

Compositional Higher-Order Model Checking via ω -Regular Games over Böhm Trees*

Takeshi Tsukada

University of Oxford
JSPS Postdoctoral Fellow for Research Abroad
tsukada@cs.ox.ac.uk

C.-H. Luke Ong

University of Oxford
Luke.Ong@cs.ox.ac.uk

Abstract

We introduce *type-checking games*, which are ω -regular games over Böhm trees, determined by a type of the Kobayashi-Ong intersection type system. These games are a higher-type extension of parity games over trees, determined by an alternating parity tree automaton. However, in contrast to these games over trees, the “game boards” of our type-checking games are composable, using the composition of Böhm trees. Moreover the winner (and winning strategies) of a composite game is completely determined by the respective winners (and winning strategies) of the component games.

To our knowledge, type-checking games give the first compositional analysis of *higher-order model checking*, or the model checking of trees generated by recursion schemes. We study a higher-type analogue of higher-order model checking, namely, the problem to decide the winner of a type-checking game over the Böhm tree generated by an arbitrary λY -term. We introduce a new type-assignment system and use it to prove that the problem is decidable.

On the semantic side, we develop a novel (two-level) arena game model for type-checking games, which is a cartesian closed category equipped with parametric monad and comonad that themselves form a parametrised adjunction.

Categories and Subject Descriptors Theory of computation [Semantics and reasoning]: Program constructs—Type structures; Theory of computation [Semantics and reasoning]: Program semantics

General Terms Theory, Verification

Keywords Higher-order model checking, Intersection type, Game semantics, Böhm tree

1. Introduction

Higher-order model checking (HOMC), or the model checking problem for trees generated by recursion schemes, is a widely studied problem in connection with the verification of higher-order programs [10, 18]. With respect to monadic second-order (MSO) prop-

erties, the model checking of trees generated by recursion schemes was first proved to be decidable in 2006 [15], using game semantics [9]. A variety of semantic and algorithmic techniques, and models of computation have since been employed to study HOMC: notably, intersection types [11], collapsible pushdown automata [8] and Krivine machines [19]. Algorithmic properties that refine and extend the MSO decidability result have also been introduced, such as *logical reflection* [1], *effective selection* [2, 7] and *transfer theorem* [20].

In both theory and practice, model checking is mainly a whole program analysis. Perhaps surprisingly, this is also true of HOMC. There are multiple decision procedures that can model check trees generated by ground-type functional programs, but not by higher-type programs, which generate *trees with binders* i.e. Böhm trees. Indeed it is open as to what is an appropriate higher-type analogue of HOMC *qua* decision problem. It seems natural to consider a Böhm tree as a graph with additional edges from bound variables to their binders, but the MSO theory is undecidable even for a λY -definable Böhm tree [4].

Symptomatic of our limited understanding of HOMC is a lack of a denotational (i.e. compositional) semantics of higher-order programs suitable for the direct analysis of model checking algorithms. A precise and abstract formulation of such a compositional semantics would be a cartesian closed category of parity games.

Our goal, then, is to build a compositional framework for HOMC, using the λY -calculus as our programming language. The primary task is to identify a well-behaved set $\mathcal{C}$ of properties τ over Böhm trees U , and a mechanism for a satisfaction relation $\models U : \tau$ whose validity is characterised by a system of witnesses $\mathfrak{s}$. We list some desiderata.

- (1) The satisfaction relation $\models$ should *conservatively extend* the MSO properties of trees (i.e. order-2 Böhm trees).
- (2) *Decidability of λY -definable Böhm trees*: It should be decidable, given a λY -term M and $\tau \in \mathcal{C}$, whether $\models \text{BT}(M) : \tau$, writing $\text{BT}(M)$ for the Böhm tree of M .
- (3) *Two-Level Compositionality*: Suppose Böhm trees U and V are composable, the set of properties satisfied by $U \circ V$ should be completely determined by those of U and of V . Further, if $\models U : \tau$ and $\models V : \sigma$ imply $\models U \circ V : \delta$, the witnesses $\mathfrak{s}_U^\tau$ of $\models U : \tau$ and $\mathfrak{s}_V^\sigma$ of $\models V : \sigma$ should be composable, generating a witness $\mathfrak{s}_U^\tau \circ \mathfrak{s}_V^\sigma$ of $\models U \circ V : \delta$.
- (4) *Effective Selection*: If $\models \text{BT}(M) : \tau$, then there should exist, constructively, a λY -definable witness of $\models \text{BT}(M) : \tau$.

To this end, our choice of $\mathcal{C}$ is the set of intersection types of the Kobayashi-Ong type system [11], as formulas of Böhm trees; we introduce a new notion of 2-player game over Böhm trees, called *type-checking games*, as the mechanism of a satisfaction relation

* This work was partially supported by JSPS KAKENHI 23220001.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9...\$15.00.
<http://dx.doi.org/10.1145/2603088.2603133>

whose validity is witnessed by P having a winning strategy for the game in question.

1.1 Contributions of the paper

We start from an algebraic abstraction of the parity conditions, called *left-closed ω -monoid*. This general notion—every ω -regular winning condition can be expressed by using a finite left-closed ω -monoid—underpins all subsequent developments: our type-checking games (Sec. 3), the concomitant type system (Sec. 4), effect arena and two-level games (Sec. 5 and 6) are all parametrised on a left-closed ω -monoids and a finite set of ground types.

We introduce *type-checking games*, which are a kind of higher-type property-checking games played over a Böhm tree, the property in question being an intersection type of the Kobayashi-Ong type system [11]. On the one hand, type-checking games generalise the property-checking games (such as parity games) which are played over trees, to games which are played over *trees with binders*; on the other, they may be viewed as an extension of Stirling’s *dependency tree automata* [21] to infinite (binding) trees with ω -regular conditions. The table below summarises the various aspects of types as a higher-order analogue of alternating parity tree automata (APT).

Trees and APT	Böhm trees and Types
Tree constructors	Order-1 free variables
Trees	Order-2 Böhm trees
Recursion schemes	λY -terms with order-1 free vars
Value tree of a grammar	Böhm tree of a term
MSO properties / APT	Order-2 types
A tree accepted by an automaton	A Böhm tree of a type

A remarkable feature of type-checking games is their *intensional* or *two-level compositionality*. Given two such games of composable simple types, not only are their underlying “game boards” composable as Böhm trees, the winning strategies of the composite game are completely determined by the respective winning strategies of the component games (Theorem 11). A key to this result is Lemma 10: certain (but enough) winning strategies of type-checking games are themselves representable as Böhm trees.

We introduce a new intersection type system that characterises the winner of type-checking games for λY -definable Böhm trees. Since HOMC can be regarded as the 2nd-order fragment of the type-checking games, this type system—which is decidable (Theorem 17)—also characterises HOMC, and may be viewed as a conservative extension and simplification of the Kobayashi-Ong type system. Precisely we prove a *transfer theorem* (Theorem 18): P wins the type-checking game determined by a given intersection type τ over the Böhm tree of a λY -term iff that term is typable by τ . The second major result is a higher-order version of *effective selection* (Theorem 20): given an inhabited intersection type, we can effectively construct a λY -term whose Böhm tree represents a P-winning strategy of the type-checking game of that type.

The semantic foundation, and the ultimate basis of the results mentioned above, is a new category of *effect arena games*. To our knowledge, this is the first example of a cartesian closed category of games and winning strategies of ω -regular conditions. We then construct a *two-level game semantics* (in the sense of [16]) as an accurate model of our type system. Our categorical constructions give HOMC examples of positive and negative actions [14] and of parametrised adjunction.

1.2 Related work

Salvati and Walukiewicz [20] have established a transfer theorem for (infinitary) λY -terms: for a fixed vocabulary of terms, the MSO properties of Böhm trees of terms are effectively MSO properties of the terms themselves. However their theorem is only applicable

to *Böhm trees without binders* (i.e. order-2 Böhm trees in our terminology). This restriction seems essential to their development. Their proof of the transfer theorem relies on the (effective) equivalence between MSOL and non-deterministic parity automata for defining tree languages. However, no such equi-expressivity results for defining Böhm-tree languages are known. Moreover, MSOL is undecidable on the binding structures of λY -definable Böhm trees [4]. In contrast, our framework, and in particular our type-based transfer theorem (Theorem 18), work for Böhm trees of all orders definable by (finitary) λY -terms.

Haddad [7] has given a new proof of *effective selection* [2] for recursion schemes, using a notion of shape-preserving scheme transformation. We (Theorem 20) extend Haddad’s result to all higher orders, based on a derivation-as-term interpretation of a new intersection type system. This approach can be found in the work of van Bakel [23] and of Kobayashi et al. [12], but applied to different problems.

Further details and all proofs are in the downloadable long version [22].

2. Monoidal-Closed Structure of Priorities

This section introduces the notion of *left-closed ω -monoid*, abstracting the algebraic properties of the parity condition (see Example 3). Types and games of this paper are parametrised by a left-closed ω -monoid.

First we review the notion of ordered ω -semigroups [17]. We write $\langle e_1, e_2, \dots \rangle$ or $\langle e_i \rangle_{i \in \omega}$ for infinite sequences.

Definition 1 (Ordered ω -semigroup). A (partially) ordered ω -semigroup is a pair $(\mathbb{E}, \mathbb{F})$, where $\mathbb{E}$ and $\mathbb{F}$ are partially ordered sets equipped with the following operations:

- An associative binary operation $\circ : \mathbb{E} \times \mathbb{E} \rightarrow \mathbb{E}$.
- An operation $\otimes : \mathbb{E} \times \mathbb{F} \rightarrow \mathbb{F}$ that defines a semigroup action of $\mathbb{E}$ to $\mathbb{F}$ (i.e. $e_1 \otimes (e_2 \otimes f) = (e_1 \circ e_2) \otimes f$).
- A surjective mapping $\pi : \mathbb{E}^\omega \rightarrow \mathbb{F}$ called *infinite product*.

