Matching Logic Explained

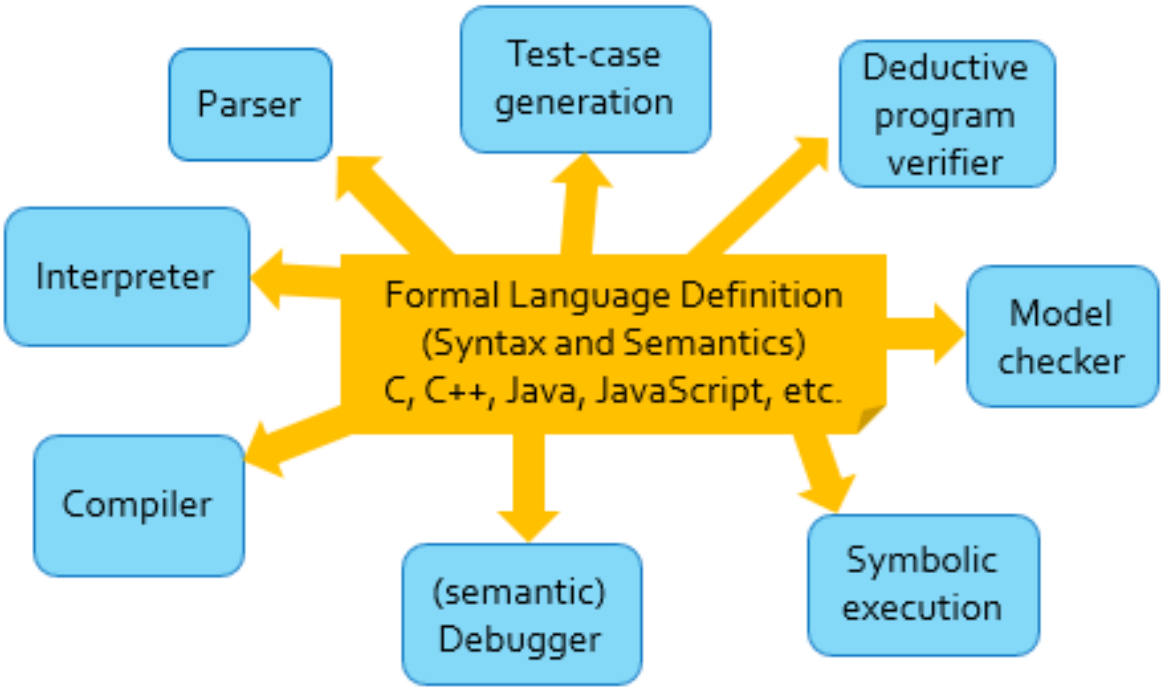
Dorel Lucanu[1]
Joint work with Xiaohong Chen[2], Grigore Roșu[2]

[1]Alexandru Ioan Cuza University of Iași
[2]University of Illinois at Urbana-Champaign

FROM, September 05, 2019

Plan

Ideal language framework: tools derived from formal language definition



A Brief History of K Framework

- 2003, Grigore Roșu at UIUC: motivated mainly by teaching programming languages and noticing that the existing semantic frameworks have limitations
- 2010-2013: joint work between Formal Systems Laboratory (FSL) from University of Illinois at Urbana-Champaign (UIUC) lead by Grigore Roșu and Formal Methods in Software Engineering (FMSE) from Al. I. Cuza University (UAIC) lead by presenter
- since 2014: joint work between FSL and Runtimeverification - a start-up founded by Grigore Roșu

A Fundamental Question

What is the best candidate for a unifying logic to be used for programming languages, specification, and verification?

Such a candidate should be able (at least)

1. to represent the structure of the programs and their configurations,
2. to specify the semantics of the language simply and in a scalable (modular) way, and
3. to support (symbolic) execution and verification, including specification of properties

None of the existing logics supplied a satisfactory answer to these requirements.

A Fundamental Question

What is the best candidate for a unifying logic to be used for programming languages, specification, and verification?
Such a candidate should be able (at least)

1. to represent the structure of the programs and their configurations,
2. to specify the semantics of the language simply and in a scalable (modular) way, and
3. to support (symbolic) execution and verification, including specification of properties

None of the existing logics supplied a satisfactory answer to these requirements.

A Fundamental Question

What is the best candidate for a unifying logic to be used for programming languages, specification, and verification?
Such a candidate should be able (at least)

1. to represent the structure of the programs and their configurations,
2. to specify the semantics of the language simply and in a scalable (modular) way, and
3. to support (symbolic) execution and verification, including specification of properties

None of the existing logics supplied a satisfactory answer to these requirements.

A Fundamental Question

What is the best candidate for a unifying logic to be used for programming languages, specification, and verification?
Such a candidate should be able (at least)

1. to represent the structure of the programs and their configurations,
2. to specify the semantics of the language simply and in a scalable (modular) way, and
3. to support (symbolic) execution and verification, including specification of properties

None of the existing logics supplied a satisfactory answer to these requirements.

A Fundamental Question

What is the best candidate for a unifying logic to be used for programming languages, specification, and verification?
Such a candidate should be able (at least)

1. to represent the structure of the programs and their configurations,
2. to specify the semantics of the language simply and in a scalable (modular) way, and
3. to support (symbolic) execution and verification, including specification of properties

None of the existing logics supplied a satisfactory answer to these requirements.

Initial Idea

- configuration: a pair term $\wedge$ constraint
  $\langle \texttt{x = x * 2; y = x +1;}, \texttt{x} \mapsto a + 3\ \texttt{y} \mapsto b \rangle \wedge a \leq b$
- language definition: rules
  $\texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \wedge B == \mathit{true} \Rightarrow S_1$
- properties: reachability formulas $\phi_1 \Rightarrow \phi_2$

Outcomes:

- symbolic execution automatically derived from definition of the semantics
- reachability logics: a couple of proof systems for reachability formulas
- implementations that showed the feasibility of the approach on real case studies (C, Java, JavaScript, etc)

Initial Idea

- configuration: a pair term $\wedge$ constraint
  $\langle \texttt{x = x * 2; y = x +1;}, \texttt{x} \mapsto a + 3\ \texttt{y} \mapsto b \rangle \wedge a \leq b$
- language definition: rules
  $\texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \wedge B == \mathit{true} \Rightarrow S_1$
- properties: reachability formulas $\phi_1 \Rightarrow \phi_2$

Outcomes:

- symbolic execution automatically derived from definition of the semantics
- reachability logics: a couple of proof systems for reachability formulas
- implementations that showed the feasibility of the approach on real case studies (C, Java, JavaScript, etc)

Current (and Final?) Status

Matching Logic (ML) (2017):

- no difference between function symbols and predicate symbols
  $\mathbb{s}(\exists\, x \wedge x > 5) \vee \mathit{plus}(x, y \wedge y < 8) \wedge x < y$

Matching μ-Logic (MmL) (2019):

- ML with least fixed-point (lfp) and greatest fixed-point (gfp) (as dual) operators

Applicative Matching Logic (AML) (2019):

- a fragment of MmL much simpler (and thus more appealing from a foundational and implementation perspectives), yet as expressive as MmL

Current (and Final?) Status

Matching Logic (ML) (2017):

- no difference between function symbols and predicate symbols
  $\mathbb{s}(\exists\, x \wedge x > 5) \vee \mathit{plus}(x, y \wedge y < 8) \wedge x < y$

Matching μ-Logic (MmL) (2019):

- ML with least fixed-point (lfp) and greatest fixed-point (gfp) (as dual) operators

Applicative Matching Logic (AML) (2019):

- a fragment of MmL much simpler (and thus more appealing from a foundational and implementation perspectives), yet as expressive as MmL

Current (and Final?) Status

Matching Logic (ML) (2017):

- no difference between function symbols and predicate symbols
  $\mathbb{s}(\exists\, x \wedge x > 5) \vee \mathit{plus}(x, y \wedge y < 8) \wedge x < y$

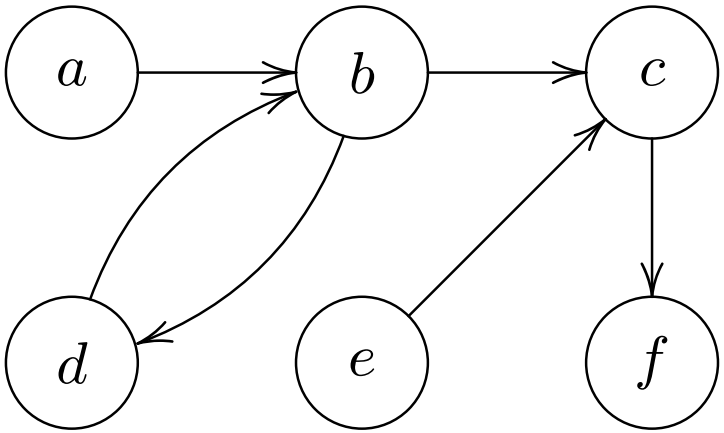
Matching μ-Logic (MmL) (2019):

- ML with least fixed-point (lfp) and greatest fixed-point (gfp) (as dual) operators

Applicative Matching Logic (AML) (2019):

- a fragment of MmL much simpler (and thus more appealing from a foundational and implementation perspectives), yet as expressive as MmL

A Taste of Matching Logic

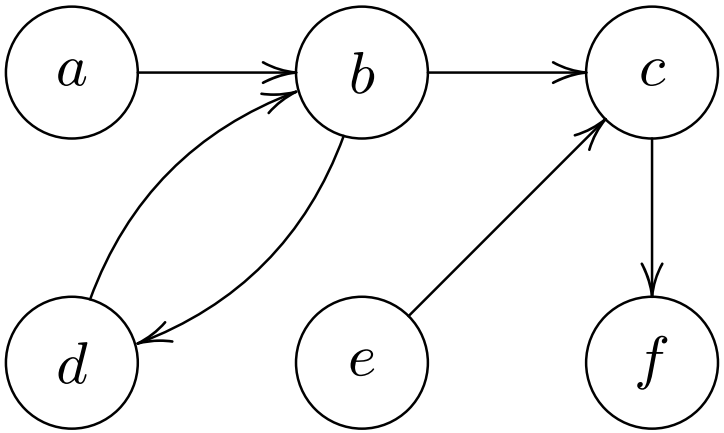


$a \in [\![State]\!] \wedge b \in [\![State]\!] \wedge \cdots$
(a, b, c, d, e, f are constants of sort $State$),
$\forall s \,.\, s \in [\![State]\!] \rightarrow \bullet\, s \subseteq [\![State]\!]$
$(\bullet\_ : [\![State]\!] \rightarrow [\![State]\!])$,

- $\bullet\, a = \bot$
- $\bullet\, d = b$
- $\bullet\, b = a \vee d$
- $\bullet\, e = \bot$
- $\bullet\, c = b \vee e$
- $\bullet\, f = c$

We may prove that there is an infinite execution starting from a:
$\Gamma \models a \rightarrow \nu\, Y \,.\, \bullet\, Y$

