Linear logic
and higher-order model checking

Paul-André Melliès
CNRS & Université Paris Diderot

Abstraction and Verification in Semantics
Institut Henri Poincaré

23 → 27 June 2014
Purpose of this talk

I. Apply the ideas of linear logic to connect
   - the type-theoretic account by Kobayashi & Ong
   - the domain-theoretic account by Salvati & Walukiewicz

of higher-order model-checking.

II. Construct a cartesian-closed category of coloured domains.

Very similar in spirit as Kazushige’s talk of this morning
Higher-order recognizability

Suppose given a set $L$ of Böhm trees of same type $A$.

Question:
When should one consider the set $L$ as a recognizable language?
Higher-order recognizability

Suppose given a set $L$ of Böhm trees of same type $A$.

**Question:**

When should one consider the set $L$ as a recognizable language?

**Tentative answer:**

Use a finite domain interpretation of types.
Higher-order recognizability

Every finite domain \( D \) induces an interpretation of \( A \) as a finite domain:

\[
\begin{align*}
\llbracket o \rrbracket & := D \\
\llbracket A \times B \rrbracket & := \llbracket A \rrbracket \times \llbracket B \rrbracket \\
\llbracket A \rightarrow B \rrbracket & := \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket
\end{align*}
\]

By continuity, every Böhm tree \( M \) of type \( A \) is interpreted as an element

\[
\llbracket M \rrbracket \in \llbracket A \rrbracket
\]

of the domain \( \llbracket A \rrbracket \).
Higher-order recognizability

Now, every finite subset \( \varphi \subseteq \llbracket A \rrbracket \) induces a set

\[
\mathcal{L}_\varphi = \{ M \mid \llbracket M \rrbracket \in \varphi \}
\]

of Böhm trees of type \( A \).

**Notation:** We write \( \models M : \varphi \) to mean that \( \llbracket M \rrbracket \in \varphi \).

**Definition.** [ adapted from Salvati 2009 ]

A set of Böhm trees \( \mathcal{L} \) is **recognizable** when it is of the form \( \mathcal{L}_\varphi \).
Refinement types

Every such pair \((D, \varphi)\) should be seen as a **predicate** over the type \(A\).

\[
\begin{array}{ccc}
\varphi & \psi \\
D & \downarrow & \uparrow \\
\downarrow & f & \downarrow \\
A & D & B
\end{array}
\]

**Pullback operation:**

Given a predicate \(\psi \subseteq \llbracket B \rrbracket\) one defines the predicate

\[f^*(\psi) := \{ x \in \llbracket A \rrbracket \mid f(x) \in \psi \}\]

in such a way that

\[\models P : \llbracket M \rrbracket^*(\psi) \iff \models MP : \psi\]

for every Böhm tree \(P\) of type \(A\).
Refinement types

Every such pair \((D, \varphi)\) should be seen as a **predicate** over the type \(A\).

\[
\begin{array}{c}
\varphi \\
D \downarrow \\
A \\
f \\
\rightarrow \\
B \\
\psi \uparrow \\
D \\
\end{array}
\]

**Pushforward operation:**

Given a predicate \(\varphi \subseteq \llbracket A \rrbracket\) one defines the predicate

\[
f(\varphi) := \{ f(x) \in \llbracket B \rrbracket \mid x \in \varphi \}
\]

in such a way that

\[
\vDash P : \varphi \quad \Rightarrow \quad \vDash MP : \llbracket M \rrbracket(\varphi)
\]

for every Böhm tree \(P\) of type \(A\).
The Scott semantics of linear logic

Well-known principle.

Every preorder \((A, \leq)\) induces a domain \(\text{Domain}(A)\) defined as follows:

- its elements are the ideals of the preorder,
- the ideals are ordered by inclusion.

Recall that a subset \(X \subseteq A\) is called an ideal of the preorder \(A\) when

\[
\forall a \in A, \forall x \in X, \quad a \leq x \Rightarrow a \in X.
\]
The Scott semantics of linear logic

Key observation.

Suppose that the base type \( o \) is interpreted as the domain of ideals

\[
\llbracket o \rrbracket = \text{Domain}(Q, \leq)
\]
generated by a preorder \( Q \) of atomic states.

