Inductive data types with negative occurrences

in HOL

Thirty Five years of Automath Workshop
Heriot-Watt University, Edinburgh, UK

by

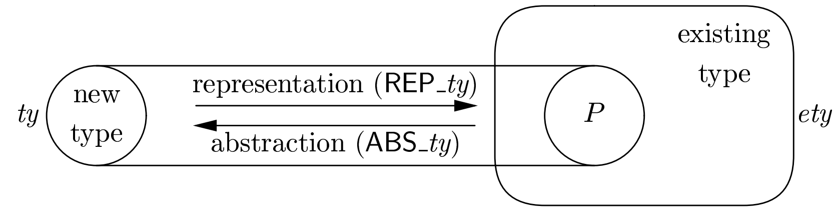
Tanja Vos (tanja@iti.upv.es)
Doaitse Swierstra (doaitse@cs.uu.nl)

Inductive Data Types (IDT)

$$ty = \mathsf{C}_1\ \tau_1^1\ \ldots\ \tau_1^{k_1} \mid \ldots \mid \mathsf{C}_m\ \tau_m^1\ \ldots\ \tau_m^{k_m}$$

- τ_i^j are admissible when:
  - non-recursive
  - ty itself.
  - $(\tau_1', \ldots, \tau_n')t'$ for t' existing IDT and admissible $\tau_1', \ldots, \tau_n'$
  - $\sigma \to \tau'$ for admissible τ' and non-recursive σ.
- $ty \to \mathsf{bool}$ not admissible, contradiction with Cantor's theorem (cardinality $\mathcal{P}(ty) >$ cardinality ty)
- $ty \to \mathsf{bool}$ is admissible when representing only **finite** sets of ty (cardinality $\mathcal{P}_{fin}(ty) =$ cardinality ty)
- `Val` = SET `Val` → bool
  | NUM `num` | LIST (`Val`)`list` | TREE (`Val`)`ltree`

The general approach



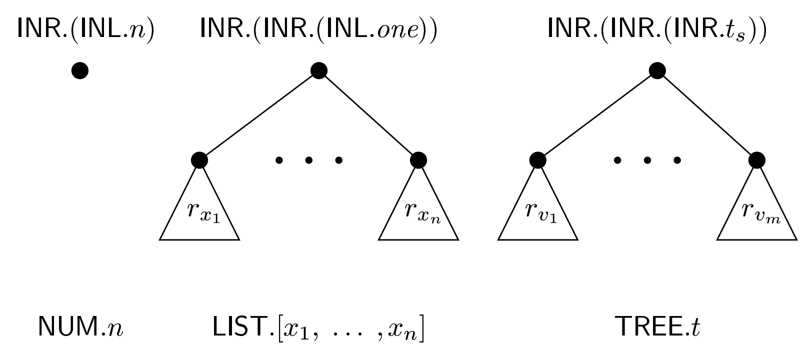
$\vdash (\forall x\ y :: (\mathsf{REP\_}ty.x = \mathsf{REP\_}ty.y) \Rightarrow x = y)$
$\wedge\ (\forall r ::\ P.r = (\exists x ::\ r = \mathsf{REP\_}ty.x))$

$\vdash (\forall a ::\ \mathsf{ABS\_}ty.(\mathsf{REP\_}ty.a) = a)$
$\wedge\ (\forall r ::\ P.r = (\mathsf{REP\_}ty.(\mathsf{ABS\_}ty.r) = r))$

Existing type labelled trees, ltree('a), 'a depends on number of and types of the constructors of ty

The representation

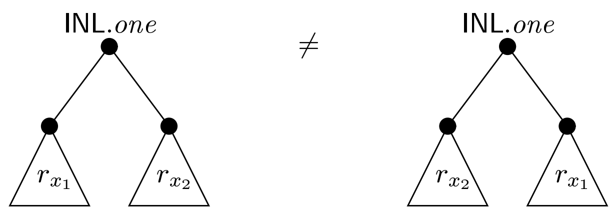
- ety is (`one` + `num` + `one` + `tree`)`ltree`
- representations of the constructors:



Problems with sets

$$\mathsf{SET}.\{x_1, x_2\} = \mathsf{SET}.\{x_2, x_1\}$$

but,



SOLUTION: take *ety* to be the equivalence classes of the type (`one + num + one + tree`)`ltree` in which the above trees are considered to be equivalent.

