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 0 and the definitions of α-equivalence, β and η-reduction. A type is 0 or A → B where A and B are types. A type A always has the form (A_1 → (⋯A_n → 0)⋯) which is usually written A_1 → ⋯ → A_n → 0. We also assume a standard definition of order: the order of 0 is 1 and the order of A_1 → ⋯ → A_n → 0 is k + 1 where k is the maximum of the orders of the A_i.

Terms are built from a countable set of variables x, y, ⋯ and constants, a, f, ⋯; 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 ∈ T (f : A ∈ T), if t : B ∈ T and x : A ∈ T, then λx.t : A → B ∈ T, and if t : A → B ∈ T and u : A ∈ T then tu : B ∈ 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, ⋯, x_n in v. A solution of a matching problem is a sequence of terms t_1, ⋯, t_n such that v[t_1/x_1, ⋯, t_n/x_n] =_η 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-equivalence [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 → ⋯ → A_n → 0 is rewritten (A_1, ⋯, A_n) → 0 and we assume that all terms in normal form are in η-long form. That is, if t : 0 then it either has the form u : 0 where u is a constant or a variable, or has the form u(t_1, ⋯, t_k) where u : (B_1, ⋯, B_k) → 0 is either a constant or a variable and each t_i : B_i.
is in \( \eta \)-long form. And if \( t : (A_1, \ldots, A_n) \to 0 \) then \( t \) has the form \( \lambda y_1 \ldots y_n . t_0 \) where \( t_0 : 0 \) is a term in \( \eta \)-long form. A term is well-named if each occurrence of a variable \( y \) within a \( \lambda \) abstraction is unique.

An interpolation equation has the form \( x(v_1, \ldots, v_n) = u \) where each \( v_i \) is a closed term in normal form and \( u : 0 \) is also in normal form. The type of the equation is the type of the free variable \( x \), which has the form \( (A_1, \ldots, A_n) \to 0 \) where \( v_i : A_i \). An interpolation problem \( P \) is a finite family of interpolation equations \( x(v_1^i, \ldots, v_n^i) = u_i, i : 1 \leq i \leq m \), 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 \( t(v_1^i, \ldots, v_n^i) = u_i \) for each \( i \). We write \( t \vdash P \) if the closed term \( t \) solves the problem \( P \).

An interpolation problem reduces to matching: there is the equivalent problem \( f(x(v_1^1, \ldots, v_n^1), \ldots, x(v_1^m, \ldots, v_n^m)) = f(u_1, \ldots, u_m) \), when \( f : 0^m \to 0 \). 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 \( x(v_1^i, \ldots, v_n^i) \neq u_i \). 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 \( t \vdash P \). 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 \( t \vdash P \) then \( t' \vdash P \) for some small term \( t' \).

\section{Preliminaries}

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

\begin{definition}

The \textbf{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 \( w = a : 0 \), then \( \text{Cl}(w, X, C) = \{a\} \)

\end{definition}
2. if \( w = f(w_1, \ldots, w_n) \), then \( \text{Cl}(w, X, C) = \{ w \} \cup \bigcup \text{Cl}(w_i, X, C) \)

3. if \( w = \lambda x_j \ldots x_j u \), then \( \text{Cl}(w, X, C) = \text{Cl}(u(c_j / x_j, \ldots, c_j / x_j), X, C) \)

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

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 \( w = x(w_1, \ldots, w_n) \), then \( \text{Sub}(w, C) = \text{Sub}'(w, C) = \{ w \} \cup \bigcup \text{Sub}(w_i, C) \)
3. if \( w = f(w_1, \ldots, w_n) \), then \( \text{Sub}(w, C) = \text{Sub}'(w, C) = \{ w \} \cup \bigcup \text{Sub}(w_i, C) \)
4. if \( w = \lambda y_1 \ldots y_n v \), then \( \text{Sub}(w, C) = \{ w \} \cup \text{Sub}(v, C) \)
5. if \( w = \lambda y_1 \ldots y_n v \), then \( \text{Sub}'(w, C) = \bigcup \text{Sub}(v(c_i / y_1, \ldots, c_i / y_n), C) \) where each \( c_i \in C \) 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 \( x(v_1^i, \ldots, v_n^i) = u_i \), \( 1 \leq i \leq m \), 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 \( R_i = \text{Cl}(u_i, X_i, C_i) \)
3. for each \( i \), the left subterms are \( L_i = \bigcup \text{Sub}(v_j^i, C_i) \cup C_i \)

### 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. Conon 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 the
binding play may jump here and there in t. Consequently, our games lack the
simple control structure of Conch 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. \( t_m = \lambda y_1 \ldots y_j \) and \( t_m \downarrow t' \) and \( q_m = q[l_1, \ldots, l_j], r \). So, \( t_{m+1} = t' \) and \( \theta_{m+1} = \theta_m \{ \gamma_{m/y_1}, \ldots, \gamma_{m/y_j} \} \) and \( q_{m+1} \) and \( \eta_{m+1} \) are by cases on \( t_{m+1} \):
1. \( a : 0 \). So, \( \eta_{m+1} = \eta_m \). If \( r = a \) then \( q_{m+1} = q[\exists] \) else \( q_{m+1} = q[\forall] \).
2. \( f : (B_1, \ldots, B_k) \rightarrow 0 \). So, \( \eta_{m+1} = \eta_m \). If \( r = f(s_1, \ldots, s_k) \) then \( q_{m+1} = q_m \)
else \( q_{m+1} = q[\forall] \).
3. \( y : B \). If \( \theta_{m+1}(y) = 0 \), then \( q_{m+1} = q[l, r] \) and \( \eta_{m+1} = \eta_y \).

