

HIGHER-ORDER MATCHING AND GAMES

Colin Stirling

School of Informatics
University of Edinburgh
email: cps@inf.ed.ac.uk

Abstract. We provide a game-theoretic characterisation of higher-order matching. The idea is suggested by model checking games. We then show that some known decidable instances of matching can be uniformly proved decidable via the game-theoretic characterisation.

Keywords: games, higher-order matching, typed lambda calculus.

1 THE MATCHING PROBLEM

Assume simply typed lambda calculus with base type $\mathbf{0}$ and the definitions of α -equivalence, β and η -reduction. A type is $\mathbf{0}$ or $A \rightarrow B$ where A and B are types. A type A always has the form $(A_1 \rightarrow (\dots A_n \rightarrow \mathbf{0}) \dots)$ which is usually written $A_1 \rightarrow \dots \rightarrow A_n \rightarrow \mathbf{0}$. We also assume a standard definition of *order*: the order of $\mathbf{0}$ is 1 and the order of $A_1 \rightarrow \dots \rightarrow A_n \rightarrow \mathbf{0}$ is $k + 1$ where k is the maximum of the orders of the A_i s.

Terms are built from a countable set of variables x, y, \dots and constants, a, f, \dots : each variable and constant is assumed to have a unique type. The set of simply typed terms is the smallest set T such that if x (f) has type A then $x : A \in T$ ($f : A \in T$), if $t : B \in T$ and $x : A \in T$, then $\lambda x.t : A \rightarrow B \in T$, and if $t : A \rightarrow B \in T$ and $u : A \in T$ then $tu : B \in T$. The *order* of a typed term is the order of its type. A typed term is *closed* if it does not contain free variables.

A *matching problem* has the form $v = u$ where $v, u : A$ for some type A , and u is closed. The *order* of the problem is the maximum of the orders of the free variables x_1, \dots, x_n in v . A *solution* of a matching problem is a sequence of terms t_1, \dots, t_n such that $v\{t_1/x_1, \dots, t_n/x_n\} =_{\beta\eta} u$. The decision question is: given a matching problem, does it have a solution? The problem is conjectured to be decidable in [3]. However, if it is decidable then its complexity is non-elementary [9, 11]. Decidability has been proven for the general problem up to order 4 and for various special cases [5, 6, 8]. Loader proved that the matching problem is undecidable for the variant definition of solution that uses just β -equality [4]. An excellent source of information about the problem is [2].

Throughout, we slightly change the syntax of terms and types. The type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow \mathbf{0}$ is rewritten $(A_1, \dots, A_n) \rightarrow \mathbf{0}$ and we assume that all terms in normal form are in *η -long form*. That is, if $t : \mathbf{0}$ then it either has the form $u : \mathbf{0}$ where u is a constant or a variable, or has the form $u(t_1, \dots, t_k)$ where $u : (B_1, \dots, B_k) \rightarrow \mathbf{0}$ is either a constant or a variable and each $t_i : B_i$

is in η -long form. And if $t : (A_1, \dots, A_n) \rightarrow \mathbf{0}$ then t has the form $\lambda y_1 \dots y_n. t_0$ where $t_0 : \mathbf{0}$ is a term in η -long form. A term is *well-named* if each occurrence of a variable y within a λ abstraction is unique.

An interpolation equation has the form $x(v_1, \dots, v_n) = u$ where each v_i is a closed term in normal form and $u : \mathbf{0}$ is also in normal form. The *type* of the equation is the type of the free variable x , which has the form $(A_1, \dots, A_n) \rightarrow \mathbf{0}$ where $v_i : A_i$. An *interpolation problem* P is a finite family of interpolation equations $x(v_1^i, \dots, v_n^i) = u_i$, $i : 1 \leq i \leq m$, all with the same free variable x . The *type* of P is the type A of the variable x and the *order* of P is the order of A . A *solution* of P of type A is a closed term $t : A$ such that $t(v_1^i, \dots, v_n^i) =_{\beta} u_i$ for each i . We write $t \models P$ if the closed term t solves the problem P .

An interpolation problem reduces to matching: there is the equivalent problem $f(x(v_1^1, \dots, v_n^1), \dots, x(v_1^m, \dots, v_n^m)) = f(u_1, \dots, u_m)$, when $f : \mathbf{0}^m \rightarrow \mathbf{0}$. Schubert shows the converse, that a matching problem of order n is reducible to an interpolation problem of order at most $n + 2$ [7]. A *dual* interpolation problem includes inequations $x(v_1^i, \dots, v_n^i) \neq u_i$. Padovani proved that a matching problem of order n is reducible to dual interpolation of the same order [6]. In the following we concentrate on the interpolation problem for orders greater than 1. If P has order 1 then it has the form $x = u_i$, $1 \leq i \leq m$. Consequently, P only has a solution if $u_i = u_j$ for each i and j .

In the following we develop a game-theoretic characterisation of $t \models P$. The idea is inspired by model-checking games (such as in [10]) where a structure, a transition graph, is navigated relative to a property and players make choices at appropriate positions. In section 2 we define some preliminary notions and in section 3 we present the term checking game and prove its correctness. Unlike transition graphs, terms t involve binding which results in moves that jump around t . The main virtue of using games is that they allow one to understand little “pieces” of a solution term t in terms of subplays and how they thereby contribute to solving P . In section 4 we identify regions of a term t that we call “tiles” and define their subplays. In section 5 we introduce four transformations on tiles that preserve a solution term: these transformations are justified by analysing subplays. In section 6 we then show that the transformations provide simple proofs of decidability for known instances of the interpolation problem via the small model property: if $t \models P$ then $t' \models P$ for some small term t' .

2 PRELIMINARIES

A right term u of an interpolation equation may contain bound variables: an example is $f(a, \lambda x_1 \dots x_4. x_1(x_1(x_2)))$. Let $X = \{x_1, \dots, x_k\}$ be the set of bound variables in u . Assume a fresh set of constants $C = \{c_1, \dots, c_k\}$ such that each c_i has the same type as x_i .

Definition 1 The *ground closure* of a closed term w , whose bound variables belong to X , with respect to C , written $\text{Cl}(w, X, C)$, is defined inductively:

1. if $w = a : \mathbf{0}$, then $\text{Cl}(w, X, C) = \{a\}$

2. if $w = f(w_1, \dots, w_n)$, then $\text{Cl}(w, X, C) = \{w\} \cup \bigcup \text{Cl}(w_i, X, C)$
3. if $w = \lambda x_{j_1} \dots x_{j_n}.u$, then $\text{Cl}(w, X, C) = \text{Cl}(u\{c_{j_1}/x_{j_1}, \dots, c_{j_n}/x_{j_n}\}, X, C)$

The ground closure of $u = f(a, \lambda x_1 \dots x_4.x_1(x_1(x_2)))$ with respect to $\{c_1, \dots, c_4\}$ is the set of ground terms $\{u, a, c_1(c_1(c_2)), c_1(c_2), c_2\}$.

Next, we wish to identify subterms of the left-hand terms v_j of an interpolation equation relative to a finite set of constants C .

Definition 2 The *subterms* of w relative to C , written $\text{Sub}(w, C)$, is defined inductively using an auxiliary set $\text{Sub}'(w, C)$:

1. if w is a variable or a constant, then $\text{Sub}(w, C) = \text{Sub}'(w, C) = \{w\}$
2. if w is $x(w_1, \dots, w_n)$ then $\text{Sub}(w, C) = \text{Sub}'(w, C) = \{w\} \cup \bigcup \text{Sub}(w_i, C)$
3. if w is $f(w_1, \dots, w_n)$, then $\text{Sub}(w, C) = \text{Sub}'(w, C) = \{w\} \cup \bigcup \text{Sub}'(w_i, C)$
4. if w is $\lambda y_1 \dots y_n.v$, then $\text{Sub}(w, C) = \{w\} \cup \text{Sub}(v, C)$
5. if w is $\lambda y_1 \dots y_n.v$, then $\text{Sub}'(w, C) = \bigcup \text{Sub}(v\{c_{i_1}/y_1, \dots, c_{i_n}/y_n\}, C)$ where each $c_{i_j} \in C$ has the same type as y_j

For the remainder of the paper we assume a fixed interpolation problem P of type A whose order is greater than 1. P has the form $x(v_1^i, \dots, v_n^i) = u_i$, $1 \leq i \leq m$, where each v_j^i and u_i are in long normal form. We also assume that terms v_j^i and u_i are well-named and that no pair share bound variables. For each i , let X_i be the (possibly empty) set of bound variables in u_i and let C_i be a corresponding set of new constants (that do not occur in P), the *forbidden* constants. We are interested in when $t \models P$ and t does not contain forbidden constants.