All the operations must respect the partial orders i.e.

- (i) $e_1 \preceq e'_1$ and $e_2 \preceq e'_2$ implies $e_1 \circ e_2 \preceq e'_1 \circ e'_2$,
- (ii) $e \preceq e'$ and $f \preceq f'$ implies $e \otimes f \preceq e' \otimes f'$, and
- (iii) $e_i \preceq e'_i$ for every i implies $\pi \langle e_i \rangle_{i \in \omega} \preceq \pi \langle e'_i \rangle_{i \in \omega}$.

Furthermore the infinite product must satisfy the following laws:

- (i) $e_0 \otimes \pi \langle e_1, e_2, \dots \rangle = \pi \langle e_0, e_1, e_2, \dots \rangle$.
- (ii) If $\langle e_i \rangle_{i \in \omega} \rightsquigarrow \langle e'_j \rangle_{j \in \omega}$, then $\pi \langle e_i \rangle_{i \in \omega} = \pi \langle e'_j \rangle_{j \in \omega}$.

Here $\rightsquigarrow$ is defined by

$$\langle e_1, \dots, e_{i_1}, e_{i_1+1}, \dots, e_{i_2}, e_{i_2+1}, \dots, e_{i_3}, \dots \rangle \rightsquigarrow \langle (e_1 \circ \dots \circ e_{i_1}), (e_{i_1+1} \circ \dots \circ e_{i_2}), (e_{i_2+1} \circ \dots \circ e_{i_3}), \dots \rangle$$

for every infinite chain $0 < i_1 < i_2 < \dots$ of natural numbers.

An *ordered ω -monoid* is an ordered ω -semigroup $(\mathbb{E}, \mathbb{F})$ such that $\mathbb{E}$ has an identity element. We write ε for the identity element. We have $\varepsilon \otimes f = f$ for every $f \in \mathbb{F}$. An ordered monoid $\mathbb{E}$ has *left-residuals* if there exists a binary operation $e \setminus e'$ s.t. $e \circ d \preceq e'$ iff $d \preceq e \setminus e'$. Then $\mathbb{E}$ as the preorder category is a left-closed (strict) monoidal category.

Definition 2 (Left-closed ω -monoid / winning condition). A *left-closed ω -monoid* is an ordered ω -monoid $(\mathbb{E}, \mathbb{F})$ with $\mathbb{E}$ having left-residuals. A *winning condition* is a triple $(\mathbb{E}, \mathbb{F}, \Omega)$, where $(\mathbb{E}, \mathbb{F})$ is a left-closed ω -monoid and $\Omega \subseteq \mathbb{F}$ is a lower-closed subset of $\mathbb{F}$ (i.e. $f \in \Omega$ and $f' \preceq f$ implies $f' \in \Omega$).

A winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$ determines the winner of an infinite play associated with an infinite sequence $\langle e_i \rangle_{i \in \omega}$ of elements in $\mathbb{E}$: we say that P wins the play just if $\pi \langle e_i \rangle_{i \in \omega} \in \Omega$.

Given a winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$, we say $\Omega \subseteq \mathbb{F}$ is *stable* if for every $e \in \mathbb{E}$ and $f \in \mathbb{F}$, $f \in \Omega \iff e \otimes f \in \Omega$. A winning condition is *finite* if $\mathbb{E}$ and $\mathbb{F}$ are finite.

Example 3. The parity condition is a finite and stable winning condition. Let $\mathbb{P} = \{0, 1, 2, \dots, 2N\}$ be the set of priorities and $\Omega_{\text{parity}} \subseteq \mathbb{P}^\omega$ be the set of all infinite sequences satisfying the parity condition (i.e. the largest priority that occurs infinitely often is even). The priorities have a monoidal product $e_1 \circ e_2 = \max(e_1, e_2)$ with 0 as the identity. Consider the *sub-priority* order [6]:

$$2N \prec \dots \prec 4 \prec 2 \prec 0 \prec 1 \prec 3 \prec \dots \prec 2N - 1.$$

The monoid product respects the sub-priority order. Priorities have left-residuals, defined by:

$$(e \setminus e') := \begin{cases} e' & \text{if } e < e' \text{ or } (e = e' \text{ and } (e \text{ is odd or } 0)) \\ e + 1 & \text{if } e > e' \text{ and } e \text{ is odd} \\ e - 1 & \text{if } e \geq e' \text{ and } e \text{ is even and } e > 0. \end{cases}$$

Consider a winning condition $(\mathbb{P}, \{\mathbf{e}, \mathbf{o}\}, \{\mathbf{e}\})$ with $\mathbf{e} \prec \mathbf{o}$. The infinite product is defined as $\pi \langle e_1, e_2, \dots \rangle = \mathbf{e}$ just if $\langle e_1, e_2, \dots \rangle \in \Omega_{\text{parity}}$ and the mixed product is defined by $e \otimes f = f$. It is finite and $\Omega = \{\mathbf{e}\}$ is stable.

It is well-known that finite ω -semigroups coincide with ω -regular languages [17]. By a similar argument, every ω -regular winning condition can be viewed as a finite winning condition and vice versa. In particular, for every finite winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$, the ω -language $\{\langle e_i \rangle_{i \in \omega} \mid \pi \langle e_i \rangle_{i \in \omega} \in \Omega\}$ is ω -regular.

In the sequel, for simplicity, Ω is assumed to be stable. See the long version [22] for the general case.

3. Type-Checking Games

This section introduces formulas describing the properties of Böhm trees, which may be viewed as a (slight) extension of the Kobayashi-Ong type system [11]. The main result is compositionality (Theorem 11), which will be proved by game semantics in Section 6.2 (proofs of other results are in the long version [22]).

Henceforth we fix a stable winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$ and a finite set Q of ground types.

3.1 Preliminary: Böhm trees

We briefly review the notion of simply-typed Böhm trees. See the long version [22] for the formal definitions.

First, we define simple types, which we shall refer to as *kinds* in order to avoid confusion with (intersection) types to be introduced later. The set of kinds is defined inductively: $\kappa ::= \mathbf{o} \mid \kappa \rightarrow \kappa$, where $\mathbf{o}$ is the unique base kind and $\kappa_1 \rightarrow \kappa_2$ is the kind for functions from κ_1 to κ_2 . A kind environment is a finite sequence of kind bindings of the form $x :: \kappa$. Kind environments are ranged over by Δ . We use $::$ for kind bindings and kind judgements in order to distinguish them from type bindings and type judgements.

We assume the standard notion of Böhm trees, which are possibly infinite trees with binders, defined co-inductively by the grammar:

$$\begin{aligned} T &::= \perp \mid x U_1 \dots U_k \\ U &::= \lambda x_1 \dots \lambda x_k. T \end{aligned}$$

where $\perp$ (meaning divergence) is a special symbol of kind $\mathbf{o}$. Böhm trees are required to be well-kinded and η -long. Kind judgements $\Delta \vdash T :: \mathbf{o}$ and $\Delta \vdash U :: \kappa$ have the expected meaning. Böhm

trees of appropriate kinds can compose: for Böhm trees U and V of kinds $\kappa \rightarrow \kappa'$ and κ respectively, we write $U @ V$ for the application of U to V , which has kind κ' . It is well-known [5, 9] that Böhm trees of a given kind corresponds bijectively to innocent strategies of the corresponding arena. This bijection bridges the gap between the Böhm-tree representation in this section and the game-semantic analysis in the following sections.

Böhm trees subsume ordinary (node-labelled, ranked) trees. Let $\Sigma = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ be a set of tree constructors and $\text{ar}(\mathbf{a}_i)$ be the arity of the constructor $\mathbf{a}_i$. We write $\Sigma^\perp$ for $\Sigma \uplus \{\perp\}$, where $\perp$ is of arity 0. A *tree over* $\Sigma^\perp$ is just a Böhm tree of 2nd-order kind $(\mathbf{o}^{\text{ar}(\mathbf{a}_1)} \rightarrow \mathbf{o}) \rightarrow \dots \rightarrow (\mathbf{o}^{\text{ar}(\mathbf{a}_n)} \rightarrow \mathbf{o}) \rightarrow \mathbf{o}$.

3.2 Kobayashi-Ong types

We use types of the Kobayashi-Ong type system [11] as formulas specifying properties of Böhm trees. Our syntax simplifies the original system [11] by removing flags from the type environments, and extends it by being parametrised by winning conditions. Furthermore we introduce new operations on types, namely, positive and negative actions of effects.

There are two classes of types, *prime types* and *intersection types*, defined by the following grammar:

$$\begin{aligned} (\text{Prime Types}) \quad \tau, \sigma &::= q \mid \alpha \rightarrow \tau \\ (\text{Intersection Types}) \quad \alpha, \beta &::= \bigwedge_{i \in I} \langle \tau_i; e_i \rangle \end{aligned}$$

where $q \in Q$, $e_i \in \mathbb{E}$ and $I = \emptyset$, ω or $\{1, 2, \dots, k\}$ for some $k \in \omega$. Precisely, the set of types is defined by induction on the structure of kinds. See the refinement relation below.

