Indexed linear logic and higher-order model-checking

Charles Grellois   Paul-André Melliès

PPS — Université Paris 7

July 18th, 2014

Model-checking higher-order programs

Usual approach in verification: model-checking.

- Build a model of a program
- Specify a property in an appropriate logic
- Make them interact in order to determine whether the program satisfies the property.

Interaction is often realized by translating the formula into an equivalent automaton, which then runs over the model.

Model-checking higher-order programs

This work is concerned with the verification of higher-order functional programs, as Java for instance.

They will be modelled by recursion schemes.

Properties will be expressed in MSO or modal μ-calculus (equi-expressive over trees).

Their automata counterpart is given by alternating parity automata (APT).

Model-checking higher-order programs

This work is concerned with the verification of higher-order functional programs, as Java for instance.

They will be modelled by recursion schemes.

Properties will be expressed in MSO or modal μ-calculus (equi-expressive over trees).

Their automata counterpart is given by alternating parity automata (APT).

Model-checking higher-order programs

This model-checking problem is decidable (Ong 2006).

Several proofs of this result were given.

Most of them rely on semantics.

Here we focus on the Kobayashi-Ong approach (2009), which uses intersection types.

Our aim is to deepen the semantic understanding we have of this result, using existing relations between intersection types, linear logic and its models.

Model-checking higher-order programs

This model-checking problem is decidable (Ong 2006).

Several proofs of this result were given.

Most of them rely on semantics.

Here we focus on the Kobayashi-Ong approach (2009), which uses intersection types.

Our aim is to deepen the semantic understanding we have of this result, using existing relations between intersection types, linear logic and its models.

Higher-order recursion schemes

Idea: it is a kind of grammar whose parameters may be functions and which generates trees.

Alternatively, it is a formalism equivalent to λY calculus with uninterpreted constants from a ranked alphabet Σ.

A very simple functional program

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \mathit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

With a recursion scheme we can model this program and produce its tree of behaviours.

Note that constants are not interpreted: in particular, a recursion scheme does not evaluate a `if`.

A very simple functional program

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \mathit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

With a recursion scheme we can model this program and produce its tree of behaviours.

Note that constants are not interpreted: in particular, a recursion scheme does not evaluate a `if`.

A very simple functional program

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen}\ x & = & \texttt{if}\ \mathit{end}\ \texttt{then}\ x\ \texttt{else Listen (data}\ x\texttt{)} \end{array}$$

formulated as a recursion scheme:

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L}\ x & = & \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ )) \end{array}$$

or, in λ-calculus style :

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L} & = & \lambda x.\texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ )) \end{array}$$

(this latter representation is a regular grammar !)

A very simple functional program

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \textit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

formulated as a recursion scheme:

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L } x & = & \texttt{if } x \,(\texttt{L (data } x\,) \end{array}$$

or, in λ-calculus style :

$$\begin{array}{lcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L} & = & \lambda x.\,\texttt{if } x \,(\texttt{L (data } x\,) \end{array}$$

(this latter representation is a regular grammar !)

A very simple functional program

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen}\ x & = & \texttt{if}\ \mathit{end}\ \texttt{then}\ x\ \texttt{else Listen (data}\ x) \end{array}$$

formulated as a recursion scheme:

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L}\ x & = & \texttt{if}\ \ x\ (\texttt{L}\ (\texttt{data}\ \ x\ ) \end{array}$$

or, in λ-calculus style :

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L} & = & \lambda x.\,\texttt{if}\ \ x\ (\texttt{L}\ (\texttt{data}\ \ x\ ) \end{array}$$

(this latter representation is a regular grammar !)

Value tree of a recursion scheme

$$\begin{array}{rcl} \mathtt{S} & = & \mathtt{L\ Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ ) \end{array}$$

generates:

S

Value tree of a recursion scheme

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L}\ x & = & \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ ) \end{array}$$

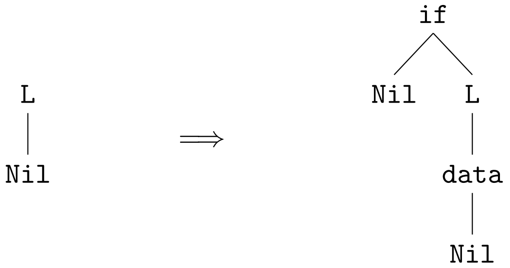
generates:

$$\texttt{S} \qquad \Longrightarrow \qquad \begin{array}{c} \texttt{L} \\ | \\ \texttt{Nil} \end{array}$$