Definition 3 Assume $P : A$ is the fixed interpolation problem:

1. T is the set of subtypes of A and the subtypes of subterms of u_i
2. for each i , the right subterms are $\mathsf{R}_i = \text{Cl}(u_i, X_i, C_i)$
3. for each i , the left subterms are $\mathsf{L}_i = \bigcup \text{Sub}(v_j^i, C_i) \cup C_i$

3 TREE-CHECKING GAMES

Using ideas suggested by model-checking we present a characterisation of interpolation. This is not the first time that such techniques have been applied to higher-order matching. Comon and Jurski define (bottom-up) tree automata for the 4th-order case that characterise all solutions to a problem [1]. The states of the automata essentially depend on Padovani's representation of the observational equivalence classes of terms up to 4th-order [6]. The existence of such an automaton not only guarantees decidability, but also shows that the set of all solutions is regular.

We now introduce a game-theoretic characterisation of interpolation for *all* orders. The idea is inspired by model-checking games where a model (a transition graph) is traversed relative to a property and players make choices at appropriate positions. Similarly, in the following game the model is a putative solution term

t that is traversed relative to the interpolation problem. However, because of binding play may jump here and there in t . Consequently, our games lack the simple control structure of Comon and Jurski's automata where flow starts at the leaves of t and proceeds to its root. Moreover, the existence of the game does not assure decidability. Its purpose is to provide a mechanism for understanding how small pieces of a solution term contribute to solving the problem.

- A. $t_m = \lambda y_1 \dots y_j$ and $t_m \downarrow_1 t'$ and $q_m = q[(l_1, \dots, l_j), r]$. So, $t_{m+1} = t'$ and $\theta_{m+1} = \theta_m \{l_1 \eta_m / y_1, \dots, l_j \eta_m / y_j\}$ and q_{m+1} and η_{m+1} are by cases on t_{m+1} .
1. $a : \mathbf{0}$. So, $\eta_{m+1} = \eta_m$. If $r = a$ then $q_{m+1} = q[\exists]$ else $q_{m+1} = q[\forall]$.
 2. $f : (B_1, \dots, B_k) \rightarrow \mathbf{0}$. So, $\eta_{m+1} = \eta_m$. If $r = f(s_1, \dots, s_k)$ then $q_{m+1} = q_m$ else $q_{m+1} = q[\forall]$.
 3. $y : B$. If $\theta_{m+1}(y) = l\eta_i$, then $q_{m+1} = q[l, r]$ and $\eta_{m+1} = \eta_i$.
- B. $t_m = f : (B_1, \dots, B_k) \rightarrow \mathbf{0}$ and $q_m = q[(l_1, \dots, l_j), f(s_1, \dots, s_k)]$. So, $\theta_{m+1} = \theta_m$ and $\eta_{m+1} = \eta_m$ and q_{m+1} and t_{m+1} are decided as follows.
1. \forall chooses a direction $i' : 1 \leq i' \leq k$ and $t_m \downarrow_{i'} t'$. So, $t_{m+1} = t'$.
If $s_{i'} : \mathbf{0}$, then $q_{m+1} = q[(\), s_{i'}]$. If $s_{i'}$ is $\lambda x_{i_1} \dots x_{i_n}.s$ then $q_{m+1} = q[(c_{i_1}, \dots, c_{i_n}), s\{c_{i_1}/x_{i_1}, \dots, c_{i_n}/x_{i_n}\}]$.
- C. $t_m = y$ and $q_m = q[l, r]$. If $l = \lambda z_1 \dots z_j.w$ and $t_m \downarrow_i t'_i$, $1 \leq i \leq j$, then $\eta_{m+1} = \eta_m \{t'_i \theta_m / z_1, \dots, t'_j \theta_m / z_j\}$ else $\eta_{m+1} = \eta_m$. The remaining components t_{m+1} , q_{m+1} and η_{m+1} are by cases on l .
1. $a : \mathbf{0}$ or $\lambda \bar{z}.a$. So, $t_{m+1} = t_m$ and $\theta_{m+1} = \theta_m$. If $r = a$ then $q_{m+1} = q[\exists]$ else $q_{m+1} = q[\forall]$.
 2. $c : (B_1, \dots, B_k) \rightarrow \mathbf{0}$. So, $\theta_{m+1} = \theta_m$. If $r \neq c(s_1, \dots, s_k)$ then $t_{m+1} = t_m$ and $q_{m+1} = q[\forall]$. If $r = c(s_1, \dots, s_k)$ then \forall chooses a direction $i' : 1 \leq i' \leq k$ and $t_m \downarrow_{i'} t'$. So, $t_{m+1} = t'$. If $s_{i'} : \mathbf{0}$, then $q_{m+1} = q[(\), s_{i'}]$. If $s_{i'}$ is $\lambda x_{i_1} \dots x_{i_n}.s$ then $q_{m+1} = q[(c_{i_1}, \dots, c_{i_n}), s\{c_{i_1}/x_{i_1}, \dots, c_{i_n}/x_{i_n}\}]$.
 3. $f(w_1, \dots, w_k)$ or $\lambda \bar{z}.f(w_1, \dots, w_k)$. So, $t_{m+1} = t_m$ and $\theta_{m+1} = \theta_m$. If $r \neq f(s_1, \dots, s_k)$, then $q_{m+1} = q[\forall]$. If $r = f(s_1, \dots, s_k)$ then \forall chooses a direction $i' : 1 \leq i' \leq k$. If $s_{i'} : \mathbf{0}$ then $q_{m+1} = q[w_{i'}, s_{i'}]$. If $w_{i'} = \lambda z'_1 \dots z'_n.w$ and $s_{i'} = \lambda x_{i_1} \dots x_{i_n}.s$, then $q_{m+1} = q[w\{c_{i_1}/z'_1, \dots, c_{i_n}/z'_n\}, s\{c_{i_1}/x_{i_1}, \dots, c_{i_n}/x_{i_n}\}]$.
 4. $z'(l_1, \dots, l_k)$ or $\lambda \bar{z}.z'(l_1, \dots, l_k)$. If $\eta_{m+1}(z') = t'\theta_i$ then $\theta_{m+1} = \theta_i$ and $t_{m+1} = t'$ and $q_{m+1} = q[(l_1, \dots, l_k), r]$.

Fig. 1. Game moves

We assume that a potential solution term t for P has the right type, is in long normal form, is well-named (with variables that are disjoint from variables in P) and does not contain forbidden constants. The term t is represented as a tree, $\text{tree}(t)$. If t is $y : \mathbf{0}$ or $a : \mathbf{0}$ then $\text{tree}(t)$ is the single node labelled with t . In the case of $u(v_1, \dots, v_k)$ when u is a variable or a constant, we assume that a dummy λ with the empty sequence of variables is placed before any subterm $v_i : \mathbf{0}$ in the tree representation. With this understanding, if t is $u(v_1, \dots, v_n)$, then $\text{tree}(t)$ consists of the root node labelled u and n -successor nodes labelled with $\text{tree}(v_i)$. We use the notation $u \downarrow_i t'$ to represent that $\text{tree}(t')$ is the i th

successor of the node u . If t is $\lambda\bar{y}.v$, where \bar{y} is a possibly empty sequence of variables $y_1 \dots y_n$, then $\text{tree}(t)$ consists of the root node labelled $\lambda\bar{y}$ and a single successor node $\text{tree}(v)$: in this case we assume $\lambda\bar{y} \downarrow_1 \text{tree}(v)$. We also assume that each node labelled with an occurrence of a variable y_j has a backward arrow \uparrow^j to the $\lambda\bar{y}$ that binds it: the index j tells us which element is y_j in \bar{y} .

