Efficient Equality Test for Recursive Second Order Types

Nadji Gauthier - François Pottier

Roadmap

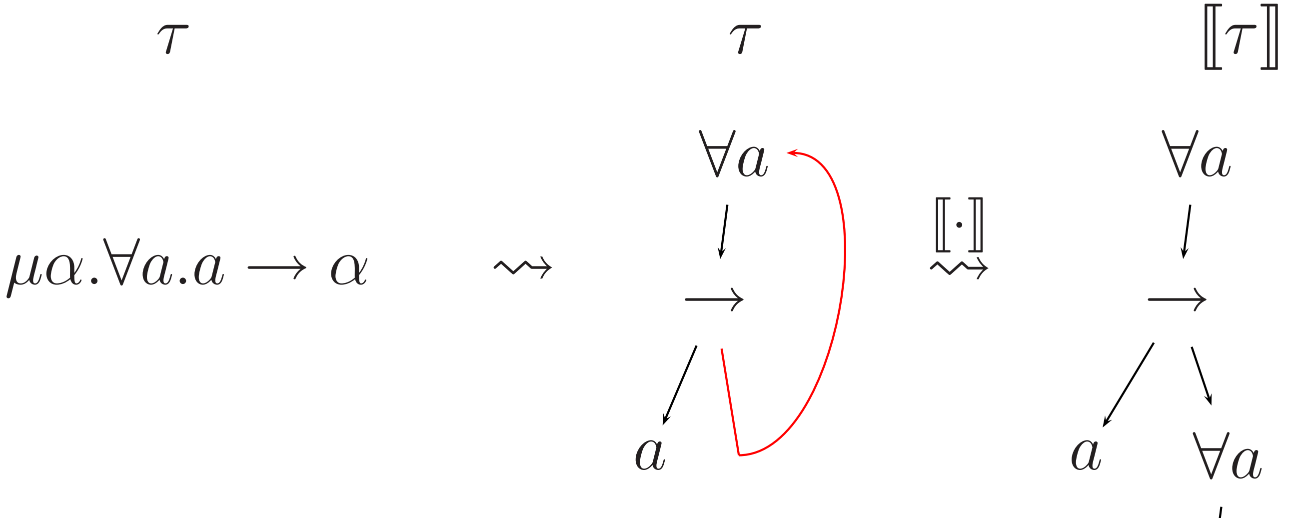
- Second order types
- Notion of equality
- Related work
- Highlight of the difficulties
- Reduction from second order to first order
- Conclusion

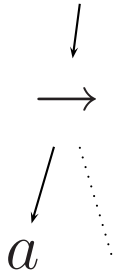
F_μ syntax

$$\begin{array}{rcl} \tau & := & \alpha \\ & \mid & a \\ & \mid & \tau \longrightarrow \tau \\ & \mid & \forall a.\tau \\ & \mid & \mu\alpha.\tau \end{array}$$

- quantification restricted on atoms
- atoms are integers

F_μ type example





- $\tau =_{2o} \tau'$ equal by definition to $[\![\tau]\!] = [\![\tau']\!]$
- Problem : given τ, τ', is $[\![\tau]\!] = [\![\tau']\!]$?

Related work (Glew[02])

- algorithm in $O(n^2)$, we propose $O(n \log n)$
- not directly applicable to unification, entailment ...
- more general problems can be decided but algorithms are more complex