In that case, the interpretation of every type \( A \) is the domain of ideals

\[
\llbracket A \rrbracket := \text{Domain}(Q_A, \leq_A)
\]
generated by a specific preorder \( Q_A \) of higher-order states.
The Scott semantics of linear logic

A series of new connectives on preorders, such as:

\[
\begin{align*}
A^\perp & := A^{op} \\
A \& B & := (A + B, \leq_A + \leq_B) \\
A \otimes B & := (A \times B, \leq_A \times \leq_B) \\
!A & := \wp_{fin}(A)
\end{align*}
\]

where the finite sets of elements of \( A \) are ordered as:

\[
\{a_1, \ldots, a_p\} \leq !_A \{b_1, \ldots, b_q\} \iff \forall i \in [p] \exists j \in [q] \ a_i \leq_A b_j
\]
The Scott semantics of linear logic

Given a preorder of **atomic states** for the base type $o$

$$Q_0 = (Q, \leq)$$

the preorder $Q_A$ of **higher-order states** is defined by induction:

$$Q_A \times B = Q_A \& Q_B$$
$$Q_A \rightarrow B = !Q_A \multimap Q_B$$

In particular, a state of the simple type $A \rightarrow B$ is of the form

$$\{q_1, \ldots, q_n\} \multimap q$$

where $q_1, \ldots, q_n$ are states of $A$ and $q$ is a state of $B$. 
What is a higher-order automaton?

Methodological question.

Given a simple type $A$, a finite preorder $(Q, \leq)$ and a subset

$$\varphi \subseteq \llbracket A \rrbracket$$

can we describe the Böhm trees of the associated language

$$\mathcal{L}_\varphi = \{ M \mid \llbracket M \rrbracket \in \varphi \} = \{ M \mid \Downarrow M : \varphi \}$$

in a more direct and automata-theoretic fashion?
What is a higher-order automaton?

Methodological question.

Given a simple type $A$, a finite preorder $(Q, \leq)$ and an element

$$q \in Q_A$$

can we describe the Böhm trees of the associated language

$$\mathcal{L}_q = \{ M \mid q \in \llbracket M \rrbracket \}$$

in a more direct and automata-theoretic fashion?
What is a higher-order automaton?

**Definition.** A higher-order automaton

\[ \mathcal{A} = \langle \Sigma, Q, \delta, q_0 \rangle \]

consists of:

- a finite signature \( \Sigma : Type \rightarrow Set \)
- a finite set of states \( Q \)
- a family of transition functions \( \delta_X : \Sigma_X \rightarrow \\llbracket X \rrbracket \)
- a higher-order initial state \( q_0 \in \llbracket A \rrbracket \)

where the interpretation \( \llbracket - \rrbracket \) of types is induced by the preorder \( Q_0 = Q \).
What is a higher-order automaton?

Suppose given a finite preorder $(Q, \leq)$.

**Adequacy Theorem.**

The interpretation of a Böhm tree $M$ is the set of its accepting states.

In other words, for every higher-order state $q \in \llbracket A \rrbracket$,

$$q \in \llbracket M \rrbracket \iff q \text{ is accepted by the automaton } \langle \emptyset, Q, \emptyset, q \rangle$$

**Corollary.**

Acceptance of a Böhm tree generated by a $\lambda Y$-term $M$ is decidable.
Higher-order recursion schemes

The infinite tree

is generated by the higher-order recursion scheme

\[
\begin{align*}
S & \rightarrow F \ a \ b \ c \\
F \ x \ y \ z & \rightarrow x \ (y \ z) \ (F \ x \ y \ (y \ z))
\end{align*}
\]
Church encoding in the $\lambda$-calculus

The higher-order recursion scheme

\[
\begin{align*}
S & \mapsto F a b c \\
F x y z & \mapsto x (y z) (F x y (y z))
\end{align*}
\]

may be seen as a $\lambda$-term of type

\[(o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o.\]

in the simply-typed $\lambda$-calculus extended with a recursion operator $Y$.