A Taste of Matching Logic

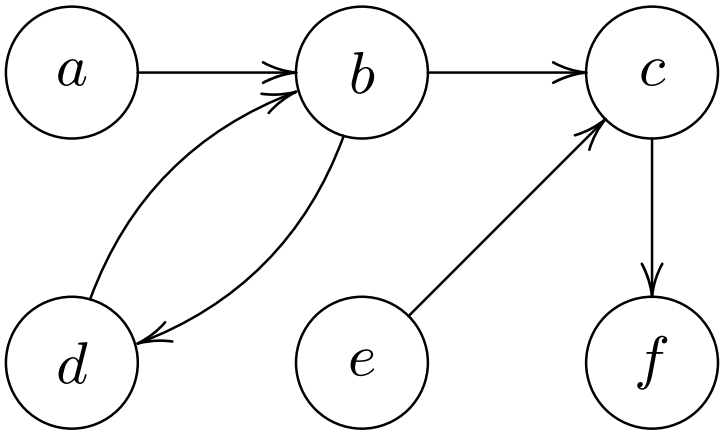


$a \in [\![State]\!] \wedge b \in [\![State]\!] \wedge \cdots$
(a, b, c, d, e, f are constants of sort $State$),
$\forall s \,.\, s \in [\![State]\!] \rightarrow \bullet\, s \subseteq [\![State]\!]$
($\bullet\_ : [\![State]\!] \rightarrow [\![State]\!]$),

- $\bullet\, a = \bot$
- $\bullet\, b = a \vee d$
- $\bullet\, c = b \vee e$
- $\bullet\, d = b$
- $\bullet\, e = \bot$
- $\bullet\, f = c$

We may prove that there is an infinite execution starting from a:
$\Gamma \models a \rightarrow \nu\, Y \,.\, \bullet\, Y$

A Taste of Matching Logic



$a \in [\![State]\!] \wedge b \in [\![State]\!] \wedge \cdots$
(a, b, c, d, e, f are constants of sort $State$),
$\forall s \,.\, s \in [\![State]\!] \rightarrow \bullet\, s \subseteq [\![State]\!]$
($\bullet\_ : [\![State]\!] \rightarrow [\![State]\!]$),

- $\bullet\, a = \bot$
- $\bullet\, d = b$
- $\bullet\, b = a \vee d$
- $\bullet\, e = \bot$
- $\bullet\, c = b \vee e$
- $\bullet\, f = c$

We may prove that there is an infinite execution starting from a:
$\Gamma \models a \rightarrow \nu\, Y \,.\, \bullet\, Y$

This Talk

- a gentle intoduction to the three components of the ML framework based on:

  Grigore Roșu. Matching logic. Logical Methods in Computer Science,

  Xiaohong Chen and Grigore Roșu. Matching mu-logic. LICS'19, 2019.

  Xiaohong Chen and Grigore Roșu. Applicative matching logic. Technical Report, `http://hdl.handle.net/2142/104616`, 2019 13(4):1-61, 2017.
- several case studies showing how the inductive reasoning and the coinductive reasoning are applied within ML (work in progress)

Plan

How We Define Syntax

Backus-Naur notation/grammar:

$$Nat ::= \mathbb{0} \mid \mathbb{s}(Nat) \mid le(Nat, Nat)$$

Inference rules:

$$\frac{}{\mathbb{0}:Nat} \qquad \frac{n:Nat}{\mathbb{s}(n):Nat} \qquad \frac{m:Nat \quad n:Nat}{le(m,n):Nat}$$

Functional-programming-language-like syntax:

```
nat : Set := Zero : nat | Succ : nat -> nat
le (m n : nat) : nat := ...
```

Signatures in ML

Σ:

- sorts: S
- symbols: $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$
- variables: $\mathrm{VAR} = \{\mathrm{VAR}_s\}_{s \in S}$

Example BNAT:

$$
\begin{aligned}
&S = \{Nat\},\\
&\Sigma_{\varepsilon,Nat} = \{\mathbb{0}\},\\
&\Sigma_{Nat,Nat} = \{\mathbb{s}\},\\
&\Sigma_{Nat\ Nat,Nat} = \{le\},\\
&\Sigma_{w,s} = \emptyset \text{ otherwise.}
\end{aligned}
$$

Patterns

$\textsc{Pattern}(\Sigma)$:

$$\varphi_s ::= x{:}s \mid \varphi_s \wedge \varphi_s \mid \neg\varphi_s \mid \exists x{:}s' \,.\, \varphi_s \mid \sigma(\varphi_{s_1}, \ldots, \varphi_{s_n})$$

where $x{:}s \in \textsc{Var}_s$, $\sigma \in \Sigma_{s_1 \cdots s_n, s}$.
Examples of BNAT patterns:

$$\mathbb{s}(\mathbb{0})$$
$$\neg\mathbb{s}(\mathbb{0})$$
$$x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat),$$
$$\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat),$$
$$\mathbb{s}(\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat))$$

Models

In a model M

- each sort $s \in S$ is interpreted as a set M_s;
- each symbol $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ as a function (relation) $M_\sigma : M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$;
- the variables are interpreted using valuations $\rho : \mathrm{VAR} \to M$, such that $\rho(x{:}s) \in M_s$ for all $x{:}s \in \mathrm{VAR}_s$, $s \in S$.

Remark

If $\sigma \in \Sigma_{\varepsilon, s}$, i.e., σ is a constant of sort s, then $M_\sigma \subseteq M_s$.

It is recommended to think that an interpretation of a symbol is a relation!

The interpretations of symbols are pointwise extended to sets:

$$M_\sigma(A_1, \ldots, A_n) = \bigcup \{ M_\sigma(a_1, \ldots, a_n) \mid a_1 \in A_1, \ldots, a_n \in A_n \}$$

Models

In a model M

- each sort $s \in S$ is interpreted as a set M_s;
- each symbol $\sigma \in \Sigma_{s_1 \dots s_n, s}$ as a function (relation) $M_\sigma : M_{s_1} \times \dots \times M_{s_n} \to \mathcal{P}(M_s)$;
- the variables are interpreted using valuations $\rho : \mathrm{VAR} \to M$, such that $\rho(x{:}s) \in M_s$ for all $x{:}s \in \mathrm{VAR}_s$, $s \in S$.

Remark

If $\sigma \in \Sigma_{\varepsilon, s}$, i.e., σ is a constant of sort s, then $M_\sigma \subseteq M_s$.

It is recommended to think that an interpretation of a symbol is a relation!

The interpretations of symbols are pointwise extended to sets:

$$M_\sigma(A_1, \dots, A_n) = \bigcup \{ M_\sigma(a_1, \dots, a_n) \mid a_1 \in A_1, \dots, a_n \in A_n \}$$

Models

In a model M

- each sort $s \in S$ is interpreted as a set M_s;
- each symbol $\sigma \in \Sigma_{s_1 \dots s_n, s}$ as a function (relation) $M_\sigma : M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$;
- the variables are interpreted using valuations $\rho : \mathrm{VAR} \to M$, such that $\rho(x{:}s) \in M_s$ for all $x{:}s \in \mathrm{VAR}_s$, $s \in S$.

Remark

If $\sigma \in \Sigma_{\varepsilon, s}$, i.e., σ is a constant of sort s, then $M_\sigma \subseteq M_s$.

It is recommended to think that an interpretation of a symbol is a relation!

The interpretations of symbols are pointwise extended to sets:

$$M_\sigma(A_1, \dots, A_n) = \bigcup \{ M_\sigma(a_1, \dots, a_n) \mid a_1 \in A_1, \dots, a_n \in A_n \}$$

Models

In a model M

- each sort $s \in S$ is interpreted as a set M_s;
- each symbol $\sigma \in \Sigma_{s_1 \dots s_n, s}$ as a function (relation) $M_\sigma : M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$;
- the variables are interpreted using valuations $\rho : \mathrm{VAR} \to M$, such that $\rho(x{:}s) \in M_s$ for all $x{:}s \in \mathrm{VAR}_s$, $s \in S$.

Remark

If $\sigma \in \Sigma_{\varepsilon, s}$, i.e., σ is a constant of sort s, then $M_\sigma \subseteq M_s$.

It is recommended to think that an interpretation of a symbol is a relation!

The interpretations of symbols are pointwise extended to sets:

$$M_\sigma(A_1, \dots, A_n) = \bigcup \{ M_\sigma(a_1, \dots, a_n) \mid a_1 \in A_1, \dots, a_n \in A_n \}$$

The First BNAT Model

$$\mathbb{M}1_{Nat} = \mathbb{N} = \{0, 1, 2, \ldots\};$$
$$\mathbb{M}1_{\mathbb{0}} = \{0\};$$
$$\mathbb{M}1_{\mathbb{s}}(n) = \{n + 1\};$$
$$\mathbb{M}1_{le}(m, n) = \begin{cases} \mathbb{N} & , m \leq n, \\ \emptyset & , \text{otherwise.} \end{cases}$$

The Third BNAT Model

$$\mathbb{M}3_{Nat} = \mathbb{N};$$

$$\mathbb{M}3_{\mathbb{0}} = \{0\};$$

$$\mathbb{M}3_{\mathbb{s}}(m) = \{n \mid n \in \mathbb{N}, m < n\};$$

$$\mathbb{M}3_{le}(m, n) = \begin{cases} \mathbb{N} & , \mathbb{M}3_{\mathbb{s}}(n) \subseteq \mathbb{M}3_{\mathbb{s}}(m), \\ \emptyset & , \text{otherwise.} \end{cases}$$

Interpretations of the Patterns

$\rho : \text{VAR} \to M$ are inductively extended to $\overline{\rho} : \text{PATTERN}(\Sigma) \to \mathcal{P}(M)$ as follows:

$$\overline{\rho}(x{:}s) = \{\rho(x)\};$$
$$\overline{\rho}(\varphi_s \wedge \varphi'_s) = \overline{\rho}(\varphi_s) \cap \overline{\rho}(\varphi'_s);$$
$$\overline{\rho}(\neg\varphi_s) = M_s \setminus \overline{\rho}(\varphi_s);$$
$$\overline{\rho}(\exists x{:}s' \,.\, \varphi_s) = \bigcup_{a \in M_s} \overline{\rho[a/x]}(\varphi);$$
$$\overline{\rho}(\sigma(\varphi_1, \ldots, \varphi_n)) = M_\sigma(\overline{\rho}(\varphi_1), \ldots, \overline{\rho}(\varphi_n));$$

where $\rho[a/x] : \text{VAR} \to M$ is the valuation defined by

$$\rho[a/x](y) = \begin{cases} a & , y = x, \\ \rho(y) & , y \neq x \end{cases}.$$

Interpretations of the Patterns: Example 1/2

$$\overline{\rho}(\mathbb{s}(\mathbb{0})) = 1 \qquad \forall \rho : \textsc{Var} \to \mathbb{M}1$$
$$\overline{\rho}(\mathbb{s}(\mathbb{0})) = \{2, 3, \ldots\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}3$$

$$\overline{\rho}(\neg\mathbb{s}(\mathbb{0})) = \mathbb{N} \setminus 1 \qquad \forall \rho : \textsc{Var} \to \mathbb{M}1$$
$$\overline{\rho}(\neg\mathbb{s}(\mathbb{0})) = \mathbb{N} \setminus \{2, 3, \ldots\} = \{0, 1\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}3$$

Interpretations of the Patterns: Example 2/2

$\overline{\rho}(x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat)) = \{0\} \cap \emptyset = \emptyset \qquad \rho : \textsc{Var} \to \mathbb{M}1, \rho(x) = 0$

$\overline{\rho}(x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat)) = \{3\} \cap \mathbb{N} = \{3\} \qquad \rho : \textsc{Var} \to \mathbb{M}1, \rho(x) = 3$

similar for $\mathbb{M}3$

$\overline{\rho}(\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat)) = \{1, 2, 3, \ldots\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}1$

$\overline{\rho}(\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat)) = \{1, 2, 3, \ldots\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}3$

$\overline{\rho}(\mathbb{s}(\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat))) = \{2, 3, \ldots\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}1$

$\overline{\rho}(\mathbb{s}(\exists x{:}Nat \,.\, x{:}Nat \wedge le(\mathbb{s}(\mathbb{0}), x{:}Nat))) = \{2, 3, \ldots\} \qquad \forall \rho : \textsc{Var} \to \mathbb{M}3$

