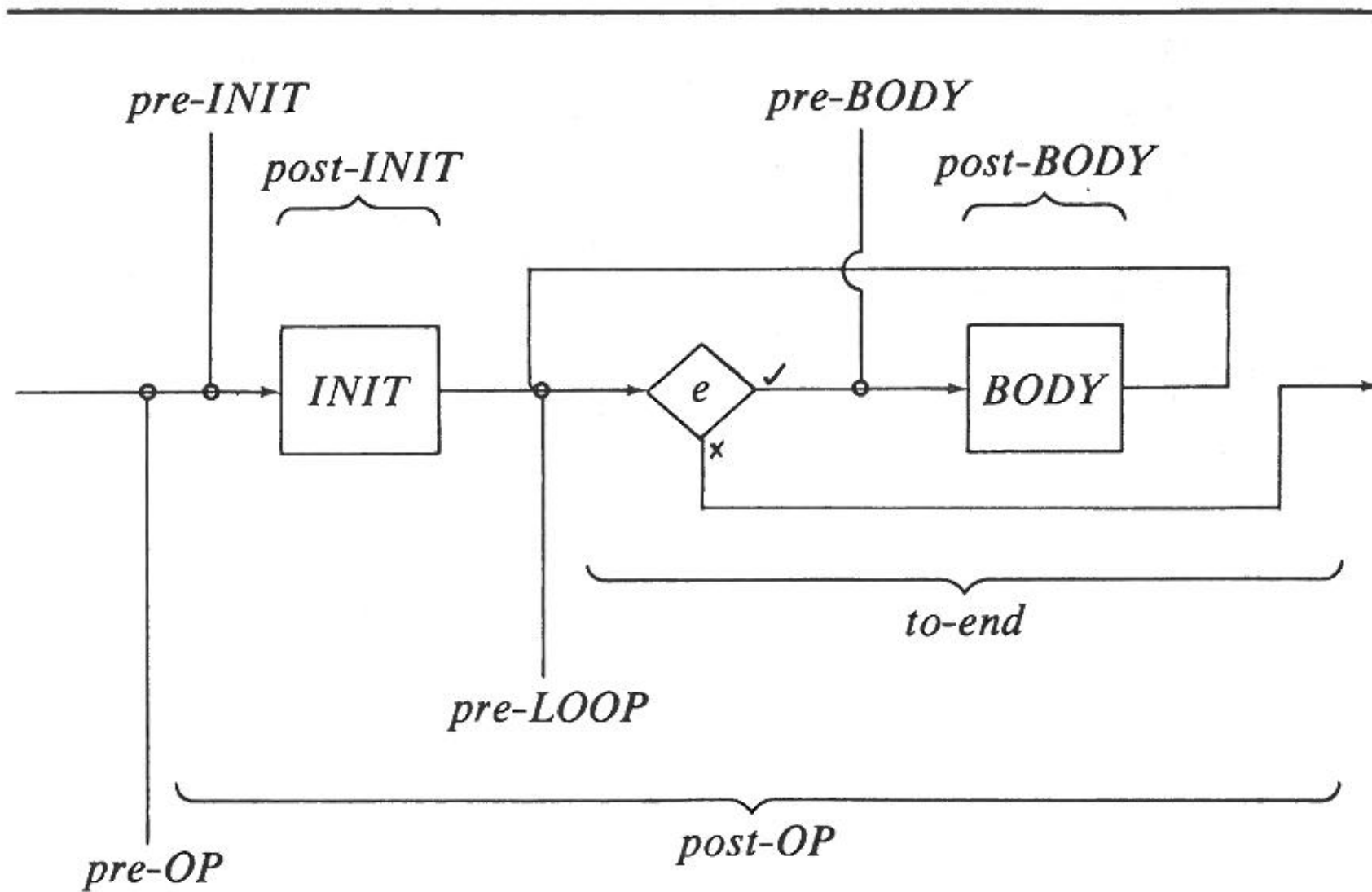
- more of a “heartfelt plea”
- formalism should support (and help check) intuition
- formalism without intuition is (IMHO) sterile