B. \( t_m = f : (B_1, \ldots, B_k) \rightarrow 0 \) and \( q_m = q[l_1, \ldots, l_j], f(s_1, \ldots, s_k) \). So, \( \theta_{m+1} = \theta_m \)
and \( \eta_{m+1} = \eta_m \) and \( q_{m+1} = q_m \) and \( \eta_{m+1} \) are by cases on \( t \):
1. \( a : 0 \) or \( \lambda x. \alpha \). 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. \( c : (B_1, \ldots, B_k) \rightarrow 0 \). So, \( \theta_{m+1} = \theta_m \). If \( r \neq c(s_1, \ldots, s_k) \) then \( t_m = t_m \) and \( \eta_{m+1} = \eta_m \). If \( r = c(s_1, \ldots, s_k) \) then \( \forall \) chooses a direction \( \delta' : 1 \leq \delta' \leq k \) and \( t_m \downarrow t' \). So, \( t_{m+1} = t' \).
3. \( f(w_1, \ldots, w_k) \) or \( \lambda x. f(w_1, \ldots, w_k) \). So, \( t_{m+1} = t_m \) and \( \theta_{m+1} = \theta_m \). If \( r \neq f(s_1, \ldots, s_k) \), then \( q_{m+1} = q[\forall] \). If \( r = f(s_1, \ldots, s_k) \) and \( \forall \) chooses a direction \( \delta' : 1 \leq \delta' \leq k \) and \( t_m \downarrow t' \). So, \( t_{m+1} = t' \).
4. \( z(l_1, \ldots, l_k) \) or \( \lambda x. z'(l_1, \ldots, l_k) \). If \( \eta_{m+1}(z') = t \theta_x \), then \( \theta_{m+1} = \theta_t \) and \( t_{m+1} = t' \) and \( q_{m+1} = q[l_1, \ldots, l_k, r] \).

Fig. 1. Game moves

We assume that a potential solution term \( t \) for \( P \) has the right \( \text{type} \), 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 \( t = y : 0 \) or \( a : 0 \) then \( \text{tree}(t) \) is the single node labelled with \( t \).
In the case of \( u(v_1, \ldots, v_k) \) when \( u \) is a variable or a constant, we assume that
a dummy \( \lambda \) with the empty sequence of variables is placed before any subterm
\( v_i : 0 \) in the tree representation. With this understanding, if \( t = u(v_1, \ldots, v_n) \),
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 \( u \downarrow t' \) to represent that tree \( t' \) is the \( i \)th
successor of the node \( u \). If \( t \) is \( \lambda \mathbf{f} n \), where \( \mathbf{f} \) is a possibly empty sequence of variables \( y_1 \ldots y_n \), then \( \text{tree}(t) \) consists of the root node labelled \( \lambda \mathbf{f} \) and a single successor node \( \text{tree}(v) \). In this case we assume \( \lambda \mathbf{f} \downarrow \lambda \text{tree}(v) \). We also assume that each node labelled with an occurrence of a variable \( y_j \) has a backward arrow \( \uparrow^j \) to the \( \lambda \mathbf{f} \) that binds it; the index \( j \) tells us which element is \( y_j \) in \( \mathbf{f} \).

The tree representation of \( \lambda_0 \mathbf{1} \mathbf{y}_2 \mathbf{f}(\mathbf{f}(\mathbf{y}_2, \mathbf{y}_1(\mathbf{y}_2))) \) is tantamount to the syntax tree of \( \lambda_0 \mathbf{1} \mathbf{y}_2 \mathbf{f}(\lambda. \mathbf{f}(\lambda \mathbf{y}_2, \lambda_0 \mathbf{y}_1(\lambda \mathbf{y}_2))) \). In the following we use \( t \) to be the \( \lambda \) term \( t \), or its \( \lambda \) tree or the label \( (a \text{ constant, variable or } \lambda \mathbf{f}) \) at its root node.

The tree-checking game \( G(t, P) \) is played by one participant, player \( V \), 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 \( L_1 \) and \( R_1 \). There are three kinds of states: argument, value and final states. Argument states have the form \( q[[t_1, \ldots, t_k], r] \) where \( i \in L_1 \) and \( k \) can be 0, and \( r \in R_1 \). Value states have the form \( q[[t, r] \) where \( t \in L_1 \) and \( r \in R_1 \). A final state is either \( q[V] \), the winning state for \( V \), or \( q[\emptyset] \), the losing state for \( V \).

The game appeals to a sequence of supplementary look-up tables \( \theta_j \) and \( \eta_j \), \( j \geq 1 \): \( \theta_j \) is a partial map from variables in \( t \) to elements \( \eta_i \) where \( i \in L_1 \) and \( k < j \), and \( \eta_j \) is a partial map from variables in \( L_1 \) to elements \( \theta_j \eta_k \) where \( \theta_j \) is a node of the tree \( t \) and \( k < j \). The initial elements \( \theta_1 \) and \( \eta_1 \) are both the empty table.

A play of \( G(t, P) \) is a sequence of positions \( t_q[\theta_1 \eta_1, \ldots, \theta_m \eta_m] \) where each \( t_q \) is a node of \( t \) and \( t_1 = \lambda \mathbf{f} \) 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: \( V \) chooses an equation \( x(e_1, \ldots, e_n) = e_0 \) from \( P \) and \( q_1 = q[[t_1, \ldots, t_k], r] \) and \( q_n = \eta[[t_1, \ldots, t_n], r] \). If the current position is \( t_m \eta_m \theta_{m+1} \eta_{m+1} \) and \( q_n \) is not a final state, then the next position is \( t_{m+1} \eta_{m+1} \theta_{m+1} \eta_{m+1} \) determined by a move of Figure 1.

Moves are divided into the three groups that depend on \( t_m \). Group A covers the case when \( t_m = \lambda \mathbf{f} \). Group B when \( t_m = f \) and group C when \( t_m = y \). We assume standard updating notation for \( \theta_{m+1} \) and \( \eta_{m+1} \): \( \beta{\alpha_1/y_1, \ldots, \alpha_n/y_m} \) is the function similar to \( \beta \) except that \( \beta(y_m) = \alpha_i \). Moreover, in the case of rules B1, C2 and C3 we assume that the constants \( c_j \) belong to the forbidden sets \( C_i \). The look-up tables are used in rules A3 and C4. If \( t_m = \lambda \mathbf{f} \) and \( t_m \downarrow t_{m+1} \), then \( \eta_{m+1} \) and \( \eta_{m+1} \) are determined by the entry for \( y \) in \( \theta_{m+1} \); if the entry is \( \eta_k \), then \( t \) is the left element of \( \eta_{m+1} \) and \( \eta_{m+1} = \eta_k \). In the case of C4, if \( t_m = y \) and \( q_m = q[[t, r] \) and \( l = t[x_1, \ldots, x_k] \) or \( \lambda \mathbf{x} \mathbf{z} \), then \( \theta_{m+1} \) and \( t_{m+1} \) are determined by the entry for \( z \) in the table \( \eta_{m+1} \); if the entry is \( \eta_k \), then \( t_{m+1} = t' \) and \( \theta_{m+1} = \theta_k \). It is this rule that allows the next move to be a jump around the term tree (to a node labelled with a \( \lambda \). 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 \( x(e) = u \) where \( e = \lambda z. z \) and \( u = f(\lambda x. x) \). Let \( X = \{x\} \) and \( C = \{\epsilon\} \) and let \( t \) be the term \( \lambda y.y(y(f(\lambda y_1.y_1))) \) and so \( \text{tree}(t) \) is

\[
(t_1)(y \downarrow t_2)(y \downarrow t_3)(y \downarrow t_4)(y \downarrow t_5)(y \downarrow t_6)(y \downarrow t_7)f \downarrow (t_1)(y \downarrow t_2)(y \downarrow t_3)(y\downarrow t_4)(y \downarrow t_5)(y \downarrow t_6)(y \downarrow t_7)(y \downarrow t_8)(y \downarrow t_9)
\]
There is just one play of $G(t, P)$, as follows.