We use the infix notation for intersection, e.g., $\langle \tau_1; e_1 \rangle \wedge \langle \tau_2; e_2 \rangle$ means $\bigwedge_{i \in \{1, 2\}} \langle \tau_i; e_i \rangle$. The intersection connective $\wedge$ is *neither* associative, commutative nor idempotent (e.g. $\langle \tau_1; e_1 \rangle \wedge \langle \tau_2; e_2 \rangle \neq \langle \tau_2; e_2 \rangle \wedge \langle \tau_1; e_1 \rangle$ and $\langle \tau; e \rangle \wedge \langle \tau; e \rangle \neq \langle \tau; e \rangle$). However these laws are true modulo the equivalence induced by the subtyping relation (defined below). Note that we allow intersection of countably infinite types $\bigwedge_{i \in \omega} \langle \tau_i; e_i \rangle$. A type is *finite* if all the index sets are finite. We define the projection p_k by $p_k(\bigwedge_{i \in I} \langle \tau_i; e_i \rangle) := \langle \tau_k; e_k \rangle$ for each $k \in I$.

We call an element $e \in \mathbb{E}$ an *effect*. The effect e of $\langle \tau; e \rangle$ is the summary of effects from a certain point to the call of τ . For example, e in the judgement $U : \langle \tau; e \rangle$ is the summary of effects from the beginning of the computation to the call of U , and e in $\lambda x. T : \langle \tau; e \rangle \rightarrow q$ is the summary of effects from the call of $\lambda x. T$ to the call of its argument x .

The Kobayashi-Ong type system considers only types that *refine* kinds. The judgement $\tau :: \kappa$ (resp. $\alpha :: \kappa$) means that the prime type τ (resp. the intersection type α) refines the kind κ . The derivation rules are as follows.

$$\frac{}{q :: \mathbf{o}} \quad \frac{\alpha :: \kappa \quad \tau :: \kappa'}{\alpha \rightarrow \tau :: \kappa \rightarrow \kappa'} \quad \frac{\forall i \in I. \tau_i :: \kappa}{\bigwedge_{i \in I} \langle \tau_i; e_i \rangle :: \kappa}$$

The *subtyping relation* is defined by the following mostly standard rules. Precisely it is a kind-indexed family of relations on prime types of the same kind and on intersection types of the same kind. We write $\tau \approx \tau'$ if $\tau \preceq \tau'$ and $\tau' \preceq \tau$.

$$\begin{aligned} \frac{}{q \preceq q} \quad \frac{\alpha' \preceq \alpha \quad \tau \preceq \tau'}{\alpha \rightarrow \tau \preceq \alpha' \rightarrow \tau'} \\ \frac{\forall j \in J. \exists i \in I. (\tau_i \preceq \sigma_j \text{ and } e_i \succeq d_j)}{\bigwedge_{i \in I} \langle \tau_i; e_i \rangle \preceq \bigwedge_{j \in J} \langle \sigma_j; d_j \rangle} \end{aligned}$$

Note that the effect annotation is *contravariant*. Intuitively this is because an effect describes a property of the *caller* (the context) who is contravariant to the callee (the term).

A *type environment* Γ is a finite sequence of intersection type bindings of the form $x : \alpha$ such that the binding variables are pairwise distinct. We write $\Gamma(x) = \alpha$ if $x : \alpha$ is in Γ . The *refinement relation* and the *subtyping relation* for environments are both defined by point-wise extension.

The *positive action* [14] of $\mathbb{E}$ to intersection types and type environments is defined by:

$$e \circledast \left(\bigwedge_{i \in I} \langle \tau_i; d_i \rangle \right) := \bigwedge_{i \in I} \langle \tau_i; e \circ d_i \rangle \quad (e \circledast \Gamma)(x) := e \circledast (\Gamma(x)).$$

Similarly the *negative action* $e \setminus -$ can be defined, e.g.,

$$e \setminus \left(\bigwedge_{i \in I} \langle \tau_i; d_i \rangle \right) := \bigwedge_{i \in I} \langle \tau_i; e \setminus d_i \rangle.$$

They form a Galois connection by the definition of $e \setminus e'$:

$$(e \setminus \Gamma_1) \preceq \Gamma_2 \text{ iff } \Gamma_1 \preceq (e \circledast \Gamma_2).$$

Here $e \setminus -$ is the left-adjoint since the effect annotation is contravariant. The positive action on type environments simplifies and generalises the flag-updating operation in the original type system [11] and the priority-updating operation in the work of Fujima et al. [6]. Its left adjoint (the negative action) is new, to the best of our knowledge.

3.3 Type-checking games over Böhm trees

It is well-known that the problem of whether a tree is accepted by a given parity tree automaton is equivalent to solving a certain parity game over the tree. This subsection extends such games over trees to games over Böhm trees. Given a Böhm tree U of kind κ and a type $\alpha :: \kappa$, we shall define a *type-checking game*, which is a game between Proponent and Opponent (henceforth, simply P and O) over the Böhm tree U . This is a conservative extension of games over trees, which can be considered as type-checking games restricted to 2nd-order types.

Definition 4 (Type-checking games). A node of the game graph is called a *position*. A position is of the form

$$\Gamma \models T : q \quad \text{or} \quad \Gamma \models (U_1, \dots, U_n) : (\alpha_1, \dots, \alpha_n)$$

where $n \geq 0$. The former is a *P-position* and the latter, written $\Gamma \models (U_i)_{i \leq n} : (\alpha_i)_{i \leq n}$, is an *O-position*. Positions must be kind-respecting: for the former, $\Gamma :: \Delta$ for some Δ , and $\Delta \vdash T :: o$; and for the latter, $\Gamma :: \Delta$ and $\Delta \models U_i :: \kappa_i$ and $\alpha_i :: \kappa_i$ for every i . In our game, *effects* (corresponding to *priorities* or *colours* in the standard terminology) are assigned to edges, instead of positions. There are three kinds of edges. The first kind has the form

$$(\Gamma \models x U_1 \dots U_l : q) \xrightarrow{\varepsilon} (\Gamma \models (U_1, \dots, U_l) : (\alpha_1, \dots, \alpha_l))$$

where $\mathbf{p}_k(\Gamma(x)) = \langle \alpha_1 \rightarrow \dots \rightarrow \alpha_l \rightarrow q; e \rangle$ with $e \succeq \varepsilon$ for some k . The second kind of edges has the form

$$(\Gamma \models (U_i)_{i \leq n} : (\alpha_i)_{i \leq n}) \xrightarrow{e} ((e \setminus \Gamma), \tilde{x} : \tilde{\beta} \models T : q)$$

where, for some $j \leq n$ and k , $U_j = \lambda x_1 \dots \lambda x_l. T$ and $\mathbf{p}_k(\alpha_j) = \langle \beta_1 \rightarrow \dots \rightarrow \beta_l \rightarrow q; e \rangle$ and $\tilde{x} : \tilde{\beta}$ means $x_1 : \beta_1, \dots, x_l : \beta_l$. The third kind of edges is for divergence,

$$(\Gamma \models \perp : q) \xrightarrow{\varepsilon} (\Gamma \models \perp : q).$$

So the unique infinite play starting from $(\Gamma \models \perp : q)$ has effects $\langle \varepsilon, \varepsilon, \dots \rangle$. The winner of an infinite play is determined by the effects of the path: P wins just if $\pi \langle e_1, e_2, \dots \rangle \in \Omega$, where $\langle e_1, e_2, \dots \rangle$ is the sequence of effects along the path. If a finite maximal play ends at a P-position (resp. O-position) then O (resp. P) wins.

Specifying the initial position determines a game. By abuse of notation, we write $\Gamma \models T : q$ and $\Gamma \models U : \alpha$ for the games starting

from these positions. Further when the game $\Gamma \models T : q$ has a winning P-strategy, we say $\Gamma \models T : q$ is *valid* and write $\Gamma \models T : q$. We abbreviate $\Gamma \models (U) : (\alpha)$ to $\Gamma \models U : \alpha$ and $\Gamma \models U : \langle \tau; \varepsilon \rangle$ to $\Gamma \models U : \tau$.

In edges from P-positions, P can use only types annotated by ε (or greater). This means that variable x is used immediately without making any effect. The effect ε in the first kind of edges is “meaningless” since $\pi \langle e_1, \varepsilon, e_2, \varepsilon, e_3, \varepsilon, \dots \rangle = \pi \langle e_1, e_2, e_3, \dots \rangle$.

The second rule can be understood as a combination of three moves. First O chooses a component $j \leq n$ and an index k of the intersection type α_j , resulting in

$$(\Gamma \models (U_i)_{i \leq n} : (\alpha_i)_{i \leq n}) \xrightarrow{\varepsilon} (\Gamma \models U_j : \mathbf{p}_k(\alpha_j)).$$

Let $\langle \tau; e \rangle = \mathbf{p}_k(\alpha_j)$. Then the effect e acts:

$$(\Gamma \models U_j : \langle \tau; e \rangle) \xrightarrow{e} (e \setminus \Gamma \models U_j : \tau).$$

Lastly new variables are introduced:

$$(e \setminus \Gamma \models U_j : \tau) \xrightarrow{\varepsilon} ((e \setminus \Gamma), \tilde{x} : \tilde{\beta} \models T : q),$$

where $U = \lambda \tilde{x}. T$ and $\tau = \beta_1 \rightarrow \dots \rightarrow \beta_l \rightarrow q$.

The third rule says that the divergence makes infinitely many ε effects. The ε effects here is significant: P wins the play if and only if $\pi \langle \varepsilon, \varepsilon, \dots \rangle \in \Omega$.

