

# Expression Decomposition in a Rely/Guarantee Context

Joey Coleman



14 January 2009

# Context & Acknowledgements

- Derives from thesis work
- EPSRC “Splitting Atoms” project
- EPSRC “TrAmS” platform grant
- Fine-grained concurrent expression evaluation
- Talk about
  - Semantics
  - Rely/Guarantee
  - Expression decomposition
  - (Atomicity refinement)
- Many small ideas

# Motivation

- Rely/Guarantee reasoning typically uses atomically-evaluated expressions
  - As do many other formalisms
- Most programming languages do not
  - Not in a concurrent context
  - i.e. shared-memory threads
- Bridging that gap

# Expression Semantics

- Each atomic evaluation step, either:
  - replace a single identifier with its *current* value
$$x < y$$
$$\rightarrow x < 42$$
  - reduce an operation to a value
$$23 < 42$$
$$\rightarrow \mathbf{true}$$
- Arbitrary evaluation order
- Interference allowed between steps
- No side-effects
- No locking primitives
- “In the spirit” of Java, etc

# Running Example

$$x < y \wedge x < z$$

- true if  $x$  is less than both  $y$  and  $z$
- The semantics gives us
  - 3 operation reductions
  - 4 variable reads —  $x$  is read twice

# Consequences of this Semantics

- $x = x$ ,  $2x = x + x$  are not tautologies
  - Interference between read-steps
- Expression evaluation is *not* a function
- It is relational; non-determinism plays a role
- And we have interference
  - arbitrary interleaving of evaluation and interference

# About Interference

- Shared-variable model
- Changes the state between evaluation steps
- Representation
  - Relational, over states
  - Transitive, Reflexive
  - Permissive, not mandatory
  - Includes the “Worst Case”
  - The R/G rely condition,  $R$

# Rely/Guarantee in Context

- $\{P, R\}$  program  $\{G, Q\}$ 
  - Assuming  $P$  and  $R$ , *program* satisfies  $G$  and  $Q$
- We can reason about programs using the semantics
  - We can also program in machine code
- Rely/Guarantee reasoning allows abstraction
  - Development rules allow the composition of programs
- R/G can extend the semantic framework
  - Requires soundness but not completeness

# (Meta-)Semantic Mismatch

- R/G
  - Predicates have meaning relative to a single state
  - Relations have meaning relative to two states
- Semantics
  - Expressions evaluated using many states
  - Interference plays a role
- Taking the meaning of an expression relative to a single state is not the same as evaluating it

# Mismatch Examples

- Evaluating  $x = y$  when  $R$  ensures that  $x = y$ 
  - In any single state,  $x = y$  holds
  - But  $x$  may not be read in the same state as  $y$ ...
- Evaluating  $x = x$  with arbitrary interference
  - R/G (atomic eval): always true
  - Semantic eval: two  $x$ s gives two arbitrary values

# (Almost) Single-State Evaluation

- Reconciles semantics to R/G
- If the expression
  - has only one variable affected by interference
  - and only one instance of said variable
- Then
  - evaluation appears to happen relative to single state
- But
  - said state is arbitrarily determined by interference
- *SingleUnstableVariable*( $e, R$ )

## Example, Continued

$$e \triangleq x < y \wedge x < z$$
$$R \triangleq I_{\{x,y\}} \wedge z \leq \frac{1}{z}$$

- $x, y$  are not affected by interference
  - $I_{x,y}$  is the identity relation on  $x$  and  $y$
- $z$  monotonically decreases
- $e$  is only affected by interference on  $z$
- $e$  has only one instance of  $z$
- Thus, *Single Unstable Variable*( $e, R$ )

# (Meta-)Semantic Mismatch in Context

$$\frac{\{P \wedge \llbracket e \rrbracket, R\} \text{ body } \{G, Q\} \quad \overleftarrow{P} \wedge \neg \llbracket e \rrbracket \Rightarrow Q}{\{P, R\} \text{ mk-If}(e, \text{body}) \{G, Q\}}$$

- Atomically-evaluated rule for *If*
- Not sound relative to our semantics
- $P \wedge \llbracket e \rrbracket$  might not hold
- Nor might  $\overleftarrow{P} \wedge \neg \llbracket e \rrbracket$

# Decomposing the Expression

$$\begin{aligned}e &\triangleq x < y \wedge x < z \\e_s &\triangleq x < y \\e_u &\triangleq x < z \\R &\triangleq I_{\{x,y\}} \wedge z \leq \overset{\leftarrow}{z}\end{aligned}$$

- $e$  is too coarse
- Introduce  $e_s$  and  $e_u$
- $e_s$  is unaffected by interference
  - $R \Rightarrow I_{\{x,y\}}$
  - or,  $R \Rightarrow I_{FV(e_s)}$
- $e_u$  is affected by interference only on  $z$ 
  - *SingleUnstableVariable*( $e_u, R$ ) holds

# Decomposed Development Rule

$$\frac{
 \begin{array}{c}
 R \Rightarrow I_{FV(e_s)} \\
 \text{SingleUnstableVariable}(e_u, R) \\
 \{P \wedge \llbracket e_s \rrbracket, R\} \text{ body } \{G, Q\} \\
 \overline{P} \wedge \neg \llbracket e_s \wedge e_u \rrbracket \Rightarrow Q
 \end{array}
 }{
 \{P, R\} \text{ mk-If}(e_s \wedge e_u, \text{body}) \{G, Q\}
 }$$

- Almost sound
  - interference can “bypass” the last hypothesis
  - we do not know how  $R$  affects  $e_u$