Value tree of a recursion scheme

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L}\ x & = & \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ )) \end{array}$$

generates:

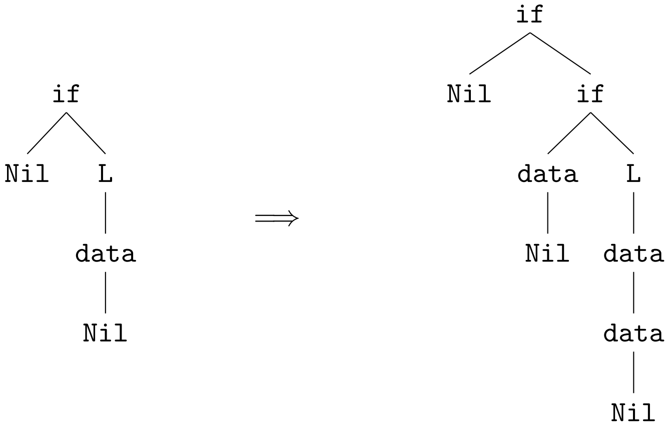


Notice that substitution and expansion occur in one same step.

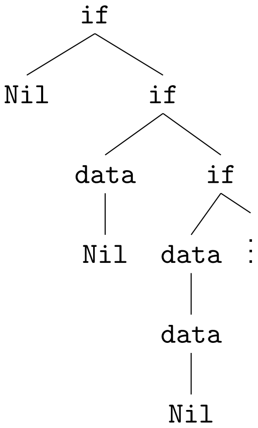
Value tree of a recursion scheme

$$
\begin{array}{rcl}
\texttt{S} & = & \texttt{L Nil} \\
\texttt{L}\ x & = & \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ ))
\end{array}
$$

generates:

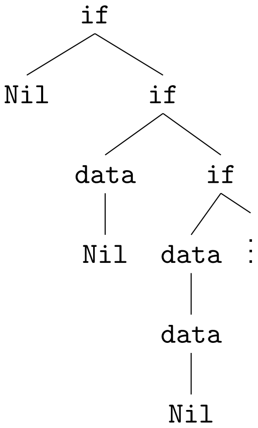


Value tree of a recursion scheme



Very simple program, yet it produces a tree which is not regular. . .

Value tree of a recursion scheme



Very simple program, yet it produces a tree which is not regular...

Alternating parity tree automata

Modal μ-calculus is an extension of boolean logic over a branching structure, with fixpoints and quantifications over the successors of the current position.

It allows to unravel some formula over the structure. This can be encoded into an alternating parity tree automata (APT).

Its states are the subformulas of the encoded formula.

Alternating parity tree automata

APT are non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Example: $\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1)$.

This is the behaviour of the exponential modality of linear logic

Alternating parity tree automata

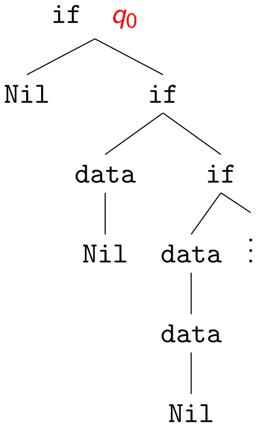
APT are non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Example: $\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1)$.

This is the behaviour of the exponential modality of linear logic

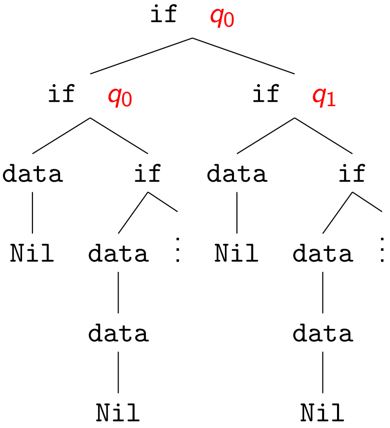
Alternating parity tree automata

$$\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1).$$



Alternating parity tree automata

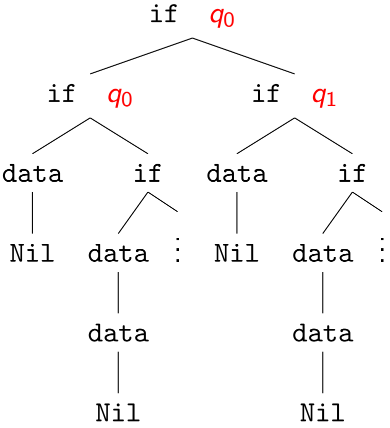
$$\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1).$$



and so on. This gives the notion of run-tree.

Alternating parity tree automata

$$\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1).$$



and so on. This gives the notion of run-tree.

