A Classical Realizability Model for a Semantical Value Restriction

UNIVERSITÉ SAVOIE MONT BLANC

Rodolphe Lepigre <rodolphe.lepigre@univ-smb.fr>

ESOP - Eindhoven, 07/04/2016

```
type rec nat = [Z | S of nat]

val rec add : nat => nat => nat =
  fun n m ->
    match n with
    | Z     -> m
    | S[n'] -> S[add n' m]
```

```
type rec nat = [Z | S of nat]

val rec add : nat => nat => nat =
  fun n m ->
    match n with
    | Z     -> m
    | S[n'] -> S[add n' m]



val addZN : (n : nat) => (add Z n == n) =
  ...
```

```
type rec nat = [Z | S of nat]

val rec add : nat => nat => nat =
  fun n m ->
    match n with
    | Z     -> m
    | S[n'] -> S[add n' m]



val addZN : (n : nat) => (add Z n == n) =
  fun n -> {}
```

```
type rec nat = [Z | S of nat]

val rec add : nat => nat => nat =
  fun n m ->
    match n with
    | Z     -> m
    | S[n'] -> S[add n' m]



val addZN : (n : nat) => (add Z n == n) =
  fun n -> {}

val rec addNZ : (n : nat) => (add n Z == n) =
  fun n ->
    match n with
    | Z     -> {}
    | S[n'] -> addNZ n'
```

An homogeneous language

Main features:

- call-by-value language with effects,
- extended type system for specification,
- proof as program (a single language).

Proofs can be composed as (and with) programs.

Dependent product type and value restriction

We would like to be able to write things like:

```
val p : (add (add Z Z) Z == add Z Z) = addNZ (add Z Z)
```

Dependent product type and value restriction

We would like to be able to write things like:

```
val p : (add (add Z Z) Z == add Z Z) = addNZ (add Z Z)
```

But the following typing rule is unsound without value restriction on u:

$$\frac{\Gamma \vdash t : \Pi_{a:A} B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B[a := u]}$$

Dependent product type and value restriction

We would like to be able to write things like:

```
val p : (add (add Z Z) Z == add Z Z) = addNZ (add Z Z)
```

But the following typing rule is unsound without value restriction on u:

$$\frac{\Gamma \vdash t : \Pi_{a:A} B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B[a := u]}$$

This breaks the compositionality of proofs and programs.

Encoding value restriction

Use two forms of judgements: $\Gamma \vdash t : A$ and $\Gamma \vdash_{\mathrm{val}} v : A$.

Encoding value restriction

Use two forms of judgements: $\Gamma \vdash t : A$ and $\Gamma \vDash_{\text{val}} v : A$.

$$\frac{}{\Gamma, x : A \vDash_{\text{val}} x : A}\,\text{Ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vDash_{\text{val}} \lambda x\, t : A \Rightarrow B}\,\rightarrow_{\text{i}}$$

$$\frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\,\rightarrow_{\text{e}}$$

Encoding value restriction

Use two forms of judgements: $\Gamma \vdash t : A$ and $\Gamma \vDash_{\text{val}} v : A$.

$$\frac{}{\Gamma, x : A \vDash_{\text{val}} x : A}\text{Ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vDash_{\text{val}} \lambda x\, t : A \Rightarrow B}\rightarrow_i$$

$$\frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\rightarrow_e$$

One more rule is required.

$$\frac{\Gamma \vDash_{\text{val}} v : A}{\Gamma \vdash v : A}\uparrow$$

Restricted type constructors

$$\frac{\Gamma \vDash_{\text{val}} v : A \qquad X \notin FV(\Gamma)}{\Gamma \vDash_{\text{val}} v : \forall X\, A}\,\forall_i \qquad \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A[X := B]}\,\forall_e$$

$$\frac{\Gamma, x : A \vdash t : B[a := x]}{\Gamma \vDash_{\text{val}} \lambda x\, t : \Pi_{a:A} B}\,\Pi_i \qquad \frac{\Gamma \vdash t : \Pi_{a:A} B \qquad \Gamma \vDash_{\text{val}} v : A}{\Gamma \vdash t\, v : B[a := v]}\,\Pi_e$$

Equivalence and semantical value restriction

With value restriction, some rules are restricted to values.

