How much do System T recursors lift to dependent types?

Hugo Herbelin

IRIF, CNRS, Université de Paris, Inria, France
hugo.herbelin@inria.fr

Consider Peano numbers (on the left) and their canonical realisability predicate (on the right):

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathbb{N}} \qquad \frac{\Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{succ}\, t : \mathbb{N}} \qquad\qquad \frac{}{\Gamma \vdash \mathsf{iszero} : \mathsf{is}\mathbb{N}\, \mathsf{zero}} \qquad \frac{\Gamma \vdash p : \mathsf{is}\mathbb{N}\, t}{\Gamma \vdash \mathsf{issucc}\, t\, p : \mathsf{is}\mathbb{N}\, (\mathsf{succ}\, t)}$$

Consider the standard recursor in $\mathbb{N}$:

$$\frac{\begin{array}{ll}\Gamma, m : \mathbb{N} \vdash A : \mathsf{U} & \Gamma \vdash u : A[m := \mathsf{zero}] \\ \Gamma \vdash t : \mathbb{N} & \Gamma, n : \mathbb{N}, a : A[m := n] \vdash v : A[n := \mathsf{succ}\, n]\end{array}}{\Gamma \vdash (\mathsf{match}\ t\ \mathsf{with}\ [\mathsf{zero} \to u \,|\, \mathsf{succ}\, n \to_a v]) : A[m := t]}$$

and the standard recursor in $\mathsf{is}\mathbb{N}$:

$$\frac{\begin{array}{ll}\Gamma, n : \mathbb{N}, y : \mathsf{is}\mathbb{N}\, n \vdash A : \mathsf{U} & \Gamma \vdash u : A[n := \mathsf{zero}][y := \mathsf{iszero}] \\ \Gamma \vdash p : \mathsf{is}\mathbb{N}\, t & \Gamma, n : \mathbb{N}, x : \mathsf{is}\mathbb{N}\, n, a : A[n := n][y := x] \vdash v : A[n := \mathsf{succ}\, n][y := \mathsf{issucc}\, n\, x]\end{array}}{\Gamma \vdash (\mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to u \,|\, \mathsf{issucc}\, n\, x \to_a v]) : A[n := t][y := p]}$$

Consider the following (trivial) program:

$$m : \mathbb{N} \vdash (\mathsf{match}\ m\ \mathsf{with}\ [\mathsf{zero} \to m \,|\, \mathsf{succ}\, n \to_a m]) : \mathbb{N}$$

It happens that the canonically derived equivalent program in $\mathsf{is}\mathbb{N}$ is ill-typed:

$$m : \mathbb{N}, y : \mathsf{is}\mathbb{N}\, m \not\vdash (\mathsf{match}\ y\ \mathsf{with}\ [\mathsf{iszero} \to y \,|\, \mathsf{issucc}\, n\, x \to_a y]) : \mathsf{is}\mathbb{N}\, m$$

Indeed, in the case of $\mathbb{N}$, the branches do not have dependencies, so the term being matched, m, can be reused. But in $\mathsf{is}\mathbb{N}$, branches have dependent types, so the term being matched, y, cannot be reused.

We describe in the next sections two standard solutions to recove typing but each of them has drawbacks. Then, we argue that an alternative approach is to introduce a notion of as-patterns in recursors, leading to a generalisation of the typing rule for induction where not only the type but also the body can depend on the term being matched.

1 Solution 1: expanding the dependent instances

The first solution is to expand the term being matched, the way it would be evaluated in each branch:

$$m : \mathbb{N}, p : \mathsf{is}\mathbb{N}\, m \vdash (\mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to \mathsf{iszero} \,|\, \mathsf{issucc}\, n\, x \to_a \mathsf{issucc}\, n\, x]) : \mathsf{is}\mathbb{N}\, m$$

It has the expected semantics, that is, if $p \equiv \mathsf{issucc}\, t\, q$, for $\equiv$ denoting convertibility, then, treating the ill-typed term as untyped, we correctly have:

$$\mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to \mathsf{iszero} \,|\, \mathsf{issucc}\, n\, x \to_a \mathsf{issucc}\, n\, x] \equiv \mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to p \,|\, \mathsf{issucc}\, n\, x \to_a p]$$

But the behaviour is different in terms of sharing the representation of the underlying constructors in the reduction: if the 3 occurrences of p have their representation shared (as in call-by-value), then the sharing in memory is lost. In particular, in guard-based typing systems such as Coq, the information that y in the branches of our running example is of the same size as y being matched is lost.

2 Solution 2: generalising the term being matched

Another solution is to generalise the term being matched to change its type:

$$m : \mathbb{N}, y : \mathsf{is}\mathbb{N}\, m \vdash (\mathsf{match}\ y\ \mathsf{with}\ [\mathsf{iszero} \to \lambda y^{\mathsf{is}\mathbb{N}\,\mathsf{zero}}.y \mid \mathsf{issucc}\, n\, x \to_a \lambda y^{\mathsf{is}\mathbb{N}\,(\mathsf{succ}\, n)}.y])\, y : \mathsf{is}\mathbb{N}\, m$$

which has also the expected semantics, that is, if $p \equiv \mathsf{issucc}\, t\, q$, then, treating the ill-typed term as untyped:

$$\mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to \lambda y.y \mid \mathsf{issucc}\, n\, x \to_{\lambda y.y}\ \mathsf{issucc}\, n\, x]\, p \equiv \mathsf{match}\ p\ \mathsf{with}\ [\mathsf{iszero} \to p \mid \mathsf{issucc}\, n\, x \to_a p]$$

This time, if all occurrences of p are shared in memory (as in call-by-value), they remain shared before and after reduction in the encoding. Also, for a guard-based typing system, if sizes are propagated along generalisations obtained by β-expanding across a case analys

However, the generalisation technically requires a stronger system than intended. For instance, if the program is primitive recursive, its encoding requires to go out of primitive recursion.