Model-checking higher-order programs

Since the trees produced by schemes are not regular, they have no good finitary representation — except the scheme which products them.

Model-checking higher-order programs

Kobayashi remarked in 2009 that a transition

$$\delta(q, a) \;=\; (1, q_0) \wedge (1, q_1) \wedge (2, q_2)$$

may be understood as a refinment of the simple typing $a : \bot \to \bot \to \bot$ with intersection types:

$$a \;:\; (q_0 \wedge q_1) \to q_2 \to q \;::\; \bot \to \bot \to \bot$$

Note that the simple type is explicited. In this approach, intersection types must refine simple types.

Model-checking higher-order programs

Kobayashi remarked in 2009 that a transition

$$\delta(q, a) \ = \ (1, q_0) \wedge (1, q_1) \wedge (2, q_2)$$

may be understood as a refinment of the simple typing $a \, : \, \perp \to \perp \to \perp$ with intersection types:

$$a \ : \ (q_0 \wedge q_1) \to q_2 \to q \ :: \ \perp \to \perp \to \perp$$

Note that the simple type is explicited. In this approach, intersection types must refine simple types.

Model-checking higher-order programs

$$a \,:\, (q_0 \wedge q_1) \rightarrow q_2 \rightarrow q \;::\; \perp \rightarrow \perp \rightarrow \perp$$

A very important consequence is that these refined types naturally lift to higher-order.

This way, the action of the APT over the infinitary, non-regular value tree of the scheme can be reflected in the finite representation that is its equivalent λY-term.

This is the core idea of the decidability result.

Model-checking higher-order programs

$$a \; : \; (q_0 \wedge q_1) \rightarrow q_2 \rightarrow q \; :: \; \perp \rightarrow \perp \rightarrow \perp$$

A very important consequence is that these refined types naturally lift to higher-order.

This way, the action of the APT over the infinitary, non-regular value tree of the scheme can be reflected in the finite representation that is its equivalent λY-term.

This is the core idea of the decidability result.

Kobayashi's intersection type system

Here is a variant of Kobayashi's original 2009 type system:

Axiom $$\frac{\vdash \ \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma\, ,\, x : \bigwedge_{j \in J} \tau_j :: \kappa \ \vdash \ x : \tau_j :: \kappa}$$

App $$\frac{\Gamma \vdash M : (\bigwedge_{j \in J} \tau_j) \rightarrow \sigma :: \kappa \rightarrow \kappa' \qquad \Gamma \vdash N : \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma \vdash MN : \sigma :: \kappa'}$$

Lambda $$\frac{\Gamma, x : \bigwedge_{j \in J} \tau_j :: \kappa \ \vdash \ M : \sigma :: \kappa'}{\Gamma \vdash \lambda x. M : (\bigwedge_{j \in J} \tau_j) \rightarrow \sigma :: \kappa \rightarrow \kappa'}$$

Note that the App rule requires several derivations for a same term.

No recursion.

Kobayashi's intersection type system

Here is a variant of Kobayashi's original 2009 type system:

$$\text{Axiom} \qquad \frac{\vdash \;\; \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma\,,\, x : \bigwedge_{j \in J} \tau_j :: \kappa \;\vdash\;\; x : \tau_j :: \kappa}$$

$$\text{App} \qquad \frac{\Gamma \vdash M : (\bigwedge_{j \in J} \tau_j) \to \sigma :: \kappa \to \kappa' \qquad \Gamma \vdash N : \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma \vdash MN : \sigma :: \kappa'}$$

$$\text{Lambda} \qquad \frac{\Gamma, x : \bigwedge_{j \in J} \tau_j :: \kappa \;\vdash\;\; M : \sigma :: \kappa'}{\Gamma \vdash \lambda x.M : (\bigwedge_{j \in J} \tau_j) \to \sigma :: \kappa \to \kappa'}$$

Note that the App rule requires several derivations for a same term.

No recursion.

Kobayashi's intersection type system

Here is a variant of Kobayashi's original 2009 type system:

$$\text{Axiom} \qquad \frac{\vdash \ \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma\,,\, x : \bigwedge_{j \in J} \tau_j :: \kappa \ \vdash \ x : \tau_j :: \kappa}$$

$$\text{App} \qquad \frac{\Gamma \vdash M : (\bigwedge_{j \in J} \tau_j) \rightarrow \sigma :: \kappa \rightarrow \kappa' \qquad \Gamma \vdash N : \tau_j :: \kappa \qquad (\forall j \in J)}{\Gamma \vdash MN : \sigma :: \kappa'}$$

$$\text{Lambda} \qquad \frac{\Gamma, x : \bigwedge_{j \in J} \tau_j :: \kappa \ \vdash \ M : \sigma :: \kappa'}{\Gamma \vdash \lambda x.M : (\bigwedge_{j \in J} \tau_j) \rightarrow \sigma :: \kappa \rightarrow \kappa'}$$

Note that the App rule requires several derivations for a same term.