$$t_1 q[(\lambda z. z), f(\lambda x. x)] \theta_1 q_1$$
$$t_2 q[\lambda z. z, f(\lambda x. x)] \theta_2 q_2 \quad \theta_2 = \theta_1 ((\lambda z. z) q_1 / y) \quad q_2 = q_1 \quad A3$$
$$t_3 q[( ), f(\lambda x. x)] \theta_3 q_3 \quad \theta_3 = \theta_2 \quad q_3 = q_2 \{ t_2 \theta_2 / z \} \quad C4$$
$$t_4 q[\lambda z. z, f(\lambda x. x)] \theta_4 q_4 \quad \theta_4 = \theta_3 \quad q_4 = q_3 \quad A3$$
$$t_5 q[( ), f(\lambda z. z)] \theta_5 q_5 \quad \theta_5 = \theta_4 \quad q_5 = q_4 \{ t_3 \theta_3 / z \} \quad C4$$
$$t_6 q[( ), f(\lambda z. z)] \theta_6 q_6 \quad \theta_6 = \theta_5 \quad q_6 = q_5 \quad A2$$
$$t_7 q[c], c \theta_7 q_7 \quad \theta_7 = \theta_6 \quad q_7 = q_6 \quad B1$$
$$t_8 q[c], c \theta_8 q_8 \quad \theta_8 = \theta_7 \{ c \eta / \eta_1 \} \quad q_8 = q_7 \quad A3$$
$$t_9 q[\exists] \theta_9 q_9 \quad \theta_9 = \theta_8 \quad q_9 = q_8 \quad C1$$

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

A partial play of $G(t, P)$ 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 $G(t, P)$ if she loses every play. The following result provides a characterisation of $t \models P$.

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

**Proof.** For any position $t q \theta_i q_i$ of a play of $G(t, P)$ 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 $t_i = \lambda y_1$ and $q = q[[l_1, \ldots, l_k], r]$ and $t_i = (t_i \theta_i) \{ l_1, \theta_1, \ldots, l_k, \theta_k \}$ then $t' = r$ ($t' \neq r$) and $t'$ normalises with $m \beta$-reductions

if $t_i = f$ and $q = q[[l_1, \ldots, l_k], r]$ and $t_i = t_i \theta_i$ then $t' = r$ ($t' \neq r$) and $t'$ normalises with $m \beta$-reductions

if $t_i = z$ and $q = q[[l, r]]$ and $t_i \downarrow t'_i$ and $t' = t_i \theta_i \{ l_1, \theta_1, \ldots, l_k, \theta_k \}$ then $t' = r$ ($t' \neq r$) and $t'$ normalises with $m \beta$-reductions.

The following are easy to show by case analysis.

1. if $t q \theta_i q_i$ $m$-holds then $q_i = q[\exists]$ or for any next position $t_{i+1} q_{i+1} \theta_{i+1} q_{i+1}$ it $m'$-holds, $m' \leq m$, or it $m'$-fails, $m' + m + 1$, and the right-term in $q_{i+1}$ is smaller than $q_i$

2. if $t q \theta_i q_i$ $m$-fails then $q_i = q[\forall]$ or there is a next position $t_{i+1} q_{i+1} \theta_{i+1} q_{i+1}$ and it $m'$-fails, $m' \leq m$, or it $m'$-fails, $m' \leq m$, or it $m'$-fails, $m' \leq m$, and the right-term in $q_{i+1}$ is smaller than $q_i$

For instance, assume $t q \theta_i q_i$ $m$-holds and $t_i = \lambda y_1 \ldots y_k$ and $t_i \downarrow \downarrow t'_i$ and $q_i = q[[l_1, \ldots, l_k], r]$. So, $\theta_{i+1} \equiv \theta_i \{ f, \theta_i / \eta_1 \}$ and $q_{i+1} = q[[l, r]]$ if $\theta_{i+1}(y) = \theta_i(y) = \eta_1$. So, $t_i \equiv \lambda y_1 \ldots y_k y(t_1, \ldots, t_m)$ and by assumption $(t_i \theta_i) \{ l_1, \theta_1, \ldots, l_k, \theta_k \} = r$. With a $\beta$-reduction we get $\theta_{i+1}(y)(t_1 \theta_1, \ldots, t_m \theta_m)$ which is $(t_{i+1}(y)(t_1 \theta_1, \ldots, t_m \theta_m)) \theta_{i+1}$ and so position $t_{i+1} q_{i+1} \theta_{i+1} q_{i+1}$ holds. Next, assume $t q \theta_i q_i$ $m$-holds, $t_i = f$, $q_i = q[[l_1, \ldots, l_k], f(s_1, \ldots, s_k)]$ and $t_i \downarrow \downarrow t'_i$. By assumption, $t'_1 \theta_1 \theta_i q_i$ and $t_{i+1} \theta_i q_{i+1}$ if $q_i = q[[l_1, \ldots, l_k], f(s_1, \ldots, s_k)]$. So, $t_{i+1} \theta_i q_{i+1}$ is any choice of next position. If $s_j = 0$ then $q_{i+1} = q[\exists, s_j], t_{i+1} \equiv \gamma_j$ and $\theta_{i+1} \equiv \theta_i$. Therefore, $t_{i+1} \theta_{i+1} = s_j$ and if $s_j$ is this next position and $m'$-holds, $m' \leq m$ or $m'$-holds and $s_j$ is smaller than $f(s_1, \ldots, s_k)$. Alternatively, $s_j = \lambda \forall s$. Therefore,
\[ t'_i = \lambda x \ell' \text{ and } t \theta_i (\eta, \eta) = s(\eta, \eta) \] where the \( c_i \)'s are new, \( m' \)-holds for \( m' \leq m \). And so \( t \theta_i (c_1, \ldots, c_n) = s(\eta, \eta) \) \((m' + 1)\)-holds, as required. Assume \( t \theta_{i_1} \eta_{i_2} \eta_{i_3} \eta_{i_4} \eta_{i_5} \eta_{i_6} \eta_{i_7} \) \( m \)-holds and \( t_i = y, q_i = q[l, r], l = \lambda z_1 \ldots z_k.w, w = z(t_1, \ldots, t_m), t_i \downarrow j t'_i \) and \( t_{i+1} \theta_{i+1} = \eta_{i+1}, (z) \). By assumption, \((\lambda z_1 \ldots z_k.w) \eta_j (t'_i \theta_i, \ldots, t'_i \theta'_i) = r \). With one \( \beta \)-reduction \( \eta_{i+1} \eta_j (z) = t_{i+1} \theta_{i+1} \) \((l_1 \eta_{i+1}, \ldots, l_m \eta_{i+1}) = r \), that is \( t_{i+1} \theta_{i+1} \eta_j (z) \) \((l_1 \eta_{i+1}, \ldots, l_m \eta_{i+1}) = r \) 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 \( t = P \) then for each initial position there is an \( m \) such that it \( m \)-holds and if \( t \not= P \) 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 \( t_0 = P \), so \( \forall \) loses the game \( G(t, P) \). The number of plays is the number of branches in the right terms of \( P \). We can index each play with \( i \alpha \) when \( \alpha \) is a branch of the right-term of the \( i \)-th equation of \( P \) containing forbidden constants: \( \pi \alpha \) is the play where all \( \forall \) choices are dictated by \( \alpha \). This means that two plays \( \pi \alpha, \pi \beta \) have a common prefix and differ after a position involving a \( \forall \) choice, when the branches \( \alpha \) and \( \beta \) diverge.