Deciding equality between second order types

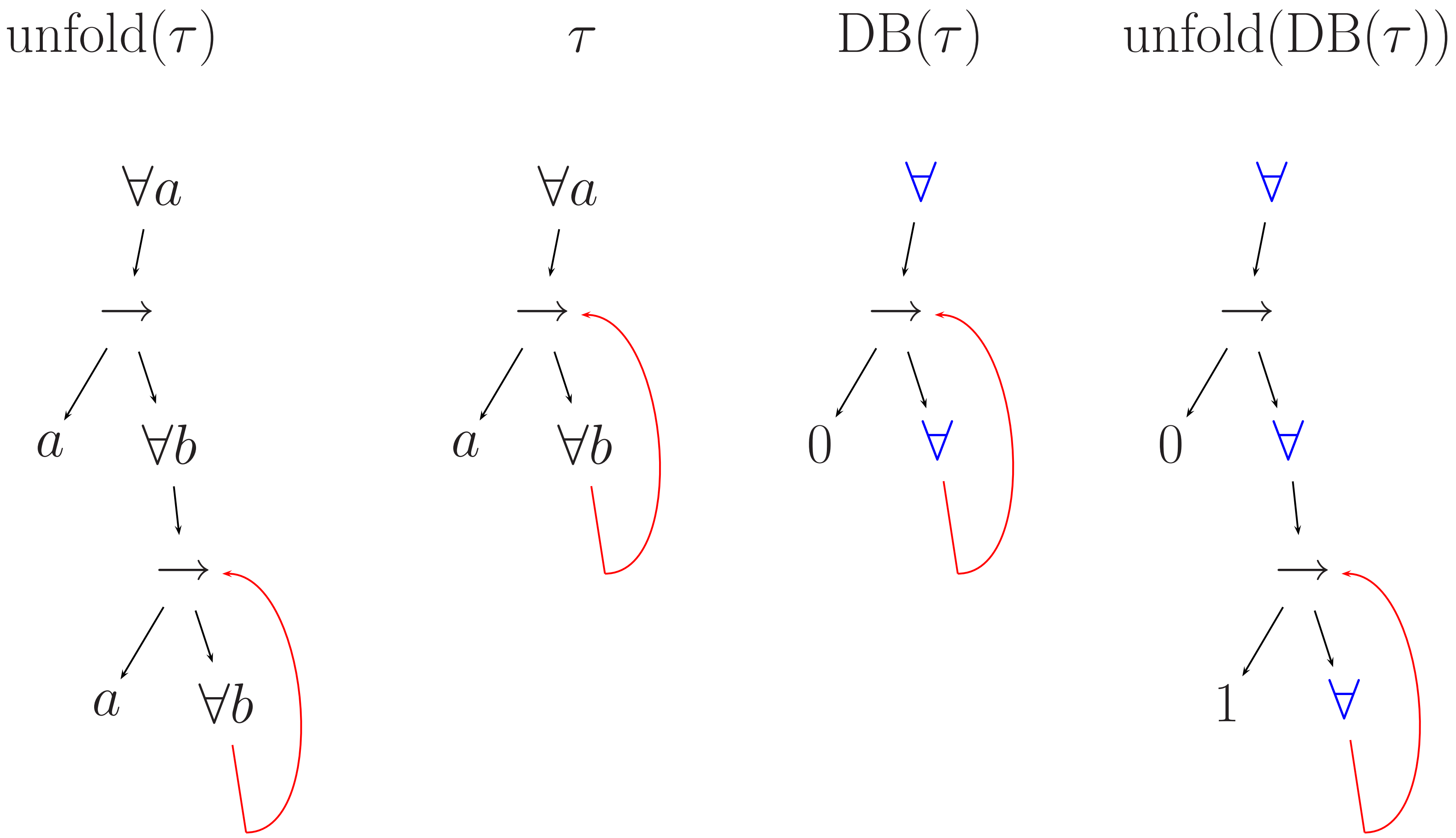
$$\forall a.\tau \overset{?}{=} \forall b.\tau' \Leftrightarrow \tau \overset{?}{=} (\tau'[b/a]), a \notin fa(\tau')$$

- avoid renamings
- in System F, it is possible with De Bruijn encoding :

$$\forall.\tau \overset{?}{=} \forall.\tau' \Leftrightarrow \tau \overset{?}{=} \tau'$$

- reused existing algorithms for deciding first order equality
- $\tau =_{2o} \tau' \Leftrightarrow enc(\tau) =_{1o} enc(\tau')$