Derived Constructs

$$\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \forall x{:}s' \,.\, \varphi \equiv \neg\exists x{:}s' \,.\, \neg\varphi$$

$$\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 \qquad \top_s \equiv \exists x{:}s \,.\, x{:}s$$

$$\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1) \qquad \bot_s \equiv \neg\top_s$$

Consequences:

$$\overline{\rho}(\varphi_1 \vee \varphi_2) = \overline{\rho}(\varphi_1) \cup \overline{\rho}(\varphi_2) \qquad \overline{\rho}(\forall x{:}s' \,.\, \varphi) = \bigcap_{a \in M_s} \overline{\rho[a/x]}(\varphi)$$

$$\overline{\rho}(\varphi_1 \rightarrow \varphi_2) = M_s \setminus (\overline{\rho}(\varphi_1) \setminus \overline{\rho}(\varphi_2)) \qquad \overline{\rho}(\top_s) = M_s$$

$$\overline{\rho}(\varphi_1 \leftrightarrow \varphi_2) = M_s \setminus (\overline{\rho}(\varphi_1) \triangle \overline{\rho}(\varphi_2)) \qquad \overline{\rho}(\bot_s) = \emptyset$$

Derived Constructs

$$\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \forall x{:}s' \,.\, \varphi \equiv \neg\exists x{:}s' \,.\, \neg\varphi$$

$$\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 \qquad \top_s \equiv \exists x{:}s \,.\, x{:}s$$

$$\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1) \qquad \bot_s \equiv \neg\top_s$$

Consequences:

$$\overline{\rho}(\varphi_1 \vee \varphi_2) = \overline{\rho}(\varphi_1) \cup \overline{\rho}(\varphi_2) \qquad \overline{\rho}(\forall x{:}s' \,.\, \varphi) = \bigcap_{a \in M_s} \overline{\rho[a/x]}(\varphi)$$

$$\overline{\rho}(\varphi_1 \rightarrow \varphi_2) = M_s \setminus (\overline{\rho}(\varphi_1) \setminus \overline{\rho}(\varphi_2)) \qquad \overline{\rho}(\top_s) = M_s$$

$$\overline{\rho}(\varphi_1 \leftrightarrow \varphi_2) = M_s \setminus (\overline{\rho}(\varphi_1) \triangle \overline{\rho}(\varphi_2)) \qquad \overline{\rho}(\bot_s) = \emptyset$$

Validity

$M \models \varphi$ (M satisfies φ): $\overline{\rho}(\varphi) = M_s$ for all $\rho : \textsc{Var} \to M$

$M \models F$, $F \subseteq \textsc{Pattern}$: $M \models \varphi$ for all $\varphi \in F$

φ valid, $\models \varphi$: $M \models \varphi$ for all models M

Examples:

$M \models \varphi_1 \wedge \varphi_2$ iff $M \models \varphi_1$ and $M \models \varphi_2$

$M \models \varphi_1 \to \varphi_2$ iff $\overline{\rho}(\varphi_1) \subseteq \overline{\rho}(\varphi_2)$ for all $\rho : \textsc{Var} \to M$

$M \models \forall x{:}s \,.\, \varphi$ iff $M \models \varphi$

$\models \varphi$, where φ is a propositional tautology

$\models \varphi_1$ and $\models \varphi_1 \to \varphi_2$ implies $\models \varphi_2$ (modus ponens)

$\models \forall x{:}s \,.\, \varphi \to \varphi[y{:}s/x{:}s]$, where $y{:}\, s \notin FV(\varphi)$ (substitution)

Validity

$M \models \varphi$ (M satisfies φ): $\overline{\rho}(\varphi) = M_s$ for all $\rho : \textsc{Var} \to M$

$M \models F$, $F \subseteq \textsc{Pattern}$: $M \models \varphi$ for all $\varphi \in F$

φ valid, $\models \varphi$: $M \models \varphi$ for all models M

Examples:

$M \models \varphi_1 \wedge \varphi_2$ iff $M \models \varphi_1$ and $M \models \varphi_2$

$M \models \varphi_1 \to \varphi_2$ iff $\overline{\rho}(\varphi_1) \subseteq \overline{\rho}(\varphi_2)$ for all $\rho : \textsc{Var} \to M$

$M \models \forall x{:}s \,.\, \varphi$ iff $M \models \varphi$

$\models \varphi$, where φ is a propositional tautology

$\models \varphi_1$ and $\models \varphi_1 \to \varphi_2$ implies $\models \varphi_2$ (modus ponens)

$\models \forall x{:}s \,.\, \varphi \to \varphi[y{:}s/x{:}s]$, where $y{:}\, s \notin FV(\varphi)$ (substitution)

Specifications

specification: (S, Σ, F)

(S, Σ, F)-model M: $M \models \varphi$ for all $\varphi \in F$

(S, Σ, F) entails φ: $M \models F$ implies $M \models \varphi$ for all models M

Examples:

$(S, \Sigma, \{\sigma(x_1{:}s_1, \ldots, x_n{:}s_n) = \top_s \vee \sigma(x_1{:}s_1, \ldots, x_n{:}s_n) = \bot_s\})$: M_σ is a predicate (holds or does not hold)

$(S, \Sigma, \{\exists\, y{:}s \,.\, \sigma(x{:}s) = y{:}s,\ \sigma(x{:}s) \wedge \sigma(y{:}s) \to \sigma(x{:}s \wedge y{:}s)\})$: M_σ is an injective function

$(S, \Sigma, \{\exists y{:}s' \,.\, \sigma(x{:}s) \to y{:}s'\})$: M_σ is a partial function

Specifications

specification: (S, Σ, F)

(S, Σ, F)-model M: $M \models \varphi$ for all $\varphi \in F$

(S, Σ, F) entails φ: $M \models F$ implies $M \models \varphi$ for all models M

Examples:

$(S, \Sigma, \{\sigma(x_1{:}s_1, \ldots, x_n{:}s_n) = \top_s \vee \sigma(x_1{:}s_1, \ldots, x_n{:}s_n) = \bot_s\})$: M_σ is a predicate (holds or does not hold)

$(S, \Sigma, \{\exists\, y{:}s \,.\, \sigma(x{:}s) = y{:}s,\ \sigma(x{:}s) \wedge \sigma(y{:}s) \rightarrow \sigma(x{:}s \wedge y{:}s)\})$: M_σ is an injective function

$(S, \Sigma, \{\exists y{:}s' \,.\, \sigma(x{:}s) \rightarrow y{:}s'\})$: M_σ is a partial function

Definedness

Motivation:

- How can we interpret patterns in a conventional, two-valued way?
- Are the patterns matching proper subsets of elements?
- How can we lift reasoning within syntactic category (sort) s_1 to syntactic category s_2?

Solution:

Consider specifications (S, Σ, F) such that, for any pair $(s_1, s_2) \in S \times S$,

- Σ includes a distinguished symbol $\lceil \_ \rceil^{s_2}_{s_1}$, called definedness
- F includes the axiom $\lceil x{:}s_1 \rceil^{s_2}_{s_1}$

Definedness

Motivation:

- How can we interpret patterns in a conventional, two-valued way?
- Are the patterns matching proper subsets of elements?
- How can we lift reasoning within syntactic category (sort) s_1 to syntactic category s_2?

Solution:

Consider specifications (S, Σ, F) such that, for any pair $(s_1, s_2) \in S \times S$,

- Σ includes a distinguished symbol $\lceil \_ \rceil^{s_2}_{s_1}$, called definedness
- F includes the axiom $\lceil x{:}s_1 \rceil^{s_2}_{s_1}$

Totality and Equality of Patterns

Motivation:

Since $\varphi \leftrightarrow \varphi'$ is not two-valued, it cannot capture the equality $\varphi = \varphi'$.

Solution:

$$\varphi =^{s_2}_{s_1} \varphi' \equiv \lfloor \varphi \leftrightarrow \varphi' \rfloor^{s_2}_{s_1}$$

where the totality symbol $\lfloor \_ \rfloor^{s_2}_{s_1}$ is the dual of the definedness:

$$\lfloor \varphi \rfloor^{s_2}_{s_1} \equiv \neg \lceil \neg \varphi \rceil^{s_2}_{s_1}$$

Explanation: (φ totally defined) $\equiv$ (it is not true that there are elements for which φ is not defined)

(φ is equal to φ' in s_2) $\equiv$ ($\varphi \leftrightarrow \varphi'$ is totally defined), i.e.,

$$M \models \varphi =^{s_2}_{s_1} \varphi' \text{ iff } M \models \varphi \leftrightarrow \varphi' \text{ for any model } M$$

Totality and Equality of Patterns

Motivation:
Since $\varphi \leftrightarrow \varphi'$ is not two-valued, it cannot capture the equality $\varphi = \varphi'$.

Solution:

$$\varphi =^{s_2}_{s_1} \varphi' \equiv \lfloor \varphi \leftrightarrow \varphi' \rfloor^{s_2}_{s_1}$$

where the totality symbol $\lfloor \_ \rfloor^{s_2}_{s_1}$ is the dual of the definedness:

$$\lfloor \varphi \rfloor^{s_2}_{s_1} \equiv \neg \lceil \neg \varphi \rceil^{s_2}_{s_1}$$

Explanation: (φ totally defined) $\equiv$ (it is not true that there are elements for which φ is not defined)
(φ is equal to φ' in s_2) $\equiv$ ($\varphi \leftrightarrow \varphi'$ is totally defined), i.e.,

$$M \models \varphi =^{s_2}_{s_1} \varphi' \text{ iff } M \models \varphi \leftrightarrow \varphi' \text{ for any model } M$$

Axiomatizing Membership and Subset

- membership: $x{:}s_1 \in^{s_2}_{s_1} \varphi \equiv \lceil x \wedge \varphi \rceil^{s_2}_{s_1}$
- inclusion: $\varphi \subseteq^{s_2}_{s_1} \varphi' \equiv \lfloor \varphi \rightarrow \varphi' \rfloor^{s_2}_{s_1}$

Axiomatizing Product Sorts

Let s_1, s_2 be two sorts in S. Then we can add the product sort $s_1 \otimes s_2$ to S with the following axiomatic definition:

Consider two auxiliary symbols:

$$\textit{pairing: } \langle \_, \_ \rangle : s_1 \times s_2 \to s_1 \otimes s_2, \text{ and}$$
$$\textit{projections: } \pi_i : s_1 \otimes s_2 \to s_i,\ i = 1, 2,$$

together with the follwing axioms:

$$\textit{injectivity: } \forall\, x_1, y_1 : s_1 \,.\, \forall\, x_2, y_2 : s_2 \,.\, \langle x_1, x_2 \rangle = \langle y_1, y_2 \rangle \to x_1 = y_1 \wedge x_2 = y_2$$
$$\textit{product: } \exists\, x_1 : s_1 \,.\, \exists\, x_2 : s_2 \,.\, \langle x_1, x_2 \rangle$$
$$\textit{diagram chasing } : \pi_i \langle x_1, x_2 \rangle = x_i,\ i = 1, 2$$

Axiomatizing Product Sorts

Let s_1, s_2 be two sorts in S. Then we can add the product sort $s_1 \otimes s_2$ to S with the following axiomatic definition:

Consider two auxiliary symbols:

$$\textit{pairing: } \langle \_, \_ \rangle : s_1 \times s_2 \to s_1 \otimes s_2, \text{ and}$$
$$\textit{projections: } \pi_i : s_1 \otimes s_2 \to s_i,\ i = 1, 2,$$

together with the follwing axioms:

$$\textit{injectivity: } \forall\, x_1, y_1 : s_1 \,.\, \forall\, x_2, y_2 : s_2 \,.\, \langle x_1, x_2 \rangle = \langle y_1, y_2 \rangle \to x_1 = y_1 \wedge x_2 = y_2$$
$$\textit{product: } \exists\, x_1 : s_1 \,.\, \exists\, x_2 : s_2 \,.\, \langle x_1, x_2 \rangle$$
$$\textit{diagram chasing } : \pi_i \langle x_1, x_2 \rangle = x_i,\ i = 1, 2$$

Plan

Motivation

Extend ML in order to

- be able to define inductive and coinductive types
- reason about the dynamic behaviour of programs

Fixed Points

Theorem (Knaster-Tarski)

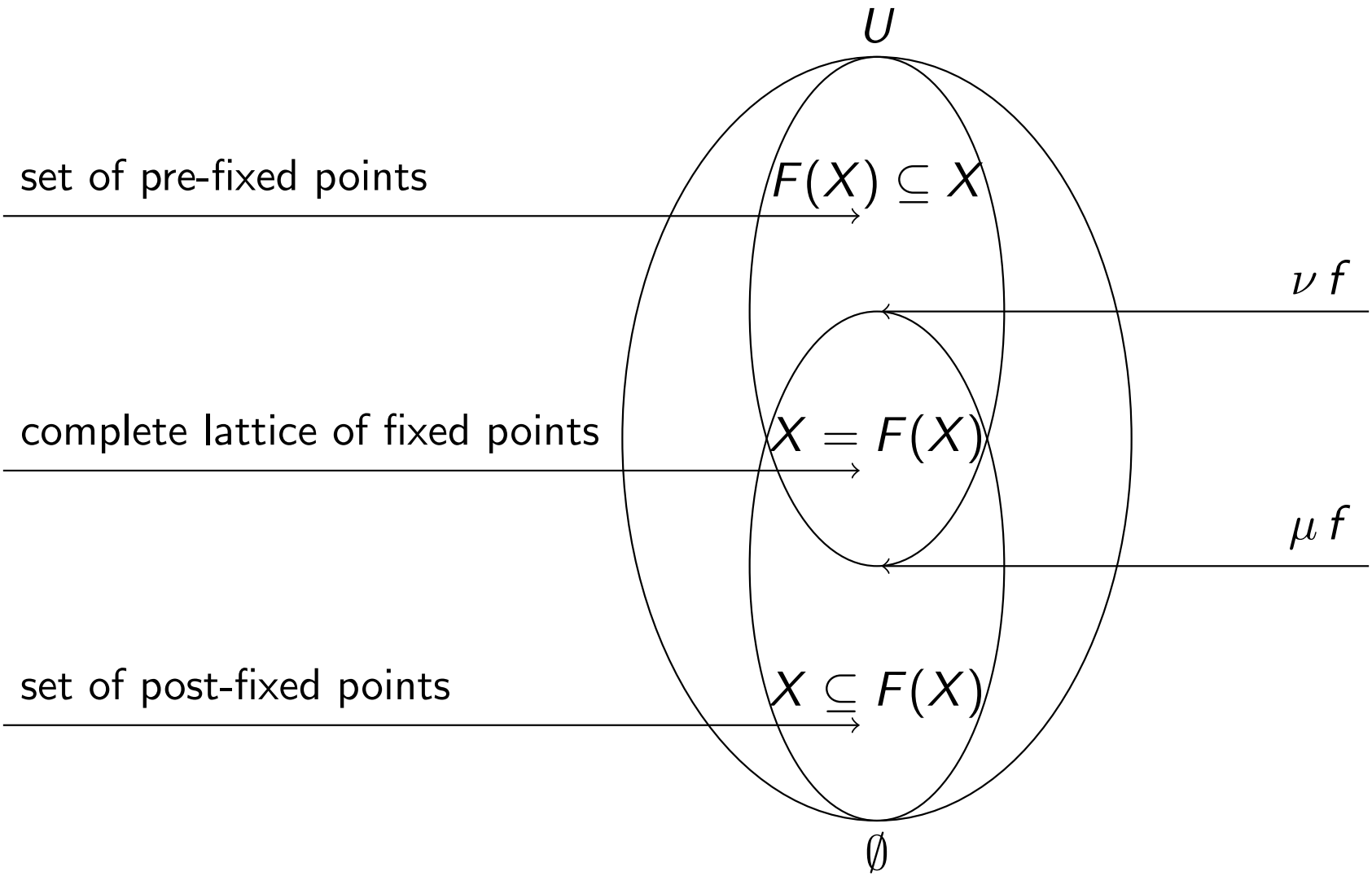
Let U be a set. Any $F : \mathcal{P}(U) \to \mathcal{P}(U)$ monotonic w.r.t. $\subseteq$ has

- *a least fixed-point $\mu\, y.\, F(y)$ (on short $\mu\, F$), and*
- *a greatest fixed-point $\nu\, y.\, F(y)$ (on short $\nu\, F$).*

Moreover,

- $\mu\, F = \bigcap \{X \mid F(X) \subseteq X\}$ *and*
- $\nu\, F = \bigcup \{X \mid X \subseteq F(X)\}$

Knaster-Tarski Theorem, Graphically



Patterns as Powerset Functionals

$\varphi \in \text{PATTERN}_s$, $\rho : \text{VAR} \to M$, $x \in FV(\varphi)$

Define $\mathcal{F}^{\rho}_{\varphi,x} : \mathcal{P}(M_s) \to \mathcal{P}(M_s)$ by

$$\mathcal{F}^{\rho}_{\varphi,x}(A) = \bigcup_{a \in A} \overline{\rho[a/x]}(\varphi)$$

Example: $\varphi = \mathbb{0} \vee \mathbb{s}(x), \rho : \text{VAR} \to \mathbb{N}$

$\mathcal{F}^{\rho}_{\varphi,x}(\emptyset) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \emptyset} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0,1\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0,1\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1, 2\}$

...

$\mathcal{F}^{\rho}_{\varphi,x}$ is monotonic, so it has a least fixed point $\mu\, \mathcal{F}^{\rho}_{\varphi,x}$ and a greatest fixed point $\nu\, \mathcal{F}^{\rho}_{\varphi,x}$.

Question: Can we define patterns matching the two fixed points?

Patterns as Powerset Functionals

$\varphi \in \textsc{Pattern}_s$, $\rho : \textsc{Var} \to M$, $x \in FV(\varphi)$

Define $\mathcal{F}^{\rho}_{\varphi,x} : \mathcal{P}(M_s) \to \mathcal{P}(M_s)$ by

$$\mathcal{F}^{\rho}_{\varphi,x}(A) = \bigcup_{a \in A} \overline{\rho[a/x]}(\varphi)$$

Example: $\varphi = \mathbb{0} \vee \mathbb{s}(x), \rho : \textsc{Var} \to \mathbb{N}$

$\mathcal{F}^{\rho}_{\varphi,x}(\emptyset) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \emptyset} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0,1\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0,1\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1, 2\}$

...

$\mathcal{F}^{\rho}_{\varphi,x}$ is monotonic, so it has a least fixed point $\mu\, \mathcal{F}^{\rho}_{\varphi,x}$ and a greatest fixed point $\nu\, \mathcal{F}^{\rho}_{\varphi,x}$.

Question: Can we define patterns matching the two fixed points?

Patterns as Powerset Functionals

$\varphi \in \textsc{Pattern}_s$, $\rho : \textsc{Var} \to M$, $x \in FV(\varphi)$

Define $\mathcal{F}^{\rho}_{\varphi,x} : \mathcal{P}(M_s) \to \mathcal{P}(M_s)$ by

$$\mathcal{F}^{\rho}_{\varphi,x}(A) = \bigcup_{a \in A} \overline{\rho[a/x]}(\varphi)$$

Example: $\varphi = \mathbb{0} \vee \mathbb{s}(x), \rho : \textsc{Var} \to \mathbb{N}$

$\mathcal{F}^{\rho}_{\varphi,x}(\emptyset) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \emptyset} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1\}$

$\mathcal{F}^{\rho}_{\varphi,x}(\{0,1\}) = \overline{\rho}(\mathbb{0}) \cup (\bigcup_{a \in \{0,1\}} \overline{\rho[a/x]}(\mathbb{s}(x))) = \{0, 1, 2\}$

...

$\mathcal{F}^{\rho}_{\varphi,x}$ is monotonic, so it has a least fixed point $\mu\, \mathcal{F}^{\rho}_{\varphi,x}$ and a greatest fixed point $\nu\, \mathcal{F}^{\rho}_{\varphi,x}$.

Question: Can we define patterns matching the two fixed points?

Matching μ-Logic (MmL), Formally 1/2

Variables: $\mathrm{VAR} = \mathrm{EVAR} \cup \mathrm{SVAR}$,
EVAR = element variables $x{:}s, y{:}s, z{:}s, \ldots$,
SVAR = set variables $X{:}s, Y{:}s, Z{:}s, \ldots$

Signatures: $\mathbb{\Sigma} = (S, \mathrm{VAR}, \Sigma)$

Patterns:

$$\varphi_s ::= x{:}s \mid X{:}s \mid \cdots$$
$$\mu X{:}s \,.\, \varphi_s \quad \text{if } \varphi_s \text{ is positive in } X{:}s$$

Matching μ-Logic (MmL), Formally 2/2

Models: similar to ML

Valuations: $\rho : \text{VAR} \to M$ such that $\rho(x{:}s) \in M_s$ for $x{:} \in \text{EVAR}$ and $\rho(X{:}s) \subseteq M_s$ for $X{:} \in \text{EVAR}$

$$\overline{\rho}(x{:}s) = \{\rho(x{:}s)\}$$
$$\overline{\rho}(X{:}s) = \rho(x{:}s)$$
$$\overline{\rho}(\mu\, X{:}s\,.\,\varphi) = \mu\, \mathcal{F}^{\rho}_{\varphi,X}$$

where $\mathcal{F}^{\rho}_{\varphi,X} : \mathcal{P}(M_s) \to \mathcal{P}M_s$, $\mathcal{F}^{\rho}_{\varphi,X}(A) = \overline{\rho[A/X](\varphi)}$

Derived pattern:

$$\nu\, X{:}s\,.\,\varphi \equiv \neg\mu\, X{:}s.\neg\varphi[\neg X{:}s/X{:}s]$$
$$\overline{\rho}(\nu\, X{:}s\,.\,\varphi) = \nu\, \mathcal{F}^{\rho}_{\varphi,X}$$

Matching μ-Logic (MmL), Formally 2/2

Models: similar to ML

Valuations: $\rho : \mathrm{VAR} \to M$ such that $\rho(x{:}s) \in M_s$ for $x{:} \in \mathrm{EVAR}$ and $\rho(X{:}s) \subseteq M_s$ for $X{:} \in \mathrm{EVAR}$

$$\overline{\rho}(x{:}s) = \{\rho(x{:}s)\}$$
$$\overline{\rho}(X{:}s) = \rho(x{:}s)$$
$$\overline{\rho}(\mu\, X{:}s\,.\,\varphi) = \mu\, \mathcal{F}^{\rho}_{\varphi,X}$$

where $\mathcal{F}^{\rho}_{\varphi,X} : \mathcal{P}(M_s) \to \mathcal{P}M_s$, $\mathcal{F}^{\rho}_{\varphi,X}(A) = \overline{\rho[A/X](\varphi)}$
Derived pattern:

$$\nu\, X{:}s\,.\,\varphi \equiv \neg\mu\, X{:}s.\neg\varphi[\neg X{:}s/X{:}s]$$
$$\overline{\rho}(\nu\, X{:}s\,.\,\varphi) = \nu\, \mathcal{F}^{\rho}_{\varphi,X}$$