Example 5. Let $\Delta = a :: o, b :: o \rightarrow ((o \rightarrow o) \rightarrow o) \rightarrow o$. Consider the order-4 Böhm tree U_0 , due to Clairambault and Murawski [4], as shown in Figure 1. Notice that in U_0 , each bound variable occurs infinitely often, and infinitely many names are required to represent variable binding. Consider the tree property $\varphi = \text{“there are only finitely many occurrences of bound variables in each branch”}$. Let U be a Böhm tree satisfying $\Delta \vdash U :: (o \rightarrow o) \rightarrow o$. Take the parity winning condition consisting of effects $2 \prec 0 \prec 1$, and a single ground type q . Then U satisfies φ iff $\Gamma \models U : (1 \rightarrow 1) \rightarrow 0$ where $\Gamma = a : 1, b : 0 \rightarrow ((1 \rightarrow 1) \rightarrow 0) \rightarrow 1$.¹ Observe that, because the effect at each contravariant position in the typing judgement is 1 and the identity effect $\varepsilon = 0 \preceq 1$, the typing does not restrict where a bound variable may occur in the input tree U . Intuitively, the effect 1 (at the underlined covariant position $\underline{1}$) in the type of b accounts for each occurrence of a variable introduced by b .

Consider the type-checking game $\Gamma \models U_0 : (1 \rightarrow 1) \rightarrow 0$. Now $(\Gamma \models (U_0) : ((1 \rightarrow 1) \rightarrow 0)) \xrightarrow{0} (0 \setminus \Gamma, x_1 : 1 \rightarrow 1 \models b(x_1 a) (\lambda x_2. -) : q)$. Since $0 \setminus \Gamma = \Gamma$ and $\varepsilon = 0 \preceq 1$, we have $(\Gamma, x_1 : 1 \rightarrow 1 \models b(x_1 a) (\lambda x_2. -) : q) \xrightarrow{\varepsilon} (\Gamma, x_1 : 1 \rightarrow 1 \models (x_1 a, \lambda x_2. -) : (0, (1 \rightarrow 1) \rightarrow 0))$. Then, with O choosing the right branch and then the left, we have

$$\begin{aligned} & (\Gamma, x_1 : 1 \rightarrow 1 \models (x_1 a, \lambda x_2. -) : (0, (1 \rightarrow 1) \rightarrow 0)) \\ & \xrightarrow{0} (\Gamma, x_1, x_2 : 1 \rightarrow 1 \models b(x_2 (x_1 a)) (\lambda x_3. -) : q) \quad \because \varepsilon \preceq 1 \\ & \xrightarrow{\varepsilon} (\Gamma, x_1, x_2 : 1 \rightarrow 1 \models (x_2 (x_1 a), (\lambda x_3. -)) : (0, (1 \rightarrow 1) \rightarrow 0)) \\ & \xrightarrow{0} (\Gamma, x_1, x_2 : 1 \rightarrow 1 \models x_2 (x_1 a) : q) \\ & \xrightarrow{\varepsilon} (\Gamma, x_1, x_2 : 1 \rightarrow 1 \models (x_1 a) : (1)) \\ & \xrightarrow{1} (\Gamma' \models x_1 a : q) \text{ where } \Gamma' = 1 \setminus (\Gamma, x_1, x_2 : 1 \rightarrow 1) \\ & \xrightarrow{\varepsilon} (\Gamma' \models (a) : (1)) \\ & \xrightarrow{1} (1 \setminus \Gamma' \models a : q) \quad \because (1 \setminus \Gamma')(a) = 1 \setminus 1 = 1 \text{ and } \varepsilon \preceq 1 \\ & \xrightarrow{\varepsilon} (1 \setminus \Gamma' \models () : ()) \end{aligned}$$

and P wins the play. Note that P has a winning strategy.

¹ Since the ground type is unique, we can abbreviate types α to $\bar{\alpha}$ whereby $\langle \alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow q; e \rangle := \bar{\alpha}_1 \rightarrow \dots \rightarrow \bar{\alpha}_n \rightarrow e$ and $\bar{\alpha} \wedge \bar{\beta} := \bar{\alpha} \wedge \bar{\beta}$.

- (ii) If $\widehat{W} \blacktriangleright (\Gamma \models U_1 @ U_2 : \tau)$, then $\widehat{V}_1 \blacktriangleright (\Gamma \models U_1 : \alpha \rightarrow \tau)$ and $\widehat{V}_2 \blacktriangleright (\Gamma \models U_2 : \alpha)$ and $\widehat{W} = \widehat{V}_1 @ \widehat{V}_2$ for some α , $\widehat{V}_1$ and $\widehat{V}_2$.

4. Type System for Model-Checking Böhm Trees

This section studies (decision procedures for) the *Model-Checking Problem for λY -definable Böhm Trees*, i.e., whether $\models \text{BT}(M) : \tau$ for a given λY -term M and type τ , where $\text{BT}(M)$ is the Böhm tree of M . We develop a type assignment system that characterises the winner of type-checking games for λY -definable Böhm trees. Since the standard higher-order model-checking can be regarded as the second-order fragment of the type-checking games, the type system also characterises higher-order model checking (for tree-generating λY -terms).

In this section, we assume that $(\mathbb{E}, \mathbb{F}, \Omega)$ is finite. Otherwise some algorithmic results would fail.

4.1 Preliminary: λY -calculus

The set of *terms* is given by:

$$M ::= x \mid M M \mid \lambda x. M \mid \mathbf{Y}_\kappa$$

where $\mathbf{Y}_\kappa$ is the fixed-point combinator of kind $(\kappa \rightarrow \kappa) \rightarrow \kappa$. The kinding rules are defined as usual. We write $M[N/x]$ for the standard capture-avoiding substitution. The reduction relation is given by the following rules

$$(\lambda x. M) N \longrightarrow M[N/x] \quad \mathbf{Y} M \longrightarrow M(\mathbf{Y} M),$$

with the η -expansion rule and the congruence rule: if $M \longrightarrow M'$, then $M N \longrightarrow M' N$ and $N M \longrightarrow N M'$.

A λY -term M generates the Böhm tree $\text{BT}(M)$ as the result of infinite reductions. An important fact is that $\text{BT}(-)$ and the application commute in the following sense.

Proposition 12. *Let $(M N)$ be a well-kinded λY -term. Then $\text{BT}(M N) = \text{BT}(M) @ \text{BT}(N)$.*

4.2 Typing rules

Definition 13. Assume a kinded term $\Delta \vdash M :: \kappa$. There are three kinds of judgements for M : *prime type judgement* $\Gamma \vdash M : \tau$, *prime effect judgement* $\Gamma \vdash M : \langle \tau; e \rangle$, and *intersection type judgement* $\Gamma \vdash M : \alpha$, where $\Gamma :: \Delta$, $\tau :: \kappa$ and $\alpha :: \kappa$. The typing rules are as follows:

$$\begin{array}{c} \frac{\langle \tau; e \rangle = \mathbf{p}_i(\Gamma(x)) \text{ for some } i \text{ and } e \succeq \varepsilon}{\Gamma \vdash x : \tau} \quad \frac{\Gamma, x : \alpha \vdash M : \tau}{\Gamma \vdash \lambda x. M : \alpha \rightarrow \tau} \\[10pt] \frac{\Gamma \vdash M : \alpha \rightarrow \tau \quad \Gamma \vdash N : \alpha}{\Gamma \vdash M N : \tau} \quad \frac{\Gamma' \preceq \Gamma \quad \Gamma \vdash M : \tau \quad \tau \preceq \tau'}{\Gamma' \vdash M : \tau'} \\[10pt] \frac{\Gamma \vdash M : \tau}{e @ \Gamma \vdash M : \langle \tau; e \rangle} \quad \frac{\forall i \in I. \Gamma \vdash M : \langle \tau_i; e_i \rangle}{\Gamma \vdash M : \bigwedge_{i \in I} \langle \tau_i; e_i \rangle} \quad \frac{}{\Gamma \vdash \mathbf{Y} : \tau} \end{array}$$

and the subtyping rule for $\Gamma \vdash M : \langle \tau; e \rangle$. The variable rule requires x to have type τ with effect ε , since x is used immediately without any effects. This rule is equivalent to the rule

$$\frac{}{x : \langle \tau; \varepsilon \rangle \vdash x : \tau}$$

followed by the subtyping rule with $\Gamma \preceq (x : \langle \tau; \varepsilon \rangle)$. The positive-action rule applies e -action to both sides. This rule would be unsound or incomplete if Ω were not stable. The rule for $\mathbf{Y}$ just checks if the judgement is true in the Böhm-tree semantics. Without relying on the type-checking game, we can define an equivalent game for $\mathbf{Y}$ on a finite graph.

Definition 14 (Game for $\mathbf{Y}$). Given a prime type τ of kind $(\kappa \rightarrow \kappa) \rightarrow \kappa$, we define the game $G(\tau)$ as follows. Here types are

considered modulo $\approx$. A P-position has the form $\alpha \models_P \sigma$, where $\alpha :: \kappa \rightarrow \kappa$ and $\sigma :: \kappa$, and an O-position has the form $\alpha \models_O \beta$, where $\alpha :: \kappa \rightarrow \kappa$ and $\beta :: \kappa$. Recall that σ, τ (resp. α, β) range over prime (resp. intersection) types. Edges are defined by:

$$(\alpha \models_P \tau) \xrightarrow{\varepsilon} (\alpha \models_O \beta),$$

whenever $\mathbf{p}_k(\alpha) = \langle \beta \rightarrow \tau'; e \rangle$ with $e \succeq \varepsilon$ and $\tau' \preceq \tau$ for some k , and

$$(\alpha \models_O \beta) \xrightarrow{e} (e \parallel \alpha \models_P \tau)$$

whenever $\mathbf{p}_k(\beta) = \langle \tau; e \rangle$ for some k . The initial position of $G(\alpha \rightarrow \tau)$ is $\alpha \models_P \tau$. The winner of an infinite play is determined by the (infinite sequence of) effects along the play.