Here, each tree-constructor $a$, $b$ and $c$ is of type:

\[
\begin{align*}
a & : o \rightarrow o \rightarrow o \\
b & : o \rightarrow o \\
c & : o
\end{align*}
\]
Higher-order recursion schemes

Signature

\[ \begin{align*}
  a & : o \rightarrow o \rightarrow o \\
  b & : o \rightarrow o \\
  c & : o 
\end{align*} \]

Non terminals

\[ \begin{align*}
  S & : o \\
  F & : o \rightarrow o 
\end{align*} \]

Rewrite rules

\[ \begin{align*}
  S & \mapsto F c \\
  F & \mapsto \lambda x. a x (F (b x)) 
\end{align*} \]

\[ S \rightarrow F c \rightarrow a c (F (b c)) \rightarrow a c (a (b c) F (b (b c))) \]
Church encoding in linear logic

The formula

\[(o \to o \to o) \to (o \to o) \to o \to o\]

traditionally translated in linear logic as

\[A = !(!o \multimap !o \multimap o) \multimap !(!o \multimap o) \multimap !o \multimap o\]

may be also translated as

\[B = !(o \multimap o \multimap o) \multimap !(o \multimap o) \multimap !o \multimap o.\]
Church encoding in linear logic

So, the same tree may be seen as a term of type

$$A = !(I o \rightarrow ! o \rightarrow o) \rightarrow !(I o \rightarrow o) \rightarrow ! o \rightarrow o$$

with tree-constructors $a$, $b$ and $c$ of type

$$a : ! o \rightarrow ! o \rightarrow o \quad b : ! o \rightarrow o \quad c : o$$

or as a term of type

$$B = !(o \rightarrow o \rightarrow o) \rightarrow !(o \rightarrow o) \rightarrow ! o \rightarrow o$$

with tree-constructors $a$, $b$ and $c$ of type

$$a : o \rightarrow o \rightarrow o \quad b : o \rightarrow o \quad c : o$$
Principle of duality

Proponent

Program

plays the formula

$A$

Opponent

Environment

plays the formula

$A^\perp$

Negation permutes the roles of Proponent and Opponent
Principle of duality

Opponent
Environment
plays the formula

$A^\perp$

Proponent
Program
plays the formula

$A$

Negation permutes the rôles of Opponent and Proponent
Duality applied to the Church encoding

**Question:** So, what is the dual of a tree?

**Answer:** Well, it should be a tree automaton!
Duality applied to the Church encoding

The formulas $A$ and $B$ have counter-formulas:

\[ A^\perp = ! ( !o \circlearrowleft !o \circlearrowleft o ) \otimes ! ( !o \circlearrowleft o ) \otimes !o \otimes o^\perp \]

\[ B^\perp = ! ( o \circlearrowleft o \circlearrowleft o ) \otimes ! ( o \circlearrowleft o ) \otimes o \otimes o^\perp \]

Claim:

- the counter-formula $B^\perp$ is the type of tree automata
- the counter-formula $A^\perp$ is the type of alternating tree automata
What is a linear higher-order automaton?

Suppose given a finite preorder \((Q, \leq)\).

**Adequacy Theorem.**

The interpretation of a Böhm tree \(M\) is the set of its accepting states.

In other words, for every higher-order state \(q \in \llbracket A \rrbracket\),

\[ q \in \llbracket M \rrbracket \iff q \text{ is accepted by the automaton } \langle \emptyset, Q, \emptyset, q \rangle \]

**Corollary.**

Acceptance of a Böhm tree generated by a \(LL_Y\)-term \(M\) is decidable.
The modal nature of priorities

A proof-theoretic account of parity tree automata
An intersection type system equivalent to the modal $\mu$-calculus

The grammar of kinds $\kappa$

\[
\kappa :: \sigma \mid \kappa \Rightarrow \kappa
\]

Naoki Kobayashi and Luke Ong [LICS 2009]
An intersection type system equivalent to the modal $\mu$-calculus

