Two-Level Game Semantics, Intersection Types, and Recursion Schemes*

C.-H. Luke Ong[1] and Takeshi Tsukada[2,3]

[1] Department of Computer Science, University of Oxford
[2] Graduate School of Information Science, Tohoku University
[3] JSPS Research Fellow

Abstract. We introduce a new cartesian closed category of *two-level* arenas and innocent strategies to model intersection types that are refinements of simple types. Intuitively a property (respectively computation) on the upper level refines that on the lower level. We prove *Subject Expansion*—any lower-level computation is closely and canonically tracked by the upper-level computation that lies over it—which is a measure of the robustness of the two-level semantics. The game semantics of the type system is *fully complete*: every winning strategy is the denotation of some derivation. To demonstrate the relevance of the game model, we use it to construct new semantic proofs of non-trivial algorithmic results in higher-order model checking.

1 Introduction

The recent development of *higher-order model checking*—the model checking of trees generated by higher-order recursion schemes (HORS) against (alternating parity) tree automata—has benefitted much from ideas and methods in semantics. Ong's proof [1] of the decidability of the monadic second-order (MSO) theories of trees generated by HORS was based on game semantics [2]. Using HORS as an intermediate model of higher-order computation, Kobayashi [3] showed that safety properties of functional programs can be verified by reduction to the model checking of HORS against trivial automata (i.e. Büchi tree automata with a trivial acceptance condition). His model checking algorithm is based on an intersection-type-theoretic characterisation of the trivial automata acceptance problem of trees generated by HORS.[1] This type-theoretic approach was subsequently refined and extended to characterise alternating parity tree automata [5], thus yielding a new proof of Ong's MSO decidability result. (Several other proofs [6,7] of the result have since been published.)

This paper was motivated by a desire to understand the connexions between the game-semantic proof [1] and the type-based proof [3,5] of the MSO decidability result. As a first step in clarifying their relationship, we construct a *two-level game semantics* to model intersection types that are refinements of simple types. Given a set Q of

* A full version with proofs is available at `http://www.cs.ox.ac.uk/people/luke.ong/personal/publications/icalp12.pdf`.

[1] Independently, Salvati [4] has proposed essentially the same intersection type system for the simply-typed λ-calculus without recursion from a different perspective.

A. Czumaj et al. (Eds.): ICALP 2012, Part II, LNCS 7392, pp. 325–336, 2012.

colours (modelling the states of an automaton), we introduce a cartesian closed category whose objects are triples (A, U, K) called *two-level arenas*, where A is a *Q-coloured arena* (modelling intersection types), K is a standard arena (modelling simple types), and U is a colour-forgetting function from A-moves to K-moves which preserves the justification relation. A map of the category from (A, U, K) to (A', U', K') is a pair of innocent and colour-reflecting strategies, $\sigma : A \longrightarrow A'$ and $\bar{\sigma} : K \longrightarrow K'$, such that the induced colour-forgetting function maps plays of σ to plays of $\bar{\sigma}$. This captures the intuition that the upper-level computation represented by σ refines (or is more constrained than) the lower-level computation represented by $\bar{\sigma}$, a semantic framework reminiscent of two-level denotational semantics in abstract interpretation as studied by Nielson [8]. Given triples $\mathcal{A}_1 = (A_1, U_1, K)$ and $\mathcal{A}_2 = (A_2, U_2, K)$ that have the same base arena K, their *intersection* $\mathcal{A}_1 \wedge \mathcal{A}_2$ is $(A_1 \times A_2, [U_1, U_2], K)$. Building on the two-level game semantics, we make the following contributions.

(i) How good is the two-level game semantics? Our answer is *Subject Expansion* (Theorem 3), which says intuitively that any computation (reduction) on the lower level can be closely and canonically tracked by the higher-level computation that lies over it. Subject Expansion clarifies the relationship between the two levels; we think it is an important measure of the robustness (and, as we shall see, the reason for the usefulness) of the game semantics.

(ii) We put the two-level game model to use by modelling Kobayashi's intersection type system [3]. Derivations of intersection-type judgements, which we represent by the terms of a new proof calculus, are interpreted by *winning strategies* i.e. compact and total (in addition to innocent and colour-reflecting). We prove that the interpretation is *fully complete* (Theorem 5): every winning strategy is the denotation of some derivation.

(iii) Finally, to demonstrate the usefulness and relevance of the two-level game semantics, we apply it to construct new semantic proofs of three non-trivial *algorithmic* results in higher-order model checking: (a) characterisation of trivial automata acceptance (existence of an accepting run-tree) by a notion of typability [3], (b) minimality of the type environment induced by traversal tree [1], and (c) completeness of GTRecS, a game-semantics based practical algorithm for model checking HORS against trivial automata [9].

Outline. In Section 2, the idea of two-level structure is explained informally. We present coloured arenas, two-level arenas, innocent strategies and related game-semantic notions in Section 3, culminating in the Subject Expansion Theorem. In Section 4 we construct a fully complete two-level game model of Kobayashi's intersection type system. Finally, Section 5 applies the game model to reason about algorithmic problems in higher-order model checking.

2 Two Structures of Intersection Type System

This section presents the intuitions behind the two levels. We explain that two different structures are naturally extracted from a derivation in an intersection type system. Here we use term representation for explanation. Two-level game semantics will be developed in the following sections based on this idea.

$$\dfrac{\dfrac{g:\tau}{g:p_1\wedge p_3\to q_1}\quad \dfrac{\dfrac{\dfrac{x:\sigma}{x:p_1}\quad\dfrac{x:\sigma}{x:p_3}}{x:p_1\wedge p_3}}{}}{}$$

$$\dfrac{\dfrac{\dfrac{g:\tau}{g:p_1\wedge p_3\to q_1}\qquad \dfrac{\dfrac{x:\sigma}{x:p_1}\quad\dfrac{x:\sigma}{x:p_3}}{x:p_1\wedge p_3}}{g\ x:q_1}\qquad \dfrac{\dfrac{g:\tau}{g:p_2\to q_2}\qquad\dfrac{x:\sigma}{x:p_2}}{g\ x:q_2}}{g\ x:q_1\wedge q_2}$$

Fig. 1. A type derivation of the intersection type system. Here type environment $\Gamma=\{g:((p_1\wedge p_3)\to q_1)\wedge(p_2\to q_2),\ x:p_1\wedge p_2\wedge p_3\}$ is omitted.

$$\dfrac{\dfrac{\dfrac{g:\tau'}{\mathsf{p}_1(g):p_1\times p_3\to q_1}\qquad \dfrac{\dfrac{x:\sigma'}{\mathsf{p}_1(x):p_1}\quad\dfrac{x:\sigma}{\mathsf{p}_3(x):p_3}}{\langle\mathsf{p}_1(x),\mathsf{p}_2(x)\rangle:p_1\times p_3}}{\mathsf{p}_1(g)\ \langle\mathsf{p}_1(x),\mathsf{p}_2(x)\rangle:q_1}\qquad \dfrac{\dfrac{g:\tau}{\mathsf{p}_2(g):p_2\to q_2}\qquad\dfrac{x:\sigma}{\mathsf{p}_2(x):p_2}}{\mathsf{p}_2(g)\ \mathsf{p}_2(x):q_2}}{\langle\mathsf{p}_1(g)\ \langle\mathsf{p}_1(x),\mathsf{p}_2(x)\rangle,\ \mathsf{p}_2(g)\ \mathsf{p}_2(x)\rangle:q_1\times q_2}$$

Fig. 2. A type derivation of the product type system, which corresponds to Fig. 1. Here $\Gamma'=\{g:((p_1\times p_3)\to q_1)\times(p_2\to q_2),\ x:p_1\times p_2\times p_3\}$ is omitted.