No recursion.

Kobayashi's intersection type system

Intersections are stable under surjective reindexing $f : J \to I$:

$$\bigwedge_{j \in J} \sigma_{f(j)} \quad = \quad \bigwedge_{i \in I} \sigma_i$$

for every family $\{\sigma_i \mid i \in I\}$ of refinement types indexed by a finite set I.

In particular, for the surjection $\{1, 2\} \to \{1\}$,

$$\sigma \wedge \sigma \quad = \quad \sigma$$

Kobayashi's 2009 result

(rephrased a little)

Let t be a term of ground type (that is, a term which evaluates to a tree).

If an alternating automaton has a run-tree over the tree obtained by evaluating t, then there exists a context Γ such that $\Gamma \vdash t \,:\, q_0 \,::\, \perp$.

Moreover, the intersection types occuring in Γ correspond to the transition function of the automaton.

The converse holds as well.

Kobayashi's 2009 result

(rephrased a little)

Let t be a term of ground type (that is, a term which evaluates to a tree).

If an alternating automaton has a run-tree over the tree obtained by evaluating t, then there exists a context Γ such that $\Gamma \vdash t \, : \, q_0 \, :: \, \perp$.

Moreover, the intersection types occuring in Γ correspond to the transition function of the automaton.

The converse holds as well.

Kobayashi's 2009 result

(rephrased a little)

Let t be a term of ground type (that is, a term which evaluates to a tree).

If an alternating automaton has a run-tree over the tree obtained by evaluating t, then there exists a context Γ such that $\Gamma \vdash t \; : \; q_0 \; :: \; \perp$.

Moreover, the intersection types occuring in Γ correspond to the transition function of the automaton.

The converse holds as well.

Models

We would like to understand this result in a model interpretation.

It suggests that model-checking a term t of simple type A may be performed compositionally

- by computing $||A||$
  (intuitively, the set of all intersection types refining A)
- then $||t|| \subseteq ||A||$
  (the intersection types for t)
- and then by checking that $||t||$ is coherent with δ.

Models

We would like to understand this result in a model interpretation.

It suggests that model-checking a term t of simple type A may be performed compositionally

- by computing $||A||$
  (intuitively, the set of all intersection types refining A)
- then $||t|| \subseteq ||A||$
  (the intersection types for t)
- and then by checking that $||t||$ is coherent with δ.

Linear decomposition of the intuitionnistic arrow

In linear logic, the intuitionnistic arrow $A \Rightarrow B$ is interpreted as

$$!A \multimap B$$

where

- $!A$ is a collection of elements of A (possibly none)
- and the linear arrow $\multimap$ uses its arguments exactly once

By interpreting the base type $\perp$ with the set of states Q of an alternating automaton, we get for example that an element of the interpretation of $\perp \Rightarrow \perp$ is some collection of states giving a new state.

This is very close to intersection types...

Linear decomposition of the intuitionnistic arrow

In linear logic, the intuitionnistic arrow $A \Rightarrow B$ is interpreted as

$$!A \multimap B$$

where

- $!A$ is a collection of elements of A (possibly none)
- and the linear arrow $\multimap$ uses its arguments exactly once

By interpreting the base type $\perp$ with the set of states Q of an alternating automaton, we get for example that an element of the interpretation of $\perp \Rightarrow \perp$ is some collection of states giving a new state.

This is very close to intersection types...

Intersection types and models of linear logic

Several interpretations of the word *collection* are possible, and give different models.

- qualitative models, in which $!A$ is interpreted as the powerset of A
- and quantitative models, in which $!A$ is interpreted as the set of multisets of elements of A

Intersection types and models of linear logic

Several interpretations of the word *collection* are possible, and give different models.

- qualitative models, in which $!A$ is interpreted as the powerset of A
- and quantitative models, in which $!A$ is interpreted as the set of multisets of elements of A

Intersection types and models of linear logic

Consider again the typing

$$a \,:\, (q_0 \wedge q_1) \rightarrow q_2 \rightarrow q \,::\, \perp \rightarrow \perp \rightarrow \perp$$

In a qualitative model, the interpretation of a will be a subset of $\mathcal{P}(Q) \times \mathcal{P}(Q) \times Q$.