3 Solution 3: adding as-variables to recursors

In pattern-matching compilation (e.g. OCaml, Haskell, Coq, ...), it is common to use as-variables. In Coq, these as-variables were thought for long as syntactic sugar and emulated with one or the other encoding above. We argue that giving a foundational status to as-variables solves our problem. We propose the following typing rules which differ from ordinary induction only in that not only the type but also the body of branches can depend on the term being matched:

$$\frac{\begin{array}{ll} \Gamma, m : \mathbb{N} \vdash A : \mathsf{U} & \Gamma, m : \mathbb{N} \vdash u : A[m := \mathsf{zero}] \\ \Gamma \vdash t : \mathbb{N} & \Gamma, n : \mathbb{N}, a : A[m := n], m : \mathbb{N} \vdash v : A[m := \mathsf{succ}\, n] \end{array}}{\Gamma \vdash (\mathsf{match}\ t\ \mathsf{as}\ m\ \mathsf{with} \begin{bmatrix} \mathsf{zero} \to u \\ \mathsf{succ}\, n \to_a v \end{bmatrix}) : A[m := t]}$$

$$\frac{\begin{array}{ll} \Gamma, n : \mathbb{N}, y : \mathsf{is}\mathbb{N}\, n \vdash A : \mathsf{U} & \Gamma, y : \mathsf{is}\mathbb{N}\,\mathsf{zero} \vdash u : A[n := \mathsf{zero}][y := \mathsf{iszero}] \\ \Gamma \vdash p : \mathsf{is}\mathbb{N}\, t & \Gamma, n : \mathbb{N}, x : \mathsf{is}\mathbb{N}\, n, a : A[y := x], y : \mathsf{is}\mathbb{N}\,(\mathsf{succ}\, n) \vdash v : A[n := \mathsf{succ}\, n][y := \mathsf{issucc}\, n\, x] \end{array}}{\Gamma \vdash (\mathsf{match}\ t\ \mathsf{as}\ y\ \mathsf{with} \begin{bmatrix} \mathsf{iszero} \to u \\ \mathsf{issucc}\, n\, x \to_a v \end{bmatrix}) : A[n := t][y := p]}$$

with reduction rules:

$$\begin{array}{lll} \mathsf{match}\ \mathsf{zero}\ \mathsf{as}\ m\ \mathsf{with} \begin{bmatrix} \mathsf{zero} \to u \\ \mathsf{succ}\, n \to_a v \end{bmatrix} & \to & u[m := \mathsf{zero}] \\ \mathsf{match}\ (\mathsf{succ}\, t)\ \mathsf{as}\ m\ \mathsf{with} \begin{bmatrix} \mathsf{zero} \to u \\ \mathsf{succ}\, n \to_a v \end{bmatrix} & \to & v[n := t][m := \mathsf{succ}\, t][a := \mathsf{match}\ t\ \mathsf{as}\ m\ \mathsf{with} \begin{bmatrix} \mathsf{zero} \to u \\ \mathsf{succ}\, n \to_a v \end{bmatrix}] \end{array}$$

$$\begin{array}{lll} \mathsf{match}\ \mathsf{iszero}\ \mathsf{as}\ y\ \mathsf{with} \begin{bmatrix} \mathsf{iszero} \to u \\ \mathsf{issucc}\, n \to_a v \end{bmatrix} & \to & u[y := \mathsf{iszero}][] \\ \mathsf{match}\ (\mathsf{issucc}\, t\, p)\ \mathsf{as}\ y\ \mathsf{with} \begin{bmatrix} \mathsf{iszero} \to u \\ \mathsf{issucc}\, n\, x \to_a v \end{bmatrix} & \to & v[n := t][x := p][y := \mathsf{issucc}\, t\, p][a := \mathsf{match}\ p\ \mathsf{as}\ y\ \mathsf{with} \begin{bmatrix} \mathsf{iszero} \to u \\ \mathsf{issucc}\, n\, x \to_a v \end{bmatrix}] \end{array}$$

As an application, we are able using these constructions to canonically derive the realisability interpretation of terms in a way which is compatible with a guard-based typing system, as it is in Coq. We rewrite the original term into:

$$m : \mathbb{N} \vdash (\mathsf{match}\ m\ \mathsf{as}\ m'\ \mathsf{with} \begin{bmatrix} \mathsf{zero} \to m' \\ \mathsf{succ}\, n \to_a m' \end{bmatrix}) : \mathbb{N}$$

and canonically derive from it the following now well-typed term:

$$m : \mathbb{N}, y : \mathsf{is}\mathbb{N}\, m \vdash (\mathsf{match}\ y\ \mathsf{as}\ y'\ \mathsf{with} \begin{bmatrix} \mathsf{iszero} \to y' \\ \mathsf{issucc}\, n\, x \to_a y' \end{bmatrix}) : \mathsf{is}\mathbb{N}\, m$$

Assuming a guard condition that recognises an as-pattern as being of the same size as the term being matched, the guardedness of the derived term derives from the guardedness of the original term.