The grammar of atomic types $\theta$ and intersection types $\tau$

\[ q_i ::_{\text{atomic}} o \]

\[ \theta_1 ::_{\text{atomic}} \kappa \quad \ldots \quad \theta_n ::_{\text{atomic}} \kappa \]
\[ (\theta_1, m_1) \land \ldots \land (\theta_n, m_n) :: \kappa \]

\[ \tau_1 :: \kappa_1 \quad \ldots \quad \tau_n :: \kappa_n \quad q ::_{\text{atomic}} o \]
\[ \tau_1 \Rightarrow \ldots \tau_k \Rightarrow q ::_{\text{atomic}} \kappa_1 \Rightarrow \ldots \Rightarrow \kappa_k \Rightarrow o \]

Naoki Kobayashi and Luke Ong [LICS 2009]
A type system equivalent to the modal $\mu$-calculus

\[
x : (\theta, \Omega[\theta]) \vdash x : \theta
\]

\[
\{ (i,q_{ij}) | 1 \leq i \leq n, 1 \leq j \leq k_i \} \text{ satisfies } \delta_A(q,a)
\]

\[
a : \bigwedge_{j=1}^{k_1} (q_{1j}, m_{1j}) \Rightarrow \cdots \Rightarrow \bigwedge_{j=1}^{k_n} (q_{nj}, m_{nj}) \Rightarrow q
\]

where $m_{ij} = \max(\Omega[q_{ij}], \Omega[q])$

\[
\Delta \vdash t : (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \Rightarrow \theta
\]

\[
\Delta_1 \vdash u : \theta_1 \quad \ldots \quad \Delta_k \vdash u : \theta_k
\]

\[
\Delta, \Delta_1 \triangleright m_1, \ldots, \Delta_k \triangleright m_k \vdash tu : \theta
\]

where $\Delta \triangleright m = \{ F : (\theta, \max(m, m')) | F : (\theta, m) \in \Delta \}$

\[
\Delta, x : \bigwedge_{i \in I} (\theta_i, m_i) \vdash t : \theta
\]

\[
\Delta \vdash \lambda x.t : \bigwedge_{i \in J} (\theta_i, m_i) \Rightarrow \theta
\]
Emulation theorem

Let $G$ be a higher-order recursion scheme.

Let $\mathcal{A}$ be an alternating parity tree automaton.

**Theorem [Kobayashi & Ong]**

The tree generated by $G$ is recognized by $\mathcal{A}$ $\iff$ The higher-order recursion scheme $G$ is typable.
Guiding idea of Kobayashi and Ong

\[(q_1, m_1) \wedge (q_2, m_2) \Rightarrow q\]
Collecting colours works in the same way as collecting levels of copies
A colour modality for intersection types

Definition. A parametric modality is a family of functors

\( \Box_m : C \rightarrow C \quad m \in \mathbb{N} \)

each of them lax monoidal:

\[
\begin{align*}
\Box_m A \otimes \Box_m B & \rightarrow \Box_m (A \otimes B) \\
1 & \rightarrow \Box_m 1
\end{align*}
\]

and defining together a parametric comonad

\[
\begin{align*}
\Box_{\max(m,m')} A & \rightarrow \Box_m \Box_{m'} A \\
\Box_0 A & \rightarrow A
\end{align*}
\]

The structure of copy management in linear logic
The exponential modality

\[ !A \otimes !B \rightarrow ! (A \otimes B) \]

\[ !A \rightarrow !!A \]

\[ !A \rightarrow A \]

The structure of copy management in linear logic
Translation

\[
\Delta \vdash t : (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \Rightarrow \theta \quad \Delta_i \vdash u : \theta_i
\]

\[
\Delta, \Delta_1 \uparrow m_1, \ldots, \Delta_k \uparrow m_k \vdash t \ u : \theta
\]

where \( \Delta \uparrow m = \{ F : (\theta, \text{max}(m, m')) \mid F : (\theta, m) \in \Delta \} \)

is translated as

\[
\Delta \vdash t : \Box_{m_1} \theta_1 \land \ldots \land \Box_{m_k} \theta_k \Rightarrow \theta \quad \Box_{m_i} \Delta_i \vdash u : \Box_{m_i} \theta_i
\]

\[
\Delta, \Box_{m_1} \Delta_1, \ldots, \Box_{m_k} \Delta_k \vdash t \ u : \theta
\]
Linear logic with colours

A domain-theoretic account of parity tree automata
A colour modality for domains

Suppose given a specific number $n$ of colours.