# A first example: relational post conditions

- Floyd's 1966 (mimeographed) paper
- my first spell in Vienna (1968-70): we found [Flo67] hard to understand
- IFIP WG 2.2 meeting April 1969
  - Strachey met Scott
  - Hoare gave (first?) talk on `axiomatic basis ...'
- we decided to invite one of the logicians from the meeting
  - Scott chosen: "but" [dBS69] ▷

# relational post conditions (ii)

- refereed Hoare's [Hoa69] for CACM
  - $\{P\} S \{Q\}$
- back in Hursley TR12.117 [Jon73]
  - used (partial) relations
  - looked at their composition
- first book on "VDM program development"
  - "SDRA" [Jon80]
  - rules for programming constructs  $\triangleright$



**Figure 24** Picture of Initialized Iteration (Down)

# relational post conditions (iii)

- Jones moved to Manchester
- Aczel [Acz82] ▷

remain fixed throughout a computation. But a more flexible and powerful approach has been advocated by Cliff Jones in his book [Jones 1980]. His approach is to allow the postcondition of a specification to depend on the starting state of a computation. So, in our example the postcondition should be

$$(y^2 \leq x < (y+1)^2) \ \& \ (x = \overleftarrow{x})$$

where we use  $\overleftarrow{x}$  to denote the value of  $x$  at the start of the computation. In his book Cliff Jones presents some rules for proving the total correctness of programs for

his notion of specification. His rules appear elaborate and unmemorable compared with the original rules for partial correctness of Hoare. Moreover they are not

# relational post conditions (iii)

- Jones moved to Manchester
- Aczel [Acz82]
  - ... Peter was too polite
  - suggested new presentation
- used in “Jones 3” [Jon86] (and since)
  - but a sneaky liking for some aspects of SDRA rules!



# *Intuition* before Formalism

- intuition
  - post conditions: more than predicates of final state
  - well-founded relations handle termination
  - must terminate (over states of pre condition) ▷

$R$  is transitive, well-founded

$$\boxed{\text{while}} \frac{\{I \wedge b\} S \{I \wedge R\}}{\{I\} \text{ while } b \text{ do } S \text{ od} \{I \wedge \neg b \wedge R^*\}}$$

# *Intuition* before Formalism

- intuition (wrt first example)
  - post conditions: more than predicates of final state
  - well-founded relations handle termination
  - must terminate (over states of pre condition)
- stick to intuition in face of messy rules
  - but grab at a neater form

# Intuition before Formalism

- can I explain what is going on to an engineer?
  - informally!
- link to “posit and prove”
- role of examples
  - for me: essential
- not: “we have a formalism ... next year we plan to try some examples”

# (Data) Abstraction (i)

- Lucas' "twin machine" proof [Luc68] ▷



courtesy: University of Graz

# (Data) Abstraction (i)

- Lucas' "twin machine" proof [Luc68]
- intuition:
  - step from abstraction to representation adds information
  - homomorphism from (more) concrete to abstract
  - using "retrieve functions" LR25.3.067 [Jon70]
- data refinement - in "SDRA" [Jon80]
  - [reification](#) before decomposition in "Jones3" [Jon86]
- but not always possible to find "most abstract" [Mar86] [Nip86]

# more generally

- top-down (TD) view
  - vs. legacy code
  - my attempts to “prove existing programs...”
  - question: is proof cost effective route to correctness?
  - has been judged cost effective for *locating* bugs
  - my belief: FM pays off in design
- TD abstractions
  - useful in BU tools

# Model-oriented specification

- in 1970s, “algebraic specs” were the rage 

$top(push(e, s)) = e$

$pop(push(e, s)) = s$

$is-empty(push(e, s)) = \mathbf{false}$

$is-empty(empty()) = \mathbf{true}$

$pop(empty()) = ?$

$top(empty()) = ??$

try understanding with  $f1/f2/f3$

Veloso's 'traversable stack'



# Model-oriented specification

- in 1970s, “algebraic specs” were the rage
  - Guttag/Zillies (cf. Lucas/Walk)
  - model-oriented approach was a minority sport
- state models were regarded as hacking!
  - but notion of “implementation bias” (1977 ms) and Jones3
  - to me, word algebra was one possible model (often biased)
  - states = equivalence class of histories

