Sikkel: Multimode Simple Type Theory as an Agda Library

Joris Ceulemans Andreas Nuyts Dominique Devriese

imec-DistriNet, KU Leuven, Belgium

Ninth Workshop on Mathematically Structured Functional Programming
Munich, Germany
2 April 2022

General Aim

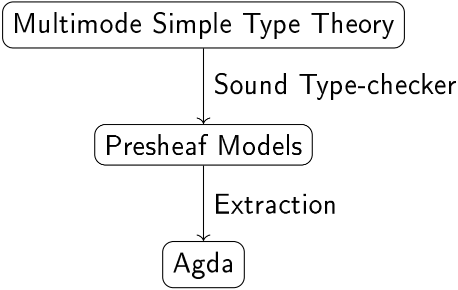
Dependently typed languages are based on type theories like MLTT or CIC.

Extensions of standard type theory with new primitives:

- guarded recursion,
- parametricity,
- univalence,
- directed type theory,
- nominal reasoning,
- ...

How to work in such extended theories within Agda, Coq, ...?

Overview of Sikkel



Overview of Sikkel

p

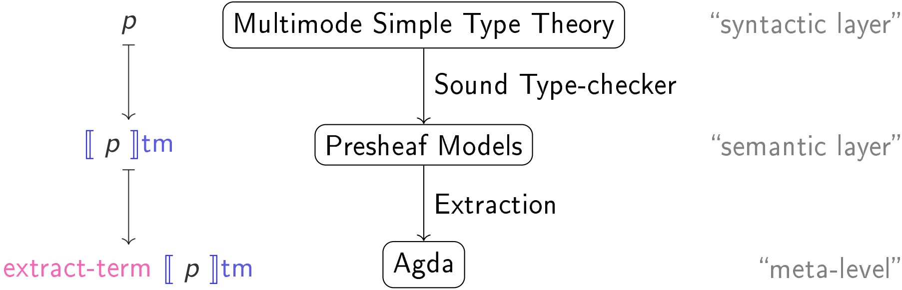
$$\begin{tikzcd} \boxed{\text{Multimode Simple Type Theory}} \arrow[d, "\text{Sound Type-checker}"] \\ \boxed{\text{Presheaf Models}} \arrow[d, "\text{Extraction}"] \\ \boxed{\text{Agda}} \end{tikzcd}$$

“syntactic layer”

Overview of Sikkel

$$\begin{tikzcd}
p \arrow[d, mapsto] & \boxed{\text{Multimode Simple Type Theory}} \arrow[d, "\text{Sound Type-checker}"] & \text{“syntactic layer”} \\
[\![\, p \,]\!]\text{tm} & \boxed{\text{Presheaf Models}} \arrow[d, "\text{Extraction}"] & \text{“semantic layer”} \\
 & \boxed{\text{Agda}} &
\end{tikzcd}$$

Overview of Sikkel



Overview of Sikkel

$$
\begin{tikzcd}
p \arrow[d, mapsto] & \boxed{\text{Multimode Simple Type Theory}} \arrow[d, "\text{Sound Type-checker}"] & \text{“syntactic layer”} \\
[\![\, p \,]\!]\text{tm} \arrow[d, mapsto] & \boxed{\text{Presheaf Models}} \arrow[d, "\text{Extraction}"] & \text{“semantic layer”} \\
\text{extract-term } [\![\, p \,]\!]\text{tm} & \boxed{\text{Agda}} & \text{“meta-level”}
\end{tikzcd}
$$

Multimode Simple Type Theory (MSTT)

$\approx$ MTT by Gratzer et al. (LICS20), restricted to simple types.

MSTT is parametrized by a mode theory ($\approx$ small category):

Mode theory	$\approx$	Small category
Modes	$\leftrightarrow$	Objects
Modalities	$\leftrightarrow$	Morphisms

Agda data types TyExpr, TmExpr, CtxExpr indexed by a mode.
MSTT is extrinsically typed.

Multimode Simple Type Theory (MSTT)

$\approx$ MTT by Gratzer et al. (LICS20), restricted to simple types.

MSTT is parametrized by a mode theory ($\approx$ small category):

Mode theory	$\approx$	Small category
Modes	$\leftrightarrow$	Objects
Modalities	$\leftrightarrow$	Morphisms

Agda data types TyExpr, TmExpr, CtxExpr indexed by a mode.
MSTT is extrinsically typed.