Idea: a term that is equivalent to a value may be considered a value.

With value restriction, some rules are restricted to values.

Idea: a term that is equivalent to a value may be considered a value.

Informal proof:

$$\dfrac{\dfrac{\dfrac{\Gamma, t \equiv v \vdash t : A}{\Gamma, t \equiv v \vdash v : A} \quad X \notin FV(\Gamma)}{\Gamma, t \equiv v \vdash v : \forall X\ A}}{\Gamma, t \equiv v \vdash t : \forall X\ A}$$

Deriving the relaxed rules

$$
\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, t \equiv v \vdash t : A}{\Gamma, t \equiv v \vdash v : A}\,\equiv}{\Gamma, t \equiv v \vdash_{\mathrm{val}} v : A}\,\downarrow \qquad X \notin FV(\Gamma)}{\dfrac{\dfrac{\Gamma, t \equiv v \vdash_{\mathrm{val}} v : \forall X\, A}{\Gamma, t \equiv v \vdash v : \forall X\, A}\,\uparrow}{\Gamma, t \equiv v \vdash t : \forall X\, A}\,\equiv}\,\forall_e
$$

Krivine machine

$$
\begin{aligned}
v, w &::= x \mid \lambda x\, t \mid C[v] \mid \{l_i = v_i\}_{i \in I} \\
t, u &::= a \mid v \mid t\, u \mid \mu\alpha\, t \mid [\pi]t \mid v.l \mid \text{case } v \text{ of } [C_i[x] \to t_i]_{i \in I} \\
\pi &::= \alpha \mid v \cdot \pi \mid [t]\pi
\end{aligned}
$$

Krivine machine

$$
\begin{aligned}
v, w &::= x \mid \lambda x\, t \mid C[v] \mid \{l_i = v_i\}_{i \in I} \\
t, u &::= a \mid v \mid t\, u \mid \mu\alpha\, t \mid [\pi]t \mid v.l \mid \text{case } v \text{ of } [C_i[x] \rightarrow t_i]_{i \in I} \\
\pi &::= \alpha \mid v \cdot \pi \mid [t]\pi
\end{aligned}
$$

$$
\begin{aligned}
t\, u * \pi &\succ u * [t]\pi \\
v * [t]\pi &\succ t * v \cdot \pi \\
(\lambda x\, t) * v \cdot \pi &\succ t[x \leftarrow v] * \pi \\
\mu\alpha\, t * \pi &\succ t[\alpha \leftarrow \pi] * \pi \\
[\pi]t * \rho &\succ t * \pi \\
\text{case } C_k[v] \text{ of } [C_i[x] \rightarrow t_i]_{i \in I} * \pi &\succ t_k[x \leftarrow v] * \pi \\
\{l_i = v_i\}_{i \in I}.l_k * \pi &\succ v_k * \pi
\end{aligned}
$$

Three levels of interpretation:

- raw semantics $[\![A]\!]$,

$$
\begin{aligned}
[\![\{l_i : A_i\}_{i \in I}]\!] &:= \left\{\{l_i = v_i\}_{i \in I} \mid \forall i \in I,\, v_i \in [\![A_i]\!]\right\} \\
[\![\forall X\, A]\!] &:= \cap_{\Phi \subseteq \mathscr{V}/\equiv} [\![A[X := \Phi]]\!] \\
[\![t \equiv u]\!] &:= [\![\{\}]\!] \quad \text{when} \quad t \equiv u \quad \text{and} \quad \emptyset \quad \text{otherwise}
\end{aligned}
$$

Three levels of interpretation:

- raw semantics $[\![A]\!]$,
- falsity value $[\![A]\!]^{\perp} = \{\pi \mid \forall v \in [\![A]\!], v * \pi \in \perp\!\!\!\perp\}$,

$$\begin{aligned}
\left[\!\!\left[\{l_i : A_i\}_{i \in I}\right]\!\!\right] &:= \left\{\{l_i = v_i\}_{i \in I} \mid \forall i \in I, v_i \in [\![A_i]\!]\right\} \\
[\![\forall X\ A]\!] &:= \cap_{\Phi \subseteq \mathscr{V}/\equiv} [\![A[X := \Phi]]\!] \\
[\![t \equiv u]\!] &:= [\![\{\}]\!] \quad \text{when} \quad t \equiv u \quad \text{and} \quad \emptyset \quad \text{otherwise}
\end{aligned}$$

Three levels of interpretation:

- raw semantics $[\![A]\!]$,
- falsity value $[\![A]\!]^{\perp} = \{\pi \mid \forall v \in [\![A]\!], v * \pi \in \perp\!\!\!\perp\}$,
- truth value $[\![A]\!]^{\perp\perp} = \{t \mid \forall \pi \in [\![A]\!]^{\perp}, t * \pi \in \perp\!\!\!\perp\}$.

$$
\begin{aligned}
\left[\!\!\left[\{l_i : A_i\}_{i \in I}\right]\!\!\right] &:= \left\{\{l_i = v_i\}_{i \in I} \mid \forall i \in I,\, v_i \in [\![A_i]\!]\right\} \\
[\![\forall X\, A]\!] &:= \cap_{\Phi \subseteq \mathscr{V}/\equiv} [\![A[X := \Phi]]\!] \\
[\![t \equiv u]\!] &:= [\![\{\}]\!] \text{ when } t \equiv u \text{ and } \emptyset \text{ otherwise}
\end{aligned}
$$

Three levels of interpretation:

- raw semantics $[\![A]\!]$,
- falsity value $[\![A]\!]^{\perp} = \{\pi \mid \forall v \in [\![A]\!], v * \pi \in \perp\!\!\!\perp\}$,
- truth value $[\![A]\!]^{\perp\perp} = \{t \mid \forall \pi \in [\![A]\!]^{\perp}, t * \pi \in \perp\!\!\!\perp\}$.

$$\begin{aligned}
[\![A \Rightarrow B]\!] &:= \{\lambda x\, t \mid \forall v \in [\![A]\!]\;\; t[x := v] \in [\![B]\!]^{\perp\perp}\} \\
\left[\!\!\left[\{l_i : A_i\}_{i \in I}\right]\!\!\right] &:= \left\{\{l_i = v_i\}_{i \in I} \mid \forall i \in I,\, v_i \in [\![A_i]\!]\right\} \\
[\![\forall X\, A]\!] &:= \cap_{\Phi \subseteq \mathscr{V}/\equiv} [\![A[X := \Phi]]\!] \\
[\![t \equiv u]\!] &:= [\![\{\}]\!] \;\text{ when }\; t \equiv u \;\text{ and }\; \emptyset \;\text{ otherwise}
\end{aligned}$$

Theorem *(Adequacy Lemma)*:

- if t is a term such that $\vdash t : A$ then $t \in [\![A]\!]^{\perp\perp}$,
- if v is a value such that $\vdash_{\mathrm{val}} v : A$ then $v \in [\![A]\!]$.

Intuition: a typed program behaves well (in any well-typed evaluation context).

Semantical value restriction

In every realizability model $[\![A]\!] \subseteq [\![A]\!]^{\perp\!\perp}$.

Semantical value restriction

In every realizability model $[\![A]\!] \subseteq [\![A]\!]^{\perp\perp}$.

This provides a semantical justification to the rule $\dfrac{\Gamma \vdash_{\text{val}} v : A}{\Gamma \vdash v : A}\uparrow$.

In every realizability model $[\![A]\!] \subseteq [\![A]\!]^{\perp\perp}$.

This provides a semantical justification to the rule $\dfrac{\Gamma \vdash_{\mathrm{val}} v : A}{\Gamma \vdash v : A}\uparrow$.

We need to have $[\![A]\!]^{\perp\perp} \cap \mathscr{V} \subseteq [\![A]\!]$ to obtain the rule $\dfrac{\Gamma \vdash v : A}{\Gamma \vdash_{\mathrm{val}} v : A}\downarrow$.