Translated to the model: $(\{q_0, q_1\}, \{q_2\}, q) \in ||a||$.

Such models contain information on the states which were used by a term, but not about how many times they did.

They correspond to idempotent types.

Intersection types and models of linear logic

Consider again the typing

$$a \;:\; (q_0 \wedge q_1) \to q_2 \to q \;::\; \perp \to \perp \to \perp$$

In a qualitative model, the interpretation of a will be a subset of $\mathcal{P}(Q) \times \mathcal{P}(Q) \times Q$.

Translated to the model: $(\{q_0, q_1\}, \{q_2\}, q) \in ||a||$.

Such models contain information on the states which were used by a term, but not about how many times they did.

They correspond to idempotent types.

Intersection types and models of linear logic

Consider again the typing

$$a \,:\, (q_0 \wedge q_1) \rightarrow q_2 \rightarrow q \,::\, \perp \rightarrow \perp \rightarrow \perp$$

In a qualitative model, the interpretation of a will be a subset of $\mathcal{P}(Q) \times \mathcal{P}(Q) \times Q$.

Translated to the model: $(\{q_0, q_1\}, \{q_2\}, q) \in ||a||$.

Such models contain information on the states which were used by a term, but not about how many times they did.

They correspond to idempotent types.

Intersection types and models of linear logic

But these models are quite complicated, because the interpretations must be closed for some order.

What about the quantitative model *Rel*, in which we could interpret the interaction of a term and an automaton ?

The relational model

We consider a relational model where

- $||\bot|| \ = \ Q$,
- $||A \multimap B|| \ = \ A \times B$,
- $||!\,A|| \ = \ \mathcal{M}_{fin}(A)$

Morally, $||A \Rightarrow B|| \ = \ \mathcal{M}_{fin}(A) \times B$.

The relational model: interpretation of intersection types

$\mathcal{M}_{fin}(A)$ is the set of finite multisets of elements of A.

What does it imply for the interpretation of intersection types ?

$$[q, q] \neq [q]$$

In other terms, the corresponding system of intersection types distinguishes $q \wedge q$ from q.

It is no longer idempotent.

The relational model: interpretation of intersection types

$\mathcal{M}_{fin}(A)$ is the set of finite multisets of elements of A.

What does it imply for the interpretation of intersection types ?

$$[q, q] \neq [q]$$

In other terms, the corresponding system of intersection types distinguishes $q \wedge q$ from q.

It is no longer idempotent.

Non-idempotent types

In a first step towards the relational model, we

- consider a sequent version of Kobayashi's system,
- with only terms in η-long β-normal form,
- and where intersections are no longer closed under surjective reindexing.

This gives the quantitative intersection type system.

Non-idempotent types

Theorem

Every derivation tree of one system may be effectively translated in the other either by lifting qualitative types or by collapsing quantitative types.

Non-idempotent types

Theorem

Every derivation tree of one system may be effectively translated in the other either by lifting qualitative types or by collapsing quantitative types:

- *If* $\Gamma \vdash t : \sigma : \kappa$ *in the quantitative system, then* $|\Gamma| \vdash t : |\sigma| :: \kappa$ *in Kobayashi's system.*

where $|\Gamma|$ and $|\sigma|$ are obtained by assuming stability under surjective reindexing.

Non-idempotent types

Theorem

Every derivation tree of one system may be effectively translated in the other either by lifting qualitative types or by collapsing quantitative types:

- *If* $\Gamma \vdash t : \sigma : \kappa$ *in the quantitative system, then* $|\Gamma| \vdash t : |\sigma| :: \kappa$ *in Kobayashi's system.*
- *If* $x_1 : \sigma_1 :: \kappa_1, \ldots, x_n : \sigma_n :: \kappa_n \vdash t : \tau : \kappa$ *in Kobayashi's system, there exists quantitative types* $\hat{\sigma}_i$ $(1 \leq i \leq n)$ *and* $\hat{\tau}$ *such that*
  - $\forall i \in \{1, \ldots, n\}$ $\hat{\sigma}_i :: \kappa_i$ *and* $\hat{\tau} :: \kappa$,
  - $\forall i \in \{1, \ldots, n\}$ $|\hat{\sigma}_i| \preccurlyeq \sigma_i$ *and* $|\hat{\tau}| \preccurlyeq \tau$,
  - $x_1 : \hat{\sigma_1} :: \kappa_1, \ldots, x_n : \hat{\sigma_n} :: \kappa_n \vdash t : \hat{\tau} : \kappa$ *in the quantitative system.*

Roughly speaking, this means that lifting a qualitative type to a quantitative type may give a more precise type.