**Definition.** The colour modality on preorders is defined as

$$\boxtimes A := A \& \cdots \& A \underbrace{}_{n}$$

As a consequence, note that

$$\text{Domain}(\boxtimes A) := \text{Domain}(A) \times \cdots \times \text{Domain}(A)$$
The colour modality

Two preliminary observations

▷ The modality □ defines a comonad.

\[
\begin{align*}
\varepsilon_A : & \quad \Box A \quad \rightarrow \quad A \\
& \quad (1, q) \quad \Rightarrow \quad q \\
\delta_A : & \quad \Box A \quad \rightarrow \quad \Box \Box A \\
& \quad (\max(m_1, m_2), q) \quad \Rightarrow \quad (m_1, (m_2, q))
\end{align*}
\]

▷ The comonad □ commutes with finite products:

\[
\Box (A \& B) \quad \cong \quad \Box A \& \Box B
\]

\[
\Box \top \quad \cong \quad \top
\]
The colour modality

A third observation

- There exists a distributivity law

\[ \lambda : ! \square \Rightarrow \square ! : \text{ScottL} \rightarrow \text{ScottL} \]

defined as follows:

\[ \lambda_A : \{(m_1, q_1), \ldots, (m_k, q_k)\} \mapsto (\max(m_1, \ldots, m_k), \{q_1, \ldots, q_k\}) \]
A colour modality

An important consequence: The composite modality

! □ : ScottL → ScottL

defines an exponential modality of linear logic.

From this follows that the Kleisli category

\( \mathcal{D} := \text{Kleisli}(\text{ScottL}, ! \square) \)

is a cartesian closed category.
A domain-theoretic formulation

The category \( \mathcal{D} \) has

- finite prime algebraic domains as objects
- continuous functions \( f : D^n \to E \) as morphisms.

Two morphisms of the category \( \mathcal{D} \)

\( f : D^n \to E \quad \text{and} \quad g : E^n \to F \)

are composed as follows:

\[
\begin{array}{c}
D^n \xrightarrow{D^{\text{max}}} D^{n \times n} \xrightarrow{f^n} E^n \xrightarrow{g} E \\
\end{array}
\]
A domain-theoretic formulation

In the case $n = 2$

$$g \circ f : (x_1, x_2) \mapsto g(f(x_1, x_2), f(x_2, x_2))$$

In the case $n = 3$

$$g \circ f : (x_1, x_2, x_3) \mapsto g(f(x_1, x_2, x_3), f(x_2, x_2, x_3), f(x_3, x_3, x_3))$$

More generally:

$$
\begin{pmatrix}
1 & 2 \\
2 & 2 \\
3 & 3 \\
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 \\
2 & 2 & 3 \\
3 & 3 & 3 \\
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
3 & 3 & 3 & 4 \\
4 & 4 & 4 & 4 \\
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 \\
2 & 2 & 3 & 4 & 5 \\
3 & 3 & 3 & 4 & 5 \\
4 & 4 & 4 & 4 & 5 \\
5 & 5 & 5 & 5 & 5 \\
\end{pmatrix}
$$
An inductive-coinductive fixpoint

For simplicity, let us assume that the number $n$ of colours is even.

Given a morphism in the category $\mathcal{D}$

$$f : D^n \rightarrow D$$

one defines the fixpoint

$$Y(f) = \nu x_n \cdot \mu x_{n-1} \cdot \nu x_{n-2} \cdots \nu x_2 \cdot \mu x_1 \cdot f(x_1, \cdots, x_n)$$

**Theorem.** This defines a categorical interpretation of the $\lambda Y$-calculus.
What is a higher-order automaton?

Suppose given a finite preorder \((Q, \leq)\).

**Adequacy Theorem.**
The interpretation of a Böhm tree \(M\) is the set of its accepting states.

In other words, for every higher-order state \(q \in \llbracket A \rrbracket\),

\[
q \in \llbracket M \rrbracket \iff q \text{ is accepted by the parity automaton } \langle \emptyset, Q, \emptyset, q \rangle
\]

**Corollary.**
Acceptance of a Böhm tree generated by a \(\lambda Y\)-term \(M\) is decidable.
Thank you !