Modality μ (from m to n) gives rise to type constructor & context operation lock ($\approx$ left adjoint).

$$
\begin{array}{ccc}
\text{CtxExpr } m & \leftarrow & \text{CtxExpr } n \\
\Gamma \text{ ,lock}\langle\ \mu\ \rangle & \leftarrow\!\shortmid & \Gamma \\
\\
\text{TyExpr } m & \rightarrow & \text{TyExpr } n \\
T & \mapsto & \langle\ \mu \mid T\ \rangle
\end{array}
$$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
  { }0

Hole	Mode	Context	Expected type
0	o	$\diamond$	$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow$ $\langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
  {lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] ?}0

Hole	Mode	Context	Expected type
0	o	$\diamond$	$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow$ $\langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \text{ (m) } \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
  lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] { }0

Hole	Mode	Context	Expected type
0	o	$\diamond$, κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$	$\langle\ \kappa \text{ (m) } \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\, \kappa \mid \langle\, \mu \mid A \,\rangle \Rrightarrow B \,\rangle \Rrightarrow \langle\, \kappa \text{ ⓜ } \mu \mid A \,\rangle \Rrightarrow \langle\, \kappa \mid B \,\rangle$$

```
mod-applicative : TmExpr o
mod-applicative =
    lam[ κ | "f" ∈ ⟨ μ | A ⟩ ⇛ B ] {lam[ κ ⓜ μ | "a" ∈ A ]
    ?}0
```

Hole	Mode	Context	Expected type
0	o	$\diamond$, κ | "f" $\in \langle\, \mu \mid A \,\rangle \Rrightarrow B$	$\langle\, \kappa \text{ ⓜ } \mu \mid A \,\rangle \Rrightarrow \langle\, \kappa \mid B \,\rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \text{ (m) } \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
  lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] lam[κ (m) μ | "a" $\in A$]
  { }0

Hole	Mode	Context	Expected type
0	o	$\diamond$, κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, κ (m) μ | "a" $\in A$	$\langle\ \kappa \mid B\ \rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ ⓜ\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
    lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] lam[κ ⓜ μ | "a" $\in A$]
    {mod⟨ κ ⟩ ?}0

Hole	Mode	Context	Expected type
0	o	◇ , κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, κ ⓜ μ | "a" $\in A$	$\langle\ \kappa \mid B\ \rangle$

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \textcircled{m} \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
    lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] lam[$\kappa \textcircled{m} \mu$ | "a" $\in A$]
    mod⟨ κ ⟩ { }0

Hole	Mode	Context	Expected type
0	n	$\diamond$, κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, $\kappa \textcircled{m} \mu$ | "a" $\in A$,lock⟨ κ ⟩	B

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
    lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] lam[κ ⓜ μ | "a" $\in A$]
    mod⟨ κ ⟩ {? ·⟨ μ ⟩ ?}0

Hole	Mode	Context	Expected type
0	n	◇ , κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, κ ⓜ μ | "a" $\in A$,lock⟨ κ ⟩	B

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \circledcirc \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

$$\begin{aligned}
&\mathsf{mod\text{-}applicative} : \mathsf{TmExpr}\ o\\
&\mathsf{mod\text{-}applicative} =\\
&\qquad \mathsf{lam}[\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ]\ \mathsf{lam}[\ \kappa \circledcirc \mu \mid \texttt{"a"} \in A\ ]\\
&\qquad \mathsf{mod}\langle\ \kappa\ \rangle\ (\ \{\ \}0\ \cdot\langle\ \mu\ \rangle\ \{\ \}1\ )
\end{aligned}$$

Hole	Mode	Context	Expected type
0	n	$\diamond\ ,\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ,$ $\kappa \circledcirc \mu \mid \texttt{"a"} \in A\ ,\mathsf{lock}\langle\ \kappa\ \rangle$	$\langle\ \mu \mid A\ \rangle \Rrightarrow B$
1	m	$\diamond\ ,\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ,$ $\kappa \circledcirc \mu \mid \texttt{"a"} \in A\ ,\mathsf{lock}\langle\ \kappa\ \rangle\ ,\mathsf{lock}\langle\ \mu\ \rangle$	A

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

$\mathsf{mod\text{-}applicative} : \mathsf{TmExpr}\ o$

$\mathsf{mod\text{-}applicative} =$

$\quad \mathsf{lam}[\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ]\ \mathsf{lam}[\ \kappa\ \text{ⓜ}\ \mu \mid \texttt{"a"} \in A\ ]$

$\quad \mathsf{mod}\langle\ \kappa\ \rangle\ (\ \{\mathsf{svar}\ \texttt{"f"}\}0\ \cdot\langle\ \mu\ \rangle\ \{\ \}1\ )$

Hole	Mode	Context	Expected type
0	n	$\diamond\ ,\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ,$ $\kappa\ \text{ⓜ}\ \mu \mid \texttt{"a"} \in A\ ,\mathsf{lock}\langle\ \kappa\ \rangle$	$\langle\ \mu \mid A\ \rangle \Rrightarrow B$
1	m	$\diamond\ ,\ \kappa \mid \texttt{"f"} \in \langle\ \mu \mid A\ \rangle \Rrightarrow B\ ,$ $\kappa\ \text{ⓜ}\ \mu \mid \texttt{"a"} \in A\ ,\mathsf{lock}\langle\ \kappa\ \rangle\ ,\mathsf{lock}\langle\ \mu\ \rangle$	A