The tree representation of $\lambda y_1 y_2. f(f(y_2, y_2), y_1(y_2))$ is tantamount to the syntax tree of $\lambda y_1 y_2. f(\lambda. f(\lambda. y_2, \lambda. y_2), \lambda. y_1(\lambda. y_2))$. In the following we use t to be the λ -term t , or its λ -tree or the label (a constant, variable or $\lambda\bar{y}$) at its root node.

The tree-checking game $G(t, P)$ is played by one participant, player \forall , the refuter who attempts to show that t is not a solution of P . The game appeals to a finite set of states involving elements of L_i and R_i . There are three kinds of states: argument, value and final states. Argument states have the form $q[(l_1, \dots, l_k), r]$ where each $l_j \in L_i$ (and k can be 0) and $r \in R_i$. Value states have the form $q[l, r]$ where $l \in L_i$ and $r \in R_i$. A final state is either $q[\forall]$, the winning state for \forall , or $q[\exists]$, the losing state for \forall .

The game appeals to a sequence of supplementary look-up tables θ_j and η_j , $j \geq 1$: θ_j is a partial map from variables in t to elements $w\eta_k$ where $w \in L_i$ and $k < j$, and η_j is a partial map from variables in L_i to elements $t'\theta_k$ where t' is a node of the tree t and $k < j$. The initial elements θ_1 and η_1 are both the empty table.

A *play* of $G(t, P)$ is a sequence of positions $t_1 q_1 \theta_1 \eta_1, \dots, t_n q_n \theta_n \eta_n$ where each t_i is a node of t and $t_1 = \lambda\bar{y}$ is the root of t , and each q_i is a state, and q_n is a final state. A node t' of the tree t may repeatedly occur in a play. The initial state is decided as follows: \forall chooses an equation $x(v_1^i, \dots, v_n^i) = u_i$ from P and $q_1 = q[(v_1^i, \dots, v_n^i), u_i]$. If the current position is $t_m q_m \theta_m \eta_m$ and q_m is not a final state, then the next position $t_{m+1} q_{m+1} \theta_{m+1} \eta_{m+1}$ is determined by a move of Figure 1.

Moves are divided into three groups that depend on t_m . Group A covers the case when $t_m = \lambda\bar{y}$, group B when $t_m = f$ and group C when $t_m = y$. We assume standard updating notation for θ_{m+1} and η_{m+1} : $\beta\{\alpha_1/y_1, \dots, \alpha_m/y_m\}$ is the function similar to β except that $\beta(y_i) = \alpha_i$. Moreover, in the case of rules B1, C2 and C3 we assume that the constants c_{i_j} belong to the forbidden sets C_i . The look-up tables are used in rules A3 and C4. If $t_m = \lambda\bar{y}$ and $t_m \downarrow_1 t_{m+1} = y$, then η_{m+1} and q_{m+1} are determined by the entry for y in θ_{m+1} : if the entry is $l\eta_i$, then l is the left element of q_{m+1} and $\eta_{m+1} = \eta_i$. In the case of C4, if $t_m = y$ and $q_m = q[l, r]$ and $l = z'(l_1, \dots, l_k)$ or $\lambda\bar{z}.z'(l_1, \dots, l_k)$, then θ_{m+1} and t_{m+1} are determined by the entry for z' in the table η_{m+1} : if the entry is $t'\theta_i$ then $t_{m+1} = t'$ and $\theta_{m+1} = \theta_i$. It is this rule that allows the next move to be a jump around the term tree (to a node labelled with a λ). The moves A1-A3, B1 and C2 traverse down the term tree while C1 and C3 remain at the current node.

Example 1 Let P be the problem $x(v) = u$ where $v = \lambda z.z$ and $u = f(\lambda x.x)$. Let $X = \{x\}$ and $C = \{c\}$ and let t be the term $\lambda y.y(y(f(\lambda y_1.y_1)))$ and so, $\text{tree}(t)$ is

$$(t_1)\lambda y \downarrow_1 (t_2)y \downarrow_1 (t_3)\lambda \downarrow_1 (t_4)y \downarrow_1 (t_5)\lambda \downarrow_1 (t_6)f \downarrow (t_7)\lambda y_1 \downarrow_1 (t_8)y_1$$

There is just one play of $\mathbf{G}(t, P)$, as follows.

$t_1 q[(\lambda z.z), f(\lambda x.x)] \theta_1 \eta_1$			
$t_2 q[\lambda z.z, f(\lambda x.x)] \theta_2 \eta_2$	$\theta_2 = \theta_1\{(\lambda z.z)\eta_1/y\}$	$\eta_2 = \eta_1$	A3
$t_3 q[(\), f(\lambda x.x)] \theta_3 \eta_3$	$\theta_3 = \theta_2$	$\eta_3 = \eta_2\{t_3\theta_2/z\}$	C4
$t_4 q[\lambda z.z, f(\lambda x.x)] \theta_4 \eta_4$	$\theta_4 = \theta_3$	$\eta_4 = \eta_1$	A3
$t_5 q[(\), f(\lambda z.z)] \theta_5 \eta_5$	$\theta_5 = \theta_4$	$\eta_5 = \eta_4\{t_5\theta_4/z\}$	C4
$t_6 q[(\), f(\lambda z.z)] \theta_6 \eta_6$	$\theta_6 = \theta_5$	$\eta_6 = \eta_5$	A2
$t_7 q[(c), c] \theta_7 \eta_7$	$\theta_7 = \theta_6$	$\eta_7 = \eta_6$	B1
$t_8 q[(c), c] \theta_8 \eta_8$	$\theta_8 = \theta_7\{c\eta_7/y_1\}$	$\eta_8 = \eta_7$	A3
$t_8 q[\exists] \theta_9 \eta_9$	$\theta_9 = \theta_8$	$\eta_9 = \eta_8$	C1

The game rule applied to produce a move is also given. □

A partial play of $\mathbf{G}(t, P)$ finishes when a final state, $q[\forall]$ or $q[\exists]$, occurs. Player \forall loses a play if the final state is $q[\exists]$ and \forall loses the game $\mathbf{G}(t, P)$ if she loses every play. The following result provides a characterisation of $t \models P$.

Theorem 1 \forall loses $\mathbf{G}(t, P)$ if, and only if, $t \models P$.

Proof. For any position $t_i q_i \theta_i \eta_i$ of a play of $\mathbf{G}(t, P)$ we say that it m -holds (m -fails) if $q = q[\exists]$ ($q = q[\forall]$) and when q_i is not final, by cases on t_i and q_i (and look-up tables become delayed substitutions)

- if $t_i = \lambda \bar{y}$ and $q_i = q[(l_1, \dots, l_k), r]$ and t' is $(t_i \theta_i)(l_1 \eta_i, \dots, l_k \eta_i)$ then $t' = r$ ($t' \neq r$) and t' normalises with m β -reductions
- if $t_i = f$ and $q_i = q[(l_1, \dots, l_k), r]$ and t' is $t_i \theta_i$ then $t' = r$ ($t' \neq r$) and t' normalises with m β -reductions
- if $t_i = z$ and $q_i = q[l, r]$ and $t_i \downarrow_j t'_j$ and t' is $l \eta_i (t'_1 \theta_i, \dots, t'_k \theta_i)$ then $t' = r$ ($t' \neq r$) and t' normalises with m β -reductions.

The following are easy to show by case analysis.