Lemma 15. *P wins $\models \text{BT}(\mathbf{Y}) : \tau$ iff P wins $G(\tau)$.*

Remark 16. Though our game $G(\tau)$ has the same flavour as the game used in the Kobayashi-Ong system [11], it requires careful effect handling. That is because $\mathbf{Y}$ can be applied to an open term, whereas the recursion of recursion schemes always takes closed terms. For a closed term M , $\vdash M : \tau$ is equivalent to $\vdash M : \langle \tau; e \rangle$ for every e , so the effect of this position is not significant. In contrast, as for open terms, the effect e of $\Gamma \vdash M : \langle \tau; e \rangle$ affects Γ . This is handled by the left-residual operation $e \parallel \alpha$ in edges from O-positions.

Assuming a finite winning condition, the set of types modulo $\approx$ is finite for each kind. It follows that $G(\tau)$ is an ω -regular game on a finite graph; hence $G(\tau)$ is decidable. Further, for every term $\Delta \vdash M :: \kappa$, there are only finitely many type judgements $\Gamma \vdash M : \tau$ modulo $\approx$. Therefore, by an induction on the structure of the term, the type-checking problem is decidable.

Theorem 17 (Decidability). *$\Gamma \vdash M : \tau$ is decidable.*

The type system is sound and complete with respect to the type-checking games. The proof is by induction on the structure of the term, using Proposition 12 and Theorem 8.

An *order- n transfer theorem* for a set $\mathcal{C}$ of properties says that the $\mathcal{C}$ -definable properties of order- n λY -definable Böhm trees are effectively the $\mathcal{C}$ -definable properties of the term-generators themselves. See the long version [22] for further details.

Theorem 18 (Transfer). *$\Gamma \vdash M : \tau$ iff $\Gamma \models \text{BT}(M) : \tau$.*

Proof. (Sketch) If $M = \mathbf{Y}$, both directions are trivial by the typing rule. Assume $M = M_1 M_2$. If $\Gamma \vdash M_1 M_2 : \tau$, then $\Gamma \vdash M_1 : \alpha \rightarrow \tau$ and $\Gamma \vdash M_2 : \alpha$ for some α by the typing rules. By the induction hypothesis, we have $\Gamma \models \text{BT}(M_1) : \alpha \rightarrow \tau$ and $\Gamma \models \text{BT}(M_2) : \alpha$. By Theorem 8, we have $\Gamma \models \text{BT}(M_1) @ \text{BT}(M_2) : \tau$. By Proposition 12, we have $\Gamma \models \text{BT}(M_1 M_2) : \tau$ as required. The converse is similar. $\square$

Example 19. Recall the Böhm tree U_0 in Example 5. Set

$$M = \mathbf{Y} (\lambda f^{\circ \rightarrow (\circ \rightarrow \circ) \rightarrow \circ}. \lambda y^{\circ}. \lambda x^{\circ \rightarrow \circ}. b(x y) (f(x y))) a$$

with $\Delta \vdash M :: (o \rightarrow o) \rightarrow o$. Then $U_0 = \text{BT}(M)$. By $\vdash \mathbf{Y} : \langle \langle \tau; 0 \rangle \rightarrow \tau; 0 \rangle \rightarrow \tau$, where $\tau = \langle q; 1 \rangle \rightarrow (\langle \langle q; 1 \rangle \rightarrow q; 1 \rangle \rightarrow q)$, we have $\Gamma \vdash M : \langle \langle \langle q; 1 \rangle \rightarrow q; 1 \rangle \rightarrow q; 0 \rangle$ (which is abbreviated to $\Gamma \vdash M : (1 \rightarrow 1) \rightarrow 0$ in Example 5). The tree $\text{BT}(M)$, due to Clairambault and Murawski [4], is interesting. When viewed as a graph, consisting of the tree edges augmented by the binding relation (so that each bound variable points to its binder), $\text{BT}(M)$ has an undecidable MSO theory [4]. However, thanks to Theorems 17 and 18, problems such as whether $\text{BT}(M)$ satisfies a given type-describable property (e.g. φ in Example 5) are decidable.

Function type with effect-annotated result type

In the current syntax of function types $\alpha \rightarrow \tau$, the result part must be a prime type, which has no effect annotation. However this is not a real restriction. Actually, one can define another kind of function types in which the result part has effect annotation, e.g., $\alpha \Rightarrow \langle \tau; e \rangle$, as a derived form using the negative action of e .

The key observation is

$$\Gamma \vdash M : \langle \tau; e \rangle \iff e \parallel \Gamma \vdash M : \tau.$$

Using this fact, it is easy to see that

$$\begin{aligned} & \Gamma, x : \alpha \vdash M : \langle \tau; e \rangle \\ \text{iff } & (e \parallel \Gamma), x : (e \parallel \alpha) \vdash M : \tau \\ \text{iff } & e \parallel \Gamma \vdash \lambda x. M : (e \parallel \alpha) \rightarrow \tau \\ \text{iff } & \Gamma \vdash \lambda x. M : \langle (e \parallel \alpha) \rightarrow \tau; e \rangle. \end{aligned}$$

So by defining $\alpha \Rightarrow \langle \tau; e \rangle := \langle (e \parallel \alpha) \rightarrow \tau; e \rangle$, we have $\Gamma, x : \alpha \vdash M : \langle \tau; e \rangle$ iff $\Gamma \vdash \lambda x. M : \alpha \Rightarrow \langle \tau; e \rangle$ as desired.

A similar construction will appear in the definition of the exponential arena in Section 5.

4.3 Effective selection

As we have seen, if P wins $\Gamma \models \text{BT}(M) : \tau$, there exists a Böhm tree representing a winning strategy. It is natural to ask: can we effectively construct a λY -term D such that $\text{BT}(D)$ represents a winning strategy of the game i.e. $\text{BT}(D) \blacktriangleright (\Gamma \models \text{BT}(M) : \tau)$? This subsection proves that D can easily be constructed from a derivation of $\Gamma \vdash M : \tau$ by induction on the derivation. This result extends the effective selection result [2, 7] to higher-order judgements.

A derivation is *finite* if all types in it are finite. If $\Gamma \vdash M : \tau$ is derivable and Γ and τ are finite, then it has a finite derivation. A *representation* of a finite derivation $\Gamma \vdash M : \tau$ is a λY -term $b(\Gamma) \vdash D : b(\tau)$ (augmented with finite products), defined by induction on the derivation. We write $D \triangleright (\Gamma \vdash M : \tau)$ if D is a representation of a derivation concluding $\Gamma \vdash M : \tau$. See the long version [22] for the complete list of rules.

$$\begin{aligned} & \frac{\text{BT}(D) \blacktriangleright (\models \text{BT}(\mathbf{Y}) : \tau)}{D \triangleright (\Gamma \vdash \mathbf{Y} : \tau)} \quad \frac{\mathbf{p}_i(\Gamma(x)) = \langle \tau; e \rangle \text{ and } e \succeq \varepsilon}{(\mathbf{p}_i x) \triangleright (\Gamma \vdash x : \tau)} \\ & \frac{D_i \triangleright (\Gamma \vdash M : \langle \tau_i; e_i \rangle) \quad (i = 1, 2, \dots, k)}{(\prod_{i \in \{1, \dots, k\}} D_i) \triangleright (\Gamma \vdash M : \bigwedge_{i \in \{1, \dots, k\}} \langle \tau_i; e_i \rangle)} \\ & \frac{D_0 \triangleright (\Gamma \vdash M : \alpha \rightarrow \tau) \quad D_1 \triangleright (\Gamma \vdash N : \alpha) \quad \text{some } \alpha}{(D_0 D_1) \triangleright (\Gamma \vdash M N : \tau)} \end{aligned}$$

The second main theorem states that a representation D actually generates a winning strategy. The proof is the same as that of the soundness direction of Theorem 18, but uses the “proof-relevant” version (Theorem 11) instead of Theorem 8.

Theorem 20. $D \triangleright (\Gamma \vdash M : \tau)$ implies $\text{BT}(D) \blacktriangleright (\Gamma \models \text{BT}(M) : \tau)$.

Now what is left is the effective construction of D for the base case of $\mathbf{Y}$. Since $G(\tau)$ is an ω -regular game, it has a finite-memory winning strategy. The expected λY -term can be extracted from this winning strategy, using mutual recursion.

Lemma 21. If P wins $\models \mathbf{Y} : \tau$, then one can effectively construct a λY -term D s.t. $\text{BT}(D) \blacktriangleright (\models \mathbf{Y} : \tau)$.

If $\Gamma \models \text{BT}(M) : \tau$, one can effectively construct a finite derivation of $\Gamma \vdash M : \tau$ and compute its term representation using Lemma 21, resulting in a λY -term D such that $D \triangleright (\Gamma \vdash M : \tau)$. Recall that, if M is of 2nd-order, $\text{BT}(M)$ is the value tree and a winning strategy is an accepting run-tree. So this result gives an

effective construction of an accepting run-tree, which is known as *effective selection* [2, 7]. In fact, our result leads to a generalisation of the previous result to arbitrary higher-order judgement.