Non-idempotent types

Interestingly enough, this order comes from a qualitative model of linear logic (the Scott model – which is very close to this variant of Kobayashi's intersection type system).

This lifting/collapse theorem is strongly related to Ehrhard's result of extensional collapse for models of linear logic.

Indexed linear logic

The quantitative type system leads naturally to an interpretation in *Rel* of the model-checking problem.

However, this model contains strictly more than denotations of terms.

Actually, if a term uses its argument several times, nothing forbids to give the denotation of a different term of the appropriated type for each occurence.

The whole relational model corresponds to denotations of the lambda calculus with resources.

Indexed linear logic

The quantitative type system leads naturally to an interpretation in *Rel* of the model-checking problem.

However, this model contains strictly more than denotations of terms.

Actually, if a term uses its argument several times, nothing forbids to give the denotation of a different term of the appropriated type for each occurence.

The whole relational model corresponds to denotations of the lambda calculus with resources.

Indexed linear logic

The quantitative type system leads naturally to an interpretation in *Rel* of the model-checking problem.

However, this model contains strictly more than denotations of terms.

Actually, if a term uses its argument several times, nothing forbids to give the denotation of a different term of the appropriated type for each occurence.

The whole relational model corresponds to denotations of the lambda calculus with resources.

Indexed linear logic

Bucciarelli and Ehrhard characterized logically the fragment of the relational model corresponding to terms.

Recall Kobayashi's application rule

$$\text{App}\quad \frac{\Gamma \vdash M : (\bigwedge_{j \in J} \tau_j) \to \sigma :: \kappa \to \kappa' \qquad \Gamma \vdash N : \tau_j :: \kappa \quad (\forall j \in J)}{\Gamma \vdash MN : \sigma :: \kappa'}$$

Intuitively, their idea is to modify the logic so that it is forced to provide a proof term of the same shape for each index j.

Indexed linear logic

Bucciarelli and Ehrhard characterized logically the fragment of the relational model corresponding to terms.

Recall Kobayashi's application rule

$$\text{App} \quad \frac{\Gamma \vdash M : (\bigwedge_{j \in J} \tau_j) \to \sigma :: \kappa \to \kappa' \qquad \Gamma \vdash N : \tau_j :: \kappa \quad (\forall j \in J)}{\Gamma \vdash MN : \sigma :: \kappa'}$$

Intuitively, their idea is to modify the logic so that it is forced to provide a proof term of the same shape for each index j.

Indexed linear logic

In this goal, proofs are parallelized.

Sequents are indexed by families I, J, $K \ldots$:

$$\Gamma \;\vdash_I\; t : \sigma_i :: A$$

This should be understood as the superposition of $|I|$ different typing proofs for a same term.

The proof of index i proves that $t : \sigma_i :: A$.

Indexed linear logic

In this goal, proofs are parallelized.

Sequents are indexed by families I, J, $K \ldots$:

$$\Gamma \ \vdash_I \ t : \sigma_i :: A$$

This should be understood as the superposition of $|I|$ different typing proofs for a same term.

The proof of index i proves that $t : \sigma_i :: A$.

Indexed linear logic

In the relational model, the exponential builds multisets.

In indexed linear logic, it should only build uniform multisets.

This is done with the Promotion rule

$$\frac{\ldots\ x_k\ :\ [\sigma_{i_k} \,|\, i_k \in I_k,\ u_k(i_k) = j]\ ::\ !_{u_k} A_k \ldots \vdash_J\ M\ :\ \tau_j\ ::\ B}{\ldots x_k\ :\ [\sigma_{i_k} \,|\, i_k \in I_k,\ v(u_k(i_k)) = l]\ ::\ !_{v \circ u_k} A_k \ldots \vdash_L\ M\ :\ [\tau_j \,|\, v(j) = l]\ ::\ !_v\ B}$$

where $v : J \to L$.

Indexed linear logic

How do we create intersection types ?

Consider $c \,:\, q_0 \wedge q_1 \,::\, \perp$ in the quantitative system. We build it with the following derivation:

$$\dfrac{\dfrac{\dfrac{(q_1,\, q_2) \in Q^2}{c \,:\, q_j \,::\, \perp_{\{1,2\}} \;\vdash_{j \in \{1,2\}}\; c \,:\, q_j \,::\, \perp_{\{1,2\}}}}{c \,:\, [q_j] \,::\, !_{id} \perp_{\{1,2\}} \;\vdash_{j \in \{1,2\}}\; c \,:\, q_j \,::\, \perp_{\{1,2\}}}}{c \,:\, [q_1, q_2] \,::\, !_{v \circ id} \perp_{\{1,2\}} \;\vdash_{\{1\}}\; c \,:\, [q_1, q_2] \,::\, !_{v} \perp_{\{1,2\}}}$$

where v is the surjection $\{1,2\} \to \{1\}$.