1. if $t_i q_i \theta_i \eta_i$ m -holds then $q_i = q[\exists]$ or for any next position $t_{i+1} q_{i+1} \theta_{i+1} \eta_{i+1}$ it m' -holds, $m' < m$, or it m' -holds, $m' \leq m + 1$, and the right-term in q_{i+1} is smaller than in q_i
2. if $t_i q_i \theta_i \eta_i$ m -fails then $q_i = q[\forall]$ or there is a next position $t_{i+1} q_{i+1} \theta_{i+1} \eta_{i+1}$ and it m' -fails, $m' < m$, or it m' -fails, $m' \leq m + 1$, and the right-term in q_{i+1} is smaller than in q_i

For instance, assume $t_i q_i \theta_i \eta_i$ m -holds and $t_i = \lambda y_1 \dots y_k$ and $t_i \downarrow_1 t_{i+1} = y$ and $t_{i+1} \downarrow_j t'_j$ and $q_i = q[(l_1, \dots, l_k), r]$. So, $\theta_{i+1} = \theta_i\{l_j \eta_i / \bar{y}_j\}$ and $q_{i+1} = q[l, r]$ if $\theta_{i+1}(y) = l \eta_n$ and $\eta_{i+1} = \eta_n$. So, $t_i = \lambda y_1 \dots y_k . y(t'_1, \dots, t'_m)$ and by assumption $(t_i \theta_i)(l_1 \eta_i, \dots, l_k \eta_i) = r$. With a β -reduction we get $\theta_{i+1}(y)(t'_1 \theta_{i+1}, \dots, t'_m \theta_{i+1})$ which is $(l \eta_{i+1})(t'_1 \theta_{i+1}, \dots, t'_m \theta_{i+1})$ and so position $t_{i+1} q_{i+1} \theta_{i+1} \eta_{i+1}$ ($m - 1$)-holds. Next, assume $t_i q_i \theta_i \eta_i$ m -holds, $t_i = f$, $q_i = q[(l_1, \dots, l_j), f(s_1, \dots, s_k)]$ and $t_i \downarrow_j t'_j$. By assumption, $f(t'_1, \dots, t'_k) \theta_i = f(s_1, \dots, s_k)$. So, $t'_j \theta_i = s_j$. Consider any choice of next position. If $s_j : \mathbf{0}$ then $q_{i+1} = q[(\), s_j]$, $t_{i+1} = t'_j$ and $\theta_{i+1} = \theta_i$. Therefore, $t'_j \theta_{i+1} = s_j$ and so this next position either m' -holds, $m' < m$ or m -holds and s_j is smaller than $f(s_1, \dots, s_k)$. Alternatively, $s_j = \lambda \bar{x}.s$. Therefore,

$t'_j = \lambda \bar{z}.t'$ and $t'\theta_i\{\bar{c}_i/\bar{z}_i\} = s\{\bar{c}_i/\bar{x}_i\}$ where the c_i s are new, m' -holds for $m' \leq m$. And so $t'_j\theta_i(c_1, \dots, c_n) = s\{\bar{c}_i/\bar{x}_i\}$ ($m' + 1$)-holds, as required. Assume $t_i q_i \theta_i \eta_i$ m -holds and $t_i = y$, $q_i = q[l, r]$, $l = \lambda z_1 \dots z_k.w$, $w = z(l_1, \dots, l_m)$, $t_i \downarrow_j t'_j$ and $t_{i+1} \theta_{i+1} = \eta_{i+1}(z)$. By assumption, $(\lambda z_1 \dots z_k.w)\eta_i(t'_1 \theta_i, \dots, t'_k \theta_i) = r$. With one β -reduction $\eta_{i+1}(z)(l_1 \eta_{i+1}, \dots, l_m \eta_{i+1}) = r$, that is $t'_{i+1} \theta_{i+1}((l_1 \eta_{i+1}, \dots, l_m \eta_{i+1})) = r$ and so the next position ($m - 1$)-holds. All other cases of 1 are similar to one of these three, and the proof of 2 is also very similar.

The result follows from 1 and 2: if $t \models P$ then for each initial position there is an m such that it m -holds and if $t \not\models P$ then there is an initial position that m -fails. \square

The tree checking game can be easily extended to characterise dual interpolation by including a second player \exists who is responsible for choices involving inequations.

Assume that $t_0 \models P$, so \forall loses the game $\mathbf{G}(t_0, P)$. The number of plays is the number of branches in the right terms of P . We can index each play with $i\alpha$ when α is a branch of the right-term of the i th equation of P containing forbidden constants: $\pi^{i\alpha}$ is the play where all \forall choices are dictated by α . This means that two plays $\pi^{i\alpha}$, $\pi^{i\beta}$ have a common prefix and differ after a position involving a \forall choice, when the branches α and β diverge.

We also allow π to range over *subplays* which are consecutive subsequences of positions of any play of $\mathbf{G}(t_0, P)$. The length of π , $|\pi|$, is the number of positions in π . We let $\pi(i)$ be the i th position of π , $\pi(i, j)$ be the interval $\pi(i), \dots, \pi(j)$ and π_i be its i th suffix, the interval $\pi(i, |\pi|)$. For ease of notation, we write $t \in \pi(i)$, $q \in \pi(i)$, $\theta \in \pi(i)$ and $\eta \in \pi(i)$ if $\pi(i) = tq\theta\eta$ and $t \notin \pi(i)$ means that $\pi(i) = t'q\theta\eta$ and $t \neq t'$. If $q = q[(l_1, \dots, l_k), r]$ or $q[l, r]$ then its *right-term* is r .

Definition 1 A subplay π is *ri, right-term invariant*, if $q \in \pi(1)$ and $q' \in \pi(|\pi|)$ share the same right-term r .

Definition 2 Table θ' *extends* θ if for all $y \in \text{dom}(\theta)$, $\theta'(y) = \theta(y)$. Similarly, η' extends η if for all $z \in \text{dom}(\eta)$, $\eta'(z) = \eta(z)$.

We widen the usage of “extends” to positions: $\pi(j)$ θ -extends $\pi(i)$ if $\theta' \in \pi(j)$ extends $\theta \in \pi(i)$, $\pi(j)$ η -extends $\pi(i)$ if $\eta' \in \pi(j)$ extends $\eta \in \pi(i)$ and $\pi(j)$ extends $\pi(i)$ if $\pi(j)$ θ -extends and η -extends $\pi(i)$.

If $\pi(i)$'s look-up table is called when move A3 or C4 produces $\pi(j)$ then $\pi(j)$ is a child of $\pi(i)$.

Definition 3 Assume $\pi \in \mathbf{G}(t_0, P)$. If $\pi(i) = tq[(l_1, \dots, l_k), r]\theta\eta$, $\pi(j) = t'q[l_m, r']\theta'\eta$, $\theta'(t') = l_m\eta$ and $t' \uparrow^m t$, then $\pi(j)$ is a *child* of $\pi(i)$. If $\pi(i) = yq[\lambda z_1 \dots \lambda z_k.w, r]\theta\eta$, $\pi(j-1) = y'q[l, r']\theta'\eta'$, $l = \lambda \bar{x}.z_m(\bar{l})$ or $\lambda \bar{x}.z_m$ or $z_m(\bar{l})$ or z_m and $\eta'(z_m) = t'\eta$ and $y \downarrow_m t'$, then $\pi(j)$ is a *child* of $\pi(i)$.

Fact 1 If $\pi(j)$ is a child of $\pi(i)$ then $\pi(j)$ extends $\pi(i)$.

4 TILES AND SUBPLAYS

Assume that $t_0 \models P$. We would like to identify regions of the tree t_0 . For this purpose, we define *tiles* that are partial trees.

Definition 1 Assume $B = (B_1, \dots, B_k) \rightarrow \mathbf{0} \in \mathbb{T}$.

1. λ is an *atomic leaf* of type $\mathbf{0}$
2. if $x_j : B_j$ then $\lambda x_1 \dots x_k$ is an *atomic leaf* of type B
3. $a : \mathbf{0}$ is a *constant* tile
4. if $f : B$ and $t_j : B_j$ are atomic leaves then $f(t_1, \dots, t_k)$ is a *constant* tile
5. $y : \mathbf{0}$ is a *simple* tile
6. if $y : B$ and $t_j : B_j$ are atomic leaves then $y(t_1, \dots, t_k)$ is a *simple* tile

A region of t_0 can be identified with a constant or simple tile. A leaf $u : \mathbf{0}$ of t_0 is the tile u . If $B \neq \mathbf{0}$ then an occurrence of $u : B$ in t_0 , $u = f$ or y , with its immediate children $\lambda \bar{x}_1, \dots, \lambda \bar{x}_k$, where \bar{x}_i may be empty, is the tile $u(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ in t_0 .