We also allow \( \pi \) to range over \textit{subplays} which are consecutive sub-sequences of positions of any play of \( G(t, P) \). The length of \( \pi \), \( |\pi| \), is the number of positions in \( \pi \). We let \( \pi(i) \) be the \( i \)-th position of \( \pi \), \( \pi(i, j) \) be the interval \( \pi(i) \ldots \pi(j) \) and \( \pi_i \) 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 \( q \in \pi(i) \) if \( \pi(i) = t q \theta \eta \) and \( t \not\in \pi(i) \) means that \( \pi(i) = t' q \theta \eta \) and \( t \not= t' \). If \( q = q[l, l_k], r \) or \( q[l, r] \) then its right-term is \( r \).

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

**Definition 2** Table \( \theta' \) extends \( \theta \) if \( \forall y \in \text{ dom}(\theta), \theta'(y) = \theta(y). \) Similarly, \( \eta' \) extends \( \eta \) if \( \forall z \in \text{ dom}(\eta), \eta'(z) = \eta(z) \).

We widen the usage of "extends" to positions: \( \pi(j) \theta \)-extends \( \pi(i) \) if \( \theta' \in \pi(j) \) extends \( \theta \in \pi(i) \), \( \pi(j) \eta \)-extends \( \pi(i) \) if \( \eta' \in \pi(j) \) extends \( \eta \in \pi(i) \) and \( \pi(j) \) extends \( \pi(i) \) if \( \pi(j) \theta \)-extends and \( \eta \)-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 \textit{child} of \( \pi(i) \).

**Definition 3** Assume \( \pi \in G(t, P) \). If \( \pi(i) = t q[l, l_k], r \theta \eta, \pi(j) = t' q[l, l'] \theta' \eta', \theta'(l') = l \text{ and } t' \downarrow m \text{ t}, \) then \( \pi(j) \) is a \textit{child} of \( \pi(i) \). If \( \pi(i) = y q[l, l_k], r \theta \eta, \pi(j - 1) = y' q[l, r'] \theta' \eta', l = \lambda x_z m(\overline{t}) \text{ or } \lambda x_z m \text{ or } z_m(\overline{t}) \) or \( z_m \) and \( \eta'(z_m) = t \eta \) and \( y \downarrow m \text{ t}' \), then \( \pi(j) \) is a \textit{child} of \( \pi(i) \).

**Fact 1** If \( \pi(j) \) is a \textit{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** Let $B = (B_1, \ldots, B_k) \rightarrow 0 \in T$.

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

A region of $t_0$ can be identified with a constant or simple tile. A leaf $u : 0$ of $t_0$ is the tile $u$. If $B \neq 0$ then an occurrence of $u : B$ in $t_0$, $u = f$ or $y$, with its immediate children $\lambda \beta_1 \ldots \beta_k$, where $\beta_i$ may be empty, is the tile $u(\lambda \beta_1 \ldots \lambda \beta_k)$ in $t_0$.

Tiles in $t_0$ induce subplays of $G(t_0, P)$. A play on $t = f(\lambda \beta_1 \ldots \lambda \beta_k)$ is a pair of positions $\pi(i, i + 1)$ with $t \in \pi(i)$: $q_1([l_1, \ldots, l_m], r) \in \pi(i)$. $r = f(s_1, \ldots, s_k)$, $\lambda \beta_j \in \pi(i + 1)$ is a leaf of $t$ and $q([], s_j)$ or $q([c_1, \ldots, c_n], s_j \{\alpha \beta / \alpha \beta\})$ is the state in $\pi(i + 1)$, depending on the type of $s_j$.

**Definition 2** A subplay $\pi$ is a play on $y(\lambda \beta_1 \ldots \lambda \beta_k)$ in $t_0$ if $y \in \pi(1)$ and $\pi([\pi])$ is a child of $\pi(1)$. It is a j-play if $\lambda \beta_j \in \pi(\pi)$.

A play $\pi$ on $y(\lambda \beta_1 \ldots \lambda \beta_k)$ in $t_0$ can have arbitrary length. It starts at $y$ and finishes at a leaf $\lambda \beta_k$. In between, flow of control can be almost anywhere in $t_0$ (including $y$). Crucially, $\pi(1)$ 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 $q_1([c_1, \ldots, c_n], r) \in \pi(1)$ also preserve their values. If $\pi \in G(t_0, P)$ and $y \in \pi(1)$ then there can be numerous plays $\pi(i, j)$ on $y(\lambda \beta_1 \ldots \lambda \beta_k)$ in $t_0$, including no plays at all. We now examine some pertinent properties of plays.