The intersection type constructor $\wedge$ of an intersection type system is characterised by the following typing rules.[2]

$$\dfrac{\Gamma\vdash t:\tau_1\qquad \Gamma\vdash t:\tau_2}{\Gamma\vdash t:\tau_1\wedge\tau_2}\qquad \dfrac{\Gamma\vdash t:\tau_1\wedge\tau_2}{\Gamma\vdash t:\tau_1}\qquad \dfrac{\Gamma\vdash t:\tau_1\wedge\tau_2}{\Gamma\vdash t:\tau_2}$$

At first glance, they resemble the rules for products. Let $\langle t_1,t_2\rangle$ be a pair of t_1 and t_2 and p_i be the projection to the ith element (for $i\in\{1,2\}$).

$$\dfrac{\Gamma\vdash t_1:\tau_1\qquad \Gamma\vdash t_2:\tau_2}{\Gamma\vdash\langle t_1,t_2\rangle:\tau_1\times\tau_2}\qquad \dfrac{\Gamma\vdash t:\tau_1\times\tau_2}{\Gamma\vdash \mathsf{p}_1(t):\tau_1}\qquad \dfrac{\Gamma\vdash t:\tau_1\times\tau_2}{\Gamma\vdash \mathsf{p}_2(t):\tau_2}$$

When we ignore terms and replace $\times$ by $\wedge$, the rules in the two groups coincide. In fact, they are so similar that a derivation of the intersection type system can be transformed to a derivation of the product type system by replacing $\wedge$ by $\times$ and adjusting terms to the rules for product. See Figures 1 and 2 for example. This is the first structure behind an intersection-type derivation, which we call the *upper-level structure.*

However the upper-level structure alone does not capture all features of the intersection type system: specifically some derivations of the product type system have no corresponding derivation in the intersection type system. For example, while the type judgement $x:p_1,y:p_2\vdash\langle x,y\rangle:p_1\times p_2$ is derivable, no term inhabits the judgement $x:p_1,y:p_2\vdash ?:p_1\wedge p_2$.

Terms in the rules explain this gap. We call them *lower-level structures.* To construct a term of type $\tau_1\times\tau_2$, it suffices to find *any* two terms t_1 of type τ_1 and t_2 of type τ_2. However to construct a term of type $\tau_1\wedge\tau_2$, we need to find a term t that has both type τ_1 and type τ_2. Thus a product type derivation has a corresponding intersection

[2] In the type system in Section 4, these rules are no longer to be independent rules, but a similar argument stands.

type derivation only if for all pairs $\langle t_1, t_2\rangle$ appearing at the derivation, the respective structures of t_1 and t_2 are "coherent".

For example, let us examine the derivation in Figure 2, which contains two pair constructors. One appears at $\langle \mathsf{p}_1(x), \mathsf{p}_3(x)\rangle : p_1 \times p_3$. Here the left argument $\mathsf{p}_1(x) : p_1$ and the right argument $\mathsf{p}_3(x) : p_3$ are "coherent" in the sense that they are the same except for details such as types and indexes of projections. In other words, by forgetting such details, $\mathsf{p}_1(x) : p_1$ and $\mathsf{p}_3(x) : p_3$ become the same term x. The other pair appears at the root and the "forgetful" map maps both the left and right arguments to $g\ x$.

This interpretation decomposes an intersection type derivation into three components: a derivation in the simple type system with product (the upper-level structure), a term (the lower-level structure) and a "forgetful" map from the upper-level structure to the lower-level structure. Since recursion schemes are simply typed, we can assume a term to also be simply typed for our purpose. Hence the resulting two-level structure consists of two derivations in the simple type system with a map on nodes from one to the other.

3 Two-Level Game Semantics

For sets A and B, we write $A + B$ for disjoint union and $A \times B$ for cartesian product. We first introduce some basic notions in game semantics [2].

We introduce Q-coloured arenas and innocent strategies, which are models of the simply-typed λ-calculus with multiple base types ranging over Q.

Definition 1 (Coloured Arena). For a set Q of symbols, a *Q-coloured arena* A is a quadruple $(M_A, \vdash_A, \lambda_A, c_A)$, where (i) M_A is a set of *moves*, (ii) $\vdash_A \subseteq M_A + (M_A \times M_A)$ is a *justification relation*, (iii) $\lambda_A : M_A \to \{P, O\}$, and (iv) $c_A : M_A \to Q$ is a *colouring*. We write $\vdash_A m$ for $m \in (\vdash_A)$ and $m \vdash_A m'$ for $(m, m') \in (\vdash_A)$. The justification relation must satisfy the conditions:

- For each $m \in M_A$, either $\vdash_A m$ or $m' \vdash_A m$ for a unique $m' \in M_A$.
- If $\vdash_A m$, then $\lambda_A(m) = O$. If $m \vdash_A m'$, then $\lambda_A(m) \neq \lambda(m')$.

For a Q-coloured arena A, the set $\mathit{Init}_A \subseteq M_A$ of *initial moves* of A is $\{m \in M_A \mid \vdash_A m\}$. A move $m \in M_A$ is called an *O-move* if $\lambda_A(m) = O$ and a *P-move* if $\lambda_A(m) = P$.

A *justified sequence* of a Q-coloured arena A is a sequence of moves such that each element except the first is equipped with a *justification pointer* to some previous move. A *play* of an arena A is a justified sequence s that satisfies: (i) Well-openness, (ii) Alternation, (iii) Justification, (iv) Visibility. A *P-strategy* (or a *strategy*) σ of an arena A is a prefix-closed subset of plays of A that satisfies Determinacy, Contingent Completeness, and

Colour Reflection. Only the opponent can change the colour i.e. for every O-move m and P-move m', if $s \cdot m \cdot m' \in \sigma$, then $c(m) = c(m')$.