Tiles in t_0 induce subplays of $\mathbf{G}(t_0, P)$. A play on $t = f(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ is a pair of positions $\pi(i, i+1)$ with $t \in \pi(i)$: $q[(l_1, \dots, l_m), r] \in \pi(i)$, $r = f(s_1, \dots, s_k)$, $\lambda \bar{x}_j \in \pi(i+1)$ is a leaf of t and $q[(\cdot), s_j]$ or $q[(c_1, \dots, c_n), s_j \{\bar{c}_{i'}/\bar{z}_{i'}\}]$ is the state in $\pi(i+1)$, depending on the type of s_j .

Definition 2 A subplay π is a *play* on $y(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ in t_0 if $y \in \pi(1)$ and $\pi(|\pi|)$ is a child of $\pi(1)$. It is a *j-play* if $\lambda \bar{x}_j \in \pi(|\pi|)$.

A play π on $y(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ in t_0 can have arbitrary length. It starts at y and finishes at a leaf $\lambda \bar{x}_i$. In between, flow of control can be almost anywhere in t_0 (including y). Crucially, $\pi(|\pi|)$ extends $\pi(1)$: the free variables in the subtree of t_0 rooted at y preserve their values, and the free variables in w when $q[\lambda z_1 \dots z_k.w, r] \in \pi(1)$ also preserve their values. If $\pi \in \mathbf{G}(t_0, P)$ and $y \in \pi(i)$ then there can be numerous plays $\pi(i, j)$ on $y(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ in t_0 , including no plays at all. We now examine some pertinent properties of plays

Proposition 1 Assume $\pi \in \mathbf{G}(t_0, P)$, $\pi(i, m)$ and $\pi(i, n)$, $n > m$, are plays on $y(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ and $\lambda \bar{x}_j \in \pi(m)$.

1. There is a position $\pi(m')$, $m' < n$, that is a child of $\pi(m)$.
2. If $\pi(m')$ is the first position that is a child of $\pi(m)$, $t' \in \pi(m')$, y_1 occurs on the branch between $\lambda \bar{x}_j$ and t' , t' is an i' -descendent of y_1 and $y_1 \downarrow_{i'} \lambda \bar{z}_{i'}$, then there is an i' -play $\pi(m_1, n_1)$ on $y_1(\lambda \bar{z}_1, \dots, \lambda \bar{z}_{k'})$ such that $m < m_1$ and $n_1 < m'$.
3. If $\pi(m+m')$ is the first position that is a child of $\pi(m)$, $\pi(m, m+m')$ is *ri* and $\pi(i, n)$ is a *j-play* then $\pi(n+m')$ is the first position that is a child of $\pi(n)$, $\pi(n, n+m')$ is *ri* and for all $n' \leq m'$, $t \in \pi(m+n')$ iff $t \in \pi(n+n')$.
4. If $\pi(m+m')$ is the first position that is a child of $\pi(m)$, $\pi(m, m+m')$ is *not ri* and $\pi(i, n)$ is a *j-play* then there is a $\pi' \in \mathbf{G}(t_0, P)$ with $\pi'(n) = \pi(n)$, $\pi'(n+m')$ is the first position that is a child of $\pi'(n)$, $\pi'(n, n+m')$ is *not ri* and for all $n' \leq m'$, $t \in \pi(m+n')$ iff $t \in \pi'(n+n')$.

Proof. 1. Assume $\pi(i) = y q[\lambda z_1 \dots z_k . w, r] \theta \eta_i$ and $\pi(i, m), \pi(i, n)$ are plays on $y(\lambda \bar{x}_1, \dots, \lambda \bar{x}_k)$ with $\lambda \bar{x}_j \in \pi(m)$. The table $\eta = \eta_i \{ \lambda \bar{x}_1 \theta / z_1, \dots, \lambda \bar{x}_k \theta / z_k \}$ belongs to $\pi(i+1)$ and positions $\pi(m-1), \pi(n-1)$ both η -extend $\pi(i+1)$ because $\pi(m), \pi(n)$ are children of $\pi(i)$. No look-up table $\eta_l \in \pi(l), l < i+1$, has these entries $\eta(z_{i'}) = \lambda \bar{x}_{i'} \theta$. Consider the first position $\pi(m_1)$ after $\pi(m)$ that is at a variable $y_1 \in \pi(m_1)$. Clearly, y_1 is a descendent of $\lambda \bar{x}_j$ in t_0 . If y_1 is bound by $\lambda \bar{x}_j$ then $\pi(m_1)$ is a child of $\pi(m)$ and the result is proved. Otherwise, there are two cases $\pi(m_1)$ is a child of $\pi(l), l < i$, and, so, by move A3 its look-up table η' cannot extend η . Play may jump anywhere in t_0 by move C4. If there is not a play $\pi(m_1, n_1)$ on the simple tile headed with y_1 then for all later positions $\pi(m_2), m_2 > m_1, \pi(m_2)$ cannot η -extend $\pi(i+1)$ which is a contradiction. Therefore, play must continue with a position $\pi(n_1)$ that is a child of $\pi(m_1)$. Secondly, y_1 is bound by a $\lambda \bar{y}$ that is below $\lambda \bar{x}_j$. But then y_1 is bound to a leaf of a constant tile that occurs between $\lambda \bar{x}_j$ and y_1 and so move C3 must apply and play proceeds to a child of y_1 . This argument is now repeated for the next position after $\pi(n_1)$ that is at a variable $y_2 \in \pi(m_2)$: y_2 must be a descendent of $\lambda \bar{x}_j$. The argument proceeds as above, except there is the new case that $\pi(m_2)$ is a child of $\pi(n_1)$. However, by move A3, $\pi(m_2)$ cannot η -extend $\pi(i+1)$. Therefore, eventually play must reach a child of $\pi(m)$.

2. This follows from the proof of 1.

3. Assume $\pi(m+m')$ is the first position that is a child of $\pi(m), \pi(m, m+m')$ is ri and $\pi(i, n)$ is a j -play. Consequently, $\pi(m) = \lambda \bar{x}_j q \theta \eta$ and $\pi(n) = \lambda \bar{x}_j q' \theta \eta'$ and both η -extend $\pi(i+1)$ because they are both children of $\pi(i)$. Consider positions $\pi(m+1), \pi(n+1)$. If $m' = 1$ the result follows. Otherwise, by move A3, $\pi(m+1) = y_1 q[l, r] \theta_1 \eta_1$ and $\pi(n+1) = y_1 q[l, r'] \theta'_1 \eta_1$. These positions have the same look-up table η_1 , the same left-terms in their state, and θ_1, θ'_1 only differ in their values for the variables that are bound by $\lambda \bar{x}_j$. Therefore, play must continue from both positions in the same way until a child of $\pi(m)$ and $\pi(n)$ is reached.

4. Assume $\pi = \pi^{i\alpha}$. The argument is similar to 3 except that the same \forall choices in the non ri play $\pi(m, m+m')$ need to be made. Therefore, there must be a $\pi' = \pi^{i\beta}$ such that $\pi'(n) = \pi(n)$ and the same \forall choices are made in $\pi'(n, n+m')$. \square

Tiles can be composed to form composite tiles. A (possibly composite) tile is a partial tree which can be extended at any atomic leaf. If $t(\lambda \bar{x})$ is a tile with leaf $\lambda \bar{x}$ and t' is a constant or simple tile, then $t(\lambda \bar{x}. t')$ is the composite tile that is the result of placing t' directly beneath $\lambda \bar{x}$ in t . Throughout, we assume that tiles are well-named. We now define a salient kind of simple or composite tile.

Definition 3 A tile is *basic* if it contains one occurrence of a free variable and does not contain any constants. A tile is an (extended) constant tile if it contains one occurrence of a constant and no occurrences of a free variable.