# two further example (skip?)

- tool support MORE
  - difficult!
  - we did/do a lot without (tools)
  - OK, tools essential for engineering use
- logics MORE
  - $5/0$  in *not* a natural number!
  - LPF progress on mechanisation [JLS13]


# Concurrency (i)

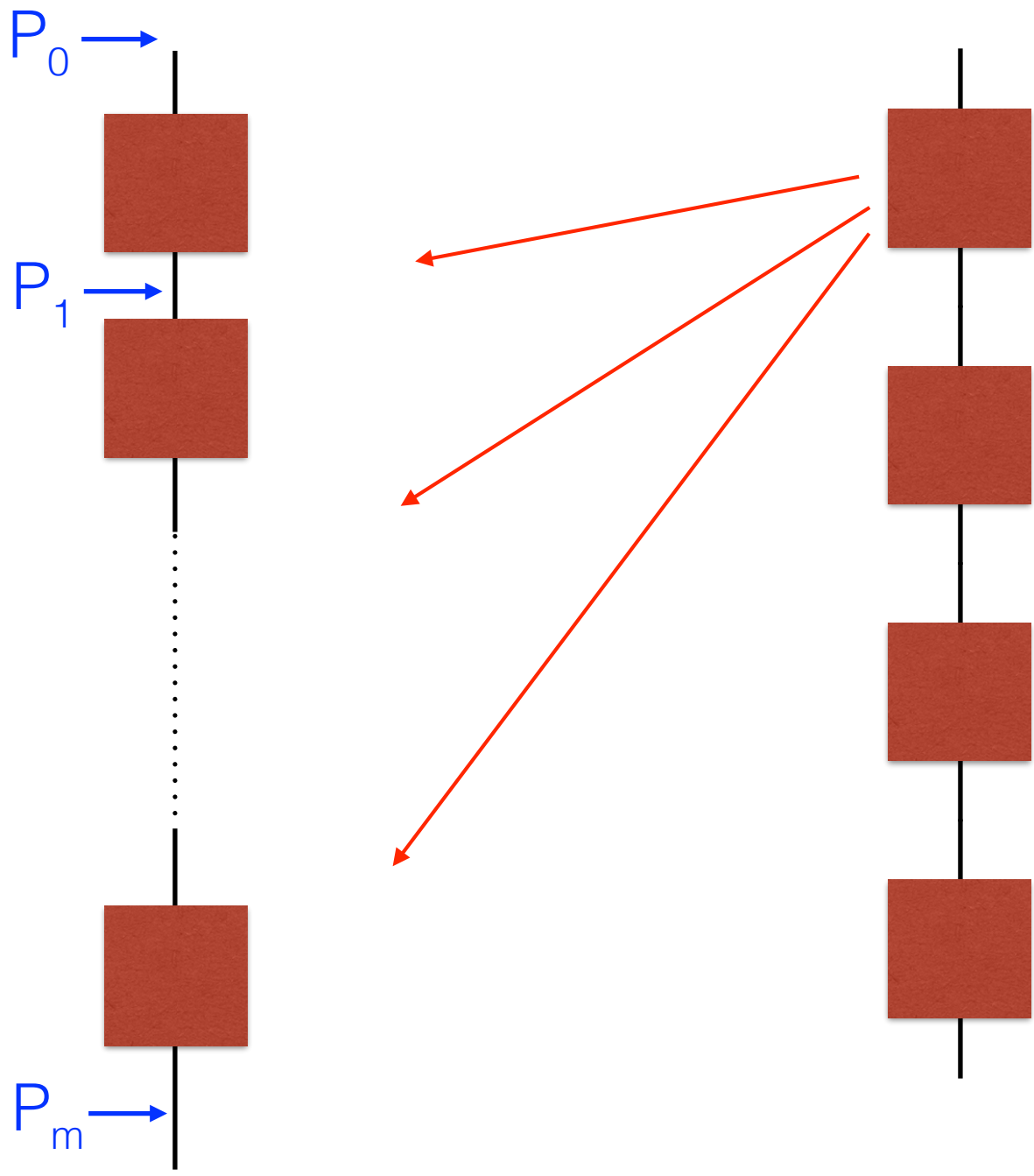
- choice of topic for (belated) doctorate
  - and much of my current research
- Vienna work had largely ignored concurrency
  - except Bekic LNCS177 [BJ84] ▶