5. Effect Arena Games

In order to prove the results in the previous sections, this section and the next develop innocent game semantics for type-checking games. Given a winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$ and a finite set Q of ground types, we construct a category $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ of effect arena games. The category $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ is cartesian closed and, when Ω is stable as we have assumed, it is equipped with an adjunction $(-\parallel -) \dashv (- \otimes -)$ parametrised by $\mathbb{E}$ which is considered a pre-order category. Furthermore $(-\otimes -) : \mathbb{E}^{\text{op}} \times \mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)} \rightarrow \mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ is itself a positive action (equivalently, a parametric comonad) [14].

An important result of this section is the well-definedness of the category $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$; the main technical lemmas are concerned with the preservation of the winning condition by strategy composition.

5.1 Effect arenas

The category $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ is based on Inn , the category of arenas (consisting of only question moves) and innocent strategies [9]. An object A of $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ is an arena $|A|$, called the *underlying arena*, equipped with an assignment of an effect for each move. The hom-set $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}(A, B)$ can naturally be viewed as a subset of $\text{Inn}(|A|, |B|)$.

Definition 22 (Effect arena). An *effect arena* (or *arena* for short) A is a tuple $(\mathcal{M}_A, \vdash_A, \lambda_A, \vartheta_A, E_A)$ where

1. $\mathcal{M}_A$ is a set of *moves*,
2. $\vdash_A \subseteq (\mathcal{M}_A \times \mathcal{M}_A)$ is an *enabling relation*,
3. $\lambda_A : \mathcal{M}_A \rightarrow \{P, O\}$ is the *ownership function*,
4. $\vartheta_A : \mathcal{M}_A \rightarrow Q$ is a *ground-type assignment*, and
5. $E_A : \mathcal{M}_A \rightarrow \mathbb{E}$ is an *effect assignment*.

We write $\vdash_A m$ if there is no m' such that $m' \vdash_A m$. The enabling relation must satisfy the conditions:

- (i) For each $m \in \mathcal{M}_A$, either $\vdash_A m$ or $m' \vdash_A m$ for a unique $m' \in \mathcal{M}_A$.
- (ii) If $\vdash_A m$, then $\lambda_A(m) = O$. If $m \vdash_A m'$, then $\lambda_A(m) \neq \lambda_A(m')$.

An effect arena is just a standard arena equipped with effect and ground-type assignments, which are used to determine the set of “good” innocent strategies, called *consistent* strategies.

For an arena A , the set $\mathcal{M}_A^{\text{init}} \subseteq \mathcal{M}_A$ of *initial moves* of A is $\{m \in \mathcal{M}_A \mid \vdash_A m\}$. A move $m \in \mathcal{M}_A$ is called an *O-move* if $\lambda_A(m) = O$ and a *P-move* if $\lambda_A(m) = P$. We often write m^P (resp. m^O) to make the owner explicit.

A *justified sequence* of an arena A is a sequence of moves such that each element except the first is equipped with a *justification pointer* to some earlier move. A *play* of an arena A is a justified sequence s that satisfies: (i) Well-openness, (ii) Alternation, (iii) Justification, and (iv) Visibility. (The definition of these standard notions can be found in [9].) A *P-strategy* (or *strategy*) $\mathfrak{s}$ of an arena A is a prefix-closed subset of plays of A that satisfies Determinacy and Contingent Completeness (see [9] for the definitions). An infinite justified sequence is said to be an *infinite play* if all its finite prefixes are plays. A strategy contains an infinite play just if it contains all its finite prefixes. An *infinite P-view* is an infinite play all of whose finite prefixes are P-views. We use bold symbols such as $\mathfrak{s}$ and $\mathfrak{p}$ for infinite plays and P-views.

Strategies are not required to be total. If a play $s \in \mathfrak{s}$ ending with an O-move is not answered by $\mathfrak{s}$, we write $s \cdot \perp \in \mathfrak{s}$. Let us write $\lceil s \rceil$ to mean the *P-view* [9] of s . A strategy $\mathfrak{s}$ is *innocent* just

if for every pair of plays $s \cdot m$, $s' \in \mathfrak{s}$ (where s and s' end with O-moves and m is a P-move), $\lceil s \rceil = \lceil s' \rceil$ implies $s' \cdot m' \in \mathfrak{s}$ and $\lceil s \cdot m \rceil = \lceil s' \cdot m' \rceil$ for some m' .

Consistency of innocent strategies has three criteria: Ground-type Reflection, Correctness of Summary and Winning Condition. Ground-type Reflection was introduced in our previous work [16] (called *Colour Reflection* therein): An innocent strategy is *ground-type reflecting* if $s \cdot m_1^O \cdot m_2^P \in \mathfrak{s}$ implies $\vartheta(m_1^O) = \vartheta(m_2^P)$. This captures well-typedness in the simple type system with multiple ground types.

Definition 23 (Correctness of Summary). Let A be an effect arena and s be a play of the underlying arena $|A|$. The effect of a P-move m^P in s is a *correct summary* if $E(m^P)$ is a summary of the respective effects of the O-moves between m and its justifier. Formally, assume $s = s_0 \cdot m^P \cdot s_1$ and let

$$\lceil s_0 \rceil \cdot m^P = p_0 \cdot n_1^O \cdot n_2^P \cdot n_3^O \cdot \dots \cdot n_{2k-1}^O \cdot m^P.$$

Then the effect $E(m^P)$ is a correct summary just if

$$\varepsilon \circ E(n_3) \circ E(n_5) \circ \dots \circ E(n_{2k-1}) \preceq E(m).$$

A play s is *summarising* if for every P-move m that occurs in s , the effect of m is a correct summary. A strategy is *summarising* just if every play in the strategy is summarising.

Notice that we only consider P-views in the definition of correctness of summary (because there may be an irrelevant O-move outside of the P-view between the moves in question).

We introduce some convenient notations. Given a finite sequence of moves $s = m_1 \cdot m_2 \cdot \dots \cdot m_k$ (not necessarily a play), we write $s|_O$ for the subsequence consisting of O-moves. When applied to plays, the resulting subsequence is no longer a play. The effect map is naturally extended to sequence of moves as

$$E^*(m_1 \cdot m_2 \cdot \dots \cdot m_k) := E(m_1) \circ E(m_2) \circ \dots \circ E(m_k)$$

(and ε if $k = 0$). Using these notations, the correctness of summary can be expressed as $p \cdot n \cdot s \cdot m \in \mathfrak{s}$ implies $E^*(s|_O) \preceq E(m)$. For an infinite sequence of moves, we define

$$E^\omega(m_1 \cdot m_2 \cdot \dots) := \pi \langle E(m_1), E(m_2), \dots \rangle.$$

Definition 24 (Winning Condition). An innocent strategy $\mathfrak{s}$ of an effect arena A satisfies the *winning condition* (or simply Ω) if the following conditions hold:

1. **Infinite P-views:** For every infinite P-view $p = m_1^O \cdot m_2^P \cdot m_3^O \cdot \dots \in \mathfrak{s}$, the sequence of effects of O-moves satisfies Ω , i.e. $\pi \langle E(m_1^O), E(m_3^O), E(m_5^O), \dots \rangle \in \Omega$.
2. **Infinite Chattering:** For every unanswered finite P-view $m_1^O \cdot \dots \cdot m_{2k-1}^O \cdot \perp \in \mathfrak{s}$, considering $\perp$ as the infinite ε s, one requires $\pi \langle E(m_1^O), \dots, E(m_{2k-1}^O), \varepsilon, \varepsilon, \dots \rangle \in \Omega$.

The condition on P-views is rephrased as $E^\omega(p|_O) \in \Omega$.

Remark 25. If $\pi \langle \varepsilon, \varepsilon, \dots \rangle \in \Omega$, then every unanswered finite P-views satisfies Ω . If not, consistent strategies must be total.

Definition 26 (Consistency). An innocent strategy $\mathfrak{s}$ of an effect arena A is *consistent* if $\mathfrak{s}$ is ground-type reflecting, summarising and satisfies the winning condition.

5.2 Products and exponentials

Given two effect arenas A and B , we construct the product $A \times B$ and the exponential $A \Rightarrow B$. The underlying arenas of the product and of the exponential are respectively the product and the exponential of underlying arenas. I.e. $|A \times B| = |A| \times |B|$ and $|A \Rightarrow B| = |A| \Rightarrow |B|$. Recall that $\mathcal{M}_{A \times B} = \mathcal{M}_A + \mathcal{M}_B$ and

$\mathcal{M}_{A \Rightarrow B} = \mathcal{M}_B^{\text{init}} \times \mathcal{M}_A + \mathcal{M}_B$. The effect assignments for $A \times B$ and $A \Rightarrow B$ are defined by:

$$E_{A \times B}(m) := \begin{cases} E_A(m) & (\text{if } m \in \mathcal{M}_A) \\ E_B(m) & (\text{if } m \in \mathcal{M}_B) \end{cases}$$

and

$$\begin{aligned} E_{A \Rightarrow B}((m, n)) &:= E_B(m) \setminus E_A(n) & (\text{if } n \in \mathcal{M}_A^{\text{init}}) \\ E_{A \Rightarrow B}((m, n)) &:= E_A(n) & (\text{if } n \notin \mathcal{M}_A^{\text{init}}). \\ E_{A \Rightarrow B}(m_B) &:= E_B(m_B) & (\text{if } m_B \in \mathcal{M}_B) \end{aligned}$$

For an explanation of the effect assignments for the exponential arena, see the argument at the end of Section 4.2.