The single occurrence of a free variable in a basic tile must be its head variable and the single occurrence of a constant in a constant tile must be its head occurrence.

A contiguous region of t_0 can be identified with a basic or constant tile: a node y with its children and some, or all, of their children and so on (as long as children of a variable $y' : B \neq \mathbf{0}$ are included) is a larger region that is a basic tile if y is its only free variable and it contains no constants. We write $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ if t is a basic tile with atomic leaves $\lambda\bar{x}_1, \dots, \lambda\bar{x}_k$. A basic or constant tile in t_0 induces subplays of $\mathbf{G}(t_0, P)$ that are compositions of plays of its component tiles.

Definition 4 A subplay π is a *play* on $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ in t_0 if $t \in \pi(1)$, for some i , $\lambda\bar{x}_i \in \pi(|\pi|)$, there is the branch $t = y_1 \downarrow_{j_1} \lambda\bar{x}_{j_1}^1 \downarrow_1 y_2 \dots y_n \downarrow_{j_n} \lambda\bar{x}_{j_n}^n = \lambda\bar{x}_i$ and π can be split into plays $\pi(i_m, j_m)$ on $y_m(\lambda\bar{x}_1^m, \dots, \lambda\bar{x}_{k_m}^m)$ where $i_1 = 1$, $i_{m+1} = j_m + 1$ and $j_n = |\pi|$. It is a *j-play* if $\lambda\bar{x}_j \in \pi(|\pi|)$.

The definition for constant tiles is similar. Properties of plays of simple tiles lift to plays of basic tiles.

Corollary 1 Assume $\pi \in \mathbf{G}(t_0, P)$, $\pi(i, m')$ and $\pi(i, n')$, $n' > m'$, are plays on $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ and $\lambda\bar{x}_j \in \pi(m')$, $t = y_1 \downarrow_{j_1} \lambda\bar{x}_{j_1}^1 \downarrow_1 y_2 \dots y_n \downarrow_{j_n} \lambda\bar{x}_{j_n}^n = \lambda\bar{x}_j$ and $\pi(i, m')$ is split into plays $\pi(i_m, j_m)$ on $y_m(\lambda\bar{x}_1^m, \dots, \lambda\bar{x}_{k_m}^m)$ where $i_1 = i$, $i_{m+1} = j_m + 1$ and $j_n = m'$.

1. $\pi(m')$ extends $\pi(i)$.
2. There is a position $\pi(m_1)$, $m' < m_1 < n'$, that is a child of $\pi(j_i)$ for some i .
3. If $\pi(m_1)$ is the first position that is a child of $\pi(j_i)$ for some i , $t' \in \pi(m_1)$, y' occurs on the branch between $\lambda\bar{x}_j$ and t' , t' is an i' -descendent of y' and $y' \downarrow_{i'} \lambda\bar{x}_{i'}$, then there is an i' -play $\pi(m_2, n_2)$ on $y'(\lambda\bar{x}_1, \dots, \lambda\bar{x}_{k'})$ such that $m' < m_2$ and $n_2 < m_1$.
4. If $\pi(m' + m_1)$ is the first position that is a child of $\pi(j_i)$, for some i , $\pi(m', m' + m_1)$ is *ri* and $\pi(i, n')$ is a j -play then $\pi(n' + m_1)$ is the first position that is a child of any position $\pi(n'')$ such that $\lambda\bar{x}_{j_i}^i \in \pi(n'')$, $\pi(n', n' + m_1)$ is *ri* and for all $n_1 \leq m_1$, $t \in \pi(m' + n_1)$ iff $t \in \pi(n' + n_1)$.
5. If $\pi(m' + m_1)$ is the first position that is a child of $\pi(j_i)$, for some i , $\pi(m', m' + m_1)$ is not *ri* and $\pi(i, n')$ is a j -play then there is a $\pi' \in \mathbf{G}(t_0, P)$ with $\pi'(n') = \pi(n')$ and $\pi'(n' + m_1)$ is the first position that is a child of any position $\pi'(n'')$ such that $\lambda\bar{x}_{j_i}^i \in \pi'(n'')$, $\pi'(n', n' + m_1)$ is not *ri* and for all $n_1 \leq m_1$, $t \in \pi(m' + n_1)$ iff $t \in \pi'(n' + n_1)$.

Definition 5 Assume π is a j -play (play) on t . It is a *shortest j-play* (play) if no proper prefix of π is a j -play (play) and it is an *ri j-play* (play) if π is also *ri*. It is a *canonical j-play* (play) if each $t' \in \pi(i)$ is a node of t . Two plays π and π' on t are *independent* if one is not contained in the other: that is, $\pi \neq \pi_1 \pi' \pi_2$ and $\pi' \neq \pi_1 \pi \pi_2$.

Definition 6 Two basic tiles t and t' in t_0 are *equivalent*, written $t \equiv t'$ if they are the same basic tiles with the same free variable y (bound to the same $\lambda\bar{y}$). A tile t' is a *j-descendent* of $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ in t_0 if there is a branch in t_0 from $\lambda\bar{x}_j$ to t' .

Definition 7 The tile $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is *j-end* in t_0 , if every free variable below $\lambda\bar{x}_j$ in t_0 is bound above t . It is an *end tile* if it is *j-end* for all j . The tile

property: if $s \mathbf{T} t$ and $s \models P$ then $t \models P$ which is proved using the game-theoretic characterisation. The first transformation is easy. Let t' be a subtree of t_0 whose root node is a variable y or a constant $f : B \neq \mathbf{0}$. $\mathbf{G}(t_0, P)$ avoids t' if $t' \not\subseteq \pi(i)$ for all positions and plays $\pi \in \mathbf{G}(t_0, P)$. Let $t_0[a/t']$ be the result of replacing t' in t_0 with the constant $a : \mathbf{0}$.

T1 If $\mathbf{G}(t_0, P)$ avoids t' then transform t_0 to $t_0[a/t']$

Assume that $t_0 \models P$. The other transformations involve basic tiles. If a j -end tile is j -directed then it is redundant and can be removed from t_0 .

T2 Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is a j -directed, j -end tile in t_0 and t' is the subtree of t_0 rooted at t . If t_j is the subtree directly beneath $\lambda\bar{x}_j$ then transform t_0 to $t_0[t_j/t']$.

The next transformation separates plays.

Definition 1 Assume $t = t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is a basic sri tile in t_0 that is not an end tile. Tile t is a *separator* if there are two independent shortest plays that end at different leaves of t .

T3 If $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is a separator in t_0 and t' is the subtree of t_0 rooted at t then transform t_0 to $t_0[t(\lambda\bar{x}_1.t', \dots, \lambda\bar{x}_k.t'/t']$.

Here, we have added an extra copy of t directly below each $\lambda\bar{x}_j$: we assume that the head variable of this copy of t is bound by the $\lambda\bar{y}$ that binds the head variable of the original t and we assume that all variables below $\lambda\bar{x}_j$ that are bound in t in t_0 are now bound in the copy of t : this means that the original t becomes an end tile.

The next transformation, in effect, allows tiles to be “lowered” in t_0 .

Definition 2 Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is j -ri and not j -end in t_0 and directly below $\lambda\bar{x}_j$ is the constant or basic tile $u(\lambda\bar{z}_1, \dots, \lambda\bar{z}_m)$ whose head variable, if there is one, is not bound in t . Tile t is *j -permutable with u* in t_0 if whenever $\pi(i, m)$ is a shortest j -play on t then either (1) there are no other j -plays $\pi(i, m')$ on t or (2) $\pi(m+1, n)$ is a shortest play on u and it is ri and u is an end tile.

T4 Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is j -permutable with $u(\lambda\bar{z}_1, \dots, \lambda\bar{z}_m)$ in t_0 and t' is the subtree rooted at u in t_0 . If t_i and t'_i are the subtrees of t_0 directly below $\lambda\bar{x}_i$ and $\lambda\bar{z}_i$ then transform t_0 to $t_0[u(\lambda\bar{z}_1.w_1, \dots, \lambda\bar{z}_m.w_m)/t']$ where $w_i = t(\lambda\bar{x}_1.t_1, \dots, \lambda\bar{x}_{j-1}.t_{j-1}, \lambda\bar{x}_j.t'_i, \lambda\bar{x}_{j+1}.t_{j+1}, \dots, \lambda\bar{x}_k.t_k)$.