Indexed linear logic

In general, this structuration rule builds parallel families of multisets.

$$\cfrac{\cfrac{(q_1,\ q_2, q_3) \in Q^2}{c\ :\ q_j\ ::\ \perp_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \perp_{\{1,2,3\}}}}{\cfrac{c\ :\ [q_j]\ ::\ !_{id} \perp_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \perp_{\{1,2,3\}}}{c\ :\ [q_j \mid v(j) = i]\ ::\ !_v \perp_{\{1,2,3\}}\ \vdash_{i \in \{1,2\}}\ c\ :\ [q_j \mid v(j) = i]\ ::\ !_v \perp_{\{1,2,3\}}}}$$

where $v\ :\ \{1, 2, 3\} \rightarrow \{1, 2\}$ maps 1 to 1, and 2 and 3 to 2.

Indexed linear logic

There is a translation theorem between the quantitative type system and the indexed linear calculus.

Moreover, the derivations of the ILC can be reconstructed from ILL — that is, removing the term and the intermediate level.

$$\dfrac{\dfrac{\dfrac{(q_1,\ q_2, q_3) \in Q^2}{c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}}}{c\ :\ [q_j]\ ::\ !_{id}\ \bot_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}}}{c\ :\ [q_j \mid v(j) = i]\ ::\ !_v\ \bot_{\{1,2,3\}}\ \vdash_{i \in \{1,2\}}\ c\ :\ [q_j \mid v(j) = i]\ ::\ !_v\ \bot_{\{1,2,3\}}}$$

Indexed linear logic

There is a translation theorem between the quantitative type system and the indexed linear calculus.

Moreover, the derivations of the ILC can be reconstructed from ILL — that is, removing the term and the intermediate level.

$$\dfrac{\dfrac{\dfrac{(q_1,\ q_2, q_3) \in Q^2}{c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}}}{c\ :\ [q_j]\ ::\ !_{id}\, \bot_{\{1,2,3\}}\ \vdash_{j \in \{1,2,3\}}\ c\ :\ q_j\ ::\ \bot_{\{1,2,3\}}}}{c\ :\ [q_j \mid v(j) = i]\ ::\ !_v\, \bot_{\{1,2,3\}}\ \vdash_{i \in \{1,2\}}\ c\ :\ [q_j \mid v(j) = i]\ ::\ !_v\, \bot_{\{1,2,3\}}}$$

Indexed linear logic

There is a translation theorem between the quantitative type system and the indexed linear calculus.

Moreover, the derivations of the ILC can be reconstructed from ILL — that is, removing the term and the intermediate level.

$$\dfrac{\dfrac{\dfrac{(q_1,\ q_2, q_3) \in Q^2}{q_j \ :: \ \bot_{\{1,2,3\}} \ \vdash_{j \in \{1,2,3\}} \ q_j \ :: \ \bot_{\{1,2,3\}}}}{[q_j] \ :: !_{id} \bot_{\{1,2,3\}} \ \vdash_{j \in \{1,2,3\}} \ q_j \ :: \ \bot_{\{1,2,3\}}}}{[q_j \mid v(j) = i] \ :: !_v \bot_{\{1,2,3\}} \ \vdash_{i \in \{1,2\}} \ [q_j \mid v(j) = i] \ :: !_v \bot_{\{1,2,3\}}}$$

We do not need terms anymore, thanks to the logical structuration of the proof.

Indexed linear logic

There is a translation theorem between the quantitative type system and the indexed linear calculus.

Moreover, the derivations of the ILC can be reconstructed from ILL — that is, removing the term and the intermediate level.

$$\dfrac{\dfrac{\dfrac{(q_1,\ q_2, q_3) \in Q^2}{\perp_{\{1,2,3\}}\ \ \vdash_{\{1,2,3\}} \perp_{\{1,2,3\}}}}{!_{id} \perp_{\{1,2,3\}}\ \ \vdash_{\{1,2,3\}} \perp_{\{1,2,3\}}}}{!_{v} \perp_{\{1,2,3\}}\ \ \vdash_{\{1,2\}} !_{v} \perp_{\{1,2,3\}}}$$

We do not need the elements of the relational model. They can be reconstructed from the Axiom information (η-long form is crucial here).