Finite Lists in MmL 1/2

$S = \{Elt, List\}$, $\Sigma = \{nil, cons\}$, and F includes the following axioms:

nil, *cons* are functions:

$$nil : \rightarrow List \qquad\qquad cons : Elt \times List \rightarrow List$$

The above statements are sugar syntax for the axioms:

$$\exists y{:}List \,.\, nil = y{:}List$$
$$\exists z{:}List \,.\, cons(x{:}Elt), y{:}List) = z{:}List$$

Finite Lists in MmL 1/2

$S = \{Elt, List\}$, $\Sigma = \{nil, cons\}$, and F includes the following axioms:

nil, $cons$ are functions:

$$nil : \rightarrow List \qquad\qquad cons : Elt \times List \rightarrow List$$

The above statements are sugar syntax for the axioms:

$$\exists y{:}List \,.\, nil = y{:}List$$
$$\exists z{:}List \,.\, cons(x{:}Elt), y{:}List) = z{:}List$$

Finite Lists in MmL 2/2

No-junk, No-confusion (*nil* and *cons* are constructors for lists):

$$\mu\, L{:}List\,.\, nil \vee cons(Elt, L)$$
$$nil \neq cons(e, L)$$
$$\forall\, e, e'{:}Elt\,.\, \forall\, \ell, \ell'{:}List\,.\, cons(e, \ell) = cons(e', \ell') \rightarrow (e = e' \wedge \ell = \ell')$$

The above faithfully reflects the slogan

> *An inductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of finitely times the constructors.*

Finite Lists in MmL 2/2

No-junk, No-confusion (*nil* and *cons* are constructors for lists):

$$\mu\, L{:}List\,.\, nil \vee cons(Elt, L)$$
$$nil \neq cons(e, L)$$
$$\forall\, e, e'{:}Elt\,.\, \forall\, \ell, \ell'{:}List\,.\, cons(e, \ell) = cons(e', \ell') \rightarrow (e = e' \wedge \ell = \ell')$$

The above faithfully reflects the slogan

> *An inductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of finitely times the constructors.*

Streams in MmL

$S = \{Bit, Stream\}$, $\Sigma = \{\mathbb{0}, \mathbb{1}, \_::\_\}$, and F includes the axioms:
$\mathbb{0}, \mathbb{1}$ and $\_::\_$ are functions:

$$\mathbb{0} : \to Bit \quad \mathbb{1} : \to Bit$$
$$\_::\_ : Bit \times Stream \to Stream$$

Inductive definition of Bit: $\mathbb{0} \vee \mathbb{1}$ (equivalent to $\mu\, B{:}Bit\,.\, \mathbb{0} \vee \mathbb{1}$) and $\mathbb{0} \neq \mathbb{1}$

No junk for streams: $\nu\, S{:}Stream\,.\, Bit :: S$

No confusion for $\_::\_$:
$\forall b, b'{:}Bit\,.\, \forall\, t, t'{:}Stream\,.\, (b :: t = b' :: t' \to b = b' \wedge t = t')$

The above faithfully reflects the slogan

> *An coinductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of possible infinitely times the constructors.*

Streams in MmL

$S = \{Bit, Stream\}$, $\Sigma = \{\mathbb{0}, \mathbb{1}, \_::\_\}$, and F includes the axioms:
$\mathbb{0}, \mathbb{1}$ and $\_::\_$ are functions:

$$\mathbb{0} : \to Bit \quad \mathbb{1} : \to Bit$$
$$\_::\_ : Bit \times Stream \to Stream$$

Inductive definition of Bit: $\mathbb{0} \vee \mathbb{1}$ (equivalent to $\mu\, B{:}Bit\,.\, \mathbb{0} \vee \mathbb{1}$) and $\mathbb{0} \neq \mathbb{1}$

No junk for streams: $\nu\, S{:}Stream\,.\, Bit :: S$

No confusion for $\_::\_$:

$\forall b, b'{:}Bit\,.\, \forall\, t, t'{:}Stream\,.\, (b :: t = b' :: t' \to b = b' \wedge t = t')$

The above faithfully reflects the slogan

> *An coinductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of possible infinitely times the constructors.*

Streams in MmL

$S = \{Bit, Stream\}$, $\Sigma = \{\mathbb{0}, \mathbb{1}, \_::\_\}$, and F includes the axioms:
$\mathbb{0}, \mathbb{1}$ and $\_::\_$ are functions:

$$\mathbb{0} : \to Bit \quad \mathbb{1} : \to Bit$$
$$\_::\_ : Bit \times Stream \to Stream$$

Inductive definition of Bit: $\mathbb{0} \vee \mathbb{1}$ (equivalent to $\mu\, B{:}Bit\,.\, \mathbb{0} \vee \mathbb{1}$) and $\mathbb{0} \neq \mathbb{1}$

No junk for streams: $\nu\, S{:}Stream\,.\, Bit :: S$
No confusion for $\_::\_$:
$\forall b, b'{:}Bit\,.\, \forall\, t, t'{:}Stream\,.\, (b :: t = b' :: t' \to b = b' \wedge t = t')$
The above faithfully reflects the slogan

> *An coinductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of possible infinitely times the constructors.*

Streams in MmL

$S = \{Bit, Stream\}$, $\Sigma = \{\mathbb{0}, \mathbb{1}, \_::\_\}$, and F includes the axioms:
$\mathbb{0}, \mathbb{1}$ and $\_::\_$ are functions:

$$\mathbb{0} : \to Bit \quad \mathbb{1} : \to Bit$$
$$\_::\_ : Bit \times Stream \to Stream$$

Inductive definition of Bit: $\mathbb{0} \vee \mathbb{1}$ (equivalent to $\mu\, B{:}Bit\,.\, \mathbb{0} \vee \mathbb{1}$) and $\mathbb{0} \neq \mathbb{1}$

No junk for streams: $\nu\, S{:}Stream\,.\, Bit :: S$
No confusion for $\_::\_$:
$\forall b, b'{:}Bit\,.\, \forall\, t, t'{:}Stream\,.\, (b :: t = b' :: t' \to b = b' \wedge t = t')$
The above faithfully reflects the slogan

> *An coinductive type (seen as a set) contains exactly those elements that we obtain by repeatedly using of possible infinitely times the constructors.*

Plan

Motivation: Subsorts

$$Nat ::= plus(Nat, Nat)$$
$$Int ::= Nat \mid plus(Int, Int)$$

The inclusion $Int ::= Nat$ can be axiomatized by $\exists x{:}Nat \,.\, x \subseteq \exists x{:}Int \,.\, x$.

But we also want to axiomatize the fact that "any pattern of sort Nat is of sort Int as well; e.g. $plus(x{:}Nat, , y{:}Nat)$.
Moreover, a pattern $plus(x{:}Nat, y{:}Int)$ is ill-formed now.

A possible solution is to consider "injections":

$$inj : Nat \to Int$$

but their axiomatization is quite challenging.

Motivation: Subsorts

$$Nat ::= plus(Nat, Nat)$$
$$Int ::= Nat \mid plus(Int, Int)$$

The inclusion $Int ::= Nat$ can be axiomatized by $\exists x{:}Nat \,.\, x \subseteq \exists x{:}Int \,.\, x$.

But we also want to axiomatize the fact that "any pattern of sort Nat is of sort Int as well; e.g. $plus(x{:}Nat, , y{:}Nat)$.
Moreover, a pattern $plus(x{:}Nat, y{:}Int)$ is ill-formed now.

A possible solution is to consider "injections":

$$inj : Nat \to Int$$

but their axiomatization is quite challenging.

Motivation: Subsorts

$$Nat ::= plus(Nat, Nat)$$
$$Int ::= Nat \mid plus(Int, Int)$$

The inclusion $Int ::= Nat$ can be axiomatized by $\exists\, x{:}Nat\,.\,x \subseteq \exists\, x{:}Int\,.\,x$.

But we also want to axiomatize the fact that "any pattern of sort Nat is of sort Int as well; e.g. $plus(x{:}Nat, , y{:}Nat)$.
Moreover, a pattern $plus(x{:}Nat, y{:}Int)$ is ill-formed now.

A possible solution is to consider "injections":

$$inj : Nat \to Int$$

but their axiomatization is quite challenging.

Motivation: Parametric Sorts

$$List\langle S\rangle ::= nil \mid cons(S, List\langle S\rangle)$$

or

$$List\langle S\rangle ::= nil\langle S\rangle \mid cons(S, List\langle S\rangle)$$

where S ranges here over sorts. E.g., we may have the sorts $List\langle Nat\rangle$, $List\langle Int\rangle$,$List\langle List\langle Nat\rangle\rangle$, and so on.

We may also want to be able to define

$$Sorts = \mu X \,.\, Nat \vee Int \vee List\langle X\rangle$$

Motivation: Parametric Sorts

$$List\langle S\rangle ::= nil \mid cons(S, List\langle S\rangle)$$

or

$$List\langle S\rangle ::= nil\langle S\rangle \mid cons(S, List\langle S\rangle)$$

where S ranges here over sorts. E.g., we may have the sorts $List\langle Nat\rangle$, $List\langle Int\rangle$,$List\langle List\langle Nat\rangle\rangle$, and so on.

We may also want to be able to define

$$Sorts = \mu X \,.\, Nat \vee Int \vee List\langle X\rangle$$

Applicative Matching Logic, Formally 1/2

Signatures: $\mathbb{\Sigma} = (\text{EVAR}, \text{SVAR}, \Sigma)$

No sorts!!! (or you may think that there is just one universal sort $*$)

Σ contains only constant symbols (i.e., no arity).

And variables are not sorted.

Patterns:

$$\varphi_s ::= x \in \text{EVAR} \mid X \in \text{SVAR} \mid \sigma \in \Sigma$$

$\varphi_1\, \varphi_2$	(application)
$\bot$	(false)
$\varphi_1 \to \varphi_2$	(implication)
$\exists x \,.\, \varphi$	(binding)
$\mu X{:}s \,.\, \varphi_s$ if $f\, \varphi_s$ is positive in $X{:}s$	(l.f.p.)

Applicative Matching Logic, Formally 1/2

Signatures: $\mathbb{\Sigma} = (\text{EVAR}, \text{SVAR}, \Sigma)$

No sorts!!! (or you may think that there is just one universal sort $*$)

Σ contains only constant symbols (i.e., no arity).

And variables are not sorted.

Patterns:

$$\varphi_s ::= x \in \text{EVAR} \mid X \in \text{SVAR} \mid \sigma \in \Sigma$$

$\varphi_1\, \varphi_2$	(application)
$\bot$	(false)
$\varphi_1 \to \varphi_2$	(implication)
$\exists x \,.\, \varphi$	(binding)
$\mu X{:}s \,.\, \varphi_s$ if $f\,\varphi_s$ is positive in $X{:}s$	(l.f.p.)

Applicative Matching Logic, Formally 2/2

Σ-Model: $(M, \_\cdot\_, \{M_\sigma \mid \sigma \in \Sigma\})$, where
M is a non-empty set,
$\_\cdot\_ : M \times M \to \mathcal{P}(M)$
$M_\sigma \subseteq M$ for each $\sigma \in \Sigma$

Abreviation: $a\,b$ for $a \cdot b$.

Derived Patterns

$$\neg\varphi \equiv \varphi \to \bot \qquad \top \equiv \neg\bot \qquad \varphi_1 \vee \varphi_2 \equiv \neg\varphi_1 \to \varphi_2$$

$$\ldots$$

Definedness[1], totality, equality, membership, inclusion,... are defined in a similar way to ML.

[1] With a small amendament.