**Proposition 1** Assume $\pi \in G(t_0, P)$, $\pi(i, m)$ and $\pi(i, n)$, $n > m$, are plays on $y(\lambda \beta_1 \ldots \lambda \beta_k)$ and $\lambda \beta_j \in \pi(m)$.

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 $\lambda \beta_j$ and $t'$, $t'$ is an $i'$-descendent of $y_1$ and $y_1 \neq \lambda \beta_i$. Then there is an $i'$-play $\pi(m_1, n_1)$ on $y_1(\lambda \beta_1 \ldots \lambda \beta_k)$ such that $m < m_1$ and $n_1 < m'$.
3. If $\pi(m + m')$ is the first position that is a child of $\pi(n)$, $\pi(m + m')$ is $r_i$ 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 $r_i$ and for all $n' \leq m'$, $t \in \pi(n + m')$ iff $t \in \pi(n + m')$.
4. If $\pi(m + m')$ is the first position that is a child of $\pi(m)$, $\pi(m + m')$ is not $r_i$ and $\pi(i, n)$ is a $j$-play then there is a $\pi' \in G(t_0, P)$ with $\pi'(0) = \pi(n)$, $\pi'(n + m')$ is the first position that is a child of $\pi'(n)$, $\pi'(n, n + m')$ is not $r_i$ and for all $n' \leq m'$, $t \in \pi(m + m')$ iff $t \in \pi'(m + m')$. 
Proof. 1. Assume $\pi(i) = q y \left[ \lambda z_{1} \cdots z_{k}, r \theta y \right]$ and $\pi(i, n)$ are plays on $y(\lambda z_{1}, \cdots, \lambda z_{k})$ with $\lambda z_{j} \in \pi(m)$. The table $\eta = \eta_{i} \{ \lambda z_{1} \theta y / z_{1}, \cdots, \lambda z_{k} \theta y / z_{k} \}$ belongs to $\pi(i + 1)$ and positions $\pi(m - 1), \pi(n - 1)$ both $\eta$-extend $\pi(i + 1)$. Because $\pi(m), \pi(n)$ are children of $\pi(i)$, no look-up table $\eta_{i} \in \pi(t), t < i + 1$, has these entries $\eta_{i}(z_{j}) = \lambda z_{j} \theta y$. 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 $\lambda z_{j}$ in $t_{0}$. If $y_{1}$ is bound by $\lambda z_{j}$ 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 $\pi(t)$, $t < i$, and so, by move A3 its look-up table $\eta'$ cannot extend $\eta$. 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 $\eta$-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 $\lambda y$ that is below $\lambda z_{j}$. But then $y_{1}$ is bound to a leaf of a constant tile that occurs between $\lambda z_{j}$ 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 $\pi(m_{1})$ that is at a variable $y_{2} \in \pi(m_{2})$: $y_{2}$ must be a descendent of $\lambda z_{j}$. 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 $\eta$-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 $r i$ and $\pi(i, n)$ is a $j$-play. Consequently, $\pi(m) = \lambda \lambda z_{j} q \theta y$ and $\pi(n) = \lambda \lambda z_{j} q' \theta y'$ and both $\eta$-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, $\pi(m + 1) = y_{1} \eta_{i} q[t, r] \theta y_{1} \eta_{1}$ and $\pi(n + 1) = y_{1} q'[t, r'] \theta y_{1} \eta_{1}$. These positions have the same look-up table $\eta_{1}$, the same left-terms in their state, and $\theta_{1}, \theta_{1}'$ only differ in their values for the variables that are bound by $\lambda z_{j}$. Therefore, play must continue from both positions in the same way until a child of $\pi(m)$ and $\pi(n)$ is reached. 4. Assume $\pi = \pi^{i_{0}}$. The argument is similar to 3 except that the same $\forall$ choices in the non $r i$ play $\pi(m, m + m')$ need to be made. Therefore, there must be a $\pi' = \pi^{i_{0}}$ such that $\pi'(n) = \pi(n)$ and the same $\forall$ choices are made in $\pi'(m, n + m')$. \qed

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 $t(\lambda z_{j})$ is a tile with leaf $\lambda z_{j}$ and $t'$ is a constant or simple tile, then $t(\lambda z_{j}, t')$ is the composite tile that is the result of placing $t'$ directly beneath $\lambda z_{j}$ 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 $y' : B \neq 0$ 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 $t(\lambda x_1, \ldots, \lambda x_k)$ if $t$ is a basic tile with atomic leaves $\lambda x_1, \ldots, \lambda x_k$. A basic or constant tile in $t_0$ induces subplays of $G(t_0, P)$ that are compositions of plays of its component tiles.

**Definition 4** A subplay $\pi$ is a play on $t(\lambda x_1, \ldots, \lambda x_k)$ in $t_0$ if $t \in \pi(1)$, for some $i$, $\lambda x_i \in \pi(1)$, there is a branch $t = y_1 \downarrow j_1 \lambda x_i \uparrow y_2 \ldots \downarrow j_n \lambda x_i = \lambda x_i$ such that $\pi$ can be split into plays $\pi(i, j_m)$ on $y_m(\lambda x_1^m, \ldots, \lambda x_k^m)$, where $i_1 = 1$, $i_{m+1} = j_m + 1$, and $j_n = |\pi|$. It is a $j$-play if $\lambda x_j \in \pi(1)$.