5.3 Category of consistent strategies

We say an arena A *satisfies the winning condition* (or simply Ω) just if $m_1 \vdash m_2 \vdash \dots$ implies $\pi \langle E(m_1), E(m_2), \dots \rangle \in \Omega$.

Definition 27. The category $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ of effect arenas and consistent strategies is defined by the following data.

- **Objects:** An effect arena that satisfies Ω .
- **Maps:** $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}(A, B)$ is the set of all innocent and consistent strategies of $A \Rightarrow B$.

The identity maps and composition are inherited from the underlying arena game model.

To prove that $\mathcal{G}_{(\mathbb{E}, \mathbb{F}, \Omega)}$ is well-defined, it suffices to show (i) consistency of the identity maps, and (ii) consistency of the composite of consistent strategies. The consistency of the identity is rather straightforward.

Lemma 28. For every effect arena A that satisfies Ω , the identity id_A of the underlying arena is consistent.

We prove that for every consistent strategies $\mathfrak{s} : A \Rightarrow B$ and $\mathfrak{t} : B \Rightarrow C$, the composite $(\mathfrak{s}; \mathfrak{t}) : A \Rightarrow C$ is consistent.

First we consider correctness of summary. An interaction sequence $s \in \text{Int}(A, B, C)$ is said to *generate a P-view* just if $s \upharpoonright_{A \Rightarrow C}$ is a P-view. Given an interaction sequence s , we write $s \upharpoonright_{A \Rightarrow C}^O$ for the subsequence of $s \upharpoonright_{A \Rightarrow C}$ consisting of O-moves. The first key lemma is about effects on an interaction sequence of summarising strategies.

Lemma 29. Assume that $\mathfrak{s} : A \Rightarrow B$ and $\mathfrak{t} : B \Rightarrow C$ are summarising strategies. Let $s \in \text{Int}(\mathfrak{s}, \mathfrak{t})$ be an interaction sequence generating a P-view. Take any move m in s and let

$$s = s_1 \cdot n \cdot s_2 \cdot m \cdot s_3.$$

Then $E_{(A \Rightarrow B) \Rightarrow C}((s_2 \cdot m) \upharpoonright_{A \Rightarrow C}^O) \preceq E_{(A \Rightarrow B) \Rightarrow C}(m)$.

Corollary 30. If $\mathfrak{s}$ and $\mathfrak{t}$ are summarising, so is $\mathfrak{s}; \mathfrak{t}$.

Now we study an infinite interaction sequence generating an infinite P-view, in order to prove the preservation of the winning condition by composition. Let $s = m_1 \cdot m_2 \cdot \dots \in \text{Int}(|A|, |B|, |C|)$ be an infinite interaction sequence. Its *sub-view* is an infinite subsequence $p = n_1 \cdot n_2 \cdot \dots$ of s s.t.

$$s = s_1 \cdot n_1 \cdot n_2 \cdot s_3 \cdot n_3 \cdot n_4 \cdot s_5 \cdot n_5 \cdot n_6 \cdot s_7 \cdot n_7 \cdot \dots,$$

where n_1 is an initial C -move or an initial B -move. A sub-view starting from an initial C -move is a P-view of $B \Rightarrow C$, and a sub-view starting from an initial B -move can be viewed as a P-view of $A \Rightarrow B$. The second key lemma says that every infinite interaction sequence has an infinite sub-view. The proof relies on a result of Clairambault and Harmer [3].

Lemma 31. Let $\mathfrak{s} : |A| \Rightarrow |B|$ and $\mathfrak{t} : |B| \Rightarrow |C|$ be innocent strategies and $s \in \text{Int}(\mathfrak{s}, \mathfrak{t})$ be an infinite interaction sequence

generating a P-view. Then s has an infinite P-view of s or t as its sub-view.

Lemma 32. *If s and t are summarising and satisfies Ω then $s; t$ satisfies Ω .*

Proof. (Sketch) Here we check that the winning condition holds for infinite P-views. Let $p \in (s; t)$ be an infinite P-view in the composite, and $s \in \text{Int}(s, t)$ be the interaction sequence such that $p = s|_{A \Rightarrow C}$. By Lemma 31, s has an infinite P-view of s or t as its sub-view. Consider the case that it is an infinite P-view q of s . The other case can be proved similarly. Assume that $s = m_1 \cdot m_2 \cdot \dots$ and $q = n_1 \cdot n_2 \cdot \dots$. Since q is a sub-view of s , we have

$$s = m_1 \cdot s_1 \cdot n_1 \cdot n_2 \cdot s_3 \cdot n_3 \cdot n_4 \cdot s_5 \cdot n_5 \cdot n_6 \cdot \dots$$

Note that n_{2k} cannot be an O-move of $A \Rightarrow C$ and hence

$$p|_{A \Rightarrow C}^O = s|_{A \Rightarrow C}^O = m_1 \cdot ((s_1 \cdot n_1)|_{A \Rightarrow C}^O) \cdot ((s_3 \cdot n_3)|_{A \Rightarrow C}^O) \cdot \dots$$

Let us define $t_{2k+1} = (s_{2k+1} \cdot n_{2k+1})|_{A \Rightarrow C}^O$. Then $p|_{A \Rightarrow C}^O = m_1 \cdot t_1 \cdot t_3 \cdot \dots$. By Lemma 29, for every k , we have

$$E_{(A \Rightarrow B) \Rightarrow C}^*(t_{2k+1}) \preceq E_{(A \Rightarrow B) \Rightarrow C}(n_{2k+1}).$$

Evaluating $E_{A \Rightarrow C}^\omega(p|_{A \Rightarrow C}^O)$ carefully, we have

$$\begin{aligned} E_{A \Rightarrow C}^\omega(p|_{A \Rightarrow C}^O) &= \pi(E_{A \Rightarrow C}(m_1) \circ E_{A \Rightarrow C}^*(t_1), E_{A \Rightarrow C}^*(t_3), \dots) \\ &\preceq \pi(E_{A \Rightarrow B}(n_1), E_{A \Rightarrow B}(n_3), \dots). \end{aligned}$$

The last expression is in Ω because s satisfies Ω . Hence by lower-closedness of Ω , we have $E_{A \Rightarrow C}^\omega(p|_{A \Rightarrow C}^O) \in \Omega$. $\square$

The next theorem follows from Corollary 30 and Lemma 32.

Theorem 33. *The composite of consistent strategies is consistent. Therefore $\mathcal{G}(\mathbb{E}, \mathbb{F}, \Omega)$ is well-defined.*

5.4 Categorical structure of $\mathcal{G}(\mathbb{E}, \mathbb{F}, \Omega)$

For simplicity, we write $\mathcal{G}$ for $\mathcal{G}(\mathbb{E}, \mathbb{F}, \Omega)$. Since an effect arena is an ordinary arena with additional information, it is natural to consider the forgetful functor $|-| : \mathcal{G} \rightarrow \mathcal{I}nn$. Because $\mathcal{G}(A, B)$ maps forgetfully to $\mathcal{I}nn(|A|, |B|)$ and the identities and strategy composition are inherited from $\mathcal{I}nn$, $|-|$ surely defines a functor.

Proposition 34. *$A \times B$ is a cartesian product and $A \Rightarrow B$ is an exponential in $\mathcal{G}$. Furthermore $|-|$ is a CCC functor.*

For an arena A and effect e , we define $e \otimes A$ to be an arena that has the same underlying arena and the same ground-type assignment as A , equipped with the effect assignment

$$E_{e \otimes A}(m) := \begin{cases} e \circ E_A(m) & (\text{if } m \in \mathcal{M}_A^{\text{init}}) \\ E_A(m) & (\text{otherwise}). \end{cases}$$

By defining $e \otimes s = s$ for strategies, $e \otimes -$ is a functor for every e . (This follows from the fact that Ω is stable.) Further it is a (strict) positive action of $\mathbb{E}^{\text{op}}$ [14].

Lemma 35. *$- \otimes -$ is a bifunctor $\mathbb{E}^{\text{op}} \times \mathcal{G} \rightarrow \mathcal{G}$. Furthermore $(e_1 \circ e_2) \otimes (-) = e_1 \otimes (e_2 \otimes (-))$ and $\varepsilon \otimes (-) = id_{\mathcal{G}}$.*

For each $e \in \mathbb{E}$, $e \otimes -$ has a left-adjoint $e \backslash - : \mathcal{G} \rightarrow \mathcal{G}$. The arena $e \backslash A$ has the same underlying arena as A . The effect assignment is given by

$$E_{e \backslash A}(m) := \begin{cases} e \backslash E_A(m) & (\text{if } m \in \mathcal{M}_A^{\text{init}}) \\ E_A(m) & (\text{otherwise}). \end{cases}$$

For strategies, $e \backslash s = s$ similarly to $e \otimes (-)$.

Lemma 36. *$- \backslash -$ is a bifunctor $\mathbb{E} \times \mathcal{G} \rightarrow \mathcal{G}$. It satisfies $(e_2 \circ e_1) \backslash - = e_1 \backslash (e_2 \backslash -)$ and $\varepsilon \backslash - = id_{\mathcal{G}}$.*

It is easy to see $e \backslash - \dashv e \otimes -$ for every $e \in \mathbb{E}$, and hence they form an $\mathbb{E}$ -parametrised adjunction.

6. Two-Level Games

Based on the effect arena games of Section 5, we shall build a game model that can interpret intersection types with effect annotations. The constructions of this section are a straightforward adaptation of the *two-level constructions* introduced in our previous work [16].