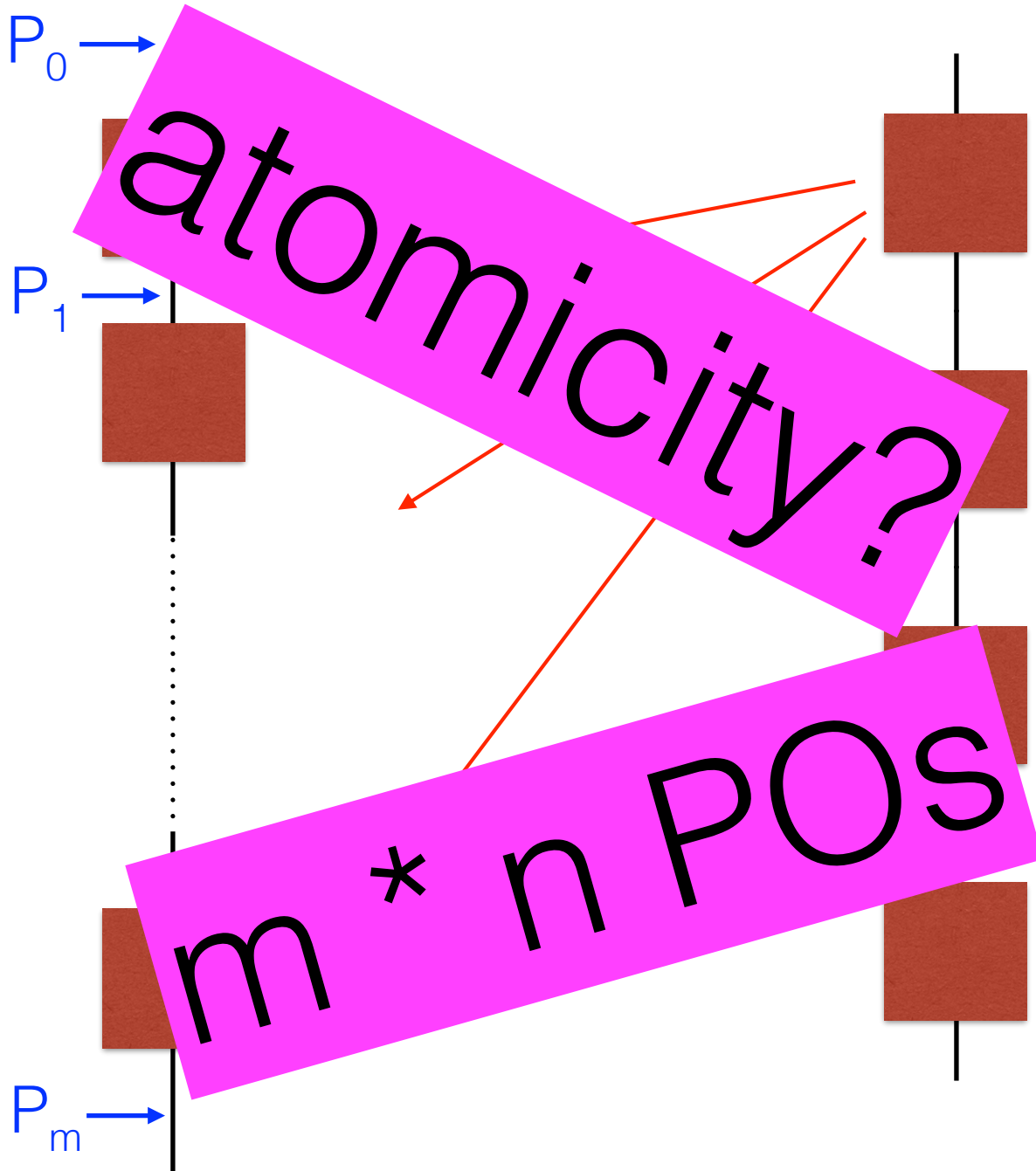


courtesy of de Roever

# Concurrency (i)

- choice of topic for (belated) doctorate
  - and much of my current research
- Vienna work had largely ignored concurrency
  - except Bekic LNCS177 [BJ84]
- as always, wanted “compositionality” (Top Down)
  - Owicki/Gries [OG76] clearly non-compositional 