The tile t is copied below u : however, in the copy of t below $\lambda\bar{z}_i$ of u t'_i (and not t_i) occurs below $\lambda\bar{x}_j$ of t . We assume that the free variables of t_i and t'_i retain their binders in the transformed term and that the copies of t below u bind the free x_j .

Consider the case when the j -ri tile t is not j -permutable with the constant tile $f(\lambda\bar{z}_1, \dots, \lambda\bar{z}_m)$. There is a shortest j -play $\pi(i, m)$ on t and another j -play $\pi(i, n)$ on t .

$$\begin{array}{cccccc} \pi(i) & \dots & \pi(m) & \pi(m+1) & \dots & \pi(n) & \pi(n+1) \\ t & & \lambda\bar{x}_j & f & & \lambda\bar{x}_j & f \end{array}$$

Consequently, permuting t with f is not permitted: the transformed term would exclude the extra play on f .

In an application of **T4**, if t is a top j -ri tile and every shortest j -play is canonical then after its application t will be j -end and j -directed, and therefore can be removed by **T2**. In this case, the tile t does percolate down the term tree t_0 .

We now show that the four transformations preserve interpolation.

Proposition 1 *For $1 \leq i \leq 4$, if $s \mathbf{T}i t$ and $s \models P$ then $t \models P$.*

Proof. This is clear when $i = 1$. Consider $i = 2$. Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is a j -directed, j -end tile in t_0 , t' is the subtree of t_0 rooted at t and t_j is the subtree directly beneath $\lambda\bar{x}_j$, $t'_0 = t_0[t_j/t']$ and $t_0 \models P$. We shall convert $\pi = \pi^{i\alpha} \in \mathbf{G}(t_0, P)$ into the play $\sigma = \sigma^{i\alpha} \in \mathbf{G}(t'_0, P)$ that \forall loses. The play π is split uniquely into regions.

$$\begin{array}{cccccccc} \pi(1) & \dots & \pi(i_1) & \dots & \pi(m_1) & \dots & \pi(i_2) & \dots & \pi(m_2) & \dots & \pi(i_n) & \dots & \pi(m_n) & \dots & \pi(|\pi|) \\ t & & \lambda\bar{x}_j & & t & & \lambda\bar{x}_j & & t & & \lambda\bar{x}_j & & t & & \lambda\bar{x}_j \end{array}$$

The play σ is just the outer subplays (modulo minor changes to the look-up tables) because each $\pi(m_k)$ extends $\pi(i_k)$.

$$\pi(1) \dots \pi(i_1 - 1)\pi(m_1 + 1) \dots \pi(i_n - 1)\pi(m_n + 1) \dots \pi(|\pi|)$$

We show, using a similar argument as is used in Proposition 1.1 of Section 4, that if s is a node in t or is a descendent of a leaf $\lambda\bar{x}_m$, $m \neq j$, of t then s cannot occur in any outer subplay of π . If s were to appear in an outer subplay then move C4 must have applied: there is then a variable y and a position in an outer subplay $y \in \pi(n)$ and $\theta \in \pi(n)$ and $\theta(y) = l\eta$ and there is a free variable z in l such that $\eta(z) = s\theta'$. However, this is impossible. Consider $\theta_1 \in \pi(i_1)$: clearly, there is no free variable in the subtree rooted at t with this property. When play reaches $\pi(m_1)$ because t is a j -end tile and because $\pi(m_1)$ extends $\pi(i_1)$ there cannot be a free variable in the subtree t_j with this property either. This argument is now repeated for subsequent positions $\pi(i_k)$ and $\pi(m_k)$.

Let $i = 3$. Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is a separator in t_0 , t' is the subtree of t_0 rooted at t and $t'_0 = t_0[t(\lambda\bar{x}_1.t', \dots, \lambda\bar{x}_k.t')/t']$. We shall convert $\pi = \pi^{i\alpha} \in \mathbf{G}(t_0, P)$ into $\sigma = \sigma^{i\alpha} \in \mathbf{G}(t'_0, P)$ that \forall loses. Consider any shortest play on t in $\pi^{i\alpha}$, $\pi(m, k)$ and assume it is a j -play. By definition this play is ri. Therefore, this interval is transformed into the following interval for t'_0 .

$$\begin{array}{cccccc} \pi(m) & \dots & \pi(k) & \pi(m) & \dots & \pi(k) \\ t & & \lambda\bar{x}_j & t & & \lambda\bar{x}_j \end{array}$$

where the second t is the copy of t directly beneath $\lambda\bar{x}_j$ in t'_0 .

Finally, $\mathbf{i} = 4$. Assume $t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ is j -permutable with $u(\lambda\bar{z}_1, \dots, \lambda\bar{z}_m)$ in t_0 , t' is the subtree rooted at u in t_0 , t_i and t'_i are the subtrees of t_0 directly below $\lambda\bar{x}_i$ and $\lambda\bar{z}_i$ and $t'_0 = t_0[u(\lambda\bar{z}_1.w_1, \dots, \lambda\bar{z}_m.w_m)/t']$ where w_i is as in **T4**. We shall convert $\pi = \pi^{i\alpha} \in \mathbf{G}(t_0, P)$ into $\sigma = \sigma^{i\alpha} \in \mathbf{G}(t'_0, P)$ that \forall loses. The play π can be divided into non-overlapping regions $\pi(i_k, m_k)$.

$$\begin{array}{cccccccc} \pi(1) & \dots & \pi(i_1) & \dots & \pi(m_1) & \pi(m_1 + 1) & \dots & \pi(i_n) & \dots & \pi(m_n) & \pi(m_n + 1) & \dots & \pi(|\pi|) \\ & & t & & \lambda\bar{x}_j & u & & t & & \lambda\bar{x}_j & u & & \end{array}$$

where $\pi(i_k, m_k)$ are shortest j -plays: such a region may also contain other shortest j -plays on t :

$$\begin{array}{cccccccc} \dots & \pi(i_k) & \dots & \pi(i') & \dots & \pi(m') & \dots & \pi(m_k) & \dots \\ & t & & t & & \lambda\bar{x}_j & & \lambda\bar{x}_j & \end{array}$$

If $u = f(\lambda\bar{z}_1, \dots, \lambda\bar{z}_n)$ is a constant tile then (1) of Definition 2 applies: so each $\pi(i_k, m_k)$ only contains a single occurrence of $\lambda\bar{x}_j$ because the play is ri. Moreover, there are no further j -plays $\pi(i_k, m')$ on t . Therefore, σ includes the following change to π for each interval $\pi(i_k, m_k)$ where we ignore the minor changes to look-up tables

$$\begin{array}{cccccccc} \pi(i_k) & \dots & \pi(m_k) & \pi(m_k + 1) & \pi(m_k + 2) & \pi(i_k) & \dots & \pi(m_k) & \pi(m_k + 3) & \dots \\ t & & \lambda\bar{x}_j & f & \lambda\bar{z}_{k_i} & t & & \lambda\bar{x}_j & t'_{k_i} & \end{array}$$

where $t'_{k_i} \in \pi(i_k)$ is the copy of t directly beneath $\lambda\bar{z}_{k_i}$ in t'_0 .