Applicative Matching Logic, Formally 2/2

Σ-Model: $(M, \_\cdot\_, \{M_\sigma \mid \sigma \in \Sigma\})$, where
M is a non-empty set,
$\_\cdot\_ : M \times M \to \mathcal{P}(M)$
$M_\sigma \subseteq M$ for each $\sigma \in \Sigma$

Abreviation: $a\ b$ for $a \cdot b$.

Derived Patterns

$$\neg\varphi \equiv \varphi \to \bot \qquad \top \equiv \neg\bot \qquad \varphi_1 \vee \varphi_2 \equiv \neg\varphi_1 \to \varphi_2$$

$\ldots$

Definedness[1], totality, equality, membership, inclusion,. . . are defined in a similar way to ML.

[1] With a small amendament.

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:
  non-empty sort: $[\![s]\!] \neq \bot$
  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$
  for each $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1\, \ldots\, x_n \subseteq [\![s]\!]$$

  or, equivalently,

$$\sigma\, [\![s_1]\!] \ldots [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:
  non-empty sort: $[\![s]\!] \neq \bot$
  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$
  for each $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1\, \ldots\, x_n \subseteq [\![s]\!]$$

  or, equivalently,

$$\sigma\, [\![s_1]\!] \ldots [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:

  non-empty sort: $[\![s]\!] \neq \bot$

  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$

  for each $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1 \ldots x_n \subseteq [\![s]\!]$$

  or, equivalently,

$$\sigma\, [\![s_1]\!] \ldots [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:
  non-empty sort: $[\![s]\!] \neq \bot$
  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$
  for each $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1\, \ldots\, x_n \subseteq [\![s]\!]$$

  or, equivalently,

$$\sigma\, [\![s_1]\!]\, \ldots\, [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:
  non-empty sort: $[\![s]\!] \neq \bot$
  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$
  for each $\sigma \in \Sigma_{s_1 \dots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1 \,\ldots\, x_n \subseteq [\![s]\!]$$

or, equivalently,

$$\sigma\, [\![s_1]\!] \,\ldots\, [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 1/2

An MmL-signature $\mathbb{\Sigma} = (\mathrm{EVAR}, \mathrm{SVAR}, S, \Sigma)$

can be encoded as an AML theory

$\mathbb{\Sigma}^{\mathrm{AML}} = (\mathrm{EVAR}^{\mathrm{AML}}, \mathrm{SVAR}^{\mathrm{AML}}, \Sigma^{\mathrm{AML}}, \Gamma^{\mathrm{AML}})$, where:

- $\mathrm{EVAR}^{\mathrm{AML}} = \{x \mid x{:}s \in \mathrm{EVAR}\}$, $\mathrm{SVAR}^{\mathrm{AML}} = \{X \mid X{:}s \in \mathrm{SVAR}\}$,
- $\Sigma^{\mathrm{AML}} = S \cup \Sigma \cup \{[\![\_]\!]\}$ (inhabitants symbol)
- Γ^{AML} includes:
  non-empty sort: $[\![s]\!] \neq \bot$
  each constant $s \in S$ is functional: $\exists\, y \,.\, s = y$
  for each $\sigma \in \Sigma_{s_1 \cdots s_n, s}$ an axiom

$$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \rightarrow \sigma\, x_1\, \ldots\, x_n \subseteq [\![s]\!]$$

or, equivalently,

$$\sigma\, [\![s_1]\!]\, \ldots\, [\![s_n]\!] \subseteq [\![s]\!]$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma\, a_1\ \ldots\ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists\, x{:}s\,.\,\varphi$ by $\exists\, x\,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu\, X{:}s\,.\,\varphi$ by $\mu\, X\,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma a_1 \ \ldots \ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists\, x{:}s \,.\, \varphi$ by $\exists\, x \,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu X{:}s \,.\, \varphi$ by $\mu X \,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma a_1 \ \ldots \ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists x{:}s \,.\, \varphi$ by $\exists x \,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu X{:}s \,.\, \varphi$ by $\mu X \,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma a_1 \ \ldots \ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists\, x{:}s \,.\, \varphi$ by $\exists\, x \,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu\, X{:}s \,.\, \varphi$ by $\mu\, X \,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma a_1 \ \ldots \ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists\, x{:}s \,.\, \varphi$ by $\exists\, x \,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu\, X{:}s \,.\, \varphi$ by $\mu\, X \,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Encoding of MmL in AML 2/2

Each Σ^{AML}-model M defines a Σ-model M' as follows:

- $M'_s = M_{[\![s]\!]}$
- $M'_\sigma(a_1, \ldots, a_n) = M_\sigma a_1 \ \ldots \ a_n$

Each Σ-pattern φ can be encoded as an Σ^{AML}-pattern φ^{AML}:

- $x{:}s$ by $x \wedge x \in [\![s]\!]$
- $\exists\, x{:}s \,.\, \varphi$ by $\exists\, x \,.\, x \in [\![s]\!] \wedge \varphi$
- $\mu\, X{:}s \,.\, \varphi$ by $\mu\, X \,.\, X \subseteq [\![s]\!] \wedge \varphi$
- the rest is straigthforward

We have

$$M' \models \varphi \ \text{ iff } \ M \models \varphi^{\mathrm{AML}}$$

Subsorting and Overloading

$$Nat ::= plus(Nat, Nat)$$
$$Int ::= Nat \mid plus(Int, Int)$$

$\Sigma = \{Nat, Int, plus\}$
Γ:
$[\![Nat]\!] \subseteq [\![Int]\!]$
$\exists\, z \,.\, plus\, x\, y = z$
$plus\, [\![Nat]\!]\, [\![Nat]\!] \subseteq [\![Nat]\!]$
$plus\, [\![Int]\!]\, [\![Int]\!] \subseteq [\![Int]\!]$

Plan

Induction Principle

Complete Lattices	MmL
$\dfrac{F(X) \subseteq X}{\mu F \subseteq X}$	$\dfrac{\varphi[\psi/X] \to \psi}{\mu X \,.\, \varphi \to \psi}$ [Knaster-Tarski]

$$M \models \varphi[\psi/X] \to \psi \text{ iff } \overline{\rho}(\varphi[\psi/X]) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$
$$\text{iff } \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$

since $\overline{\rho}(\varphi[\psi/X]) = \overline{\rho[\rho(\psi)/X]}(\varphi) = \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi))$.

$\mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi)$ means that $\overline{\rho}(\psi)$ is a *pre-fixed point* of $\mathcal{F}^{\rho}_{\varphi,X}$.

Induction Principle

Complete Lattices	MmL
$\dfrac{F(X) \subseteq X}{\mu F \subseteq X}$	$\dfrac{\varphi[\psi/X] \to \psi}{\mu X \,.\, \varphi \to \psi}$ [Knaster-Tarski]

$$M \models \varphi[\psi/X] \to \psi \text{ iff } \overline{\rho}(\varphi[\psi/X]) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$
$$\text{iff } \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$

since $\overline{\rho}(\varphi[\psi/X]) = \overline{\rho[\rho(\psi)/X]}(\varphi) = \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi))$.

$\mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi)$ means that $\overline{\rho}(\psi)$ is a *pre-fixed point* of $\mathcal{F}^{\rho}_{\varphi,X}$.

Induction Principle

Complete Lattices	MmL
$\dfrac{F(X) \subseteq X}{\mu F \subseteq X}$	$\dfrac{\varphi[\psi/X] \to \psi}{\mu X \,.\, \varphi \to \psi}$ [Knaster-Tarski]

$$M \models \varphi[\psi/X] \to \psi \text{ iff } \overline{\rho}(\varphi[\psi/X]) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$
$$\text{iff } \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi) \text{ for any } \rho : \text{Var} \to M$$

since $\overline{\rho}(\varphi[\psi/X]) = \overline{\rho[\rho(\psi)/X]}(\varphi) = \mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi))$.

$\mathcal{F}^{\rho}_{\varphi,X}(\overline{\rho}(\psi)) \subseteq \overline{\rho}(\psi)$ means that $\overline{\rho}(\psi)$ is a *pre-fixed point* of $\mathcal{F}^{\rho}_{\varphi,X}$.

Induction on Finite Lists

Σ: *Sorts*, *Nat*, *Int*, *List*, *nil*, *cons*

Γ:

$\exists\, y \,.\, Nat = y$, $\exists\, y \,.\, Int = y$, $\forall\, s \,.\, \exists\, y \,.\, List\ s = y$

$\exists\, y \,.\, nil = y$, $\forall\, x \,.\, \forall\, \ell \,.\, \exists\, y \,.\, cons\ x\ \ell = y$

$Sorts = \mu\, S \,.\, Int \lor Nat \lor List\ S$

$\forall\, s \,.\, [\![List\ s]\!] = \mu\, L \,.\, nil \lor cons\ [\![s]\!]\ L$

Theorem

$$F \models (nil \in P \land cons\ [\![s]\!]\ P \subseteq P) \to [\![List\ s]\!] \subseteq P \qquad (\spadesuit)$$

Sugar syntax: $cons(e, \ell) \equiv cons\ e\ \ell$

Induction on Finite Lists

Σ: $\mathit{Sorts}, \mathit{Nat}, \mathit{Int}, \mathit{List}, \mathit{nil}, \mathit{cons}$

Γ:
$\exists\, y \,.\, \mathit{Nat} = y$, $\exists\, y \,.\, \mathit{Int} = y$, $\forall\, s \,.\, \exists\, y \,.\, \mathit{List}\, s = y$
$\exists\, y \,.\, \mathit{nil} = y$, $\forall\, x \,.\, \forall\, \ell \,.\, \exists\, y \,.\, \mathit{cons}\, x\, \ell = y$
$\mathit{Sorts} = \mu\, S \,.\, \mathit{Int} \vee \mathit{Nat} \vee \mathit{List}\, S$
$\forall\, s \,.\, [\![\mathit{List}\, s]\!] = \mu\, L \,.\, \mathit{nil} \vee \mathit{cons}\, [\![s]\!]\, L$

Theorem

$$F \models (\mathit{nil} \in P \wedge \mathit{cons}\, [\![s]\!]\, P \subseteq P) \rightarrow [\![\mathit{List}\, s]\!] \subseteq P \qquad (\spadesuit)$$

Sugar syntax: $\mathit{cons}(e, \ell) \equiv \mathit{cons}\, e\, \ell$

Induction on Finite Lists

Σ: $Sorts, Nat, Int, List, nil, cons$

Γ:
$\exists\, y \,.\, Nat = y$, $\exists\, y \,.\, Int = y$, $\forall\, s \,.\, \exists\, y \,.\, List\ s = y$
$\exists\, y \,.\, nil = y$, $\forall\, x \,.\, \forall\, \ell \,.\, \exists\, y \,.\, cons\ x\ \ell = y$
$Sorts = \mu\, S \,.\, Int \vee Nat \vee List\ S$
$\forall\, s \,.\, [\![List\ s]\!] = \mu\, L \,.\, nil \vee cons\ [\![s]\!]\ L$

Theorem

$$F \models (nil \in P \wedge cons\ [\![s]\!]\ P \subseteq P) \rightarrow [\![List\ s]\!] \subseteq P \qquad (\spadesuit)$$

Sugar syntax: $cons(e, \ell) \equiv cons\ e\ \ell$

Proof of $rev(rev(\ell)) = \ell$

Specification of *app* and *rev*:

$$\forall\, e{:}Elt\,.\, app(nil, e) = cons(e, nil)$$
$$\forall\, e, e'{:}Elt\,.\,\forall\, \ell{:}List\,.\, app(cons(e, \ell), e') = cons(e, app(\ell, e'))$$
$$rev(nil) = nil$$
$$\forall\, e{:}Elt\,.\,\forall\, \ell{:}List\,.\, rev(cons(e, \ell)) = app(rev(\ell), e)$$

Since

$$F \models \forall\, \ell{:}List\,.\, rev(rev(\ell)) = \ell \text{ iff}$$
$$F \models \forall\, \ell{:}List\,.\, \ell \in \exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

we may use the list coinduction principle considering

$$\exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

as instance of P.

