That SnS Can Be Modally Characterized

Hans Hüttel
Laboratory for Foundations of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ
SCOTLAND

November 21, 1996

Abstract

We show that a modal mu-calculus with label set $\{1, \ldots, n\}$ can define the Rabin recognizable tree languages up to an equivalence similar to the observational equivalence of Milner.

1 Introduction

In [11] it was shown that the temporal logic ETL [10] can define exactly the class of ω-regular languages. In [7] it was shown that a fixed-point calculus whose signature apart from maximal and minimal fixed points and disjunction includes the usual operators on trees can define exactly the sets of infinite trees recognized by Rabin tree automata [8]; this class of sets corresponds to the class of structures definable in the second-order monadic theory of n successors, SnS .

It would be nice if one could show that a branching time temporal logic has the same expressive power as SnS ; after all, branching time temporal logics are interpreted on computation trees. In [4] it is shown that a restricted version of SnS with set quantification restricted to paths is expressively equivalent to CTL^* for binary tree models. However, as was pointed out in [2], [1] (and independently shown in [4]) the full SnS can express properties which have no correlate in a branching time temporal logic with a *finite* number of branching time operators. The reason for this is that the semantics of any branching time operator uses finitely many first-order quantifiers, so the logic in question has a particular upper bound on the number of quantifiers. But to express the property that there are k nodes in a tree which are incomparable to a node x w.r.t. to the ancestral ordering, $\leq$, one needs k quantifiers:

$$A(x) \stackrel{\text{def}}{=} \exists x_1, \ldots, x_k. \bigwedge_{i=1}^{k} (x \not\leq x_i \wedge x_i \not\leq x)$$

In this paper we show that a modal fixed-point calculus that incorporates a notion of counting descendants characterizes the Rabin recognizable tree languages of [8] up to an equivalence of parental information.

The paper is organized as follows: we first outline the syntax and semantics of SnS and the fixed-point calculus, here referred to as CML . We then show that SnS is at least as expressive as CML . Finally we use Rabin's tree theorem to establish the result by showing how to encode the acceptance condition of Rabin automata in CML such that a set of trees equivalent to the SnS -recognizable set satisfies the acceptance encoding formula. Our notion of equivalence resembles the observational equivalence of [6] and the tree equivalence of [3].

2 Syntax and semantics of SnS and CML

2.1 SnS

The second order monadic theory of n successors, SnS , has its set of terms, TERM_{SnS} , given by the abstract syntax

$$T ::= x \mid \epsilon \mid Ti$$

where $i \in \{1, \ldots, n\}$ and x ranges over a set of first-order variables, Var_1. Its formulae, FORM_{SnS} , are given by the abstract syntax

$$AF ::= T_1 = T_2 \mid T_1 < T_2 \mid T \in X \mid T \in P_i$$

for atomic formulae. Here X ranges over a set of second-order variables, Var_2 and P_i ranges over a set of atomic predicates. For composite formulae the syntax is

$$F ::= AF \mid F_1 \vee F_2 \mid \neg F \mid \forall x.F \mid \forall X.F$$

SnS is interpreted in the structures in STRUCT_{SnS} , which are structures of the form $\mathcal{M} = (\{0, \ldots, n-1\}^*, succ_0, \ldots, succ_{n-1}, <, P_1, \ldots, P_m)$. Here ϵ is the empty string. $succ_0, \ldots, succ_{n-1}$ are the successor functions defined by $succ_i(w) = wi$, i.e. by concatenation. $<$ is the prefix order on $\{0, \ldots, n-1\}^*$ and $P_1, \ldots, P_m$ are subsets of $\{0, \ldots, n-1\}^*$.

The semantics of SnS is defined relative to two variable assigments, $\sigma \in Ass_1 : Var_1 \rightarrow \{0, \ldots, n-1\}^*$ and $\rho \in Ass_2 : Var_2 \rightarrow \mathbf{2}^{\{0,\ldots,n-1\}^*}$. The semantics of TERM_{SnS} is given by $[\![\ ]\!] : \text{TERM}_{SnS} \rightarrow Ass_1 \rightarrow \{0, \ldots, n-1\}^*$ as defined by

$$\begin{array}{rcl} [\![x]\!]\sigma & = & \sigma(x) \\ [\![\epsilon]\!]\sigma & = & \epsilon \\ [\![Ti]\!]\sigma & = & ([\![T]\!]\sigma)i \end{array}$$