Next, let u be a basic tile. To obtain σ we iteratively do additions and deletions to π starting with $\pi(i_1, m_1)$ and then recursively transforming inner j -plays on t within this region. Let π be the result of the changes to the initial π for the intervals $\pi(i_j, m_j)$, $j < k$. Consider the interval $\pi(i_k, m_k)$. Consider case (1) of Definition 2. Let $\pi(m_k + 1, n_k^i)$ be all plays on $u \in \pi(m_k + 1)$. If there are no plays then π is initially unchanged. Otherwise, π has the following structure:

$$\begin{array}{cccccccc} \dots & \pi(i_k) & \dots & \pi(m_k) & \pi(m_k + 1) & \dots & \pi(n_k^i) & \pi(n_k^i + 1) & \dots \\ & t & & \lambda\bar{x}_j & u & & \lambda\bar{z}_{k_i} & t'_{k_i} & \end{array}$$

To obtain the new π , we do the following addition for each i

$$\begin{array}{cccccccc} \pi(i_k) & \dots & \pi(m_k) & \pi(m_k + 1) & \dots & \pi(n_k^1) & \dots & \pi(n_k^i) & \pi(i_k) & \dots & \pi(m_k) & \pi(n_k^i + 1) & \dots \\ t & & \lambda\bar{x}_j & u & & \lambda\bar{z}_{k_1} & & \lambda\bar{z}_{k_i} & t & & \lambda\bar{x}_j & t'_{k_i} & \end{array}$$

where t immediately after $\pi(n_k^i)$ is its copy in t'_0 directly beneath $\lambda\bar{z}_{k_i}$.

Finally, we consider the case that u is an end tile. Let $\pi(m_k + 1, m_k + n)$ be the unique play on u with $\lambda\bar{z}_i \in \pi(m_k + n)$. Consider all j -plays $\pi(i_k, m_k^i)$ on $t \in \pi(i_k)$ where $m_k^1 = m_k$:

$$\begin{array}{cccccccc} \pi(i_k) & \dots & \pi(m_k^1) & \dots & \pi(m_k^i) & \pi(m_k^i + 1) & \dots & \pi(m_k^i + n) & \pi(m_k^i + n + 1) & \dots \\ t & & \lambda\bar{x}_j & & \lambda\bar{x}_j & u & & \lambda\bar{z}_i & t'_i & \end{array}$$

There must be the same play on u at each $\pi(m_k^i + 1)$ because the value of the head variable of u is always the same and u is an end tile. So initially we do the following addition

$$\begin{array}{cccccccc} \pi(i_k) & \dots & \pi(m_k^1) & \pi(m_k^1 + 1) & \dots & \pi(m_k^1 + n) & \pi(i_k) & \dots & \pi(m_k^1) & \pi(m_k^1 + n + 1) \\ t & & \lambda\bar{x}_j & u & & \lambda\bar{z}_i & t & & \lambda\bar{x}_j & t'_i \end{array}$$

where the second $t \in \pi(i_k)$ is the copy of t directly below $\lambda\bar{z}_i$ in t'_0 , and for subsequent $i > 1$ we delete the ri region $\pi(m_k^i + 1, m_k^i + n)$. To complete the argument, we recursively apply this technique to shortest j -plays on t within $\pi(i_k, m_k^1)$: note that j -plays on t below $\lambda\bar{z}_i$ within $\pi(i_k, m_k^1)$ will include additional ri plays on t and on u . \square

6 DECIDABLE INSTANCES

We now briefly sketch how the the game-theoretic characterisation of matching provides uniform decidability proofs for two instances of interpolation that are known to be decidable, the 4th-order problem and the atoms case where in each equation $x(v_1, \dots, v_n) = u$ the term u is a constant $a : \mathbf{0}$ [5, 6]. In both cases the proof establishes the *small model property* (if $t_0 \models P$ then there is a small $t \models P$) via the transformations of the previous section. In neither case do we need to appeal to observational equivalence.

Figure 2 presents the algorithm for both cases. The procedure is initiated by marking all leaves of t_0 and recursively proceeds towards its root. At each stage, a lowest marked node u is examined for transformations: the algorithm has, therefore, already ascended all branches below u .

Assume $t_0 \models P$

1. mark all leaves $u : \mathbf{0}$ of t_0
2. choose a marked node u such that no descendent of u is marked
3. if $t_0 \mathbf{T1}t'$ at u then $t_0 = t'$ and unmark all nodes and return to 1
4. identify basic or constant tile $t = t(\lambda\bar{x}_1, \dots, \lambda\bar{x}_k)$ rooted at u
5. if $t_0 \mathbf{Ti}t'$ at t for $i \in \{2, 3\}$ then $t_0 = t'$ and unmark all nodes and return to 1
6. identify successor basic or constant tiles t_i below $\lambda\bar{x}_i$
7. if $t_0 \mathbf{T4}t'$ at t and a successor then $t_0 = t'$ and unmark all nodes and return to 1.
8. if $u' \downarrow_{i_1} \lambda\bar{y} \downarrow_1 u$ then unmark u and mark u' and return to 2
9. finish

Fig. 2. The algorithm

Clearly, the procedure must terminate with $t_0 \models P$ and where no transformation applies anywhere in t_0 . Assume t_0 is such a term.

Proposition 1 *If t' is a subterm of t_0 such that t' only contains sri tiles, leaves $y : \mathbf{0}$ and $a : \mathbf{0}$ then t' consists of sri end tiles and leaves $a : \mathbf{0}$.*

Proof. By a simple induction. A leaf u may be a constant or a variable. Consider u' such that $u' \downarrow_{i_1} \lambda \bar{y} \downarrow_1 u$. By repeating the argument for other directions i_j from u' , the tile rooted at u' will be an end tile. Consider the first time that a tile isn't an end tile. Either **T3** or **T4** must apply, which is a contradiction.

Hence for the atoms case, as all tiles are sri, every end tile is also a top tile. There can be at most m separators where m is the number of equations. Finally, Proposition 2 of Section 4 provides a simple upper bound both on the size of an end tile in t_0 and the number of embedded end tiles. The details are straightforward.

Next we consider the 4th-order case. The term t_0 consists of top tiles, leaves and constant tiles. Shortest plays on a top tile are canonical. The number of top tiles that are not sri is bounded (by the sum of the sizes of the sets R_i of section 2). Again there can be at most m separators. Now, the crucial property is that given a sequence of sri top tiles $t_i(\lambda \bar{x}_1^i, \dots, \lambda \bar{x}_{k_i}^i)$ such that for each i , t_{i+1} is directly below $\lambda \bar{x}_{j_i}^i$, then most of the tiles t_i are n_i -end and n_i -directed for some n_i which follows easily from Proposition 1 of Section 4. (If a shortest ri j -play on t_i , $\pi(k, m)$, is such that there is a child $\pi(m')$ of $\pi(m)$, so $y : \mathbf{0} \in \pi(m')$, then every j -play $\pi(k, n)$ of t_i is such that there is a child $\pi(n')$ of $\pi(n)$ and $y \in \pi(n')$ or $\pi(k, m')$ is not ri and for some n' , $\pi(k, n')$ is also not ri.)

References

1. Comon, H. and Jurski, Y. Higher-order matching and tree automata. *Lecture Notes in Computer Science*, **1414**, 157-176, (1997).
2. Dowek, G. Higher-order unification and matching. In A. Robinson and A. Voronkov ed. *Handbook of Automated Reasoning*, Vol 2, 1009-1062, North-Holland, 2001.
3. Huet, G. *Résolution d'équations dans les langages d'ordre 1, 2, ... ω* . Thèse de doctorat d'état, Université Paris VII, (1976).
4. Loader, R. Higher-order β -matching is undecidable, *Logic Journal of the IGPL*, **11(1)**, 51-68, (2003).
5. Padovani, V. Decidability of all minimal models. *Lecture Notes in Computer Science*, **1158**, 201-215, (1996).
6. Padovani, V. Decidability of fourth-order matching. *Mathematical Structures in Computer Science*, **10(3)**, 361-372, (2001).
7. Schubert, A. Linear interpolation for the higher-order matching problem. *Lecture Notes in Computer Science*, **1214**, 441-452.
8. Schmidt-Schauß, M. Decidability of arity-bounded higher-order matching. *Lecture Notes in Artificial Intelligence*, **2741**, 488-502, (2003).
9. Statman, R. The typed λ -calculus is not elementary recursive. *Theoretical Computer Science*, **9**, 73-81, (1979).
10. Stirling, C. *Modal and Temporal Properties of Processes*, Texts in Computer Science, Springer, 2001.
11. Wierzbicki, T. Complexity of higher-order matching. *Lecture Notes in Computer Science*, **1632**, 82-96, (1999).