Cycles and the De Bruijn encoding



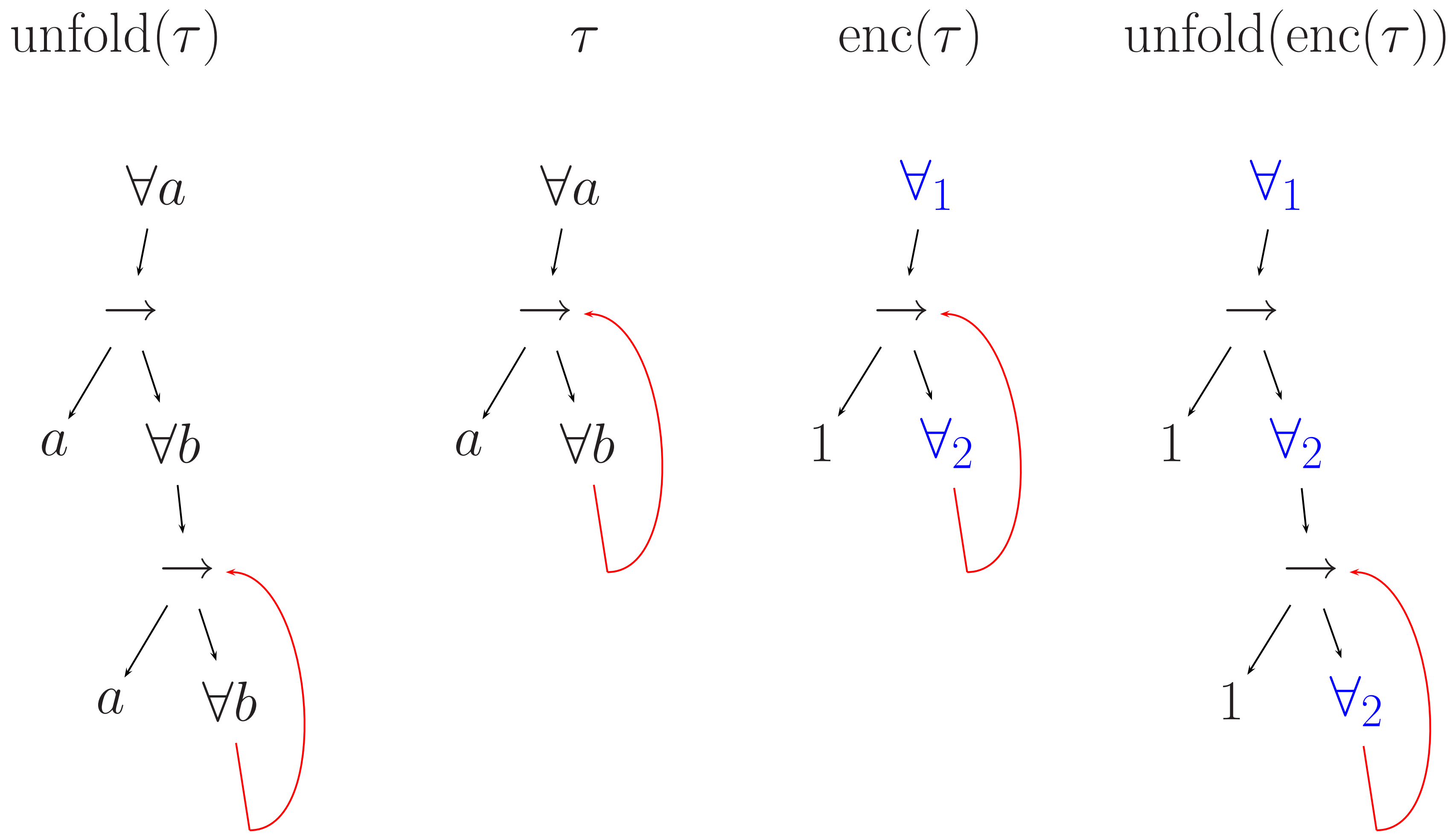
First order target syntax

$$
\begin{array}{rcl}
\sigma & := & \alpha \\
& | & a \\
& | & \sigma \rightarrow \sigma \\
& | & \mu\alpha.\sigma \\
& | & \forall_1.\sigma | \forall_2.\sigma | \ldots
\end{array}
$$

