Relational decomposition

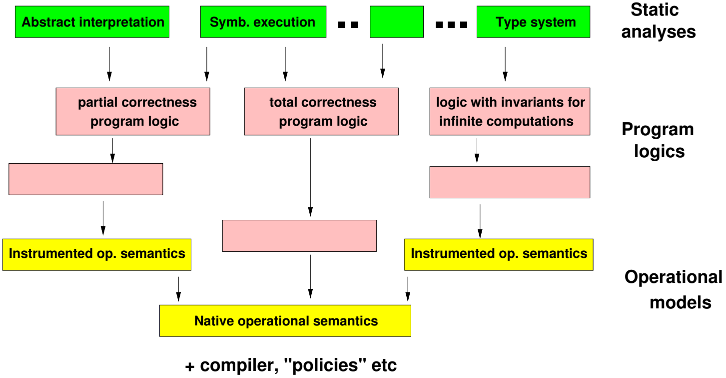
Lennart Beringer

Department of Computer Science
Princeton University

ITP 2011, August 22-26, Nijmegen

Partially funded by AFOSR FA9550-09-1-0138 and NSF CNS-0910448

Program certification stacks: (F)PCC, VST,. . .



- Formal embedding in theorem prover (Isabelle/HOL, Coq,. . .)
  - interactive/automated discharging of VCGens
  - construction of semantic model / soundness proof
  - exploit expressivity of meta-logic in interpretation of assertions
  - use program extraction, reflective inference, proof checking . . .

Aim: reuse of formalisms

- amortize the formalization effort and TCB infrastructure

(Rel.) completeness: "sanity check" for given proof rules

- specific to format & interpretation of assertions & judgements
- typically, interpretation concerns single program execution

Observation: many program properties are relational

- extensional interpretations of program analyses: liveness, def-use chains, (in-)dependencies, slicing...
- security: noninterference, fault-tolerance
- program equivalences (correctness of compiler transformations and translations, bisimulations)
- PL theory: polymorphism/parametricity, types & effects...

Aim: reuse of formalisms

- amortize the formalization effort and TCB infrastructure

(Rel.) completeness: "sanity check" for given proof rules

- specific to format & interpretation of assertions & judgements
- typically, interpretation concerns single program execution

Observation: many program properties are relational

- extensional interpretations of program analyses: liveness, def-use chains, (in-)dependencies, slicing...
- security: noninterference, fault-tolerance
- program equivalences (correctness of compiler transformations and translations, bisimulations)
- PL theory: polymorphism/parametricity, types & effects...

Reasoning about two-execution properties

- Syntactic approaches (CFG, program points, paths): translation validation, Voronkov$^+$,...
- Relational program logics (Benton, Yang, Amtoft$^+$)
  - judgements over pairs of program phrases
  - pre-/postconditions are state relations
- Self-composition (Barthe$^+$, Joshi$^+$, Darvas$^+$, Beringer$^+$)
  - reduce two-execution property to some one-execution property of a different program (syntactic translation)
  - then use existing unary verification calculi

Example: c noninterferent iff $\{\overline{l_i = l_i'}\}c; c'\{\overline{l_i = l_i'}\}$
  - algorithmic improvements: Terauchi$^+$

Contribution

Relational decomposition

- technique for deriving relational (2-execution) program logics from unary logics (Hoare, VDM, wp)
- provides a compositional analysis of self-composition at level of program logics
- applicable across different languages / op. semantics / states

Rest of talk:

1. Formal definitions & abstract results
2. Instantiation: compositional derivation of RHL
   - termination-insensitive interpretation
   - novel rules for dissonant loops

Formalization in Isabelle/HOL

Formal setup: transition systems, simulations

Starting point: unary program logics $\rhd C : A$

- Big-step operational semantics (LTS): $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{P} \times \mathcal{S}$
- Curried unary assertions $A \in \mathcal{S} \Rightarrow \mathcal{S} \Rightarrow \mathbb{T}$

Partial-correctness judgement $\models^{\mathcal{T}} C : A$

If $(\sigma, C, \tau) \in \mathcal{T}$ then $A\ \sigma\ \tau$.