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \text{ ⓜ } \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

mod-applicative : TmExpr o
mod-applicative =
  lam[κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$] lam[κ ⓜ μ | "a" $\in A$]
  mod$\langle\ \kappa\ \rangle$ (svar "f" $\cdot\langle\ \mu\ \rangle$ { }1)

Hole	Mode	Context	Expected type
1	m	$\diamond$, κ | "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, κ ⓜ μ | "a" $\in A$,lock$\langle\ \kappa\ \rangle$,lock$\langle\ \mu\ \rangle$	A

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ ⓜ\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

```
mod-applicative : TmExpr o
mod-applicative =
    lam[ κ | "f" ∈ ⟨ μ | A ⟩ ⇛ B ] lam[ κ ⓜ μ | "a" ∈ A ]
    mod⟨ κ ⟩ (svar "f" ·⟨ μ ⟩ {svar "a"}1 )
```

Hole	Mode	Context	Expected type
1	m	$\diamond$, $\kappa \mid$ "f" $\in \langle\ \mu \mid A\ \rangle \Rrightarrow B$, κ ⓜ $\mu \mid$ "a" $\in A$,lock$\langle\ \kappa\ \rangle$,lock$\langle\ \mu\ \rangle$	A

MSTT by Example

$$m \xrightarrow{\mu} n \xrightarrow{\kappa} o$$

Constructing a function of type

$$\langle\ \kappa \mid \langle\ \mu \mid A\ \rangle \Rrightarrow B\ \rangle \Rrightarrow \langle\ \kappa \ \text{ⓜ}\ \mu \mid A\ \rangle \Rrightarrow \langle\ \kappa \mid B\ \rangle$$

```
mod-applicative : TmExpr o
mod-applicative =
    lam[ κ | "f" ∈ ⟨ μ | A ⟩ ⇛ B ] lam[ κ ⓜ μ | "a" ∈ A ]
    mod⟨ κ ⟩ (svar "f" ·⟨ μ ⟩ svar "a")
```

Hole	Mode	Context	Expected type

Running Example: Guarded Recursion

Coinductive streams in Agda:

```
record Stream (A : Set) : Set where
  coinductive
  field
    head : A
    tail : Stream A

zeros : Stream ℕ
head zeros = 0
tail zeros = zeros

map : (A → B) → Stream A → Stream B
head (map f as) = f (head as)
tail (map f as) = map f (tail as)
```

Running Example: Guarded Recursion

$\boxed{\text{nats}}$: Stream $\mathbb{N}$
head nats = 0
tail nats = map suc $\boxed{\text{nats}}$

$$
\begin{array}{ccccccc}
\text{nats} & & 0 & . & . & . & \ldots \\
\Big\downarrow {\scriptstyle \text{map suc}} & & \downarrow & \downarrow & \downarrow & & \\
\text{tail nats} & & . & . & . & \ldots &
\end{array}
$$

Running Example: Guarded Recursion

$\boxed{\mathsf{nats}} : \mathsf{Stream}\ \mathbb{N}$
$\mathsf{head}\ \mathsf{nats} = 0$
$\mathsf{tail}\ \mathsf{nats} = \mathsf{map}\ \mathsf{suc}\ \boxed{\mathsf{nats}}$

$$
\begin{array}{ccccccc}
\mathsf{nats} & & 0 & 1 & \cdot & \cdot & \ldots \\
\mathsf{map\ suc} \downarrow & & \downarrow & \downarrow & \downarrow & & \\
\mathsf{tail\ nats} & & 1 & \cdot & \cdot & \ldots &
\end{array}
$$

Running Example: Guarded Recursion

nats : Stream ℕ
head nats = 0
tail nats = map suc nats

$$\begin{array}{rccccccc} & \text{nats} & 0 & 1 & 2 & \cdot & \dots \\ \text{map suc} & \big\downarrow & \big\downarrow & \big\downarrow & \big\downarrow & & \\ & \text{tail nats} & 1 & 2 & \cdot & \dots & \end{array}$$

Running Example: Guarded Recursion

$\boxed{\text{nats}} : \text{Stream}\ \mathbb{N}$
$\text{head nats} = 0$
$\text{tail nats} = \text{map suc}\ \boxed{\text{nats}}$

$$\begin{tikzcd} \text{nats} \arrow[d, mapsto, "\text{map suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \dots \\ \text{tail nats} & 1 & 2 & 3 & \dots & \end{tikzcd}$$

Running Example: Guarded Recursion

$\boxed{\mathsf{nats}} : \mathsf{Stream}\ \mathbb{N}$

$\mathsf{head}\ \mathsf{nats} = 0$

$\mathsf{tail}\ \mathsf{nats} = \mathsf{map}\ \mathsf{suc}\ \boxed{\mathsf{nats}}$

$$\begin{tikzcd}
\mathsf{nats} \arrow[d, mapsto, "\mathsf{map}\ \mathsf{suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \ldots \\
\mathsf{tail}\ \mathsf{nats} & 1 & 2 & 3 & \ldots &
\end{tikzcd}$$