A strategy σ is *innocent* just if for every pair of plays $s \cdot m,\ s' \cdot m' \in \sigma$ ending with P-moves m and m', $\ulcorner s \urcorner = \ulcorner s' \urcorner$ implies $\ulcorner s \cdot m \urcorner = \ulcorner s' \cdot m' \urcorner$, writing $\ulcorner s \urcorner$ to mean the *P-view* of s. Further σ is *winning* just if the following hold:

Compact. The domain $\mathrm{dom}(f_\sigma)$ of the view function of σ is a finite set.
Total. If $s \cdot m \in \sigma$ for an O-move m, then $s \cdot m \cdot m' \in \sigma$ for some P-move m'.

Given Q-coloured arenas A and B, the *product* $A \times B$ and *function space arena* $A \Rightarrow B$ are standard. We define a category whose objects are Q-coloured arenas; maps from A to B are innocent strategies of the arena $A \Rightarrow B$. The category is cartesian closed, and is thus a model of the simply-typed lambda calculus with (indexed) products.

Theorem 1. *For every set Q, the category of Q-coloured arenas and innocent strategies is cartesian closed with the product $A \times B$ and function space $A \Rightarrow B$.*

Definition 2 (Two-Level Arenas). An *two-level arena* based on Q is a triple $\mathcal{A} = (A, U, K)$, where A is a Q-colored arena, K is a $\{o\}$-colored arena (i.e. an ordinary arena, which we call the *base arena* of $\mathcal{A}$) and U is a map from M_A to M_K that satisfies: (i) $\lambda_A(m) = \lambda_K(U(m))$ (ii) If $m \vdash_A m'$ then $U(m) \vdash_K U(m')$; and if $\vdash_A m$ then $\vdash_K U(m)$.

For a justified sequence $s = m_1 \cdot m_2 \cdots m_k$, we write $U(s)$ to mean the justified sequence $U(m_1) \cdot U(m_2) \cdots U(m_k)$ whose justification pointers are induced by those of s. Let $\mathcal{A} = (A, U, K)$ be a two-level arena. It is easy to see that if s is a play of A, then $U(s)$ is a play of K.

For a strategy σ of A, $U(\sigma) := \{U(s) \mid s \in \sigma\}$ is a set of plays of K, which is not necessarily a strategy, since $U(s)$ may not satisfy determinacy. (Recall that some upper-level structure has no corresponding lower-level structure.)

Definition 3 (Strategies). A *strategy* of a two-level arena (A, U, K) is a pair $(\sigma, \bar{\sigma})$ of strategies of A and K respectively such that $U(\sigma) \subseteq \bar{\sigma}$; it is *innocent* just if σ and $\bar{\sigma}$ are innocent as strategies of A and K respectively.

Let $\mathcal{A}_i = (A_i, U_i, K_i)$ where $i = 1, 2$ be two-level arenas. We define product, function space and intersection constructions as follows.

Product. $\mathcal{A}_1 \times \mathcal{A}_2 := (A_1 \times A_2, U, K_1 \times K_2)$, where $U : (M_{A_1} + M_{A_2}) \to (M_{K_1} + M_{K_2})$ is defined as $U_1 + U_2$.
Function Space. $\mathcal{A}_1 \Rightarrow \mathcal{A}_2 := (A_1 \Rightarrow A_2, U, K_1 \Rightarrow K_2)$, where $U : ((M_{A_1} \times \mathit{Init}_{A_2}) + M_{A_2}) \to ((M_{K_1} \times \mathit{Init}_{K_2}) + M_{K_2})$ is defined as $U_1 \times U_2 + U_2$.
Intersection. *Provided* $K_1 = K_2 = K$, define $\mathcal{A}_1 \wedge \mathcal{A}_2 := (A_1 \times A_2, U, K)$, where $U : (M_{A_1} + M_{A_2}) \to M_K$ is defined as $[U_1, U_2]$.

We can now define a category whose objects are two-level arenas, and maps $\mathcal{A}_1 \to \mathcal{A}_2$ are innocent strategies of $\mathcal{A}_1 \Rightarrow \mathcal{A}_2$. The composite of $(\sigma_1, \bar{\sigma}_1) : \mathcal{A}_1 \Rightarrow \mathcal{A}_2$ and $(\sigma_2, \bar{\sigma}_2) : \mathcal{A}_2 \Rightarrow \mathcal{A}_3$ is defined as $(\sigma_1; \sigma_2,\, \bar{\sigma}_1; \bar{\sigma}_2) : \mathcal{A}_1 \Rightarrow \mathcal{A}_3$. Let $\top$ be the terminal object in the category of Q-coloured arenas.

Theorem 2. *(i) Let $\mathcal{A}_i = (A_i, U_i, K)$. Then $\mathcal{A}_1 \wedge \mathcal{A}_2$ is the pullback of*
$\mathcal{A}_1 \xrightarrow{(!_{A_1}, \mathrm{id}_K)} (\top, \emptyset, K) \xleftarrow{(!_{A_2}, \mathrm{id}_K)} \mathcal{A}_2$. *(ii) The category of two-level arenas and innocent strategies is cartesian closed.*

Finally we introduce *Subject Expansion* which is a property that relates the two levels of game semantics. The name originates from a characteristic property of (intersection) type systems, which states that for any terms t and t', type environment Γ and type τ, if $t \longrightarrow t'$ and $\Gamma \vdash t' : \tau$, then $\Gamma \vdash t : \tau$. Subject expansion plays a central rôle in completeness of intersection type systems.

In two-level game semantics, it seems best to formulate Subject Expansion as a kind of factorisation theorem, stated as follows.

Theorem 3 (Subject Expansion). *Let $\mathcal{A}_i = (A_i, U_i, K_i)$ be a two-level arena for $i = 1, 2$ and K be a base arena. If*

$$\begin{tikzcd} \mathcal{A}_1 \arrow[rr, "{(\sigma,\bar{\sigma})}"] & & \mathcal{A}_2 \end{tikzcd} \qquad \textit{and} \qquad \begin{tikzcd} K_1 \arrow[rr, "\bar{\sigma}"] \arrow[dr, "\bar{\sigma}_1"'] & \circlearrowleft & K_2 \\ & K \arrow[ur, "\bar{\sigma}_2"'] & \end{tikzcd}$$

(a map of two-level arenas) *(maps of base arenas)*

then there are a two-level arena $\mathcal{A}$ whose base arena is K and strategies $\sigma_1 : A_1 \to A$ and $\sigma_2 : A \to A_2$ such that

$$\begin{tikzcd} \mathcal{A}_1 \arrow[rr, "{(\sigma,\bar{\sigma})}"] \arrow[dr, "{(\sigma_1,\bar{\sigma}_1)}"'] & \circlearrowleft & \mathcal{A}_2 \\ & \mathcal{A} \arrow[ur, "{(\sigma_2,\bar{\sigma}_2)}"'] & \end{tikzcd}$$

Moreover, there is a canonical triple $(\sigma_1, \mathcal{A}, \sigma_2)$: for every triple $(\sigma_1', \mathcal{A}', \sigma_2')$ that satisfies the requirement, there exists a mapping φ from moves of $\mathcal{A}$ to moves of $\mathcal{A}'$ such that $[\mathrm{id}_{\mathcal{A}_1}, \varphi](\sigma_1) \subseteq \sigma_1'$ and $[\varphi, \mathrm{id}_{\mathcal{A}_2}](\sigma_2) \subseteq \sigma_2'$.