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

**Corollary 1** Assume $\pi \in G(t_0, P)$, $\pi(i, m')$ and $\pi(i, n')$, $m' > n'$, are plays on $t(\lambda x_1, \ldots, \lambda x_k)$ and $\lambda x_j \in \pi(m')$, $t = y_1 \downarrow j_1 \lambda x_j \uparrow y_2 \ldots \downarrow j_n \lambda x_j = \lambda x_j$ and $\pi(i, m')$ is split into plays $\pi(i, j_m)$ on $y_m(\lambda x_1^m, \ldots, \lambda x_k^m)$, where $i_1 = i$, $i_{m+1} = j_m + 1$, $j_n = m'$. Then:

1. $\pi(m')$ extends $\pi(i)$.
2. There is a position $\pi(m_1)$, $m' < m_1 < n'$, that is a child of $\pi(j_1)$ for some $i$.
3. If $\pi(m_1)$ is the first position that is a child of $\pi(j_1)$ for some $i$, $t' \in \pi(m_1)$, $y'$ occurs on the branch between $\lambda x_j$ and $t'$, $t'$ is an $i'$-descendent of $y'$ and $y' \downarrow j \lambda x_j$, then there is an $i'$-play $\pi(m_2, n_2)$ on $y'/\lambda x_1^m, \ldots, \lambda x_k^m$ such that $m' < m_2, n_2 < m_1$.
4. If $\pi(m' + 1)$ is the first position that is a child of $\pi(j_1)$, for some $i$, $\pi(m', m' + 1)$ is $i$-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 $\lambda x_j \in \pi(n''')$, $\pi(n', n' + m_1)$ is $i$-ri and for all $n_1 \leq m_1$, $t \in \pi(m' + n_1)$ iff $t \in \pi(n' + n_1)$.
5. If $\pi(m' + 1)$ is the first position that is a child of $\pi(j_1)$, for some $i$, $\pi(m', m' + 1)$ is not $i$-ri and $\pi(i, n')$ is a $j$-play then there is a $\pi' \in G(t_0, P)$ with $\pi'(n') = \pi(m' + 1)$ and $\pi'(n' + m_1)$ is the first position that is a child of any position $\pi'(n''')$ such that $\lambda x_j \in \pi'(n''')$, $\pi'(n', n' + m_1)$ is not $i$-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 $\pi$ is a $j$-play (play) on $t$. It is a shortest $j$-play (play) if no proper prefix of $\pi$ is a $j$-play (play) and it is an $i$- $j$-play (play) if $\pi$ is also $i$-ri. It is a canonical $j$-play (play) if each $t' \in \pi(i)$ is a node of $t$. Two plays $\pi$ and $\pi'$ on $t$ are independent if one is not contained in the other: that is, $\pi \neq \pi_1 \pi'_2$, and $\pi' \neq \pi_1 \pi'_2$.

**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 $\lambda y$).

A tile $t'$ is a $j$-descendent of $t(\lambda x_1, \ldots, \lambda x_k)$ in $t_0$ if there is a branch in $t_0$ from $\lambda x_j$ to $t'$.

**Definition 7** The tile $t(\lambda x_1, \ldots, \lambda x_k)$ is $j$-end in $t_0$, if every free variable below $\lambda x_j$ in $t_0$ is bound above $t$. It is an end tile if it is $j$-end for all $j$. The tile
$t(\lambda \forall_1, \ldots, \lambda \forall_k)$ is a top tile in $t_0$ if its free variable $y$ is bound by the initial lambda $\lambda y$ of $t_0$.

A shortest play on a top tile is canonical. The following is a simple consequence of Corollary 1.

**Fact 1** If $\pi \in G(t_0, P)$ and $t$ is a j-end tile and $t \in \pi(i)$, then there is at most one j-play $\pi(i, m)$ on $t$.

We also want to classify tiles according to their plays.

**Definition 8** The tile $t(\lambda \forall_1, \ldots, \lambda \forall_k)$ is sri if every shortest play on $t$ is ri. It is j-ri if every shortest j-play on it is ri.

**Definition 9** Assume $t(\lambda \forall_1, \ldots, \lambda \forall_k)$ is a basic tile in $t_0$ and $\pi$ is a subplay. We inductively define when $t$ is j-directed in $\pi$:

1. If $t \not\ni \pi(i)$ for all $i$, then $t$ is j-directed in $\pi$.
2. If $\pi(i)$ is the first position with $t \in \pi(i)$ and there is a shortest j-play $\pi(i, m)$ on $t$ and $\pi(i, m)$ is ri and $t$ is j-directed in $\pi_{m+1}$, then $t$ is j-directed in $\pi$.

**Definition 10** Tile $t$ is j-directed in $t_0$ if it is j-directed in every $\pi \in G(t_0, P)$.

If $t$ is j-directed in $t_0$ then $\pi \in G(t_0, P)$ is partitioned uniquely into a sequence of ri inner regions $\pi(i_k, m_k)$ which are shortest j-plays on $t$.

$$\pi(1) \ldots \pi(i_1) \ldots \pi(m_1) \ldots \pi(i_n) \ldots \pi(m_n) \ldots \pi(|\pi|)$$

$t \leftarrow \lambda \forall_j \quad t \leftarrow \lambda \forall_j$

By definition, $t$ cannot occur outside these regions. If $\pi = \pi^{i_0}$ then any play $\pi^{i_j}$ will have the same intervals $\pi^{i_j}(i_k, m_k)$ until the point that $\pi^{i_j}$, $\pi^{i_j}$ diverge (which is outside a region). A tile can be j-directed in $t_0$ for multiple $j$.

We now pick out an interesting feature about embedded end tiles.

**Proposition 2** If $t_1 \equiv t_2$ are end tiles in $t_0$ and $t_2$ is a j-descendent of $t_1$, then either $t_2$ is j-directed in $t_0$ or there are $\pi, \pi' \in G(t_0, P)$ and j-plays $\pi(m_1, n_1)$ on $t_1$, $\pi'(m_2, n_2)$ on $t_2$ that are not ri and $m_2 > n_1$.

**Proof.** Assume $t_1 \equiv t_2$ are end tiles and $t_2$ is a j-descendent of $t_1$. Both $t_1$ and $t_2$ have the same head variable bound to the same $\lambda y$ above $t_1$ in $t_0$. Let $\pi \in G(t_0, P)$. Consider the first position $t_2 \in \pi(m)$. There must be an earlier position $t_1 \in \pi(i)$ such that $\pi(m)$ extends $\pi(i)$ and a j-play $\pi(i, i + k)$ on $t_1$. If this play is ri then because $t_1 \equiv t_2$ are end tiles there is the same j-play on $t_2$, $\pi(m, m + k)$. This argument is repeated for subsequent plays or until the j-play on $t_1$ is not ri. If the play on $t_1$ is not ri then for some play $\pi'$ with $\pi'(m) = \pi(m)$ there is the same j-play $\pi'(m, m + k)$ on $t_2$. \qed

5 TRANSFORMATIONS

In this section we define four transformations. A transformation $T$ changes a tree $s$ into a tree $t$, written $s \xrightarrow{T} t$. Each transformation preserves the crucial
property: if \( s \vdash t \) and \( s \vdash P \) then \( t \vdash P \) which is proved using the game-theoretic characterisation. The first transformation is easy. Let \( t' \) be a sub-tree of \( t_0 \) whose root node is a variable \( y \) or a constant \( f : B \neq 0 \). Then \( G(t_0, P) \) avoids \( t' \) if \( t' \notin \pi(i) \) for all positions and plays \( \pi \in G(t_0, P) \). Let \( t_0[a/t'] \) be the result of replacing \( t' \) in \( t_0 \) with the constant \( a : 0 \).