$\boxed{\mathsf{problem}} : \mathsf{Stream}\ \mathbb{N}$

$\mathsf{head}\ \mathsf{problem} = 0$

$\mathsf{tail}\ \mathsf{problem} = \mathsf{flipFst}\ \boxed{\mathsf{problem}}$

$$\begin{tikzcd}
\mathsf{trouble} \arrow[d, mapsto, "\mathsf{flipFst}"'] & 0 \arrow[dr] & \cdot \arrow[dl] & \cdot \arrow[d] & \cdot & \ldots \\
\mathsf{tail}\ \mathsf{trouble} & \cdot & \cdot & \cdot & \ldots &
\end{tikzcd}$$

Running Example: Guarded Recursion

$\boxed{\text{nats}}$: Stream $\mathbb{N}$
head nats = 0
tail nats = map suc $\boxed{\text{nats}}$

$$\begin{tikzcd}[row sep=large]
\text{nats} \arrow[d, mapsto, "\text{map suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \ldots \\
\text{tail nats} & 1 & 2 & 3 & \ldots &
\end{tikzcd}$$

$\boxed{\text{problem}}$: Stream $\mathbb{N}$
head problem = 0
tail problem = flipFst $\boxed{\text{problem}}$

$$\begin{tikzcd}[row sep=large]
\text{trouble} \arrow[d, mapsto, "\text{flipFst}"'] & 0 \arrow[dr] & \cdot \arrow[dl] & 0 \arrow[d] & \cdot & \ldots \\
\text{tail trouble} & \cdot & 0 & \cdot & \ldots &
\end{tikzcd}$$

Running Example: Guarded Recursion

$\boxed{\mathsf{nats}}$: Stream $\mathbb{N}$
head nats = 0
tail nats = map suc $\boxed{\mathsf{nats}}$

$$
\begin{tikzcd}
\mathsf{nats} \arrow[d, mapsto, "\mathsf{map\ suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \dots \\
\mathsf{tail\ nats} & 1 & 2 & 3 & \dots &
\end{tikzcd}
$$

$\boxed{\mathsf{problem}}$: Stream $\mathbb{N}$
head problem = 0
tail problem = flipFst $\boxed{\mathsf{problem}}$

$$
\begin{tikzcd}
\mathsf{trouble} \arrow[d, mapsto, "\mathsf{flipFst}"'] & 0 \arrow[dr] & \cdot \arrow[dl] & 0 \arrow[d] & 0 & \dots \\
\mathsf{tail\ trouble} & \cdot & 0 & 0 & \dots &
\end{tikzcd}
$$

Running Example: Guarded Recursion

$\boxed{\text{nats}}$: Stream $\mathbb{N}$
head nats = 0
tail nats = map suc $\boxed{\text{nats}}$

$$
\begin{tikzcd}
\text{nats} \arrow[d, mapsto, "\text{map suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \dots \\
\text{tail nats} & 1 & 2 & 3 & \dots &
\end{tikzcd}
$$

$\boxed{\text{problem}}$: Stream $\mathbb{N}$
head problem = 0
tail problem = flipFst $\boxed{\text{problem}}$

$$
\begin{tikzcd}
\text{trouble} \arrow[d, mapsto, "\text{flipFst}"'] & 0 \arrow[dr] & ? \arrow[dl] & 0 \arrow[d] & 0 & \dots \\
\text{tail trouble} & ? & 0 & 0 & \dots &
\end{tikzcd}
$$

Running Example: Guarded Recursion

$\boxed{\mathsf{nats}} : \mathsf{Stream}\ \mathbb{N}$

$\mathsf{head\ nats} = 0$

$\mathsf{tail\ nats} = \mathsf{map\ suc}\ \boxed{\mathsf{nats}}$

$\mathsf{map\ suc} : \mathsf{Stream}\ \mathbb{N} \to \mathsf{Stream}\ \mathbb{N}$

$$\begin{tikzcd}
\mathsf{nats} \arrow[d, mapsto, "\mathsf{map\ suc}"'] & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 & \dots \\
\mathsf{tail\ nats} & 1 & 2 & 3 & \dots &
\end{tikzcd}$$

$\boxed{\mathsf{problem}} : \mathsf{Stream}\ \mathbb{N}$

$\mathsf{head\ problem} = 0$

$\mathsf{tail\ problem} = \mathsf{flipFst}\ \boxed{\mathsf{problem}}$

$\mathsf{flipFst} : \mathsf{Stream}\ \mathbb{N} \to \mathsf{Stream}\ \mathbb{N}$

$$\begin{tikzcd}
\mathsf{trouble} \arrow[d, mapsto, "\mathsf{flipFst}"'] & 0 \arrow[dr] & ? \arrow[dl] & 0 \arrow[d] & 0 & \dots \\
\mathsf{tail\ trouble} & ? & 0 & 0 & \dots &
\end{tikzcd}$$

Guarded Streams in MSTT

Approach based on Veltri and and van der Weide (FSCD19) & Gratzer et al. (LICS20).

