Everything is a lens

Taming partiality using invariants and non-determinism

Nuno Macedo
HASLab
INESC TEC & Universidade do Minho

Bidirectional Transformations

- Transforming data between different structures is a typical computing problem;
- Transformations should be *bidirectional*: changes in either structure should be propagated to the other;
- Defined between *sources* and *views* (views being typically more abstract), and consist of a *forward* and a *backward* transformations;
- Example: view-update problem from databases.

Lenses

- *Lenses* are one of the most successful approaches;
- A lens $l : S \trianglerighteq V$ comprises transformations

$$get : S \to V \qquad put : V \times S \to S$$

- *put* uses the original source to generate the updated source.

Lenses

- A lens is said to be *well-behaved* if it satisfies the following round-tripping laws:

$$put\,(s, get\,s) \sqsubseteq s$$

$$get\,(put\,(v, s)) \sqsubseteq v$$

- A lens is *total* if both *get* and *put* are.

Lenses

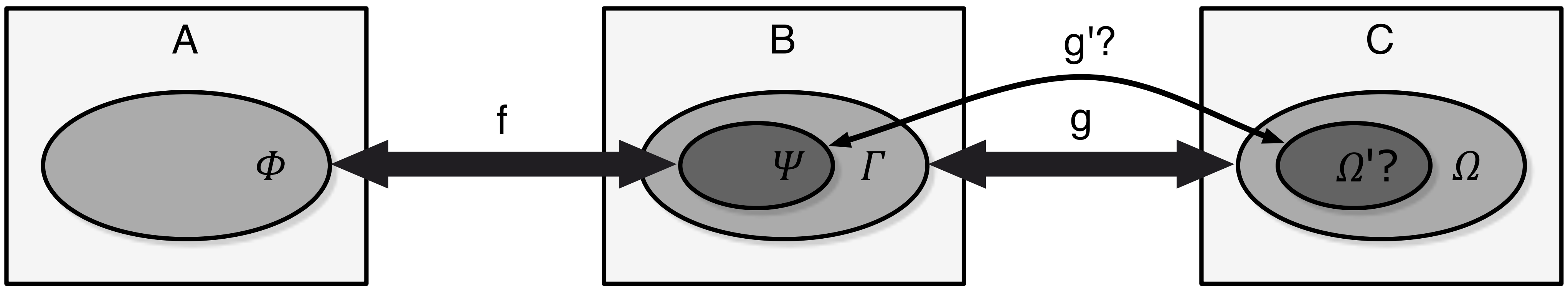
- The round-tripping and totality laws restrict the expressiveness of the language;
- For instance, duplication of data and conditionals are not allowed;
- Existing approaches are either less expressive, or relax the previous laws.

Lenses

- What if we refine the types to well-defined subsets for which the laws hold?
- We enhance the type system with invariants, in order to more precisely characterize the types of the transformations;
- All properties (including totality) are now defined with regard to the new types.

Example

- For instance, composition $g \cdot f$



- Since the range of *f* is smaller than the domain of g, the overall range of $g \cdot f$ must be restricted.

Example

- Duplication: $dup\, a = (a, a)$;
- If only one element of the pair is modified, round-tripping laws are broken;
- But if we restrict the codomain to pairs whose elements are equal, every value can be transformed back.

Relational Calculus

- Point-free relational calculus allows the easy calculation and reasoning about properties;
- Invariants and partial transformations are easily represented;
- Moreover, all transformations have well-defined converses, non-determinism being natural.

Relational Calculus

- Invariants are represented by coreflexives $(\Phi, \Psi, \ldots)$, relations smaller than the identity that act as filters;
- Coreflexives on pairs are denoted as [R], stating that the elements of the pair are related by *R*, i.e., $(a, b) \in [R] \equiv a\, R\, b$.
- The type of a transformation restricted by Φ and Ψ is denoted by $f : \Phi \rightarrow \Psi$.

nd-lenses

- A well-behaved *nd-lens* $l : \Phi \trianglerighteq \Psi$ comprises two relations $Get : \Phi \to \Psi$ and $Put : \Psi \times \Phi \to \Phi$ such that:

$$Get\,(Put\,(v,s)) \sqsubseteq \Phi\,v \qquad \Psi \times \Phi \subseteq \delta Put$$

$$Put\,(s, Get\,s) \sqsubseteq \Psi\,s \qquad \Phi \subseteq \delta Get$$

- Consists of the original properties restricted by the invariants.

Every transformation is a lens

- It is known that every *total* and *surjective* transformation gives rise to a well-behaved lens;
- So, if $Get : \Phi \rightarrow \Psi$ is total and surjective, we can always derive a nd-lens;
- The largest subset for which a relation is total and surjective, is *exactly* its range and domain.

Every transformation is a lens

- Every transformation *f* can be lifted to a well-behaved nd-lens $\lfloor f \rfloor : \delta f \to \rho f$, with

$$
\begin{aligned}
&Get = f \\
&Put\,(v, s) = \begin{cases} s & \text{if } f\ s = v \\ f^{\circ}\ v & \text{otherwise} \end{cases}
\end{aligned}
$$

- Definition arises directly from the round-tripping properties.