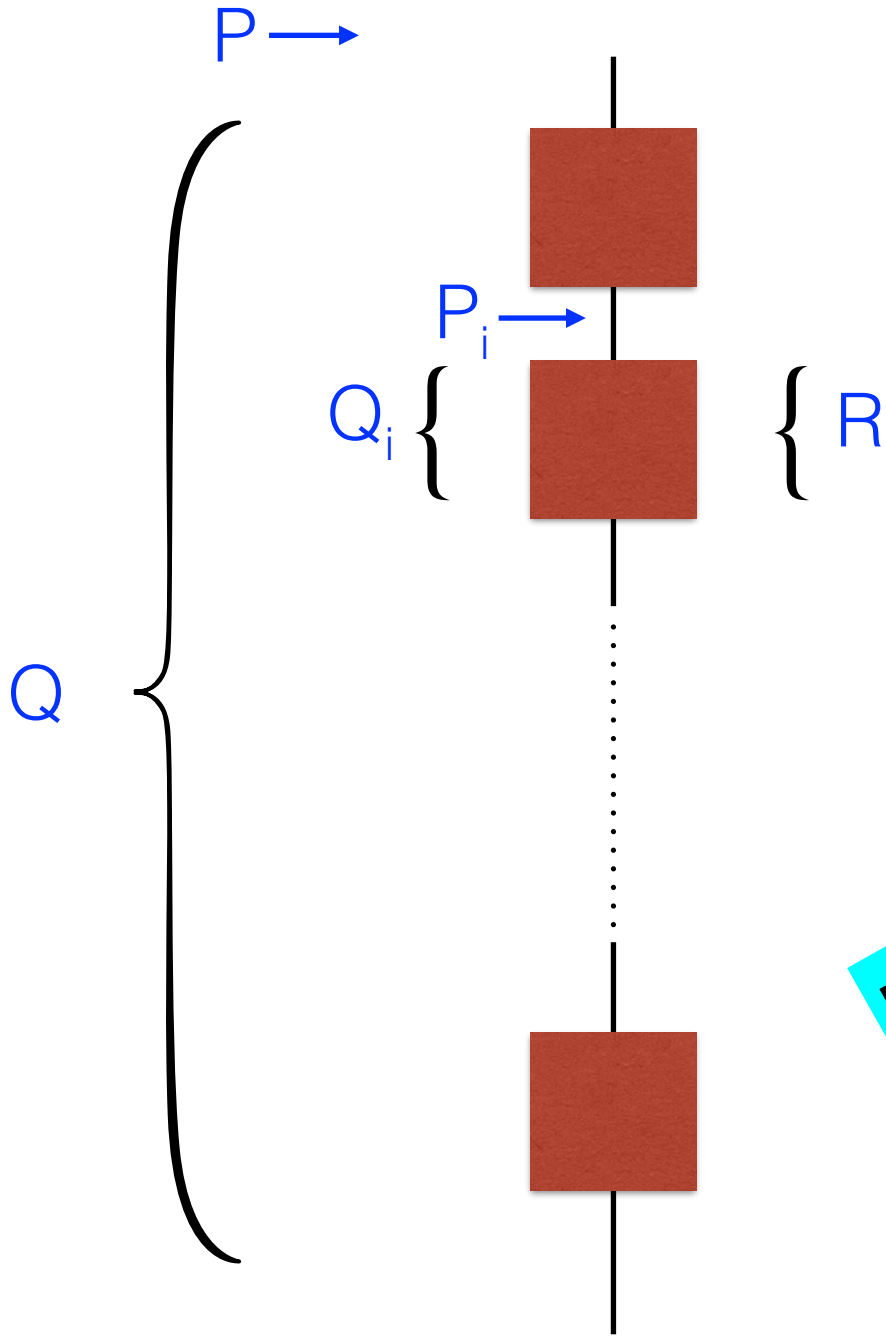
atomicity?