Proof of $rev(rev(\ell)) = \ell$

Specification of *app* and *rev*:

$$\forall\, e{:}Elt\,.\, app(nil, e) = cons(e, nil)$$
$$\forall\, e, e'{:}Elt\,.\,\forall\, \ell{:}List\,.\, app(cons(e, \ell), e') = cons(e, app(\ell, e'))$$
$$rev(nil) = nil$$
$$\forall\, e{:}Elt\,.\,\forall\, \ell{:}List\,.\, rev(cons(e, \ell)) = app(rev(\ell), e)$$

Since

$$F \models \forall\, \ell{:}List\,.\, rev(rev(\ell)) = \ell \text{ iff}$$
$$F \models \forall\, \ell{:}List\,.\, \ell \in \exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

we may use the list coinduction principle considering

$$\exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

as instance of P.

Proof of $rev(rev(\ell)) = \ell$

Specification of *app* and *rev*:

$$\forall\, e{:}Elt\,.\, app(nil, e) = cons(e, nil)$$
$$\forall\, e, e'{:}Elt\,.\, \forall\, \ell{:}List\,.\, app(cons(e, \ell), e') = cons(e, app(\ell, e'))$$
$$rev(nil) = nil$$
$$\forall\, e{:}Elt\,.\, \forall\, \ell{:}List\,.\, rev(cons(e, \ell)) = app(rev(\ell), e)$$

Since

$$F \models \forall\, \ell{:}List\,.\, rev(rev(\ell)) = \ell \text{ iff}$$
$$F \models \forall\, \ell{:}List\,.\, \ell \in \exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

we may use the list coinduction principle considering

$$\exists\, \ell'{:}List\,.\, \ell' \wedge rev(rev(\ell')) = \ell'$$

as instance of P.

Mutual Inductive Types

- Σ: $Even, Odd, \mathbb{0}, \mathbb{s}$
- Γ:
  $\mathbb{0} : \rightarrow Even$, $\mathbb{s} : Even \rightarrow Odd$, $\mathbb{s} : Odd \rightarrow Even$,
  $\forall x . \mathbb{0} \neq \mathbb{s}\, x$,
  $[\![Even]\!] = ??$
  $[\![Odd]\!] = ??$

So, the question is how to specify $[\![Even]\!]$ and $[\![Odd]\!]$?

They have to satisfy the equalities

$$[\![Even]\!] = \mathbb{0} \vee \mathbb{s}\, [\![Odd]\!]$$
$$[\![Odd]\!] = \mathbb{s}\, [\![Even]\!]$$

and

both of them include exactly those elements that we obtain by repeatedly using of finitely times the corresponding constructors.

Mutual Inductive Types

- Σ: $\mathit{Even}, \mathit{Odd}, \mathbb{0}, \mathbb{s}$
- Γ:
  $\mathbb{0} : \rightarrow \mathit{Even}$, $\mathbb{s} : \mathit{Even} \rightarrow \mathit{Odd}$, $\mathbb{s} : \mathit{Odd} \rightarrow \mathit{Even}$,
  $\forall x \,.\, \mathbb{0} \neq \mathbb{s}\, x$,
  $[\![\mathit{Even}]\!] = ??$
  $[\![\mathit{Odd}]\!] = ??$

So, the question is how to specify $[\![\mathit{Even}]\!]$ and $[\![\mathit{Odd}]\!]$?
They have to satisfy the equalities

$$[\![\mathit{Even}]\!] = \mathbb{0} \vee \mathbb{s}\,[\![\mathit{Odd}]\!]$$
$$[\![\mathit{Odd}]\!] = \mathbb{s}\,[\![\mathit{Even}]\!]$$

and

> *both of them include exactly those elements that we obtain by repeatedly using of finitely times the corresponding constructors.*

A possible answer

- $\Sigma \cup \{Even \otimes Odd, \langle \_, \_ \rangle, \pi_1, \pi_2\}$
- $\Gamma \cup$
  $\langle \_, \_ \rangle : Even \times Odd \to Even \otimes Odd$,
  $\langle x_1, x_2 \rangle = \langle y_1, y_2 \rangle \to x_1 = y_1 \wedge x_2 = y_2$
  $\pi_1 : Even \otimes Odd \to Even$, $\pi_2 : Even \otimes Odd \to Odd$,
  $\pi_i \ \langle x_1, x_2 \rangle = x_i$, $i = 1, 2$,

$[\![Even \otimes Odd]\!] = \mu X \,.\, \langle \mathbb{0}, \mathbb{s}\, \mathbb{0} \rangle \vee \langle \mathbb{s}\, \pi_2\, X, \mathbb{s}\, \pi_1\, X \rangle$,

Since

$$[\![Even \otimes Odd]\!] = [\![Even]\!] \times [\![Odd]\!]$$

we obtain

$[\![Even]\!] = \pi_1\, [\![Even \otimes Odd]\!]$, $[\![Odd]\!] = \pi_2\, [\![Even \otimes Odd]\!]$

A possible answer

- $\Sigma \cup \{Even \otimes Odd, \langle \_, \_ \rangle, \pi_1, \pi_2\}$
- $\Gamma \cup$
  $\langle \_, \_ \rangle : Even \times Odd \to Even \otimes Odd$,
  $\langle x_1, x_2 \rangle = \langle y_1, y_2 \rangle \to x_1 = y_1 \wedge x_2 = y_2$
  $\pi_1 : Even \otimes Odd \to Even$, $\pi_2 : Even \otimes Odd \to Odd$,
  $\pi_i \ \langle x_1, x_2 \rangle = x_i$, $i = 1, 2$,

$$[\![Even \otimes Odd]\!] = \mu X . \langle \mathbb{0}, \mathbb{s}\, \mathbb{0} \rangle \vee \langle \mathbb{s}\, \pi_2\, X, \mathbb{s}\, \pi_1\, X \rangle,$$

Since

$$[\![Even \otimes Odd]\!] = [\![Even]\!] \times [\![Odd]\!]$$

we obtain

$$[\![Even]\!] = \pi_1\, [\![Even \otimes Odd]\!],\ [\![Odd]\!] = \pi_2\, [\![Even \otimes Odd]\!]$$

A possible answer

- $\Sigma \cup \{Even \otimes Odd, \langle \_, \_ \rangle, \pi_1, \pi_2\}$
- $\Gamma \cup$
  $\langle \_, \_ \rangle : Even \times Odd \to Even \otimes Odd$,
  $\langle x_1, x_2 \rangle = \langle y_1, y_2 \rangle \to x_1 = y_1 \wedge x_2 = y_2$
  $\pi_1 : Even \otimes Odd \to Even$, $\pi_2 : Even \otimes Odd \to Odd$,
  $\pi_i \ \langle x_1, x_2 \rangle = x_i$, $i = 1, 2$,

$[\![Even \otimes Odd]\!] = \mu X . \langle \mathbb{0}, \mathbb{s}\, \mathbb{0} \rangle \vee \langle \mathbb{s}\, \pi_2\, X, \mathbb{s}\, \pi_1\, X \rangle$,

Since

$$[\![Even \otimes Odd]\!] = [\![Even]\!] \times [\![Odd]\!]$$

we obtain

$[\![Even]\!] = \pi_1\, [\![Even \otimes Odd]\!]$, $[\![Odd]\!] = \pi_2\, [\![Even \otimes Odd]\!]$

Induction Principle for Mutual Inductive Types

Now we obtained for free induction principle for *Even* and *Odd*:

$$(\langle \mathbb{0}, \mathbb{s}(\mathbb{0}) \rangle \in P \wedge \mathbb{s}\, P \subseteq P) \rightarrow \langle [\![Even]\!], [\![Odd]\!] \rangle \subseteq P$$

where $\mathbb{s}\,\langle x, y \rangle = \langle \mathbb{s}\, x, \mathbb{s}\, y \rangle$

For instance, the proof of $\forall\, x\,.\, x \in [\![Even]\!] \rightarrow \mathbb{s}\,\mathbb{s}\, x \in [\![Even]\!]$ should be proved in parallel with $\forall\, y\,.\, y \in [\![Odd]\!] \rightarrow \mathbb{s}\,\mathbb{s}\, y \in [\![Odd]\!]$, i.e., we have to prove by induction that

$$\Gamma \models \langle \exists\, x\,.\, x \wedge (x \in [\![Even]\!] \rightarrow \mathbb{s}\,\mathbb{s}\, x \in [\![Even]\!]),$$
$$\exists\, y\,.\, y \wedge (y \in [\![Odd]\!] \rightarrow \mathbb{s}\,\mathbb{s}\, y \in [\![Odd]\!]) \rangle$$

Induction Principle for Mutual Inductive Types

Now we obtained for free induction principle for *Even* and *Odd*:

$$(\langle \mathbb{0}, \mathbb{s}(\mathbb{0}) \rangle \in P \wedge \mathbb{s}\, P \subseteq P) \rightarrow \langle [\![Even]\!], [\![Odd]\!] \rangle \subseteq P$$

where $\mathbb{s}\, \langle x, y \rangle = \langle \mathbb{s}\, x, \mathbb{s}\, y \rangle$
For instance, the proof of $\forall\, x\, .\, x \in [\![Even]\!] \rightarrow \mathbb{s}\, \mathbb{s}\, x \in [\![Even]\!]$ should be proved in parallel with $\forall\, y\, .\, y \in [\![Odd]\!] \rightarrow \mathbb{s}\, \mathbb{s}\, y \in [\![Odd]\!]$, i.e., we have to prove by induction that

$$\Gamma \models \langle \exists\, x\, .\, x \wedge (x \in [\![Even]\!] \rightarrow \mathbb{s}\, \mathbb{s}\, x \in [\![Even]\!]),$$
$$\exists\, y\, .\, y \wedge (y \in [\![Odd]\!] \rightarrow \mathbb{s}\, \mathbb{s}\, y \in [\![Odd]\!]) \rangle$$

Plan

Coinduction Principle

Complete Lattices

$$\frac{X \subseteq F(X)}{X \subseteq \nu F}$$

In order to prove that $x \in \nu F$:

1. find a subset X;
2. show that X is a post-fixed point: $X \subseteq F(X)$;
3. show that $x \in X$.

MmL

$$\frac{\psi \to \varphi[\psi/X]}{\psi \to \nu X . \varphi} \text{ [Knaster-Tarski]}$$

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point": $F \models \psi' \to \varphi[\psi'/X]$;
3. show that $F \models \psi \to \psi'$.

This can be expressed in MmL by the following

Theorem

$$F \models (P \to P' \wedge P' \to \varphi[P'/X]) \to (P \to \nu X . \varphi)$$

Coinduction Principle

Complete Lattices

$$\frac{X \subseteq F(X)}{X \subseteq \nu F}$$

In order to prove that $x \in \nu F$:

1. find a subset X;
2. show that X is a post-fixed point: $X \subseteq F(X)$;
3. show that $x \in X$.

MmL

$$\frac{\psi \to \varphi[\psi/X]}{\psi \to \nu X . \varphi} \text{ [Knaster-Tarski]}$$

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point": $F \models \psi' \to \varphi[\psi'/X]$;
3. show that $F \models \psi \to \psi'$.