Mode Theory:

$$\begin{tikzcd} \omega \arrow[loop left, "\mathsf{later}"] \arrow[r, bend left, "\mathsf{forever}"] & \star \arrow[l, bend left, "\mathsf{constantly}"] \end{tikzcd}$$

$$\mathsf{forever} \circledcirc \mathsf{later} \simeq^{\mathsf{m}} \mathsf{forever}$$
$$\mathsf{forever} \circledcirc \mathsf{constantly} \simeq^{\mathsf{m}} \mathbb{1}$$

Guarded Streams in MSTT

Approach based on Veltri and and van der Weide (FSCD19) & Gratzer et al. (LICS20).

Mode Theory:

$$\begin{tikzcd} \omega \arrow["\mathsf{later}"', loop left] \arrow[r, bend left, "\mathsf{forever}"] & \star \arrow[l, bend left, "\mathsf{constantly}"] \end{tikzcd}$$

$$\mathsf{forever} ⓜ \mathsf{later} \simeq^{\mathsf{m}} \mathsf{forever}$$
$$\mathsf{forever} ⓜ \mathsf{constantly} \simeq^{\mathsf{m}} \mathbb{1}$$

New non-modal type/term constructors:

$$\begin{tikzcd} \mathsf{GStream}\ A \arrow[r, bend left, "(\mathsf{g\text{-}head},\ \mathsf{g\text{-}tail})"] & \langle\ \mathsf{constantly} \mid A\ \rangle \times \langle\ \mathsf{later} \mid \mathsf{GStream}\ A\ \rangle \arrow[l, bend left, "\mathsf{g\text{-}cons}"] \end{tikzcd}$$

$$\frac{\Gamma\ ,\ \mathsf{later} \mid x \in T \vdash t : T \ @\ \omega}{\Gamma \vdash \mathsf{löb}[\mathsf{later} \mid\ x \in\ T\ ]\ t :\ T\ @\ \omega}$$

Implementation of nats in MSTT

g-nats : TmExpr ω
g-nats = $\boxed{\{\ \}0}$

Hole	Mode	Context	Expected type
0	ω	$\diamond$	GStream Nat

Implementation of nats in MSTT

$$\text{g-nats} : \text{TmExpr}\ \omega$$
$$\text{g-nats} = \boxed{\{\text{löb[later}|\ \texttt{"s"} \in \text{GStream Nat}\ ]\ ?\}0}$$

Hole	Mode	Context	Expected type
0	ω	$\diamond$	GStream Nat

Implementation of nats in MSTT

g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat] { }0

Hole	Mode	Context	Expected type
0	ω	$\diamond$, later | "s" ∈ GStream Nat	GStream Nat

Implementation of nats in MSTT

$\Gamma \vdash$ g-cons : $\langle$ constantly $|$ A $\rangle \Rrightarrow \langle$ later $|$ GStream A $\rangle \Rrightarrow$ GStream A

g-nats : TmExpr ω
g-nats = löb[later| "s" $\in$ GStream Nat] {g-cons
·⟨ constantly ⟩ ?
·⟨ later ⟩ ?}0

Hole	Mode	Context	Expected type
0	ω	$\diamond$, later | "s" $\in$ GStream Nat	GStream Nat

Implementation of nats in MSTT

$\Gamma \vdash$ g-cons : ⟨ constantly | A ⟩ ⇛ ⟨ later | GStream A ⟩ ⇛ GStream A

g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat] g-cons
·⟨ constantly ⟩ { }0
·⟨ later ⟩ { }1

Hole	Mode	Context	Expected type
0	$\star$	◇ , later | "s" ∈ GStream Nat ,lock⟨ constantly ⟩	Nat
1	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle\ \text{constantly} \mid A\ \rangle \Rrightarrow \langle\ \text{later} \mid \text{GStream}\ A\ \rangle \Rrightarrow \text{GStream}\ A$$

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ {lit 0}0
      ·⟨ later ⟩ { }1
```

Hole	Mode	Context	Expected type
0	⋆	◇ , later | "s" ∈ GStream Nat ,lock⟨ constantly ⟩	Nat
1	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle\ \text{constantly} \mid A\ \rangle \Rrightarrow \langle\ \text{later} \mid \text{GStream}\ A\ \rangle \Rrightarrow \text{GStream}\ A$$

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ lit 0
      ·⟨ later ⟩ { }1
```

Hole	Mode	Context	Expected type
1	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle \text{ constantly} \mid A \rangle \Rrightarrow \langle \text{ later} \mid \text{GStream } A \rangle \Rrightarrow \text{GStream } A$$

$$\Gamma \vdash \text{g-map} : \langle \text{ constantly} \mid A \Rrightarrow B \rangle \Rrightarrow \text{GStream } A \Rrightarrow \text{GStream } B$$

(Implementation: see paper.)