$m * n$  POs

# Concurrency (ii)

- R/G concept is simple: thesis = [Jon81], ..., [BKP84] ▷





Laws for distributing R/G

# Concurrency (ii)

- R/G concept is simple: thesis = [Jon81], ..., [BKP84]
  - but still understanding it!
- “completeness”
  - no it’s not!
  - “auxiliary variables considered harmful” [Jon10]
  - expressive weakness may be its strength?
- concept “pulled apart” [JHC15] ▷

from Morgan's refinement calculus:  $[p, q] \subset \text{Commands}$   
 for any  $c$ , can write: **guar**  $g \cdot c$  or **rely**  $r \cdot c$

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

Intro-G:  $c \sqsubseteq (\mathbf{guar} \ g \cdot c)$

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

Intro-multi-Par:  $\wedge_i [q_i] \sqsubseteq \parallel_i (\mathbf{guar} \ g_i \cdot (\mathbf{rely} \ g_i \cdot [q_i]))$

$[q_1 \wedge q_2] \sqsubseteq (\mathbf{guar} \ g_1 \bullet (\mathbf{rely} \ g_2 \bullet [q_1])) \parallel (\mathbf{guar} \ g_2 \bullet (\mathbf{rely} \ g_1 \bullet [q_2]))$

# Concurrency (iii)

- best/worst info about environment
  - = (non-deterministic) code thereof
- R/G conditions abstract from that
  - (as do post conditions for sequential)
  - “auxiliary variables considered harmful” [Jon10]
  - expressive weakness may be its strength?
- cautiously extending expressiveness (PosVals) ▷

# If ...

- ... I don't have time to talk about semantics
- see Graz paper [Jon01]
- and forthcoming (?) paper on history
  - some history resources at:
  - <http://homepages.cs.ncl.ac.uk/cliff.jones/semantics-library/>
- MORE

# Some (unexpectedly) successful gambles

- VDM-E (Brussels 1987); FM-E; FM
- going straight into industry
  - a late doctorate with Hoare in Oxford
  - but no Research Council funding!
- publication of VDM with Bjørner
  - LNCS61 [BJ78] morphed into [BJ82]
- *Formal Aspects of Computing*
  - Shaw's promise!
  - Vol 26 and counting

# Thanks

- super colleagues along the way
  - ... (far too many to list)
  - especially: “no” men
- FM for this honour!

# Pointers

Intuition before Formalism

References

June 22, 2015

- tool support [MORE](#)
- logics [MORE](#)
- semantics [MORE](#)

**Relational post conditions:** [Flo67, dBS69, Hoa69, Jon73, Jon80, Acz82, Jon86]  
**Data abstraction/reification:** [Luc68, Jon70, Jon80, Jon86, Mar86, Nip86, Jon07]  
**Tool support:** [Kin69, JLM91]  
**LPF:** [GS96, Kle52, BCJ84, JM94, JLS13]  
**Concurrency:** [BJ84, OG76, Jon77, Jon81, BKP84, Jon10, JHC15, San99, Jon07, JY15]  
**Semantics:** [Jon01, JL71, Jon99, ACJ72, BBH<sup>+</sup>74, BJ78, BJ82]  
**History resources:** <http://homepages.cs.ncl.ac.uk/cliff.jones/semantics-library/>  
**History:** [MJ84, Jon03]

## References

- [ACJ72] C. D. Allen, D. N. Chapman, and C. B. Jones. A formal definition of ALGOL 60. Technical Report 12.105, IBM Laboratory Hursley, August 1972.
- [Acz82] P. Aczel. A note on program verification. (private communication) Manuscript, Manchester, January 1982.
- [BBH<sup>+</sup>74] H. Bekič, D. Bjørner, W. Henhapl, C. B. Jones, and P. Lucas. A formal definition of a PL/I subset. Technical Report 25.139, IBM Laboratory Vienna, December 1974.
- [BCJ84] H. Barringer, J.H. Cheng, and C. B. Jones. A logic covering undefinedness in program proofs. *Acta Informatica*, 21(3):251–269, 1984.
- [BJ78] D. Bjørner and C. B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *Lecture Notes in Computer Science*. Springer-Verlag, 1978.
- [BJ82] Dines Bjørner and Cliff B. Jones, editors. *Formal Specification and Software Development*. Prentice Hall International, 1982.