**T1** If \( G(t_0, P) \) avoids \( t' \) then transform \( t_0 \) to \( t_0[a/t'] \)

Assume that \( t_0 \vdash P \). 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 \( t(\lambda x_1, \ldots, \lambda x_k) \) is a \( j \)-directed, \( j \)-end tile in \( t_0 \) and \( t' \) is the sub-tree of \( t_0 \) rooted at \( t \). If \( t_j \) is the sub-tree directly beneath \( \lambda x_j \) then transform \( t_0 \) to \( t_0[t_j/t'] \).

The next transformation separates plays.

**Definition 1** Assume \( t = t(\lambda x_1, \ldots, \lambda x_k) \) is a basic scri 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 \( t(\lambda x_1, \ldots, \lambda x_k) \) is a separator in \( t_0 \) and \( t' \) is the sub-tree of \( t_0 \) rooted at \( t \) then transform \( t_0 \) to \( t_0[t(\lambda x_1, \ldots, \lambda x_k, t'/t')] \).

Here, we have added an extra copy of \( t \) directly below each \( \lambda x_j \): we assume that the head variable of this copy of \( t \) is bound by the \( \lambda x_j \) that binds the head variable of the original \( t \) and we assume that all variables below \( \lambda x_j \) 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 \( t(\lambda x_1, \ldots, \lambda x_k) \) is \( j \)-ri and not \( j \)-end in \( t_0 \) and directly below \( \lambda x_j \) is the constant or basic tile \( u(\lambda x_1, \ldots, \lambda x_m) \) 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 \( t(\lambda x_1, \ldots, \lambda x_k) \) is \( j \)-permutable with \( u(\lambda x_1, \ldots, \lambda x_m) \) in \( t_0 \) and \( t' \) is the sub-tree rooted at \( u \) in \( t_0 \). If \( t_i \) and \( t'_i \) are the sub-trees of \( t_0 \) directly below \( \lambda x_i \) and \( \lambda x_i \), then transform \( t_0 \) to \( t_0[u(\lambda x_1, w_1, \ldots, \lambda x_m, w_m) / t'] \) where \( w_i = t(\lambda x_1, t_1, \ldots, \lambda x_{i-1}, t_{i-1}, \lambda x_i, t_i', \lambda x_{i+1}, t_{i+1}, \ldots, \lambda x_k, t_k) \).

The tile \( t \) is copied below \( u \); however, in the copy of \( t \) below \( \lambda x_i \) of \( u \) \( t'_i \) (and not \( t_i \)) occurs below \( \lambda x_i \) 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 \( f(\lambda x_1, \ldots, \lambda x_m) \). There is a shortest \( j \)-play \( \pi(i, m) \) on \( t \) and another \( j \)-play \( \pi(i, n) \) on \( t \).
\[ \pi(i) \ldots \pi(m) \pi(m + 1) \ldots \pi(n) \pi(n + 1) \]
\[ t \quad \lambda x_j \quad f \quad \lambda x_j \quad f \]

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 \( s \text{ Ti } t \) and \( s \models P \) then \( t \models P \).

**Proof.** This is clear when \( i = 1 \). Consider \( i = 2 \). Assume \( t(\lambda x_1, \ldots, \lambda x_k) \) 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 \( \lambda x_j \), \( t'_0 = t_0[t_j/t'] \) and \( t_0 \models P \). We shall convert \( \pi = \pi^n \in G(t_0, P) \) into the play \( \sigma = \sigma^n \in G(t'_0, P) \) that \( \forall \) loses. The play \( \pi \) is split uniquely into regions.

\[
\pi(1) \ldots \pi(i_1) \ldots \pi(m_1) \ldots \pi(i_2) \ldots \pi(m_2) \ldots \pi(i_n) \ldots \pi(m_n) \ldots \pi(|\pi|) 
\]
\[ t \quad \lambda x_j \quad t \quad \lambda x_j \quad t \quad \lambda x_j \]

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

\[
\pi(1) \ldots \pi(i_1 - 1)\pi(m_1 + 1) \ldots \pi(i_n - 1)\pi(m_n + 1) \ldots \pi(|\pi|) 
\]

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 \( \lambda x_m \), \( m \neq j \), of \( t \) then \( s \) cannot occur in any outer subplay of \( \pi \). 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 \( \theta(y) = t_\eta \) and there is a free variable \( z \) in \( l \) such that \( \eta(z) = \phi' \). However, this is impossible. Consider \( \eta(1) \in \pi(i_1) \); 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 \( t(\lambda x_1, \ldots, \lambda x_k) \) is a separator in \( t_0 \), \( t' \) is the subtree of \( t_0 \) rooted at \( t \) and \( t'_0 = t_0[t(\lambda x_1, t', \ldots, \lambda x_k, t')/t'] \). We shall convert \( \pi = \pi^n \in G(t_0, P) \) into \( \sigma = \sigma^n \in G(t'_0, P) \) that \( \forall \) loses. Consider any shortest play on \( t \) in \( \pi^n \), \( \pi(m, k) \) and assume it is a \( j \)-play. By definition this play is ri. Therefore, this interval is transformed into the following interval for \( t'_0 \).

\[
\pi(m) \ldots \pi(k) \pi(m) \ldots \pi(k) 
\]
\[ t \quad \lambda x_j \quad t \quad \lambda x_j \]

where the second \( t \) is the copy of \( t \) directly beneath \( \lambda x_j \) in \( t'_0 \).
Finally, \(i = 4\). Assume \(t(\lambda \mathcal{F}_1, \ldots, \lambda \mathcal{F}_k)\) is \(j\)-permutable with \(u(\lambda \mathcal{E}_1, \ldots, \lambda \mathcal{E}_n)\) 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 \(\lambda \mathcal{E}_i\) and \(\lambda \mathcal{F}_i\) and \(t'_0 = t_0[u(\lambda \mathcal{E}_1, w_1, \ldots, \lambda \mathcal{E}_n, w_m)]/t'\) where \(w_i\) as is in \(T4\). We shall convert \(\pi = \pi^{i_0} \in G(t_0, P)\) into \(\sigma = \sigma^{i_0} \in G(t'_0, P)\) that \(\forall\) loses. The play \(\pi\) can be divided into non-overlapping regions \(\pi(i_k, m_k)\).