The equivalence relation

Def. $\mathsf{equiv}.(\mathsf{Node}.v_1.tl_1).(\mathsf{Node}.v_2.tl_2) =$

$(v_1 = v_2)$

$\wedge$

$(\quad tl_1 = tl_2 \wedge \exists n :: \ v_1 = \mathsf{INR}.(\mathsf{INL}.n)$

$\vee$

$\mathsf{map.equiv}.tl_1 = \mathsf{map.equiv}.tl_2$

$\wedge (v_1 = \mathsf{INR}.(\mathsf{INR}.(\mathsf{INL}.one)) \vee \exists t :: \ v_1 = \mathsf{INR}.(\mathsf{INR}.(\mathsf{INR}.t)))$

$\vee$

$\mathsf{image.equiv}.(\mathsf{l2s}.tl_1) = \mathsf{image.equiv}.(\mathsf{l2s}.tl_2) \wedge \mathsf{ISL}.v_1$

$)$

Thm. $\mathsf{equiv}.t_1.t_2 = (\mathsf{equiv}.t_1 = \mathsf{equiv}.t_2)$

The subset predicate

Def. which trees?

$Q.(\mathsf{Node}.v.tl) =$

$(\exists n :: (v = \mathsf{INR}.(\mathsf{INL}.n))) \Rightarrow tl = []$

$\wedge$

$(\exists t :: v = \mathsf{INR}.(\mathsf{INR}.(\mathsf{INR}.t))) \Rightarrow \mathsf{Is\_ltree}.(\mathsf{OUTR}.(\mathsf{OUTR}.(\mathsf{OUTR}.v)), tl)$

$\wedge$

$(\forall t :: t \in tl \Rightarrow Q.t)$

Def. so take their equivalence classes

$P \ = \ (\lambda s. \exists t :: \ (s \ = \ \mathsf{equiv}.t) \wedge (Q.t))$

Proceed as usual

- Extend syntax of logical types to include new type `Val`, and get the type bijections ABS_`Val` and REP_`Val` between `Val` and P.
- Define the constructors using the bijections:

$\mathsf{NUM}.n = \mathsf{ABS\_Val}.(\mathsf{equiv}.(\mathsf{Node}.(\mathsf{INR}.(\mathsf{INL}.n)) .[]))$

$\mathsf{SET}.s = \mathsf{ABS\_Val}.(\mathsf{equiv}.(\mathsf{Node}.(\mathsf{INL}.one) .(\mathsf{map}.(\mathsf{pick} \circ \mathsf{REP\_Val}).(\mathsf{s2l}.s))))$

(LIST and TREE are similar)

- Prove the initiality theorem (i.e. the existence of a paramorphism)

The paramorphism (or initiality theorem)

$$
\begin{aligned}
&\forall f_n\ f_s\ f_l\ f_t :: \\
&\quad \exists!\mathsf{para} :: \ (\forall n :: \ \mathsf{para}.(\mathsf{NUM}.n) = f_n.n) \\
&\qquad \wedge \ (\forall s :: \mathsf{finite}.s) \Rightarrow (\mathsf{para}.(\mathsf{SET}.s) = f_s.(\mathsf{image}.(\mathsf{split}.\mathsf{para}).s)) \\
&\qquad \wedge \ (\forall l :: \mathsf{para}.(\mathsf{LIST}.l) = f_l.(\mathsf{map}.(\mathsf{split}.\mathsf{para}).l)) \\
&\qquad \wedge \ (\forall t :: \mathsf{para}.(\mathsf{TREE}.t) = f_t.(\mathsf{map\_tree}.(\mathsf{split}.\mathsf{para}).t))
\end{aligned}
$$

Concluding remarks

- We have used a data type like `Val` to model the value space of programs that have different variables taking different types.
- Although the paper is about the theorem prover HOL, the results can easily be re-used within Isabelle and PVS.
- HOL-script (HOL90 version 7) available from: www.cs.uu.nl/~wishnu/research/hol_downloads/about.html