Relational simulation: pairs of LTS's, relational assertions R, S:

$$\begin{tikzcd}
\sigma \arrow[r, dash, "R"] \arrow[d, "C"'] & \sigma' \arrow[d, "C'"] \\
\tau \arrow[r, dash, "S"'] & \tau'
\end{tikzcd}$$

Term.-insensitive simulation $\models^{\mathcal{T}'}_{\mathcal{T}} C \sim C' : R \Rightarrow S$

If $(\sigma, C, \tau) \in \mathcal{T}$ and $(\sigma', C', \tau') \in \mathcal{T}'$ and $\sigma R \sigma'$ then $\tau S \tau'$.

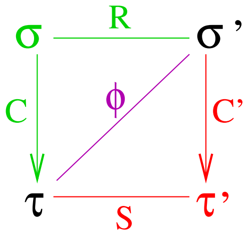
Research task

Characterize relational simulation without direct reference to operational semantics, just with respect to one-execution logic.

Relational decomposition

Given relation $\phi \subseteq \mathcal{S} \times \mathcal{S}'$, define the operators Dec_{L} and Dec_{R}

$$Dec_{\mathsf{L}}\ R\ \phi\ \sigma\ \tau \quad = \quad \forall \sigma'.\ \sigma R \sigma' \to \tau \phi \sigma'$$



$$Dec_{\mathsf{R}}\ S\ \phi\ \sigma'\ \tau' \quad = \quad \forall \tau.\ \tau \phi \sigma' \to \tau S \tau'$$

The operators yield unary specifications for the one-executions:

- Dec_{L}: ϕ is postcondition for R along C, for fixed primed state
- Dec_{R}: ϕ is precondition for S along C' for fixed nonprimed state

Relational decomposition: soundness

Soundness

Suppose $\models^{\mathcal{T}} C : Dec_{\mathsf{L}}\ R\ \phi$ and $\models^{\mathcal{T}'} C' : Dec_{\mathsf{R}}\ S\ \phi$. Then $\models C \sim C' : R \Rightarrow S$.

How to obtain diagonal relations ϕ?

- Constructively: construct ϕ once and for all in derivation of proof rules for relational logics, compositionally along phrase structures
- Conceptually: existence of extremal witnesses (completeness)

Extremal witnesses, completeness

Strongest postcondition of R along C is min. witness $\phi_{\mathsf{L}}^{\mathcal{T}}\ C\ R$:

- Satisfies $\phi_{\mathsf{L}}^{\mathcal{T}}\ C\ R \subseteq \phi$ whenever $\models^{\mathcal{T}} C : Dec_{\mathsf{L}}\ R\ \phi$

Weakest lib. precondition of S along C' is max. witness $\phi_{\mathsf{R}}^{\mathcal{T}'}\ C'\ S$:

- Satisfies $\phi \subseteq \phi_{\mathsf{R}}^{\mathcal{T}'}\ C'\ S$ whenever $\models^{\mathcal{T}'} C' : Dec_{\mathsf{R}}\ S\ \phi$

Thus, *any* witness ϕ satisfies $\phi_{\mathsf{L}}^{\mathcal{T}}\ C\ R \subseteq \phi \subseteq \phi_{\mathsf{R}}^{\mathcal{T}'}\ C'\ S$.

Completeness

Let $\models C \sim C' : R \Rightarrow S$. Any relation $\phi_{\mathsf{L}}^{\mathcal{T}}\ C\ R \subseteq \phi \subseteq \phi_{\mathsf{R}}^{\mathcal{T}'}\ C'\ S$ satisfies $\models^{\mathcal{T}} C : Dec_{\mathsf{L}}\ R\ \phi$ and $\models^{\mathcal{T}'} C' : Dec_{\mathsf{R}}\ S\ \phi$.

Corollary: witness-free characterization

$\models C \sim C' : R \Rightarrow S$ is equivalent to $\phi_{\mathsf{L}}^{\mathcal{T}}\ C\ R \subseteq \phi_{\mathsf{R}}^{\mathcal{T}'}\ C'\ S$.

