Korrekte Software: Grundlagen und Methoden
Vorlesung 15 vom 30.06.16: Separation Logic
Slides courtesy of Rajeev Goré, ANU, Australia

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2016

Hoare Logic: syntax, semantics and calculus

Syntax	Semantics	Calculus
$\neg \wedge \vee \Rightarrow \forall \exists$	FOL	N/A
$= + - \geq \leq \cdots$	Arithmetic	N/A
`:= ; while` `if then else`	State maps variables to values (no pointers)	N/A
$\{P\}S\{Q\}$	if initial state satisfies P and S terminates then final state satisfies Q	6 Inference Rules

Axiom Schemas

$$p_1 * p_2 \Leftrightarrow p_2 * p_1$$
$$(p_1 * p_2) * p_3 \Leftrightarrow p_1 * (p_2 * p_3)$$
$$p * \textbf{emp} \Leftrightarrow p$$
$$(p_1 \vee p_2) * q \Leftrightarrow (p_1 * q) \vee p_2 * q)$$
$$(p_1 \wedge p_2) * q \Leftrightarrow (p_1 * q) \wedge p_2 * q)$$
$$(\exists x.p) * q \Leftrightarrow \exists x.(p * q) \quad \text{when } x \text{ not free in } q$$
$$(\forall x.p) * q \Leftrightarrow \forall x.(p * q) \quad \text{when } x \text{ not free in } q$$

Unsound axiom schemas

$$p \Rightarrow p * p \quad \text{(Contraction)}$$
$$p * p \Rightarrow p \quad \text{(Weakening)}$$

More valid axiom schemas

$$p_1 \wedge p_2 \Rightarrow p_1 * p_2 \quad \text{when } p_1 \text{ or } p_2 \text{ pure}$$
$$p_1 * p_2 \Rightarrow p_1 \wedge p_2 \quad \text{when } p_1 \text{ and } p_2 \text{ pure}$$
$$(p \wedge q) * r \Rightarrow p \wedge (q * r) \quad \text{when } p \text{ pure}$$

Pure Expressions

An expression e is pure, if it does neither contain $\mapsto$, $\rightsquigarrow$ nor **emp**.

Showing $x = y$

$$(6)\{x = x_1 \wedge x \mapsto v * y = y_1 \wedge y \mapsto v\}x := [x]; y = y$$
$$\{x = v \wedge x_1 \mapsto v * y = v \wedge y_1 \mapsto v\}$$

$x = v \wedge x_1 \mapsto v * y = v \wedge y_1 \mapsto v$
$\Rightarrow x = v * x_1 \mapsto v * y = v \wedge y_1 \mapsto v$ — $x = v$ pure
$\Rightarrow x = v * x_1 \mapsto v * y = v * y_1 \mapsto v$ — $y = v$ pure
$\Rightarrow (x = v * y = v) * x_1 \mapsto v * y_1 \mapsto v$
$\Rightarrow (x = v \wedge y = v) * x_1 \mapsto v * y_1 \mapsto v x = v$ and $y = v$ pure
$\Rightarrow x = y * x_1 \mapsto v * y_1 \mapsto v$

Mutation

$$\{e \mapsto -\}[e] := e'\{e \mapsto e'\}$$

Example $[x] := \text{cpns}(3, 4)$

	St	Hp		$x := \text{cpns}(3,4)$	St	Hp	
	$x = 20$	20	21		$x = 20$	20	21
	$*1$	2			3	4	

Axiom Instance

$$\{x \mapsto 20, 21\}[x] := cons(3, 4)\{x \mapsto 3, 4\}$$

Mutation (backwards)

$$\{e \mapsto - * (e \mapsto e' -\!\!* p)\}[e] := e'\{p\}$$

Example $[x] := \text{cpns}(3, 4)$

	St	Hp		$x := \text{cpns}(3,4)$	St	Hp	
	$x = 20$	20	21		$x = 20$	20	21
	$*1$	2			3	4	

Axiom Instance

$$\{x \mapsto 20, 21 * (x \mapsto 3, 4 -\!\!* x \mapsto 3 \wedge x + 1 \mapsto 4)\}[x] := cons(3, 4)\{x \mapsto 3 \wedge x + 1 \mapsto 4\}$$

Summary

- Separation logic is the method to really handle point structures
- Can also handle function and procedure calls.
- Needs to be adapted for C