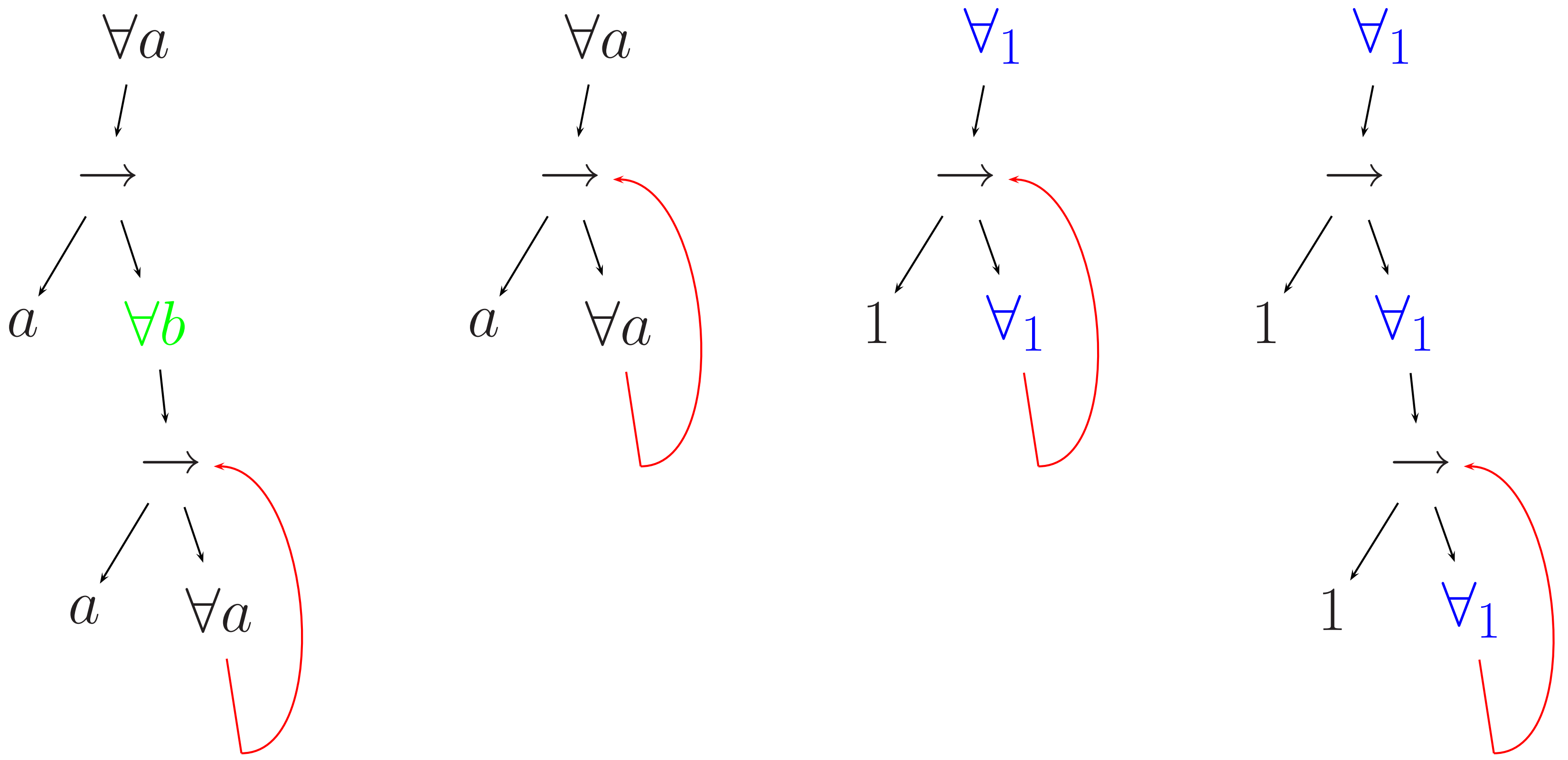
$$
\begin{array}{lcl}
Q & : & \sigma \rightarrow \tau \\
Q(\alpha) & = & \alpha \\
Q(a) & = & a \\
Q(\sigma_1 \rightarrow \sigma_2) & = & Q(\sigma_1) \rightarrow Q(\sigma_2) \\
Q(\mu\alpha.\sigma) & = & \mu\alpha.Q(\sigma) \\
Q(\forall_a.\sigma) & = & \forall a.Q(\sigma)
\end{array}
$$

- $\forall_1.1 \not=_{1o} \forall_2.2$
- $Q(\forall_1.1) = \forall a.a = Q(\forall_2.2)$