This can be expressed in MmL by the following

Theorem

$$F \models (P \to P' \wedge P' \to \varphi[P'/X]) \to (P \to \nu X . \varphi)$$

Coinduction Principle

Complete Lattices

$$\frac{X \subseteq F(X)}{X \subseteq \nu F}$$

In order to prove that $x \in \nu F$:

1. find a subset X;
2. show that X is a post-fixed point: $X \subseteq F(X)$;
3. show that $x \in X$.

MmL

$$\frac{\psi \to \varphi[\psi/X]}{\psi \to \nu X . \varphi} \text{ [Knaster-Tarski]}$$

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point": $F \models \psi' \to \varphi[\psi'/X]$;
3. show that $F \models \psi \to \psi'$.

This can be expressed in MmL by the following

Theorem

$$F \models (P \to P' \wedge P' \to \varphi[P'/X]) \to (P \to \nu X . \varphi)$$

Coinduction Principle

Complete Lattices

$$\frac{X \subseteq F(X)}{X \subseteq \nu F}$$

In order to prove that $x \in \nu F$:

1. find a subset X;
2. show that X is a post-fixed point: $X \subseteq F(X)$;
3. show that $x \in X$.

MmL

$$\frac{\psi \to \varphi[\psi/X]}{\psi \to \nu X . \varphi} \text{ [Knaster-Tarski]}$$

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point": $F \models \psi' \to \varphi[\psi'/X]$;
3. show that $F \models \psi \to \psi'$.

This can be expressed in MmL by the following

Theorem

$$F \models (P \to P' \wedge P' \to \varphi[P'/X]) \to (P \to \nu X . \varphi)$$

Coinduction Principle on Streams

Σ: $Bit, Stream$

Γ: $[\![Stream]\!] = \nu\, S\, .\, [\![Bit]\!] :: S$

In order to prove that $F \models \psi \rightarrow [\![Stream]\!]$:

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point":
   $F \models \psi' \rightarrow [\![Bit]\!] :: \psi'$;
3. show that $F \models \psi \rightarrow \psi'$.

This can be expressed by the following

Theorem

$$F \models (P \subseteq P' \wedge P' \subseteq [\![Bit]\!] :: P') \rightarrow (P \subseteq [\![Stream]\!])$$

Coinduction Principle on Streams

Σ: $Bit, Stream$

Γ: $[\![Stream]\!] = \nu\, S\, .\, [\![Bit]\!] :: S$

In order to prove that $F \models \psi \rightarrow [\![Stream]\!]$:

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point":
   $F \models \psi' \rightarrow [\![Bit]\!] :: \psi'$;
3. show that $F \models \psi \rightarrow \psi'$.

This can be expressed by the following

Theorem

$$F \models (P \subseteq P' \wedge P' \subseteq [\![Bit]\!] :: P') \rightarrow (P \subseteq [\![Stream]\!])$$

Equality on Streams, Coinductively

$$BEQ_{Stream} \equiv \nu\, R{:}Stream \otimes Stream \,.\, [\![Bit]\!] :: R$$

where $b :: \langle s_1, s_2 \rangle = \langle b :: s_1, b :: s_2 \rangle$

We have

$$F \models \forall s_1, s_2{:}Stream \,.\, s_1 = s_2 \text{ iff } \langle s_1, s_2 \rangle \in BEQ_{Stream}$$

Mutual Coinductive Types

- Σ: $Tree, EList, Elt, nil, cons, node$
- Γ:
  $\exists\, y\,.\, EList = y$, $\exists\, y\,.\, Tree = y$, $\exists\, y\,.\, Elt = y$, $nil : \rightarrow [\![EList]\!]$,
  $cons : [\![Tree]\!] \times [\![EList]\!] \rightarrow [\![EList]\!]$,
  $node : [\![Elt]\!] \times [\![EList]\!] \rightarrow [\![Tree]\!]$,
  $[\![EList]\!] = ??$
  $[\![Tree]\!] = ??$

How to specify $[\![EList]\!]$ and $[\![Tree]\!]$?

We want they satisfy the equalities

$$[\![EList]\!] = nil \vee cons\, [\![Tree]\!]\, [\![EList]\!]$$
$$[\![Tree]\!] = node\, [\![Elt]\!]\, [\![EList]\!]$$

and

both of them include exactly those elements that we obtain by repeatedly using of possible infinitely times the corresponding constructors.

Mutual Coinductive Types

- Σ: *Tree*, *EList*, *Elt*, *nil*, *cons*, *node*
- Γ:
  $\exists\, y\,.\, EList = y$, $\exists\, y\,.\, Tree = y$, $\exists\, y\,.\, Elt = y$, $nil : \rightarrow [\![EList]\!]$,
  $cons : [\![Tree]\!] \times [\![EList]\!] \rightarrow [\![EList]\!]$,
  $node : [\![Elt]\!] \times [\![EList]\!] \rightarrow [\![Tree]\!]$,
  $[\![EList]\!] = ??$
  $[\![Tree]\!] = ??$

How to specify $[\![EList]\!]$ and $[\![Tree]\!]$?
We want they satisfy the equalities

$$[\![EList]\!] = nil \vee cons\, [\![Tree]\!]\, [\![EList]\!]$$
$$[\![Tree]\!] = node\, [\![Elt]\!]\, [\![EList]\!]$$

and

> *both of them include exactly those elements that we obtain by repeatedly using of possible infinitely times the corresponding constructors.*

A Possible Solution

- Σ: $\mathit{Tree} \otimes \mathit{EList}$, $\langle \_, \_ \rangle$, π_1, π_2
- Γ:
  $\langle \_, \_ \rangle : \mathit{Tree} \times \mathit{EList} \to \mathit{Tree} \otimes \mathit{EList}$,
  $\pi_1 : \mathit{Tree} \otimes \mathit{EList} \to \mathit{Tree}$, $\pi_2 : \mathit{Tree} \otimes \mathit{EList} \to \mathit{EList}$,
  $\exists\, y \,.\, \langle x_1, x_2 \rangle = y$, $\exists\, y \,.\, \pi_i(x) = y$, $i = 1, 2$,
  $\pi_i(\langle x_1, x_2 \rangle) = x_i$, $i = 1, 2$,

$$[\![\mathit{Tree} \otimes \mathit{EList}]\!] = \nu\, X \,.\, \langle \mathit{node}\, [\![\mathit{Elt}]\!]\, \pi_2\, X, \mathit{nil} \vee \mathit{cons}\, (\pi_1\, X)\, \pi_2\, X \rangle$$

We have

$$[\![\mathit{Tree} \otimes \mathit{EList}]\!] = \langle [\![\mathit{Tree}]\!], [\![\mathit{EList}]\!] \rangle$$

i.e. $[\![\mathit{Tree}]\!] = \pi_1([\![\mathit{Tree} \otimes \mathit{EList}]\!])$ and $[\![\mathit{EList}]\!] = \pi_2([\![\mathit{Tree} \otimes \mathit{EList}]\!])$.

A Possible Solution

- Σ: $\mathit{Tree} \otimes \mathit{EList}$, $\langle \_, \_ \rangle$, π_1, π_2
- Γ:
  $\langle \_, \_ \rangle : \mathit{Tree} \times \mathit{EList} \to \mathit{Tree} \otimes \mathit{EList}$,
  $\pi_1 : \mathit{Tree} \otimes \mathit{EList} \to \mathit{Tree}$, $\pi_2 : \mathit{Tree} \otimes \mathit{EList} \to \mathit{EList}$,
  $\exists\, y \,.\, \langle x_1, x_2 \rangle = y$, $\exists\, y \,.\, \pi_i(x) = y$, $i = 1, 2$,
  $\pi_i(\langle x_1, x_2 \rangle) = x_i$, $i = 1, 2$,

$$[\![\mathit{Tree} \otimes \mathit{EList}]\!] = \nu\, X \,.\, \langle \mathit{node}\, [\![\mathit{Elt}]\!]\, \pi_2\, X, \mathit{nil} \vee \mathit{cons}\, (\pi_1\, X)\, \pi_2\, X \rangle$$

We have

$$[\![\mathit{Tree} \otimes \mathit{EList}]\!] = \langle [\![\mathit{Tree}]\!], [\![\mathit{EList}]\!] \rangle$$

i.e. $[\![\mathit{Tree}]\!] = \pi_1([\![\mathit{Tree} \otimes \mathit{EList}]\!])$ and $[\![\mathit{EList}]\!] = \pi_2([\![\mathit{Tree} \otimes \mathit{EList}]\!])$.

Equality, Coinductively

$$BEQ_{Tree \otimes EList} =$$

$$\nu\, R{:}(Tree \otimes Tree) \otimes (EList \otimes EList)\,.$$
$$\langle node\, [\![Elt]\!]\, (\pi_2\, R), \langle nil, nil\rangle \vee cons\, (\pi_1\, R)\, (\pi_2\, R)\rangle$$

where $node\, x\, \langle \ell_1, \ell_2\rangle = \langle node\, x\, \ell_1, node\, x\, \ell_2\rangle$ and
$cons\, \langle t_1, t_2\rangle\, \langle \ell_1, \ell_2\rangle = \langle cons\, t_1\, \ell_1, cons\, t_2\, \ell_2\rangle$

We have

$$F \models \forall\, t_1, t_2{:}Tree\,.\, t_1 = t_2 \leftrightarrow \langle t_1, t_2\rangle \in \pi_1\, BEQ_{Tree \otimes EList}$$
$$F \models \forall\, \ell_1, \ell_2{:}EList\,.\, \ell_1 = \ell_2 \leftrightarrow \langle \ell_1, \ell_2\rangle \in \pi_2\, BEQ_{Tree \otimes EList}$$

Coinduction Principle for Mutual Coinductive Types

In order to prove that $F \models \psi \rightarrow [\![\mathit{Tree} \otimes \mathit{EList}]\!]$:

1. find a suitable pattern ψ';
2. show that ψ' is a "post-fixed point" for $\langle \mathit{node}\, [\![\mathit{Elt}]\!]\, (\pi_2\, X), \mathit{nil} \vee \mathit{cons}\, (\pi_1\, X)\, (\pi_2\, X) \rangle$:

$$F \models \psi' \rightarrow \langle \mathit{node}\, [\![\mathit{Elt}]\!]\, (\pi_2\, \psi'), \mathit{nil} \vee \mathit{cons}\, (\pi_1\, \psi')\, (\pi_2\, \psi') \rangle$$

3. show that $F \models \psi \rightarrow \psi'$.

This can be expressed by the following

Theorem

$$F \models (P \subseteq P' \wedge P' \subseteq \langle \mathit{node}\, [\![\mathit{Elt}]\!]\, (\pi_2\, P'), \mathit{nil} \vee \mathit{cons}\, (\pi_1\, P')\, (\pi_2\, P') \rangle \rightarrow (P \subseteq [\![\mathit{Tree} \otimes \mathit{EList}]\!])$$

Plan

Concluding remarks

- we presented only basics of ML
- ML is engaging,
- . . . and quite appealing after you understand it and you discover its expressivity
  - ML: strong enough to encode FOL, Separation Logic, Hybrid Modal Logic,. . .
  - MmL: strong enough to encode FOL with flp and glp, Separation Logic with recursion, temporal logics,. . .
  - AML: strong enough to encode (easier) type systems, many-sorted and order-sorted algebra. . .
- AML is the best candidate for an implementation
- since many logics can be encoded in ML, proofs in those logics can be borrowed

Not Included in this Talk

- proof systems of the three MLs
- encodings of other logics in ML
- generating proof certificates (FM 2019)
- implementation (`https://github.com/kframework/kore`)

Future Work

- a full formalization in ML of the induction and coinduction principles
- include these principles into the ML prover

Questions?

Thanks!