# Decomposed Development Rule

$$\begin{array}{c}
 R \Rightarrow (\neg \overset{\leftarrow}{e}_u \Rightarrow \neg e_u) \\
 R \Rightarrow I_{FV(e_s)} \\
 \text{SingleUnstableVariable}(e_u, R) \\
 \{P \wedge \llbracket e_s \rrbracket, R\} \text{ body } \{G, Q\} \\
 \frac{\overset{\leftarrow}{P} \wedge \neg \llbracket e_s \wedge e_u \rrbracket \Rightarrow Q}{\{P, R\} \text{ mk-If}(e_s \wedge e_u, \text{body}) \{G, Q\}}
 \end{array}$$

- Almost sound
  - interference can “bypass” the last hypothesis
  - we do not know how  $R$  affects  $e_u$
- Ensure that interference keeps  $e_u$  false
  - Requires *SingleUnstableVariable* to work

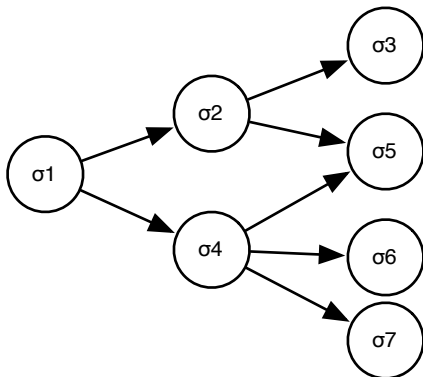
# Advertising

- In the paper...
  - Connection to Atomicity Refinement
  - Actual semantic rules
  - Specific R/G framework
  - Development of the R/G rules
  - Transformation of  $x < \min(y, z)$  into  $x < y \wedge x < z$
- LNCS 5295 (VSTTE'09 Proceedings)

# Current Work — Atomicity Refinement

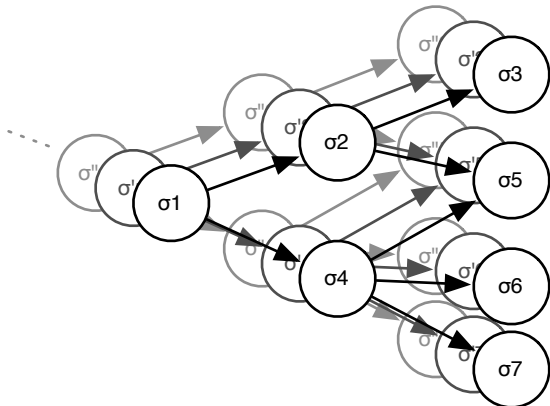
- The transformation of  $e$  into  $e_s \wedge e_u$ 
  - Less interested in the R/G aspects *per se*
- Looking at creating a refinement relation
  - $e' \sqsubseteq_R e$
  - meaning  $e'$  is “less non-deterministic” than  $e$
  - or “less affected by interference”
- My aim is to find a way of determining if an  $e'$  is a good replacement for  $e$

# An Evaluation Model



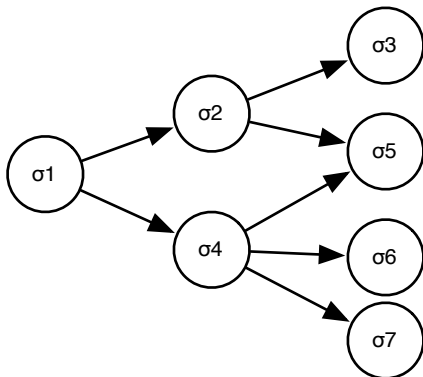
- Nodes are states, arrows correspond to  $R$
- Interference from an initial state with a specific  $R$

# An Evaluation Model



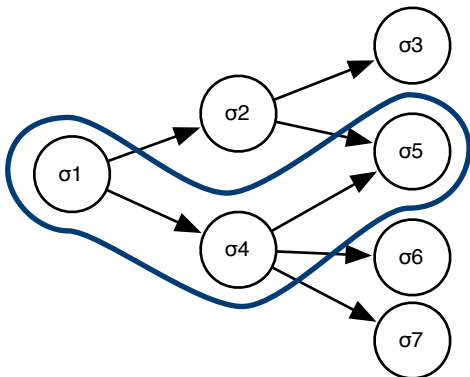
- Nodes are states, arrows correspond to  $R$
- Many possible initial states

# An Evaluation Model



- Nodes are states, arrows correspond to  $R$

# An Evaluation Model



- Nodes are states, arrows correspond to  $R$
- A specific path of interference

## Using this Model

- Each interference path becomes a vector
  - $\{(\sigma_1, \sigma_4, \sigma_5, \dots), \dots\}$
- If we use *SingleUnstableVariable* then the meaning of  $e$  relative to  $\sigma_i$  is equivalent to the evaluation of  $e$  starting in  $\sigma_1$  if the unstable variable is read from  $\sigma_i$
- This gives us a set of vectors of values for each expression,
  - $e \longrightarrow \{(v_1, v_4, v_5, \dots), \dots\}$
  - $e' \longrightarrow \{(v'_1, v'_4, v'_5, \dots), \dots\}$
  - Correspond to the paths of interference
- Refinement is a comparison of the value vectors

## Caveat comparator...

- This model is doubly-quantified
  - Refinement if *all* paths from *all* initial states give corresponding value vectors which compare favourably
- The comparison is straightforward
- But, actual use in a proof is probably ugly
- So, current effort is on
  - looking at examples
  - a simpler set of rules that are sound relative to this

Thank you.  
Questions?