Naming

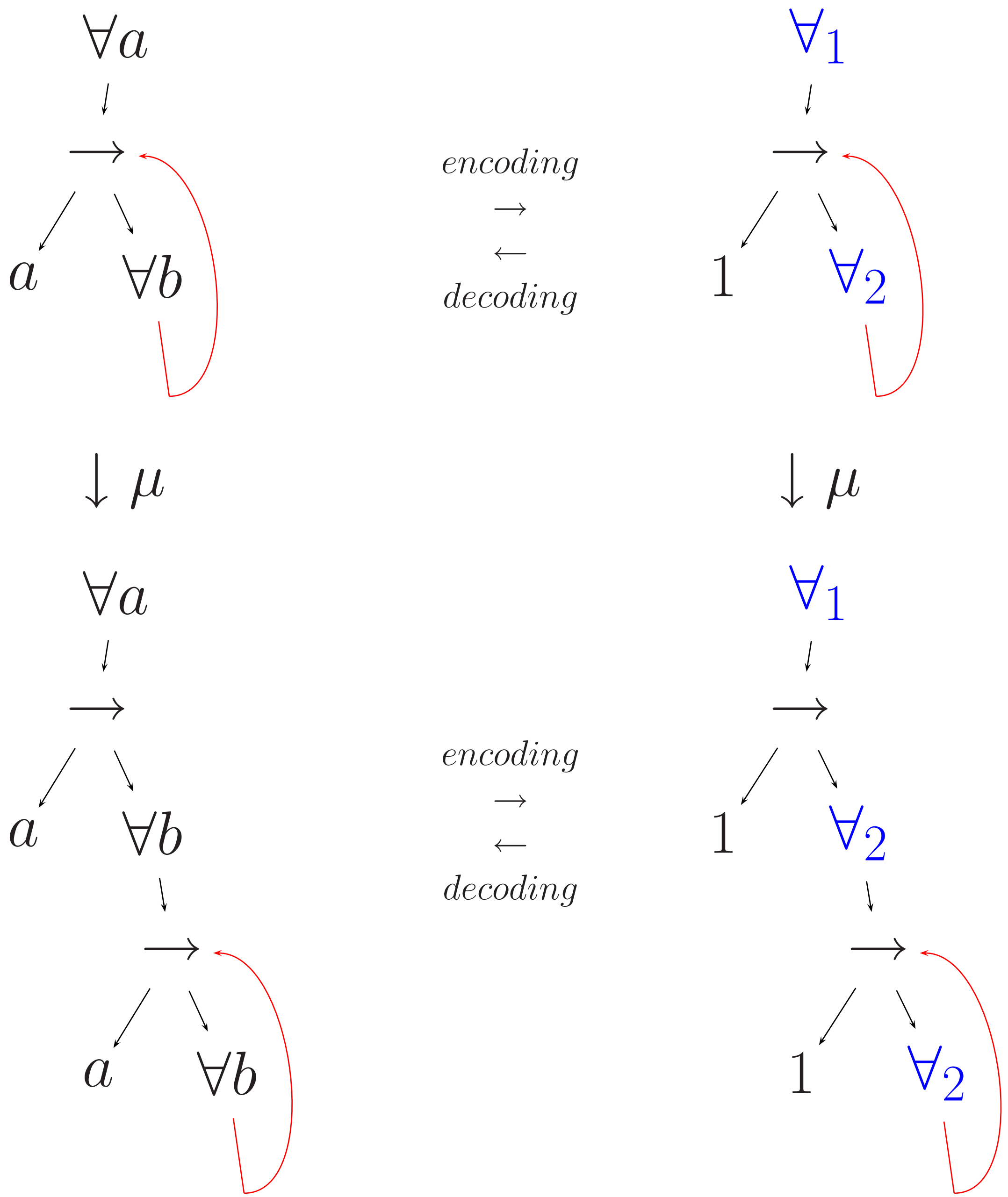


Naming



- $\forall a.\mu\alpha.a \to \forall a.\alpha =_{2o} \forall a.\,((a \to \forall a.\alpha)[\mu\alpha.a \to \forall a.\alpha/\alpha])$

Commuting property



Conclusion

- encoding an F_μ type can be done in $O(n \log n)$
- second order equality testing can be reduced to the first order
- similar reductions exists for the unification and entailment problem
- compatible with rows
- `http://pauillac.inria.fr/~gauthier/`