The semantics of FORM_{SnS} is defined by $\models \subseteq \text{STRUCT}_{SnS} \times Ass_1 \times Ass_2 \times \text{FORM}_{SnS}$. The clauses are [1] :

$$\begin{array}{rcl} \mathcal{M} \models_{\sigma,\rho} T_1 = T_2 & \iff & [\![T_1]\!]\sigma = [\![T_2]\!]\sigma \\ \mathcal{M} \models_{\sigma,\rho} T_1 < T_2 & \iff & [\![T_1]\!]\sigma < [\![T_2]\!]\sigma \\ \mathcal{M} \models_{\sigma,\rho} T \in X & \iff & [\![T]\!]\sigma \in \rho(X) \\ \mathcal{M} \models_{\sigma,\rho} T \in P & \iff & [\![T]\!]\sigma \in P \\ \mathcal{M} \models_{\sigma,\rho} \neg F & \iff & \text{not } \mathcal{M} \models_{\sigma,\rho} F \\ \mathcal{M} \models_{\sigma,\rho} F_1 \vee F_2 & \iff & \mathcal{M} \models_{\sigma,\rho} F_1 \text{ or } \mathcal{M} \models_{\sigma,\rho} F_2 \\ \mathcal{M} \models_{\sigma,\rho} \forall x.F & \iff & \text{for all } v \in \{0, \ldots, n-1\}^* : \mathcal{M} \models_{\sigma\{v/x\},\rho} F \\ \mathcal{M} \models_{\sigma,\rho} \forall X.F & \iff & \text{for all } V \in \mathbf{2}^{\{0,\ldots,n-1\}^*} : \mathcal{M} \models_{\sigma,\rho\{V/X\}} F \end{array}$$

When the atomic predicates are explicitly chosen from among $P_1, \ldots, P_m$, we shall refer to the language as $SnS_{P_1,\ldots,P_m}$.

2.2 CML

We now introduce a modal logic, CML , for describing tree properties. The syntax of its formulae, FORM_{CML} , is given by

$$F ::= P \mid F \vee F \mid \neg F \mid z \mid \nu z.F \mid ⓘ F$$

where $i \in \{0, \ldots, n-1\}$, P ranges over a set ATOMIC_{CML} of atomic predicates on $\{0, \ldots, n-1\}^*$ and z ranges over a set of recursion variables, $CMLVar$. We assume that all recursions variables are within the scope of an even number of

[1] $\sigma\{v/x\}$ is the assignment in Ass_1 that maps x to v and otherwise agrees with σ. Similarly, $\rho\{V/X\}$ is the assignment in Ass_2 that maps X to V and otherwise agrees with ρ.

negations; this will ensure the well-definedness of $\| \ \|$ given below. Note that CML can be seen as a version of the propositional mu-calculus of [5] with label set $\{0, \ldots, n-1\}$.

We interpret CML in STRUCT_{SnS} ; we now think of the nodes in a tree as possible worlds and $succ_i$ as relations between possible worlds. The intended semantics of the (i) modality in CML is that $(i)\, F$ holds if F holds in the ith successor-world.

The semantic function $\| \ \|$ is seen relative to an assignment of the recursion variables, $\rho \in Ass_{CMLVar}$. It therefore has type $\| \ \| : \text{FORM}_{CML} \rightarrow Ass_{CMLVar} \rightarrow \mathbf{2}^{\{0,\ldots,n-1\}^*}$. It is defined relative to a structure $\mathcal{M}$ by

$$\begin{aligned}
\|P_j\|_\rho^\mathcal{M} &= P_j \\
\|F_1 \vee F_2\|_\rho^\mathcal{M} &= \|F_1\|_\rho^\mathcal{M} \cup \|F_2\|_\rho^\mathcal{M} \\
\|\neg F\|_\rho^\mathcal{M} &= \{0, \ldots, n-1\}^* \setminus (\|F_1\|_\rho^\mathcal{M}) \\
\|(i)\, F\|_\rho^\mathcal{M} &= \{w \mid w' = succ_i(w) \Rightarrow w' \in \|F\|_\rho^\mathcal{M}\} \\
\|z\|_\rho^\mathcal{M} &= \rho(z) \\
\|\nu z.F\|_\rho^\mathcal{M} &= \bigcup \{V \mid V \subseteq \|F\|_{\rho\{V/z\}}^\mathcal{M}\}
\end{aligned}$$

Here, when the atomic predicates are explicitly chosen from among $P_1, \ldots, P_m$, we shall refer to our language as $\text{FORM}_{CML_{P_1,\ldots,P_m}}$.