As an application, we shall use Subject Expansion to prove the completeness of the intersection type system for recursion scheme model checking (Theorem 6).

4 Interpretation of Intersection Types

In this section, we interpret Kobayashi's intersection type system [3] in the two-level game model, and show that the interpretation is *fully complete* i.e. every winning strategy is the denotation of some derivation.

We consider the standard Church-style simply-typed lambda calculus. However, to avoid confusion with intersection types, we henceforth refer to simple types as *kinds*, defined by $\kappa ::= o \mid \kappa_1 \to \kappa_2$. Let Δ be a *kind environment* i.e. a set of variable-kind bindings, $x : \kappa$. We write $\Delta \vdash t :: \kappa$ to mean t has kind κ under the environment Δ. Fix a set Q of symbols, ranged over by q. *Intersection pre-types* are defined by $\tau, \sigma ::= q \mid \tau \to \sigma \mid \bigwedge_{i \in I} \tau_i$. The *well-kindedness relation* $\tau :: \kappa$ is defined by the following rules.

$$\frac{}{q :: o} \qquad \frac{\tau_i :: \kappa \quad (\text{for all } i \in I) \qquad \sigma :: \kappa'}{(\bigwedge_{i \in I} \tau_i) \to \sigma :: \kappa \to \kappa'}$$

An *intersection type* is an intersection pre-type τ such that $\tau :: \kappa$ for some κ.

An *(intersection) type environment* Γ is a set of variable-type bindings, $x : \bigwedge_{i \in I} \tau_i$. We write $\Gamma :: \Delta$ just if $x : \bigwedge_{i \in I} \tau_i \in \Gamma$ implies that for some κ, $x : \kappa \in \Delta$ and $\tau_i :: \kappa$ for all $i \in I$. *Valid typing sequents* are defined by induction over the following rules.

$$\frac{}{\Gamma, x : \bigwedge_{i \in I} \tau_i \vdash x : \tau_i} \qquad \frac{\Gamma, x : \bigwedge_{i \in I} \tau_i \vdash t : \sigma \qquad \tau_i :: \kappa \quad (\text{for all } i \in I)}{\Gamma \vdash \lambda x^{\kappa}.t : (\bigwedge_{i \in I} \tau_i) \to \sigma}$$

$$\frac{\Gamma \vdash t_1 : (\bigwedge_{i \in I} \tau_i) \to \sigma \qquad \Gamma \vdash t_2 : \tau_i \quad \text{(for all } i \in I)}{\Gamma \vdash t_1\, t_2 : \sigma}$$

Lemma 1. *If $\Delta \vdash t :: \kappa$ and $\Gamma :: \Delta$ and $\Gamma \vdash t : \tau$, then $\tau :: \kappa$.*

For notational convenience, we use a Church-style simply-kinded lambda calculus with (indexed) product as a term representation of derivations. The raw terms are defined as follows.

$$M \ ::= \ \mathsf{p}_i(x) \ \mid \ \lambda x^{\bigwedge_{i \in I} \tau_i}.M \ \mid \ M_1\, M_2 \ \mid \ \prod_{i \in I} M_i$$

where I is a finite indexing set. We omit I and simply write $\lambda x^{\bigwedge_i \tau_i}$ and so on if I is clear from the context or unimportant. We say a term M is *well-formed* just if for every application subterm $M_1\, M_2$ of M, M_2 has the form $\prod_{i \in I} N_i$. We consider only well-formed terms. By abuse of notation, we write $\top$ for $\prod \emptyset$.

We give a type system for terms of the calculus, which ressemble the intersection type system, but is syntax directed, i.e., a term completely determines the structure of a derivation.

$$\frac{}{\Gamma, x : \bigwedge_{i \in I} \tau_i \Vdash \mathsf{p}_i(x) : \tau_i} \qquad \frac{\Gamma \Vdash M_i : \tau_i \qquad \tau_i :: \kappa \qquad \text{(for all } i)}{\Gamma \Vdash \prod_i M_i : \bigwedge_i \tau_i}$$

$$\frac{\Gamma \Vdash M_1 : (\bigwedge_i \tau_i) \to \sigma \qquad \Gamma \Vdash M_2 : \bigwedge_i \tau_i}{\Gamma \Vdash M_1\, M_2 : \sigma} \qquad \frac{\Gamma, x : \bigwedge_{i \in I} \tau \Vdash M : \sigma}{\Gamma \Vdash \lambda x^{\bigwedge_{i \in I} \tau_i}.M : (\bigwedge_{i \in I} \tau_i) \to \sigma}$$

We call a term-in-context $\Gamma \Vdash M : \tau$ a *proof term*. Observe that a proof term is essentially a typed lambda term with (indexed) product. Here an intersection type $\tau_1 \wedge \cdots \wedge \tau_n$ is interpreted as a product type $\tau_1 \times \cdots \times \tau_n$ and a proof term $M_1 \sqcap \cdots \sqcap M_n$ is a tuple $\langle M_1, \ldots, M_n \rangle$. Then all variables are bound to tuples and a proof term $\mathsf{p}_i(x)$ is a projection into the ith element.

Unfortunately, not all the proof terms correspond to a derivation of the intersection type system. For example, $\lambda f^{(q_1 \wedge q_2) \to p}.\lambda x^{q_1}.\lambda y^{q_2}.f(\mathsf{p}(x) \sqcap \mathsf{p}(y))$ is a proof term of the type $((q_1 \wedge q_2) \to p) \to q_1 \to q_2 \to p$, but there is no inhabitant of that type. In the intersection type system, $t : \tau \wedge \sigma$ only if $t : \tau$ and $t : \sigma$ for the same term t, but the proof term $\mathsf{p}(x) \sqcap \mathsf{p}(y)$ violates the requirement.