\[
\pi(1) \ldots \pi(i_1) \ldots \pi(m_1) \pi(m_1 + 1) \ldots \pi(i_n) \ldots \pi(m_n) \pi(m_n + 1) \ldots \pi(|\pi|)
\]

\[
t \quad \lambda \mathcal{F}_j \quad u \quad t \quad \lambda \mathcal{F}_j' \quad u
\]

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

\[
\ldots \pi(i_k) \ldots \pi(i') \ldots \pi(m') \ldots \pi(m_k) \ldots
\]

\[
t \quad t \quad \lambda \mathcal{F}_j \quad \lambda \mathcal{F}_j'
\]

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

\[
\pi(i_k) \ldots \pi(m_k) \pi(m_k + 1) \pi(m_k + 2) \pi(i_k) \ldots \pi(m_k) \pi(m_k + 3) \ldots
\]

\[
t \quad \lambda \mathcal{F}_j \quad f \quad \lambda \mathcal{E}_{k_i} \quad t \quad \lambda \mathcal{F}_j' \quad t'_k
\]

where \(t'_k \in \pi(i_k)\) is the copy of \(t\) directly beneath \(\lambda \mathcal{E}_{k_i}\) in \(t'_0\).

Next, let \(u\) be a basic tile. To obtain \(\sigma\) we iteratively do additions and deletions to \(\pi\) starting with \(\pi(i_1, m_1)\) and then recursively transforming inner \(j\)-plays on \(t\) within this region. Let \(\pi\) be the result of the changes to the initial \(\pi\) 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 \(\pi(m_k + 1, n_k)\) be all plays on \(u \in \pi(m_k + 1)\). If there are no plays then \(\pi\) is initially unchanged. Otherwise, \(\pi\) has the following structure:

\[
\ldots \pi(i_k) \ldots \pi(m_k) \pi(m_k + 1) \ldots \pi(n_k) \ldots \pi(n_k + 1) \ldots
\]

\[
t \quad \lambda \mathcal{F}_j \quad u \quad \lambda \mathcal{E}_{k_i} \quad t \quad \lambda \mathcal{F}_j' \quad t'_k
\]

To obtain the new \(\pi\), we do the following addition for each \(i\)

\[
\pi(i_k) \ldots \pi(m_k) \pi(m_k + 1) \ldots \pi(n_k) \ldots \pi(n_k + 1) \ldots
\]

\[
t \quad \lambda \mathcal{F}_j \quad u \quad \lambda \mathcal{E}_{k_i} \quad \lambda \mathcal{E}_{k_i} \quad t \quad \lambda \mathcal{F}_j \quad t'_k
\]

where \(t\) immediately after \(\pi(n_k)\) is its copy in \(t'_0\) directly beneath \(\lambda \mathcal{E}_{k_i}\).

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 \(\lambda \mathcal{E}_{k_i} \in \pi(m_k + n)\). Consider all \(j\)-plays \(\pi(i_k, m'_k)\) on \(t \in \pi(i_k)\) where \(m'_k = m_k - k:\)

\[
\pi(i_k) \ldots \pi(m_k) \ldots \pi(n_k) \ldots \pi(n_k + 1) \ldots \pi(m_k + n) \ldots \pi(m_k + n + 1) \ldots
\]

\[
t \quad \lambda \mathcal{F}_j \quad \lambda \mathcal{F}_j \quad u \quad \lambda \mathcal{E}_{k_i} \quad t'_k
\]
There must be the same play on \( u \) at each \( \pi(m_i^k + 1) \) 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

\[
\pi(i_k) \ldots \pi(m_i^k) \pi(m_i^k + 1) \ldots \pi(m_i^k + n) \pi(i_k) \ldots \pi(m_i^k) \pi(m_i^k + n + 1)
\]

where the second \( t \in \pi(i_k) \) is the copy of \( t \) directly below \( \lambda\pi_i \) in \( t_0 \), and for subsequent \( i > 1 \) we delete the \( \text{ri} \) region \( \pi(m_j^k + 1, m_j^k + n) \). To complete the argument, we recursively apply this technique to shortest \( j \)-plays on \( t \) within \( \pi(i_k, m_i^k) \); note that \( j \)-plays on \( t \) below \( \lambda\pi_i \) within \( \pi(i_k, m_i^k) \) will include additional \( \text{ri} \) 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 \( x(v_1, \ldots, v_n) = u \) the term \( u \) is a constant \( a : 0 \) [3, 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 \( u : 0 \) of \( t_0 \)
2. choose a marked node \( u \) such that no descendent of \( u \) is marked
3. if \( t_0 T u \) then \( t_0 = t' \) and unmark all nodes and return to 1
4. identify basic or constant tile \( t = t(\lambda\pi_1, \ldots, \lambda\pi_k) \) rooted at \( u \)
5. if \( t_0 T u' \) 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 \( \lambda\pi_i \)
7. if \( t_0 T A u' \) at \( t \) and a successor then \( t_0 = t' \) and unmark all nodes and return to 1.
8. if \( u' \lambda_i \lambda\pi_1 u \) then unmark \( u \) and mark \( u' \) and return to 2
9. finish

![Fig. 2. The algorithm](image)

Clearly, the procedure must terminate with \( t_0 = P \) 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 \( y : 0 \) and \( a : 0 \) then \( t' \) consists of sri end tiles and leaves \( a : 0 \).
Proof. By a simple induction. A leaf $u$ may be a constant or a variable. Consider $u'$ such that $u' \downarrow_i u$. 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 $T_3$ or $T_4$ 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 $t_i(\lambda x_i^1, \ldots, \lambda x_i^{n_i})$ such that for each $i$, $t_{i+1}$ is directly below $\lambda x_i^n$, 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 $y : \emptyset \in \pi(m')$, 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 $\pi(k, n')$ is not ri and for some $n'$, $\pi(k, n')$ is also not ri.)

References

3. Huet, G. Résolution d'équations dans les langages d'ordre 1, 2, ... $\omega$. Thèse de doctorat d'état, Université Paris VII, (1976).
7. Schubert, A. Linear interpolation for the higher-order matching problem. Lecture Notes in Computer Science, 1214, 441-452.