With this rule we can derive relaxed typing rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, t \equiv v \vdash t : A}{\Gamma, t \equiv v \vdash v : A}\equiv}{\Gamma, t \equiv v \vdash_{\mathrm{val}} v : A}\downarrow \qquad X \notin FV(\Gamma)}{\Gamma, t \equiv v \vdash_{\mathrm{val}} v : \forall X\, A}\forall_e}{\Gamma, t \equiv v \vdash v : \forall X\, A}\uparrow}{\Gamma, t \equiv v \vdash t : \forall X\, A}\equiv$$

The new instruction trick

The property $[\![A]\!]^{\perp\perp} \cap \mathscr{V} \subseteq [\![A]\!]$ is not true in every realizability model.

The new instruction trick

The property $[\![A]\!]^{\perp\perp} \cap \mathscr{V} \subseteq [\![A]\!]$ is not true in every realizability model.

To obtain it we extend the system with a new term constructor $\delta_{v,w}$.

The new instruction trick

The property $[\![A]\!]^{\perp\perp} \cap \mathscr{V} \subseteq [\![A]\!]$ is not true in every realizability model.

To obtain it we extend the system with a new term constructor $\delta_{v,w}$.

We will have $\delta_{v,w} * \pi \succ v * \pi$ if and only if $v \not\equiv w$.

The new instruction trick

The property $[\![A]\!]^{\perp\perp} \cap \mathscr{V} \subseteq [\![A]\!]$ is not true in every realizability model.

To obtain it we extend the system with a new term constructor $\delta_{v,w}$.

We will have $\delta_{v,w} * \pi \succ v * \pi$ if and only if $v \not\equiv w$.

Idea of the proof:

- suppose $v \notin [\![A]\!]$ and show $v \notin [\![A]\!]^{\perp\perp}$,
- we need to find π such that $v * \pi \notin \perp\!\!\!\perp$ and $\forall w \in [\![A]\!], w * \pi \in \perp\!\!\!\perp$,
- we can take $\pi = [\lambda x\, \delta_{x,v}]\varepsilon$,
- $v * [\lambda x\, \delta_{x,v}]\varepsilon \succ \lambda x\, \delta_{x,v} * v.\varepsilon \succ \delta_{v,v} * \varepsilon$,
- $w * [\lambda x\, \delta_{x,v}]\varepsilon \succ \lambda x\, \delta_{x,v} * w.\varepsilon \succ \delta_{w,v} * \varepsilon \succ w * \varepsilon$.

Stratified reduction and equivalence

Problem: the definitions of $(\succ)$ and $(\equiv)$ are circular.

Problem: the definitions of $(\succ)$ and $(\equiv)$ are circular.

We need to rely on a stratified construction of the two relations

$$(\twoheadrightarrow_i) = (\succ) \cup \left\{(\delta_{v,w} * \pi,\ v * \pi) \mid \exists j < i,\ v \not\equiv_j w\right\}$$

$$(\equiv_i) = \left\{(t,\ u) \mid \forall j \leqslant i, \forall \pi, \forall \sigma,\ t\sigma * \pi \Downarrow_j \Leftrightarrow u\sigma * \pi \Downarrow_j\right\}$$

We then take

$$(\twoheadrightarrow) = \bigcup_{i \in \mathbb{N}} (\twoheadrightarrow_i) \qquad (\equiv) = \bigcap_{i \in \mathbb{N}} (\equiv_i)$$

Current and future work

Subtyping without coercions:

- useful for programming (modules, classes...),
- provide injections between types for free,
- interprets $\vdash A \subseteq B$ as $[\![A]\!] \subseteq [\![B]\!]$ in the semantics.

Implementation (in progress):

- the types $\mu X\, A$ and $\nu X\, A$ will be handled by subtyping,
- we need to extend the language with a fixpoint,
- termination needs to be ensured to preserve soundness.

Theoretical investigation (for later):

- can we use $\delta_{v,w}$ to realize new formulas,
- how do we encode real maths in the system?