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ lit 0
      ·⟨ later ⟩ {(g-map ·⟨ constantly ⟩ ? · ?)}1
```

Hole	Mode	Context	Expected type
1	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle\ \text{constantly} \mid A\ \rangle \Rrightarrow \langle\ \text{later} \mid \text{GStream}\ A\ \rangle \Rrightarrow \text{GStream}\ A$$

$$\Gamma \vdash \text{g-map} : \langle\ \text{constantly} \mid A \Rrightarrow B\ \rangle \Rrightarrow \text{GStream}\ A \Rrightarrow \text{GStream}\ B$$

(Implementation: see paper.)

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ lit 0
      ·⟨ later ⟩ (g-map ·⟨ constantly ⟩ { }1 · { }2)
```

Hole	Mode	Context	Expected type
1	$\star$	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩ ,lock⟨ constantly ⟩	Nat ⇛ Nat
2	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle\ \text{constantly} \mid A\ \rangle \Rrightarrow \langle\ \text{later} \mid \text{GStream}\ A\ \rangle \Rrightarrow \text{GStream}\ A$$

$$\Gamma \vdash \text{g-map} : \langle\ \text{constantly} \mid A \Rrightarrow B\ \rangle \Rrightarrow \text{GStream}\ A \Rrightarrow \text{GStream}\ B$$

(Implementation: see paper.)

g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat] g-cons
  ·⟨ constantly ⟩ lit 0
  ·⟨ later ⟩ (g-map ·⟨ constantly ⟩ {suc}1 · { }2)

Hole	Mode	Context	Expected type
1	$\star$	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩ ,lock⟨ constantly ⟩	Nat ⇛ Nat
2	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle\ \text{constantly} \mid A\ \rangle \Rrightarrow \langle\ \text{later} \mid \text{GStream}\ A\ \rangle \Rrightarrow \text{GStream}\ A$$

$$\Gamma \vdash \text{g-map} : \langle\ \text{constantly} \mid A \Rrightarrow B\ \rangle \Rrightarrow \text{GStream}\ A \Rrightarrow \text{GStream}\ B$$

(Implementation: see paper.)

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ lit 0
      ·⟨ later ⟩ (g-map ·⟨ constantly ⟩ suc · { }2 )
```

Hole	Mode	Context	Expected type
2	ω	◇ , later | "s" ∈ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle \text{ constantly } \mid A \rangle \Rrightarrow \langle \text{ later } \mid \text{GStream } A \rangle \Rrightarrow \text{GStream } A$$

$$\Gamma \vdash \text{g-map} : \langle \text{ constantly } \mid A \Rrightarrow B \rangle \Rrightarrow \text{GStream } A \Rrightarrow \text{GStream } B$$

(Implementation: see paper.)