# Role of tool support (this is a difficult one)

- we did/do a lot without (tools)
- OK, tools essential for engineering use
- FM tools must be integrated with standard tools

# dangers of tools

- focus on what we can handle
  - like bibliometrics (what we can count)
  - prove absence of some problems
- tool dictates development steps
- “I proved it with (e.g.) PVS ... but not what I thought I was proving”

# Need tools that support intuition

- King's Effigy [Kin69]
- FDSS in IBM
- mural [JJLM91]
- Rodin tools: [www.event-b.org](http://www.event-b.org)

# AI4FM

- AI from Bundy's team; FM from Jones'
  - Bundy was thinking about "mining proofs"
  - Jones argued for mining "proof process"
- hypothesis:
  - *enough information can be extracted from one proof to make failing proofs of same family work*
- after 4 years: we really understand the task!

# AI4FM

- AI from Bundy's team; FM from Jones'
  - Bundy was thinking about "mining proofs"
  - Jones argued for mining "proof process"
- hypothesis:
  - *enough information can be extracted from one proof to make failing proofs of same family work*
- after 4 years: we really understand the task!
- RETURN

# 5/0 is *not* a natural number!

- McCarthy's conditional operators problematic
  - tried a combination
  - also tried by Gries [GS96] ▷

$a$  **cor**  $b$       if  $a$  then true else  $b$

$a$  **cand**  $b$       if  $a$  then  $b$  else false

not commutative!

$\neg (a \vee (b \text{ **cand** } c))$        $\neg a \wedge (\neg b \text{ **cor** } \neg c)$

$a \wedge (\neg a \text{ **cor** } b)$        $a \text{ **cand** } b$

Quantifiers!?

$\vee$	<b>true</b>	$\perp$	<b>false</b>	$\wedge$	<b>true</b>	$\perp$	<b>false</b>
<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	$\perp$	<b>false</b>
$\perp$	<b>true</b>	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	<b>false</b>
<b>false</b>	<b>true</b>	$\perp$	<b>false</b>	<b>false</b>	<b>false</b>	<b>false</b>	<b>false</b>

Quantifiers are natural generalisations of  $\wedge / \vee$



# 5/0 is *not* a natural number!

- McCarthy's conditional operators problematic
  - tried a combination
  - also tried by Gries [GS96]
- Kleene's 3-valued propositions
  - [Kle52] (blue book) attributes to Łukasiewicz
- “LPF” (untyped) [BCJ84]; (typed) [JM94] ▷
  - recent progress on mechanisation [JLS13]

$$\boxed{\wedge\text{-comm}} \frac{a \wedge b}{b \wedge a}$$

$$\boxed{\text{contrapositive}} \frac{a \Rightarrow b}{\neg b \Rightarrow \neg a}$$

$$\boxed{\text{deMorgan}(i)} \frac{\neg (\exists x \in X \cdot e(x))}{\forall x \in X \cdot \neg e(x)}$$

• RETURN

# Semantics (of PLs)

- intuition
  - we'd tried “debugging a compiler” (1968)
  - some senior IBM managers ...
  - base a *process* on a semantic description?
  - took me to a VDL course (April)
  - then to assignment (August)

# Semantics (ii)

- VDL state (like Karlskirche) too Baroque
  - stack of environments
  - hardest lemmas in [JL71] (in “LNCS0”)
  - control tree - manipulated for goto
  - vs. “exit” mechanism
- Hursley TR12.105 [ACJ72]
  - “functional semantics”

# Semantics (iii)

- exchange of letters 1971/2
  - Bekic had been in UK with Landin
  - Jones attended some of Strachey's lectures
- that phone call (end 1972)!
  - transfer to Wien
  - new team
  - “denotational” ... VDM TR25.139 [BBH+74]
  - “combinators” - including exit (cf. Mosses on Monads [Mos14])
- “FS” machine dropped!

# Aside

- proof of exits vs. continuations
  - nice teasing apart of issues [Jon78]
  - unread?
  - poor sales pitch [Jon82]

# Semantics (iv)

- I taught denotational in Manchester for 15 years
  - may the students forgive me!
- in Newcastle, I now teach SOS
  - because it gives students an *intuitive* tool
  - with a minimum of extra mathematical weight
- RETURN