Conditional Parametricity in

Jan Gilcher, Andreas Lochbihler, Dmitriy Traytel
Institute of Information Security, Department of Computer Science, ETH Zurich

Relational parametricity

Informal definition: parametric ≈ “truly” polymorphic
Parametric functions, whose type contains type variables, behave exactly the
same, no matter what concrete type they are actually used with at run-time.

Examples and non-examples

<table>
<thead>
<tr>
<th>function</th>
<th>parametric in α β</th>
<th>reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>(·) : α → α list → α list</td>
<td>✗</td>
<td>-</td>
</tr>
<tr>
<td>length : α list → N</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>map : (α → β) → α list → β list</td>
<td>✗</td>
<td>constant iff</td>
</tr>
<tr>
<td>(·) : α → α → β</td>
<td>✗</td>
<td>head [] under-specified</td>
</tr>
<tr>
<td>lookup : (α × β) list → α → β option</td>
<td>✗</td>
<td>needs (=) on α</td>
</tr>
</tbody>
</table>

Applications in

• data refinement
• theorem transfer across subtypes and quotients
• productivity of non-primitively corecursive definitions
• nonuniform (co)datatypes

Formal definition and examples

Types as relations, type constructors as relations
(a ⊗ β denotes the type of binary relations between α and β)

function space: related inputs are mapped to related outputs
⇒ : α → α′ → β → β′ → (α → β) → (α′ → β′)

list: same length and position-wise related elements
relₜ : α → α → α relₜ α relₜ α
((relₜ R) ys in [x₁, x₂, ... , xₙ] ∧ ∀ i ≤ n, xᵢ R yᵢ) ⇒ [y₁, y₂, ... , yₙ]

any “well-behaved” type constructor π T:
relₜ requires same T-shape and position-wise related elements
relₜ : α₁ ⊗ α₁′ → · · · → αₙ ⊗ αₙ′ ⇒ π relₜ αₙ ⊗ π relₜ αₙ′

(Conditional) Parametricity

c ≪ π T in parametric iff (c ≪ π T) (relₜ R₁ · · · Rₙ) (π ≪ T) for all π T

c is C-parametric if the same property holds for those π T that satisfy C π T

Examples

1. Infer parametricity relation ?R and conditions ?C
2. Simplify inferred conditions C by
   • removing duplicates and
   • applying rules for type constructors
3. Enter parametricity theorem C ⇒ c R ∈ in database DBparam

Evaluation

Inferred parametricity for 730 non-recursive and primitively recursive
polyomorphism definitions in the Isabelle/HOL standard library using the
existing database DBparam of parametricity theorems.

| trivial | inferred theorem is c = c |
| existing | rediscovered previously |
| better | inferred theorem has fewer |
| new | definition without existing |
| better | parametricity theorem |

Comparison with other parametricity provers

parametric_constant sum_def — "Our tool"

lemma — transfer_prover (Huffman, Kuncar)
includes lifting_syntax
assumes [transfer_rule]: (R 0 0) (R ≡ R) (op +) (op ×)
shows (list_all2 R ≡ R) (sum sum)
unfolding sum_def by transfer_prover

lemma — Autoref (Lammich)
assumes (0, 0) ∈ R (op +, op ×) ∈ R −→ R −→ R
shows (sum, sum) ∈ (R list_rel −→ R)
using assms unfolding sum_def by parametricity