```
g-nats : TmExpr ω
g-nats = löb[later| "s" ∈ GStream Nat ] g-cons
      ·⟨ constantly ⟩ lit 0
      ·⟨ later ⟩ (g-map ·⟨ constantly ⟩ suc · {svar "s"}2 )
```

Hole	Mode	Context	Expected type
2	ω	$\diamond$, later | "s" $\in$ GStream Nat ,lock⟨ later ⟩	GStream Nat

Implementation of nats in MSTT

$$\Gamma \vdash \text{g-cons} : \langle \text{ constantly} \mid A \rangle \Rrightarrow \langle \text{ later} \mid \text{GStream } A \rangle \Rrightarrow \text{GStream } A$$

$$\Gamma \vdash \text{g-map} : \langle \text{ constantly} \mid A \Rrightarrow B \rangle \Rrightarrow \text{GStream } A \Rrightarrow \text{GStream } B$$

(Implementation: see paper.)

g-nats : TmExpr ω
g-nats = löb[later| "s" $\in$ GStream Nat] g-cons
 ·⟨ constantly ⟩ lit 0
 ·⟨ later ⟩ (g-map ·⟨ constantly ⟩ suc · svar "s")

Hole	Mode	Context	Expected type

Implementation of nats in MSTT

What about flipFst?

g-flipFst : GStream Nat $\Rrightarrow$ $\langle$ later | GStream Nat $\rangle$

g-map $\cdot\langle$ constantly $\rangle$ suc : GStream Nat $\Rrightarrow$ GStream Nat

Implementation of nats in MSTT

What about flipFst?

$\text{g-flipFst} : \text{GStream Nat} \Rrightarrow \langle\ \text{later} \mid \text{GStream Nat}\ \rangle$

$\text{g-map} \cdot\langle\ \text{constantly}\ \rangle\ \text{suc} : \text{GStream Nat} \Rrightarrow \text{GStream Nat}$

Guarded streams behave differently than Agda streams.

$\text{Stream'} : \text{TyExpr}\ \star \to \text{TyExpr}\ \star$

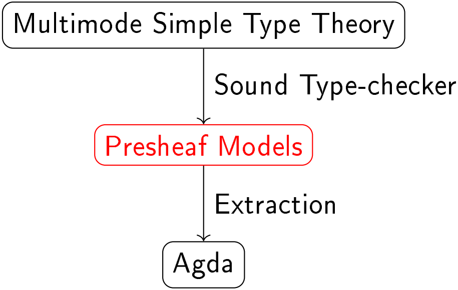
$\text{Stream'}\ A = \langle\ \text{forever} \mid \text{GStream}\ A\ \rangle$

$$\begin{tikzcd}
\omega \arrow[loop left, "\text{later}"] \arrow[r, bend left, "\text{forever}"] & \star \arrow[l, bend left, "\text{constantly}"]
\end{tikzcd}$$

$\text{nats} : \text{TmExpr}\ \star$

$\text{nats} = \text{mod}\langle\ \text{forever}\ \rangle\ \text{g-nats}$

Overview of Sikkel



Presheaf Models: Intuition

Interpretation of GStream A:

$$\mathsf{Vec}_1\, A \xleftarrow{\mathsf{init}} \mathsf{Vec}_2\, A \xleftarrow{\mathsf{init}} \ldots \xleftarrow{\mathsf{init}} \mathsf{Vec}_n\, A \xleftarrow{\mathsf{init}} \ldots$$

Presheaf Models: Intuition

Interpretation of GStream A and of $\langle$ later $|$ GStream A $\rangle$:

$$\mathsf{Vec}_1\, A \xleftarrow{\mathsf{init}} \mathsf{Vec}_2\, A \xleftarrow{\mathsf{init}} \dots \xleftarrow{\mathsf{init}} \mathsf{Vec}_n\, A \xleftarrow{\mathsf{init}} \dots$$

$$\top \longleftarrow \mathsf{Vec}_1\, A \xleftarrow{\mathsf{init}} \dots \xleftarrow{\mathsf{init}} \mathsf{Vec}_{n-1}\, A \xleftarrow{\mathsf{init}} \dots$$

Presheaf Models: Intuition

Interpretation of GStream A and of $\langle$ later | GStream A $\rangle$:

$$
\begin{tikzcd}
\mathrm{Vec}_1\, A & \mathrm{Vec}_2\, A \arrow[l, "\mathrm{init}"'] & \dots \arrow[l, "\mathrm{init}"'] & \mathrm{Vec}_n\, A \arrow[l, "\mathrm{init}"'] & \dots \arrow[l, "\mathrm{init}"'] \\
\top \arrow[u] & \mathrm{Vec}_1\, A \arrow[l] \arrow[u] & \dots \arrow[l, "\mathrm{init}"'] & \mathrm{Vec}_{n-1}\, A \arrow[l, "\mathrm{init}"'] \arrow[u] & \dots \arrow[l, "\mathrm{init}"']
\end{tikzcd}
$$

Löb induction boils down to induction on natural numbers.

Presheaf Models: Intuition

Interpretation of GStream A and of $\langle$ later $\mid$ GStream A $\rangle$:

$$\mathsf{Vec}_1\ A \xleftarrow{\mathsf{init}} \mathsf{Vec}_2\ A \xleftarrow{\mathsf{init}} \ldots \xleftarrow{\mathsf{init}} \mathsf{Vec}_n\ A \xleftarrow{\mathsf{init}} \ldots$$