We introduce a judgement $M :: t$ that means the structure of M coincides with the structure of t. By definition, $\top :: t$ for every term t.

$$\begin{array}{ll} \mathsf{p}_i(x) :: x & \lambda x^{\bigwedge_i \tau_i}.M :: \lambda x^{\kappa}.t \ := \ M :: t \ \wedge \ \forall i \,.\, \tau_i :: \kappa \\ \prod_i M_i :: t \ := \ \forall i \,.\, M_i :: t & M_1\, M_2 :: t_1\, t_2 \ := \ M_1 :: t_1 \ \wedge \ M_2 :: t_2 \end{array}$$

We write $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ just if $\Gamma :: \Delta$, $M :: t$, $\tau :: \kappa$, $\Delta \vdash t{::}\kappa$ and $\Gamma \Vdash M : \tau$. Let t be a term such that $\Delta \vdash t :: \kappa$. It is easy to see that there is a one-to-one correspondence between a derivation of $\Gamma \vdash t : \tau$ and a proof term M such that $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$.

Example 1. Let $Q = \{ q_1, q_2 \}$ and take $\theta \to (q_1 \wedge q_2) \to q_1 :: (o \to o) \to o \to o$ where $\theta = (q_1 \to q_1) \wedge (q_2 \to q_1) \wedge (q_1 \wedge q_2 \to q_1)$ and terminal $f : q_1 \to q_2$. Set $M := \lambda x^{\theta}\, y^{q_1 \wedge q_2}.\mathsf{p}_2(x)(f^{q_1 \to q_2}(\mathsf{p}_1(x)(\mathsf{p}_3(x)(\mathsf{p}_1(y) \sqcap \mathsf{p}_2(y)))))$. Then we have $M :: \lambda xy.x(f(x(x\, y))$.

A two-level arena represents a proof of well-kindedness, $\tau :: \kappa$. The interpretation is straightforward since we have arena constructors $\Rightarrow$ and $\wedge$:

$$[\![q :: o]\!] := ([\![q]\!], U, [\![o]\!]) \qquad [\![(\bigwedge_{i\in I} \tau_i) \to \sigma :: \kappa \to \kappa']\!] := (\bigwedge_{i\in I} [\![\tau_i :: \kappa]\!]) \Rightarrow [\![\sigma :: \kappa']\!]$$

where $[\![q]\!]$ is a Q-coloured arena with a single move of the colour q, $[\![o]\!]$ is a $\{o\}$-coloured arena with a single move, and U maps the unique move of $[\![q]\!]$ to the unique move of $[\![o]\!]$. Let $\Gamma = x_1 : \bigwedge_{i\in I_1} \tau_i^1, \dots, x_n : \bigwedge_{i\in I_n} \tau_i^n$ be a type environment with $\Gamma :: \Delta$ where $\Delta = x_1 : \kappa_1, \dots, x_n : \kappa_n$. Then $[\![\Gamma :: \Delta]\!] := \prod_{j\le n}(\bigwedge_{i\in I_j} [\![\tau_i^j :: \kappa_i]\!])$.

A proof $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$, which is equivalent to a derivation of $\Gamma \vdash t : \tau$, is interpreted as a strategy of the two-level arena $[\![\Gamma :: \Delta]\!] \Rightarrow [\![\tau :: \kappa]\!]$, defined by the following rules (for simplicity, we write $[\![M :: t]\!]$ instead of $[\![[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]]\!]$):

$$[\![\mathsf{p}_i(x) :: x]\!] := \pi_x ; \mathsf{p}_i \qquad [\![M_1\ M_2 :: t_1\ t_2]\!] := \langle [\![M_1 :: t_1]\!], [\![M_2 :: t_2]\!] \rangle ; \mathbf{eval}$$
$$[\![\textstyle\prod_i M_i :: t]\!] := \textstyle\prod_i [\![M_i :: t]\!] \qquad [\![\lambda x.M :: \lambda x.t]\!] := \Lambda([\![M :: t]\!])$$

where π_x is the projection $[\![(\Gamma, x : \bigwedge_i \tau_i) :: (\Delta, x : \kappa)]\!] \longrightarrow [\![\bigwedge_i \tau_i :: \kappa]\!]$ and for strategies $\sigma_i : [\![\Gamma :: \Delta]\!] \longrightarrow [\![\tau_i :: \kappa]\!]$ indexed by i, the strategy $\prod_i \sigma_i : [\![\Gamma :: \Delta]\!] \longrightarrow \bigwedge_i [\![\tau_i :: \kappa]\!]$ is the canonical map of the pullback.

Lemma 2 (Componentwise Interpretation). *Let* $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ *be a derivation. Then* $[\![M :: t]\!] = ([\![M]\!], [\![t]\!])$.

Theorem 4 (Adequacy). *Let* $[\Gamma :: \Delta] \vdash [M_1 :: t_1] : [\tau :: \kappa]$ *and* $[\Gamma :: \Delta] \vdash [M_2 :: t_2] : [\tau :: \kappa]$ *be two proofs such that* $[M_1 :: t_1] =_{\beta\eta} [M_2 :: t_2]$. *Then* $[\![M_1 :: t_1]\!] = [\![M_2 :: t_2]\!]$.

Theorem 5 (Definability). *Let* $(\sigma, \bar\sigma) : [\![\Gamma :: \Delta]\!] \to [\![\tau :: \kappa]\!]$ *be a winning strategy. There is a derivation* $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ *such that* $(\sigma, \bar\sigma) = [\![M :: t]\!]$.

Proof. (Sketch) By the standard argument of definability [2], we have a proof term M and a simply-typed lambda term t such that $[\![M]\!] = \sigma : [\![\Gamma]\!] \longrightarrow [\![\tau]\!]$ and $[\![t]\!] = \bar\sigma : [\![\Delta]\!] \longrightarrow [\![\kappa]\!]$, where $[\![\cdot]\!]$ is the standard interpretation of typed lambda terms (here intersection $\wedge$ in Γ and τ is interpreted as a product). If $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ is a valid derivation, by Lemma 2, we have $[\![M :: t]\!] = (\sigma, \bar\sigma)$ as required. Thus it suffices to show that $M :: t$, which can be shown by an easy induction. $\square$

We can use Church-style type-annotated terms in β-normal η-long form, called *canonical terms*, to represent winning strategies, which are terms-in-context of the form: $\Gamma \Vdash \mathsf{p}_i(x)\, M_1 \cdots M_n : q$ where $\Gamma = \cdots, x : \bigwedge_i \alpha_i, \cdots$ and $\alpha_i = \tau_1 \to \cdots \to \tau_n \to q$, and for each $k \in \{1, \dots, n\}$,

$$M_k \ = \prod_{j\in J_k} \lambda y_{kj1}^{\tau_{kj1}} \dots y_{kjr}^{\tau_{kjr}}.N_{kj} \ : \ \bigwedge_{j\in J_k} \beta_{kj} \ = \ \tau_k$$

such that for each $j \in J_k$, $\beta_{kj} = \tau_{kj1} \to \cdots \to \tau_{kjr} \to q_{kj}$ with $r = r_{kj}$ and $\Gamma, y_{kj1} : \tau_{kj1}, \cdots, y_{kjr} : \tau_{kjr} \Vdash N_{kj} : q_{kj}$ is a canonical term. (We assume that canonical terms are proof terms that represent derivations.)