Indexed linear logic

There is a translation theorem between the quantitative type system and the indexed linear calculus.

Moreover, the derivations of the ILC can be reconstructed from ILL — that is, removing the term and the intermediate level.

As a consequence, we obtain a translation between derivations in intersection type systems and relational denotations of terms.

Higher-order model checking

Two major issues of the model-checking problem were not adressed so far:

- recursion
- and parity conditions

Higher-order model checking

Recursion can be added with the rule

$$\text{Fix} \quad \frac{\Gamma \vdash_K M : [\tau_j \,|\, j \in u^{-1}(k)] \multimap \sigma_k :: !_u C \multimap A \qquad \Delta \vdash_J YM : \tau_j :: C}{\Gamma,\ !_u \Delta \vdash_K YM : \sigma_k :: A}$$

where C and A need to refine the same simple type.

This rule reflects recursion in an infinitary variant of the relational model.

Higher-order model checking

We discovered recently that the parity condition of the APT can be integrated within typing derivations.

A new modality, carrying the colouring information, is introduced. It is a comonad and thus acts in a way which is similar to the exponential of linear logic.

A notion of winning derivation is then defined — it is a parity condition over derivation trees.

Again, this leads to extensions of the relational model and of ILL, which preserve their mutual connection.

Higher-order model checking

We discovered recently that the parity condition of the APT can be integrated within typing derivations.

A new modality, carrying the colouring information, is introduced. It is a comonad and thus acts in a way which is similar to the exponential of linear logic.

A notion of winning derivation is then defined — it is a parity condition over derivation trees.

Again, this leads to extensions of the relational model and of ILL, which preserve their mutual connection.

Conclusions and perspectives

- We studied the relation between different intersection policies and logical models.
- We related
  - qualitative models and idempotent intersection types,
  - quantitative models and non-idempotent intersection types,
  - idempotent and non-idempotent intersection types, a result whose model-theoretic counterpart is Ehrhard's extensional collapse theorem.
- Logical characterizations permitted us to extend models with constructions for interpreting verification problems (recursion, infinite multisets, colouration).
- Moreover, this semantic investigation lead us to a better understanding and a clearer version of the Kobayashi-Ong type system, with strong links to game semantics.

Conclusions and perspectives

- We studied the relation between different intersection policies and logical models.
- We related
  - qualitative models and idempotent intersection types,
  - quantitative models and non-idempotent intersection types,
  - idempotent and non-idempotent intersection types, a result whose model-theoretic counterpart is Ehrhard's extensional collapse theorem.
- Logical characterizations permitted us to extend models with constructions for interpreting verification problems (recursion, infinite multisets, colouration).
- Moreover, this semantic investigation lead us to a better understanding and a clearer version of the Kobayashi-Ong type system, with strong links to game semantics.

Conclusions and perspectives

- We studied the relation between different intersection policies and logical models.
- We related
  - qualitative models and idempotent intersection types,
  - quantitative models and non-idempotent intersection types,
  - idempotent and non-idempotent intersection types, a result whose model-theoretic counterpart is Ehrhard's extensional collapse theorem.
- Logical characterizations permitted us to extend models with constructions for interpreting verification problems (recursion, infinite multisets, colouration).
- Moreover, this semantic investigation lead us to a better understanding and a clearer version of the Kobayashi-Ong type system, with strong links to game semantics.

Conclusions and perspectives

- We studied the relation between different intersection policies and logical models.
- We related
  - qualitative models and idempotent intersection types,
  - quantitative models and non-idempotent intersection types,
  - idempotent and non-idempotent intersection types, a result whose model-theoretic counterpart is Ehrhard's extensional collapse theorem.
- Logical characterizations permitted us to extend models with constructions for interpreting verification problems (recursion, infinite multisets, colouration).
- Moreover, this semantic investigation lead us to a better understanding and a clearer version of the Kobayashi-Ong type system, with strong links to game semantics.

Conclusions and perspectives

- We studied the relation between different intersection policies and logical models.
- We related
  - qualitative models and idempotent intersection types,
  - quantitative models and non-idempotent intersection types,
  - idempotent and non-idempotent intersection types, a result whose model-theoretic counterpart is Ehrhard's extensional collapse theorem.
- Logical characterizations permitted us to extend models with constructions for interpreting verification problems (recursion, infinite multisets, colouration).
- Moreover, this semantic investigation lead us to a better understanding and a clearer version of the Kobayashi-Ong type system, with strong links to game semantics.