Instantiation: $\mathbf{IMP}$ + simple objects

Benton/Yang-style relational logic $\vdash C \sim C' : R \Rightarrow S$, but

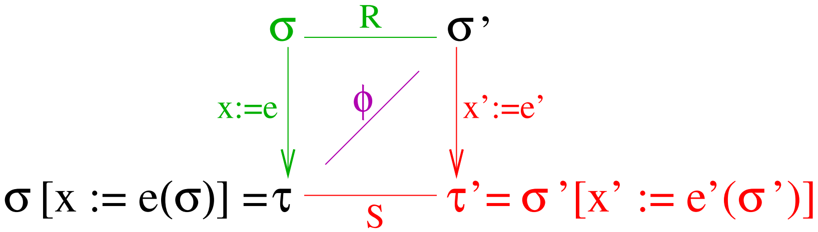
- termination-insensitive interpretation
- justification of rules exhibits witnesses ϕ
- formally: define judgement form $\vdash C \sim C' : R \Rightarrow S$ as

$$\exists\, \phi.\ \rhd C : Dec_{\mathsf{L}}\ R\ \phi \wedge \rhd C' : Dec_{\mathsf{R}}\ S\ \phi$$

  and *derive* proof rules
  - minor differences due to termination, PER-ness
  - perfect decomposition w.r.t L/R

Derived RHL: Assign-Assign

$$\text{R-AssAss}\ \frac{}{\vdash x{:=}e \sim x'{:=}e' : S[e/x, e'/x'] \Rightarrow S}$$

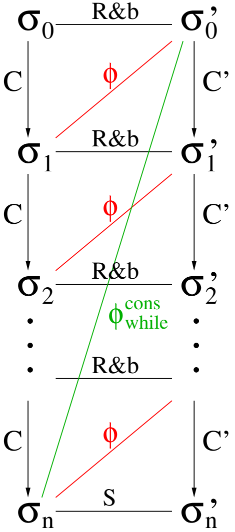


$$\phi = \{(\tau, \sigma').\ \tau \mathbf{S} (\sigma'[\mathbf{x}' := \mathbf{e}'(\sigma')])\}$$

Nonatomic statements & transformation rules : synthesize witnesses for concluding judgement from witnesses of hypothetical judgements

Consonant loops $\vdash \textbf{While}\ b\ \textbf{do}\ C \sim \textbf{While}\ b'\ \textbf{do}\ C' : R \Rightarrow S$

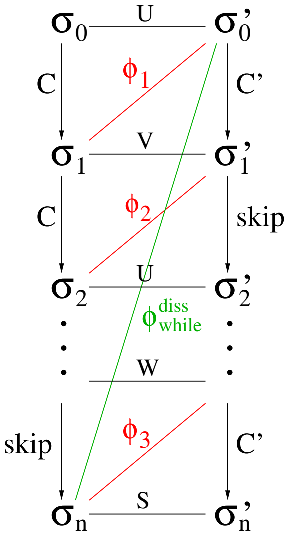
Benton-Yang: iterations must proceed in lock-step



R = T & b=b'
S = R & ~b

Dissonant loops $\vdash$ While b do $C \sim$ While b' do $C' : R \Rightarrow S$

New rule: split R into homogeneous U and inhomogeneous V, W



U = R & b & b'
V ⊆ R & b
W ⊆ R & b'
S = R & ~b & ~b'
R ⊆ U ∪ V ∪ W ∪ S

Discussion

Additional material:

- In paper:
  - new loop rule in action
  - parametrized relational decomposition (aux. state)
  - more details on assertion/predicate transformers
- In formalization:
  - RD for logics with fault states
  - derivation of unary and relational separation logics

Future work:

- instantiation for unstructured code, compiler correctness
- algorithmic reformulation (Terauchi-Aiken), product programs (Barthe-Crespo-Kunz)
- point-free formulation
- termination-sensitive relational decomposition
- scale up to non-toy languages
- HW equivalence checking?