By definition, canonical terms are not λ-abstractions. We call terms-in-context such as M_k above canonical terms in (partially) *curried form*; they have the shape $\Gamma \Vdash \lambda\overline{x}.M : \tau_1 \to \cdots \to \tau_n \to q$. Note that in case $n = 0$, the curried form retains an outermost "dummy lambda" $\Gamma \Vdash \lambda.M : q$. With this syntactic convention, we obtain a tight correspondence between syntax and semantics.

Lemma 3. *Let $\tau :: \kappa$. There is a one-to-one correspondence between winning strategies over the two-level arena $[\![\tau :: \kappa]\!]$ and canonical terms in curried form of the shape $\emptyset \Vdash M : \tau$ (with η-long β-normal simply-typed term t such that $M :: t$).*

A strategy $(\sigma, \bar{\sigma})$ of $\mathcal{A} = (A, U, K)$ is *P-full* (respectively *O-full*) just if every P-move (respectively O-move) of A occurs in σ. Suppose $(\sigma, \bar{\sigma})$ is a winning strategy of $[\![\tau :: \kappa]\!]$. Then: (i) If $(\sigma, \bar{\sigma})$ is P-full, then it is also O-full. (ii) There is a subtype $\tau' :: \kappa$ of τ such that $(\sigma, \bar{\sigma})$ is winning and P-full over $[\![\tau' :: \kappa]\!]$.

A derivation $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ is *relevant* just if for each abstraction subterm $\lambda x^{\bigwedge_{i \in I} \tau_i}.M'$ of M and $i \in I$, M' has a free occurrence of $\mathsf{p}_i(x)$.

Lemma 4. $[\Gamma :: \Delta] \vdash [M :: t] : [\tau :: \kappa]$ *is relevant iff* $[\![M :: t]\!]$ *is P-full.*

5 Applications to HORS Model-Checking

Fix a ranked alphabet Σ and a HORS $G = \langle \Sigma, \mathcal{N}, S, \mathcal{R} \rangle$ we first give the game semantics $[\![G]\!]$ of G (see [1] for a definition of HORS). Let $\mathcal{N} = \{ F_1 : \kappa_1, \ldots, F_n : \kappa_n \}$ with $F_1 = S$ (start symbol), and $\Sigma = \{ a_1 : r_1, \ldots, a_m : r_m \}$ where each $r_i = ar(a_i)$, the arity of a_i. Writing $[\![\Sigma]\!] := \prod_{i=1}^{m} [\![o^{r_i} \to o]\!]$ and $[\![\mathcal{N}]\!] := \prod_{i=1}^{n} [\![\kappa_i]\!]$, the *game semantics* of G, $[\![G]\!] : [\![\Sigma]\!] \longrightarrow [\![o]\!]$, is the composite

$$[\![\Sigma]\!] \overset{\Lambda(\mathbf{g})}{\longrightarrow} ([\![\mathcal{N}]\!] \Rightarrow [\![\mathcal{N}]\!]) \overset{Y}{\longrightarrow} [\![\mathcal{N}]\!] \overset{\{ S::o \}}{\longrightarrow} [\![o]\!]$$

in the cartesian closed category of o-coloured arenas and innocent strategies, where $\mathbf{g} = \langle g_1, \ldots, g_n \rangle : [\![\Sigma]\!] \times [\![\mathcal{N}]\!] \longrightarrow [\![\mathcal{N}]\!]$ with $g_i = [\![\Sigma \cup \mathcal{N} \vdash \mathcal{R}(F_i) :: \kappa_i]\!]$ and Λ(-) is currying; Y is the standard fixpoint strategy (see [2, §7.2]); and $\{ S :: o \} = \pi_1 : [\![\mathcal{N}]\!] \longrightarrow [\![o]\!]$ is the projection map.

Remark 1. Since the set of P-views of $[\![G]\!]$ coincide with the branch language[3] of the *value tree* of G (i.e. the Σ-labelled tree generated by G; see [1]) and an innocent strategy is determined by its P-views, we identify the map $[\![G]\!]$ with the value tree of G.

Now fix a trivial automaton $\mathcal{B} = \langle Q, \Sigma, q_I, \delta \rangle$. We extend the game-semantic account to express the run tree of $\mathcal{B}$ over the value tree $[\![G]\!]$ in the category of Q-based two-level arenas and innocent strategies. First set

$$[\![\delta :: \Sigma]\!] := \prod_{a \in \Sigma} \bigwedge_{(q, a, \overline{q}) \in \delta} [\![q_1 \to \ldots \to q_{ar(a)} \to q :: o^{ar(a)} \to o]\!] = ([\![\delta]\!], U, [\![\Sigma]\!])$$

[3] Let m be the maximum arity of the symbols in Σ, and write $[m] = \{ 1, \cdots, m \}$. The *branch language* of $t : dom(t) \longrightarrow \Sigma$ consists of (i) $(f_1, d_1)(f_2, d_2) \cdots$ if there exists $d_1\, d_2 \cdots \in [m]^\omega$ s.t. $t(d_1 \cdots d_i) = f_{i+1}$ for every $i \in \omega$; and (ii) $(f_1, d_1) \cdots (f_n, d_n)\, f_{n+1}$ if there exists $d_1 \cdots d_n \in [m]^*$ s.t. $t(d_1 \cdots d_i) = f_{i+1}$ for $0 \leq i \leq n$, and the arity of f_{n+1} is 0.

where $[\![\delta]\!]$ is the Q-coloured arena $\prod_{a\in\Sigma}\prod_{(q,a,\overline{q})\in\delta}[\![q_1 \to \ldots \to q_{ar(a)} \to q]\!]$ and $\overline{q} = q_1, q_2, \cdots, q_{ar(a)}$.

A *run tree* of $\mathcal{B}$ over $[\![G]\!]$ is just an innocent strategy $(\rho, [\![G]\!])$ of the arena $[\![\delta :: \Sigma]\!] \Rightarrow [\![q_I :: o]\!] = ([\![\delta]\!] \Rightarrow [\![q_I]\!], V, [\![\Sigma]\!] \Rightarrow [\![o]\!])$. Every P-view $\bar{p} \in [\![G]\!]$ has a unique "colouring" i.e. a P-view $p \in \rho$ such that $V(p) = \bar{p}$. This associates a colour (state) with each node of the value tree, which corresponds to a run tree in the concrete presentation.