$$\top \longleftarrow \mathsf{Vec}_1\ A \xleftarrow{\mathsf{init}} \ldots \xleftarrow{\mathsf{init}} \mathsf{Vec}_{n-1}\ A \xleftarrow{\mathsf{init}} \ldots$$

Löb induction boils down to induction on natural numbers.

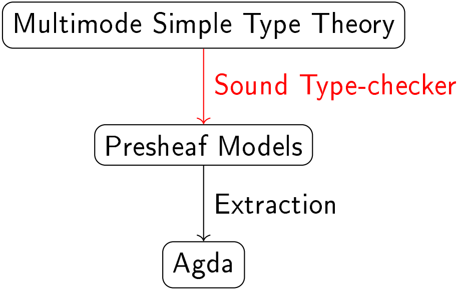
General presheaf model: contexts & types represented as diagrams, shape determined by base category.

Presheaf Models of Type Theory

MSTT	$\rightsquigarrow$	Presheaf model
Mode	$\rightsquigarrow$	Base category C
Context	$\rightsquigarrow$	Presheaf Γ : $\mathsf{Ctx}\ C$
Type	$\rightsquigarrow$	"Dependent presheaf" T : $\mathsf{Ty}\ \Gamma$
Term	$\rightsquigarrow$	"Dependent presheaf morphism" t : $\mathsf{Tm}\ \Gamma\ T$
Modality	$\rightsquigarrow$	Dependent right adjoint (DRA, Birkedal et al. (2020))

Note: Semantic types can depend on variables.

Overview of Sikkel

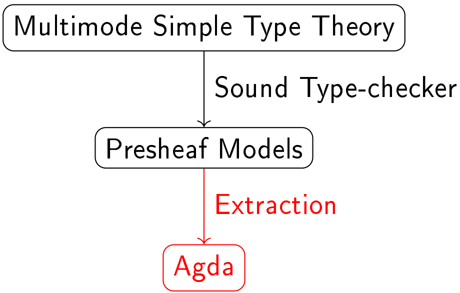


Sound Type-checker

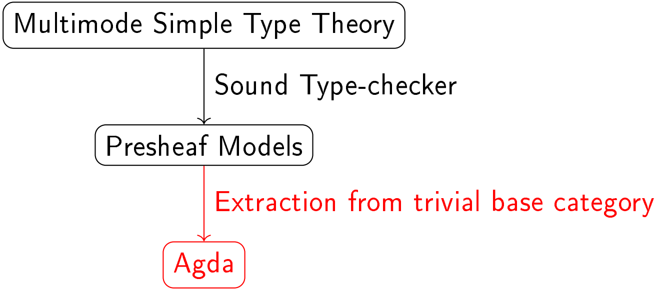
$$
\begin{array}{c}
(\Gamma : \mathsf{CtxExpr}\ m) \times (t : \mathsf{TmExpr}\ m) \\
\Big\downarrow \text{infer-interpret} \\
\mathsf{Maybe}\ (\Sigma[\ T \in \mathsf{TyExpr}\ m\ ]\ (\mathsf{Tm}\ \{\llbracket\ m\ \rrbracket\mathsf{mode}\}\ \llbracket\ \Gamma\ \rrbracket\mathsf{ctx}\ \llbracket\ T\ \rrbracket\mathsf{ty}))
\end{array}
$$

User implementing new type theory must specify interpretations $\llbracket\_\rrbracket\mathsf{mode}$, $\llbracket\_\rrbracket\mathsf{modality}$, ...

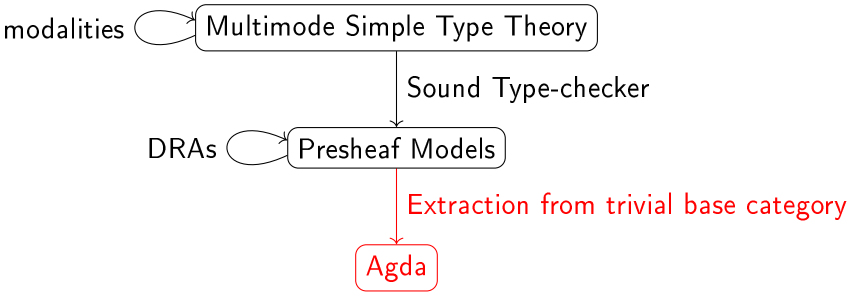
Overview of Sikkel



Overview of Sikkel



Overview of Sikkel



Extraction to the Meta-level

Problem: Agda types obtained by directly unpacking semantic types (trivial base category) are not always as intended, only isomorphic.

$$
\begin{aligned}
[\![\text{ Stream' Nat }]\!]\text{ty} \quad &= \quad \forall\ n \rightarrow \text{Vec}\ \mathbb{N}\ (\text{suc}\ n) \qquad \text{(+ naturality condition)} \\
&\cong \quad \text{Stream}\ \mathbb{N}
\end{aligned}
$$

Extraction to the Meta-level

Problem: Agda types obtained by directly unpacking semantic types (trivial base category) are not always as intended, only isomorphic.

$$\begin{array}{rcll} [\![\text{ Stream' Nat }]\!]\text{ty} & = & \forall\ n \to \text{Vec } \mathbb{N}\ (\text{suc } n) & \text{(+ naturality condition)} \\ & \cong & \text{Stream } \mathbb{N} & \end{array}$$

Solution: type class Extractable for closed semantic types.

Instance provides intended translated type and function extract-term to apply isomorphism.

Constructing the Agda stream of natural numbers:

nats-agda : Stream $\mathbb{N}$
nats-agda = extract-term $[\![$ nats $]\!]$tm

Conclusion

Sikkel allows to work in a **multi-modal** setting within **off-the-shelf** Agda and to interpret programs as Agda programs.

- MSTT parametrized by **general mode theory**.
- **Easy to extend** with extra (non-modal) primitives.
- Semantic layer fully general in **base category**. Extra example in paper: parametricity.
- Elegant **extraction mechanism** thanks to modes and modalities.

Future Work

- Support for **dependent types** at syntactic layer.
  - Challenging to implement interpretation that passes Agda's termination checker.
  - Intermediate step: **extensible program logic**, orthogonal to Sikkel's current 3 layers.
- Implementation of **Hofmann-Streicher universe** at semantic layer.
- Investigating **other applications** (ongoing exploration of nominal type theory).

Thanks for listening!
Questions?

https://github.com/JorisCeulemans/sikkel/releases/tag/v1.0