6.1 Two-level effect arena games

Definition 37 (Two-Level Arenas). A *two-level arena* is a triple (A, ϖ, K) , where A is an effect arena, K is an arena without effects, and $\varpi : \mathcal{M}_A \rightarrow \mathcal{M}_K$ is a map of moves that preserves the enabling relation i.e. $\vdash_A m$ implies $\vdash_K \varpi(m)$ and $m \vdash_A m'$ implies $\varpi(m) \vdash_K \varpi(m')$. In case the map ϖ is clear from the context, we abbreviate (A, ϖ, K) to $A :: K$.

The map of moves can be extended naturally to a map of plays, $\varpi(m_1 \cdot m_2 \cdot \dots \cdot m_k) := \varpi(m_1) \cdot \varpi(m_2) \cdot \dots \cdot \varpi(m_k)$. Notice that the RHS is actually a play of K , since the enabling relations are preserved by ϖ . Take a two-level arena (A, ϖ, K) and strategies s of A and t of K . We write $s \sqsubset_{\varpi} t$ just if $s \in s$ implies $\varpi(s) \in t$ for every s . We omit ϖ and simply write $s \sqsubset t$ if ϖ is clear from the context.

Definition 38 (Two-Level Strategies). A *two-level strategy* of a two-level arena (A, ϖ, K) is just a pair $s :: t$ of innocent strategies of A and of K respectively such that $s \sqsubset_{\varpi} t$. It is *consistent* if s is a consistent strategy of A . It is *relatively total* if $s \cdot \perp \in s$ implies $\varpi(s) \cdot \perp \in t$. A consistent and relatively total two-level strategy is said to be *winning*.

The *product* (resp. *exponential*) of two-level arenas is defined as component-wise products (resp. exponentials) equipped with the obvious map of moves. The category $\mathcal{G} :: \mathcal{I}nn$ has two-level arenas as objects and the maps from $(A :: J)$ to $(B :: K)$ are the two-level winning strategies of $(A :: J) \Rightarrow (B :: K)$. Composition is defined component-wise.

Lemma 39. *Let $s_1 :: t_1$ and $s_2 :: t_2$ be winning two-level strategies. Then the composite $(s_1; s_2) :: (t_1; t_2)$ is winning.*

The category of two-level games is a CCC by the products and exponentials introduced in the preceding. Projection on the second component $p : (\mathcal{G} :: \mathcal{I}nn) \rightarrow \mathcal{I}nn : (A :: K) \mapsto K$ is a strict CCC functor. This functor is neither a fibration nor an opfibration, but satisfies a weaker property that suffices for completeness.

Lemma 40 (De-substitution). *Let $(s :: t) : (A :: I) \rightarrow (C :: K)$. Then for any decomposition of t in $\mathcal{I}nn$, i.e., $I \xrightarrow{t} K = I \xrightarrow{t_1} J \xrightarrow{t_2} K$, we have a decomposition of $s :: t$ in $\mathcal{G} :: \mathcal{I}nn$*

$$(A :: I) \xrightarrow{s :: t} (C :: K) = (A :: I) \xrightarrow{s_1 :: t_1} (B :: J) \xrightarrow{s_2 :: t_2} (C :: K)$$

for some B , s_1 and s_2 .

Two-level arenas (A_1, ϖ_1, K) and (A_2, ϖ_2, K) sharing the same K have another kind of the product, defined by

$$(A_1, \varpi_1, K) \wedge (A_2, \varpi_2, K) := (A_1 \times A_2, \varpi, K),$$

where $\varpi := [\varpi_1, \varpi_2] : \mathcal{M}_{A_1} + \mathcal{M}_{A_2} \rightarrow \mathcal{M}_K$. This two-level arena is the pull-back of a certain diagram.

Positive action $e \otimes -$ is defined by $e \otimes (A :: K) := (e \otimes A) :: K$ on two-level arenas and $e \otimes (s :: t) := s :: t$ on two-level strategies. Negative action is defined similarly.

6.2 Connection to type-checking games

The results in Section 3 described by Böhm trees are just a rephrasing of results in this section described by the two-level arena game

model, via the bijective correspondence between Böhm trees and innocent strategies.

The game semantics of types mapping types to arenas is straightforwardly defined by using the CCC structure, fibred products and effect actions. Given $q \in Q$, we define the two-level arena $\mathfrak{D}_q = (A, \varpi, K)$, where A consists of a unique O-move of effect ε and ground type q , K consists of a unique O-move and ϖ is the canonical map. The two-level arena interpretation $\llbracket \cdot \rrbracket$ of types is given by:

$$\begin{aligned} \llbracket q :: \circ \rrbracket &:= \mathfrak{D}_q \\ \llbracket \bigwedge_{i \in I} \langle \tau_i; e_i \rangle :: \kappa \rrbracket &:= \bigwedge_{i \in I} (e_i \circ \llbracket \tau_i :: \kappa \rrbracket) \\ \llbracket (\alpha \rightarrow \tau) :: (\kappa \rightarrow \kappa') \rrbracket &:= \llbracket \alpha :: \kappa \rrbracket \Rightarrow \llbracket \tau :: \kappa' \rrbracket. \end{aligned}$$

A type environment is interpreted as the product of its components. We simply write $\llbracket \tau \rrbracket$ leaving the kind implicit.

Assume the map $\langle \cdot \rangle$ from kinds to (ordinary) arenas, and the bijection from the Böhm trees of kind κ to the innocent strategies of the arena $\langle \kappa \rangle$. By abuse of notation, this bijective map is also written as $\langle \cdot \rangle$. The next lemma bridges the gap between the type-checking games and the two-level game model. Theorem 11 is a corollary of Lemma 39, Lemma 40 and Lemma 41.

Lemma 41. $\hat{V} \blacktriangleright (\Gamma \vdash U : \tau)$ iff $(\langle \hat{V} \rangle :: \langle U \rangle) : \llbracket \Gamma \rrbracket \longrightarrow \llbracket \tau \rrbracket$.

Further directions

The type-describable properties seem not to be closed under union and negation. The introduction of negation to higher-order types could be of practical significance, since negative ground types have played a key rôle in PREFACE, a state-of-the-art model checker [18]. It would be interesting to understand the general semantic picture when $\mathbb{E}$ is a category, rather than a preorder.

References

- [1] C. H. Broadbent, A. Carayol, C.-H. L. Ong, and O. Serre. Recursion schemes and logical reflection. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010)*, pages 120–129. IEEE Computer Society, 2010.
- [2] A. Carayol and O. Serre. Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS 2012)*, pages 165–174. IEEE Computer Society, 2012.
- [3] P. Clairambault and R. Harmer. Totality in arena games. *Ann. Pure Appl. Logic*, 161(5):673–689, 2010.
- [4] P. Clairambault and A. S. Murawski. Böhm trees as higher-order recursive schemes. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *LIPICs*, pages 91–102. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- [5] P.-L. Curien. Abstract Böhm trees. *Mathematical Structures in Computer Science*, 8(6):559–591, 1998.
- [6] K. Fujima, S. Ito, and N. Kobayashi. Practical alternating parity tree automata model checking of higher-order recursion schemes. In *Proceedings of 11th Asian Symposium on Programming Languages and Systems (APLAS 2013)*, volume 8301 of *LNCS*. Springer, 2013. ISBN 978-3-319-03541-3.
- [7] A. Haddad. Model checking and functional program transformations. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *LIPICs*, pages 115–126. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- [8] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*, pages 452–461. IEEE Computer Society, 2008.
- [9] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.
- [10] N. Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3):20, 2013.
- [11] N. Kobayashi and C.-H. L. Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 179–188. IEEE Computer Society, 2009.
- [12] N. Kobayashi, K. Matsuda, and A. Shinohara. Functional programs as compressed data. In *Proceedings of the ACM SIGPLAN 2012 Workshop on Partial Evaluation and Program Manipulation (PEPM 2012)*, pages 121–130. ACM, 2012.
- [13] P.-A. Mellies. Asynchronous games 2: The true concurrency of innocence. *Theor. Comput. Sci.*, 358(2-3):200–228, 2006.
- [14] P.-A. Mellies. Parametric monads and enriched adjunctions. Preprint, 28 pages, 2012.
- [15] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proceedings of 21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 81–90. IEEE Computer Society, 2006.
- [16] C.-H. L. Ong and T. Tsukada. Two-level game semantics, intersection types, and recursion schemes. In *Proceedings of 39th International Colloquium on Automata, Languages, and Programming (ICALP 2012)*, volume 7392 of *LNCS*, pages 325–336. Springer, 2012.
- [17] D. Perrin and J. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Academic Press, 2004.
- [18] S. J. Ramsay, R. P. Neatherway, and C.-H. L. Ong. A type-directed abstraction refinement approach to higher-order model checking. In *Proceedings of the 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2014)*, pages 61–72. ACM, 2014.
- [19] S. Salvati and I. Walukiewicz. Krivine machines and higher-order schemes. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6756 of *LNCS*, pages 162–173. Springer, 2011.
- [20] S. Salvati and I. Walukiewicz. Evaluation is MSOL-compatible. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *LIPICs*, pages 103–114. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- [21] C. Stirling. Dependency tree automata. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS 2009)*, volume 5504 of *LNCS*, pages 92–106. Springer, 2009.
- [22] T. Tsukada and C.-H. L. Ong. Compositional higher-order model-checking via ω -regular games over Böhm trees (long version). Preprint, 2014. URL <http://www.kb.is.s.u-tokyo.ac.jp/~tsukada/papers/effect-arena-long.pdf>.
- [23] S. van Bakel. The heart of intersection type assignment: Normalisation proofs revisited. *Theor. Comput. Sci.*, 398(1-3):82–94, 2008.