Characterisation by Complete Type Environment. Using G and $\mathcal{B}$ as before, Kobayashi [3] showed that $[\![G]\!]$ is accepted by $\mathcal{B}$ if, and only if, there is a *complete type environment* Γ, meaning that (i) $S : q_I \in \Gamma$, (ii) $\Gamma \vdash \mathcal{R}(F) : \theta$ for each $F : \theta \in \Gamma$. As a first application of two-level arena games, we give a semantic counterpart of the characterisation. Let $\Gamma = \{\, F_1 : \bigwedge_{j\in I_1} \tau_{1j} :: \kappa_1, \ldots, F_n : \bigwedge_{j\in I_n} \tau_{nj} :: \kappa_n \,\}$ be a type environment of G. Set $[\![\Gamma :: \mathcal{N}]\!] := \prod_{i=1}^{n} \bigwedge_{j\in I_i} [\![\tau_{ij} :: \kappa_i]\!] = ([\![\Gamma]\!], U_1, [\![\mathcal{N}]\!])$ where $[\![\Gamma]\!] := \prod_{i=1}^{n}\prod_{j\in I_i}[\![\tau_{ij}]\!]$.

Theorem 6. *Using Σ, G and $\mathcal{B}$ as before, $[\![G]\!]$ is accepted by $\mathcal{B}$ if, and only if, there exists Γ such that (i) $S : q_I \in \Gamma$, and (ii) there exists a strategy σ (say) of the Q-coloured arena $[\![\delta]\!] \times [\![\Gamma]\!] \Rightarrow [\![\Gamma]\!]$ such that $(\sigma, \mathbf{g})$ defines a winning strategy of the two-level arena $([\![\delta :: \Sigma]\!] \times [\![\Gamma :: \mathcal{N}]\!]) \Rightarrow [\![\Gamma :: \mathcal{N}]\!]$. (Thanks to Theorem 5, (ii) is equivalent to: $\Gamma \vdash \mathcal{R}(F) : \theta$ for each $F : \theta$ in Γ; hence Γ is complete.)*

Proof. (Sketch) We use Subject Expansion (Theorem 3) to prove the left-to-right direction. To prove the right-to-left, consider the composite

$$[\![\delta :: \Sigma]\!] \xrightarrow{\Lambda(\sigma,\mathbf{g})} ([\![\Gamma :: \mathcal{N}]\!] \Rightarrow [\![\Gamma :: \mathcal{N}]\!]) \xrightarrow{(Y,Y)} [\![\Gamma :: \mathcal{N}]\!] \xrightarrow{(\{S:q_I\},\{S::o\})} [\![q_I :: o]\!]. \quad \square$$

Minimality of Traversals-induced Typing Using the same notation as before, interaction sequences from $\mathbf{Int}(\Lambda(\mathbf{g}), \mathbf{fix}) \subseteq \mathrm{Int}([\![\Sigma]\!], [\![\mathcal{N}]\!] \Rightarrow [\![\mathcal{N}]\!], [\![o]\!])$ form a tree, which is (in essence) the *traversal tree* in the sense of Ong [1].

Prime types, which are intersection types of the form $\theta = \bigwedge_{i\in I_1}\theta_{1i} \to \cdots \to \bigwedge_{i\in I_n}\theta_{ni} \to q$, are equivalent to *variable profiles* (or simply *profiles*) [1]. Precisely θ corresponds to profile $\widehat{\theta} := (\{\, \widehat{\theta_{1i}} \mid i \in I_1 \,\}, \cdots, \{\, \widehat{\theta_{ni}} \mid i \in I_n \,\}, q)$. We write profiles of ground kind as q, rather than (q). Henceforth, we shall use prime types and profiles interchangeably.

Tsukada and Kobayashi [10] introduced (a kind-indexed family of) binary relations $\leq_\kappa$ between profiles of kind κ, and between sets of profiles of kind κ, by induction over the following rules.

(i) If for all $\theta \in A$ there exists $\theta' \in A'$ such that $\theta \leq_\kappa \theta'$ then $A \leq_\kappa A'$.
(ii) If $A_i \leq_{\kappa_i} A'_i$ for each i then $(A_1, \ldots, A_n; q) \leq_{\kappa_1\to\cdots\to\kappa_n\to o} (A'_1, \ldots, A'_n, q)$.

A *profile annotation* (or simply *annotation*) of the traversal tree $\mathbf{Int}(\Lambda(\mathbf{g}), \mathbf{fix})$ is a map of the nodes (which are move-occurrences of $M_{[\![\Sigma]\!]} + M_{[\![\mathcal{N}]\!]\Rightarrow[\![\mathcal{N}]\!]} + M_{[\![o]\!]}$) of the tree to profiles. We say that an annotation of the traversal tree is *consistent* just if whenever a move m, of kind $\kappa_1 \to \cdots \to \kappa_n \to o$ and simulates q, is annotated with profile $(A_1, \cdots, A_n, q')$, then (i) $q' = q$, (ii) for each i, A_i is a set of profiles of kind κ_i, (iii) if m' is annotated with θ and i-points to m, then $\theta \in A_i$. Now consider *annotated* *moves*, which are moves paired with their annotations, written (m, θ). We say that a profile annotation is *innocent* just if whenever $u_1 \cdot (m_1, \theta_1)$ and $u_2 \cdot (m_2, \theta_2)$ are even-length paths in the annotated traversal tree such that $\ulcorner u_1 \urcorner = \ulcorner u_2 \urcorner$, then $m_1 = m_2$ and $\theta_1 = \theta_2$.

Every consistent (and innocent) annotation α of an (accepting) traversal tree gives rise to a typing environment, written Γ_α, which is the set of bindings $F_i : \theta$ where $i \in \{1, \ldots, n\}$ and θ is the profile that annotates an occurrence of an initial move of $[\![\kappa_i]\!]$. Note that Γ_α is finite because there are only finitely many types of a given kind. We define a relation between annotations: $\alpha_1 \leq \alpha_2$ just if for each occurrence m of a move of kind κ in the traversal tree, $\alpha_1(m) \leq_\kappa \alpha_2(m)$.

Theorem 7. *(i) Let α be a consistent and innocent annotation of a traversal tree. Then Γ_α is a complete type environment.*

(ii) There is $\leq$-minimal consistent and innocent annotation, written $\alpha_{\min}$. Then $\Gamma_{\alpha_{\min}} \leq \Gamma_\alpha$ meaning that for all $F : \theta \in \Gamma_{\alpha_{\min}}$ there exists $F : \theta' \in \Gamma_\alpha$ such that $\theta \leq \theta'$.

(iii) Every complete type environment Γ determines a consistent and innocent annotation α_Γ of the traversal tree.