Calculating invariants

- The range and domain can be easily calculated using the PF calculus:

$$
\begin{array}{cc|cc}
\delta \mathsf{id} = \mathsf{id} & \delta \Phi = \Phi & \rho \mathsf{id} = \mathsf{id} & \rho \Phi = \Phi \\
\delta (f \vartriangle g) = \delta f \cap \delta g & \delta (f \triangledown g) = \delta f + \delta g & \rho (f \vartriangle g) = [g \circ f^\circ] & \rho (f \triangledown g) = \rho f \cup \rho g \\
\delta \pi_1 = \mathsf{id} & \delta \pi_2 = \mathsf{id} & \rho \pi_1 = \mathsf{id} & \rho \pi_2 = \mathsf{id} \\
\delta i_1 = \mathsf{id} & \delta i_2 = \mathsf{id} & \rho i_1 = \mathsf{id} + \bot & \rho i_2 = \bot + \mathsf{id} \\
\delta i_1^\circ = \mathsf{id} + \bot & \delta i_2^\circ = \bot + \mathsf{id} & \rho i_1^\circ = \mathsf{id} & \rho i_2^\circ = \mathsf{id} \\
\delta \underline{k} = \mathsf{id} & \delta ! = \mathsf{id} & \rho \underline{k} = \underline{k} \cap \mathsf{id} & \rho ! = \mathsf{id}_1 \\
\delta \Phi? = \mathsf{id} & \delta (R \circ S) = \delta (\delta R \circ S) & \rho \Phi? = \Phi + (\mathsf{id} \cap \overline{\Phi}) & \rho (R \circ S) = \rho (R \circ \rho S)
\end{array}
$$

Backward transformation

- In order to guarantee that the domains are as large as possible, we allow non-determinism;
- Generic definition is highly inefficient;
- We define equivalent definitions where the invariants are propagated down the expression;
- Non-determinism is reduced to a minimum.

Backward transformation

- Optimized *Put:*

$$
\begin{array}{ll}
\mathsf{Put}_{\mathsf{id}:\Phi\to\Psi} = \Phi\circ\pi_1 & \mathsf{Put}_{\Omega:\Phi\to\Psi} = \Phi\circ\pi_1 \\
\mathsf{Put}_{f\circ g:\Phi\to\Psi} = \mathsf{Put}_{g:\Phi\to\delta(\Psi\circ f)}\circ((\mathsf{Put}_{f:\rho(g\circ\Phi)\to\Psi}\circ(\mathsf{id}\times g))\vartriangle\pi_2) & \\
\mathsf{Put}_{\Omega?:\Phi\to\Psi} = (\Omega\mathbin{\triangledown}(\mathsf{id}\cap\overline{\Omega}))\circ\pi_1 & \\
\mathsf{Put}_{\pi_1:[R]\to\Psi} = (\pi_2\mathbin{\triangledown}(\mathsf{id}\vartriangle R)\circ\pi_1)\circ[\pi_1^\circ]? & \mathsf{Put}_{\pi_2:[R]\to\Psi} = (\pi_2\mathbin{\triangledown}(R^\circ\vartriangle\mathsf{id})\circ\pi_1)\circ[\pi_2^\circ]? \\
\mathsf{Put}_{f\triangledown g:\Phi+\Psi\to\Omega} = ((i_1\circ\mathsf{Put}_{f:\Phi\to\Omega}\cup i_2\circ\mathsf{Put}_{g:\Psi\to\Omega}\circ(\mathsf{id}\times\top)\circ[\overline{f}])\mathbin{\triangledown} & \\
\qquad (i_2\circ\mathsf{Put}_{g:\Psi\to\Omega}\cup i_1\circ\mathsf{Put}_{f:\Phi\to\Omega}\circ(\mathsf{id}\times\top)\circ[\overline{g}]))\circ\mathsf{distr} & \\
\mathsf{Put}_{i_1:\Phi\to\Psi} = (\Phi\mathbin{\triangledown}\bot)\circ\pi_1 & \mathsf{Put}_{i_2:\Phi\to\Psi} = (\bot\mathbin{\triangledown}\Phi)\circ\pi_1 \\
\mathsf{Put}_{i_1^\circ:\Phi\to\Psi} = \Phi\circ i_1\circ\pi_1 & \mathsf{Put}_{i_2^\circ:\Phi\to\Psi} = \Phi\circ i_2\circ\pi_1 \\
\mathsf{Put}_{\underline{k}:\Phi\to\Psi} = \pi_2 & \mathsf{Put}_{!:\Phi\to\Psi} = \pi_2
\end{array}
$$

Recursion Patterns

- Although we are able to calculate the domains of recursion patterns, they do not present a normal form;
- There are however rules that allow simplifications in some cases;
- In those cases, our system is able to simplify and evaluate the transformation.

Example

- The transformation $tail \triangle length$ calculates the tail *t* and length *n* of a list;
- Can be lifted to the nd-lens

$$\lfloor tail \triangle length \rfloor : id \trianglerighteq [succ \cdot length]$$

- Meaning that $length\, t = n + 1$;
- E.g. $Put(([2,3],3),[1,2,3]) = [1,2,3]$

$$Put(([2,0],3),[1,2,3]) = [0,2,0],[1,2,0],[2,2,0],\ldots$$

Conclusions

- We defined and implemented a bidirectional language over data-types with invariants, more expressive than previously existing;
- PF calculus allows the easy calculation of invariants and simplifications;
- The specialized backward transformation makes non-deterministic evaluation viable.