Game-Semantic Proof of Completeness of GTRecS. GTRecS [9] is a higher-order model checker proposed by Kobayashi. Although GTRecS is inspired by game-semantics, the formal development of the algorithm is purely type-theoretical and no concrete relationship to game semantics is known. Here we give a game-semantic proof of completeness of GTRecS based on two-level arena games.

The novelty of GTRecS lies in a function on type bindings, named $\mathbf{Expand}$. For a set Γ of nonterminal-type bindings, $\mathbf{Expand}(\Gamma)$ is defined as

$$\Gamma \cup \bigcup \{\Gamma' \cup \{F_i : \tau'\} \mid \Gamma \preceq_P \Gamma' \wedge \Gamma' \vdash \mathcal{R}(F_i) : \tau' \wedge \Gamma \preceq_O \{F_i : \tau'\}\},$$

where $\Gamma' \vdash \mathcal{R}(F_i) : \tau'$ is relevant. Here for types τ_1 and τ_2, $\tau_1 \preceq_P \tau_2$ if the arena $[\![\tau_2]\!]$ is obtained by adding only proponent moves to $[\![\tau_1]\!]$. For example, $(\bigwedge \emptyset) \to q \preceq_P ((\bigwedge \emptyset) \to q') \to q$ but $(\bigwedge \emptyset) \to q \not\preceq_P (q'' \to q') \to q$, since q' is at the proponent position and q'' at the opponent position. $\Gamma \preceq_P \Gamma'$ is defined as $\forall F : \tau' \in \Gamma'. \exists F : \tau \in \Gamma. \tau \preceq_P \tau'$. Similarly, $\tau \preceq_O \tau'$ and $\Gamma \preceq_O \Gamma'$ are defined.

Our goal is to analyse $\mathbf{Expand}$ game theoretically. The result is Lemma 5, which states that $\mathbf{Expand}$ overapproximates one step interaction of two strategies, σ and $\mathbf{fix}$. Completeness of GTRecS is a corollary of Lemma 5.

Assume that G is typable and fix a type environment $\Gamma = \{F_i : \bigwedge_j \tau_{i,j} \mid F_i \in \mathcal{N}\}$ such that $\vdash G : \Gamma$. Let $\sigma : [\![\delta]\!] \longrightarrow (\Gamma^1 \Rightarrow \Gamma^2)$ (here we use superscripts to distinguish occurrences of Γ) be the winning strategy that is induced from the derivation of $\vdash G : \Gamma$. The strategy $\mathbf{fix} : ([\![\Gamma^1]\!] \Rightarrow [\![\Gamma^2]\!]) \longrightarrow [\![q_I]\!]$ is defined as the composite of $([\![\Gamma]\!]^1 \Rightarrow [\![\Gamma]\!]^2) \xrightarrow{Y} [\![\Gamma]\!] \xrightarrow{\{S:q_I\}} [\![q_I]\!]$. For $n \in \{1, 2, \ldots\}$, the *nth approximation of* $\mathbf{fix}$ is defined by $\lfloor \mathbf{fix} \rfloor_n = \{s \in \mathbf{fix} \mid |s| \leq 2n + 1\}$. Thus $\lfloor \mathbf{fix} \rfloor_n$ is a strategy that behaves like $\mathbf{fix}$ until the nth interaction, but stops after that.

Let $n \in \{0, 1, 2, \ldots\}$. The n-th approximation of $\mathbf{fix}$ induces approximation of arenas and type environments. An arena $\lfloor \Gamma^1 \Rightarrow \Gamma^2 \rfloor_n$ is defined as the restriction of

$\Gamma^1 \Rightarrow \Gamma^2$ that consists of only moves appearing at $\mathbf{Int}(\sigma, \lfloor \mathbf{fix} \rfloor_n)$. The arena $\lfloor \Gamma^1 \Rightarrow \Gamma^2 \rfloor_n$ is decomposed as $\prod_{i,j}(\lfloor \Gamma^1 \rfloor_{n,i,j} \Rightarrow \lfloor \tau_{i,j} \rfloor_n)$. Let $\lfloor \Gamma^1 \rfloor_n$ be the union of variable-type bindings corresponding to $\bigcup_{i,j} \lfloor \Gamma^1 \rfloor_{n,i,j}$ and $\lfloor \Gamma^2 \rfloor_n$ be the set of type bindings $\{F_i : \lfloor \tau_{i,j} \rfloor_n\}_{i,j}$.

Lemma 5. $\lfloor \Gamma^1 \rfloor_n \cup \lfloor \Gamma^2 \rfloor_n \subseteq \mathbf{Expand}^n(\{S : q_0\})$.

Conclusions and Further Directions. Two-level arena games are an accurate model of intersection types. Thanks to Subject Expansion, they are a useful semantic framework for reasoning about higher-order model checking.

For future work, we aim to (i) consider properties that are closed under disjunction and quantifications, and (ii) study a call-by-value version of intersection games. In orthogonal directions, it would be interesting to (iii) construct an intersection game model for untyped recursion schemes [10], and (iv) build a CCC of intersection games parameterised by an alternating parity tree automaton, thus extending our semantic framework to mu-calculus properties.

Acknowledgement. This work is partially supported by Kakenhi 22 · 3842 and EPSRC EP/F036361/1. We thank Naoki Kobayashi for encouraging us to think about game-semantic proofs and for insightful discussions.

References

1. Ong, C.H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS, pp. 81–90. IEEE Computer Society (2006)
2. Hyland, J.M.E., Ong, C.H.L.: On full abstraction for PCF: I, II, and III. Inf. Comput. 163(2), 285–408 (2000)
3. Kobayashi, N.: Types and higher-order recursion schemes for verification of higher-order programs. In: Shao, Z., Pierce, B.C. (eds.) POPL, pp. 416–428. ACM (2009)
4. Salvati, S.: On the membership problem for non-linear abstract categorial grammars. Journal of Logic, Language and Information 19(2), 163–183 (2010)
5. Kobayashi, N., Ong, C.H.L.: A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In: LICS, pp. 179–188. IEEE Computer Society (2009)
6. Hague, M., Murawski, A.S., Ong, C.H.L., Serre, O.: Collapsible pushdown automata and recursion schemes. In: LICS, pp. 452–461 (2008)
7. Salvati, S., Walukiewicz, I.: Krivine Machines and Higher-Order Schemes. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 162–173. Springer, Heidelberg (2011)
8. Nielson, F.: Two-level semantics and abstract interpretation. Theor. Comput. Sci. 69(2), 117–242 (1989)
9. Kobayashi, N.: A Practical Linear Time Algorithm for Trivial Automata Model Checking of Higher-Order Recursion Schemes. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 260–274. Springer, Heidelberg (2011)
10. Tsukada, T., Kobayashi, N.: Untyped Recursion Schemes and Infinite Intersection Types. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 343–357. Springer, Heidelberg (2010)