Note that all 'superfluous' logical operators, including $\vee$, $\oplus$ (exclusive or) and $\mu z.F$ (least fixed point), are easily derived. The same goes for the temporal operators of CTL [2]. For instance, $\forall G$ is defined by $\forall GP \stackrel{\text{def}}{=} \nu z.P \wedge \bigwedge_{i=0}^{n-1} (i)\, z$. We shall feel free to use all these derived operators as convenient abbreviations.

3 SnS is at least as expressive as CML

In what follows we shall often refer to trees. An n-ary infinite labelled tree t whose labels are in the alphabet A can be seen as a function $t : \{0, \ldots, n-1\}^* \rightarrow A$; we define $dom(t) \stackrel{\text{def}}{=} \{0, \ldots, n-1\}^*$. The set of all A-labelled trees is denoted by T_A^ω. A set of n-ary A-labelled trees is called a *tree language* over A.

Labelled trees and the structures in STRUCT_{SnS} can in an obvious way be regarded as one and the same thing. For a structure $\mathcal{M} = (\{0, \ldots, n-1\}^*, succ_0, \ldots, succ_{n-1}, <, P_1, \ldots, P_m)$ can be seen as an infinite n-ary tree $t_\mathcal{M} : \{0, \ldots, n-1\}^* \rightarrow \mathbf{2}^{\{P_1,\ldots,P_m\}}$ with $t(w) = P_i$ iff $w \in P_i$. And a tree t labelled by $A = \{\alpha_1 \ldots, \alpha_m\}$ determines a structure $\mathcal{M}_t = (\{0, \ldots, n-1\}^*, succ_0, \ldots, succ_{n-1}, <, P_{\alpha_1}, \ldots, P_{\alpha_m})$ with $w \in P_{\alpha_j}$ iff $t(w) = \alpha_j$.

It is therefore natural to talk about the tree languages definable in SnS and CML . A tree language L is SnS -definable iff there is an SnS -formula whose models are the trees in L:

Definition 1 *An A-labelled tree language L is SnS -definable if there is a formula $\phi \in \text{FORM}_{SnS}$ with one free first order variable, x, and closed w.r.t. second-order variables such that*

$$L = \{t \mid \mathcal{M}_t \models_{\emptyset\{\epsilon/x\},\emptyset} \phi(x)\}$$

The single free first order variable is to denote the root of the tree, ϵ. Putting all this slightly differently with a slight abuse of notation, we can define

$$\|\phi(x)\| \stackrel{\text{def}}{=} \{t \mid \mathcal{M}_t \models_{\emptyset\{\epsilon/x\},\emptyset} \phi(x)\}$$

Then L is SnS -definable just in case there is a formula $\phi(x)$ in SnS such that $L = \|\phi(x)\|$.

We say that L is CML -definable if there is a formula in CML which is true in the roots of the trees in L and none others:

Definition 2 *An A-labelled tree language L is CML -definable if there is a closed formula $\psi \in \text{FORM}_{CML}$ such that*[2]

$$L = \{t \mid \epsilon \in \|\psi\|_{\rho}^{t_{\mathcal{M}}}\}$$

Again, putting all this slightly differently, we can define

$$\|\psi\| \stackrel{\text{def}}{=} \{t \mid \epsilon \in \|\psi\|^{\mathcal{M}^t}\}$$

Since the metalanguage used in defining the semantics of CML is not far from SnS , it is easy to see that any tree language definable in CML is also definable in SnS .

Lemma 1 *For any closed $F \in \text{FORM}_{CML_{P_1,\ldots,P_m}}$ there is a $\Theta(F) \in \text{FORM}_{SnS_{P_1,\ldots,P_m}}$ with $\|F\| = \|\Theta(F)\|$.*

---

[2]The assignment ρ is inessential, since ψ is closed, and will henceforth be omitted for closed formulae.

PROOF: We exhibit a direct translation $\Theta : \text{FORM}_{CML_{P_1,\ldots,P_m}} \rightarrow \text{FORM}_{SnS_{P_1,\ldots,P_m}}$ that gives us a formula in SnS with one free variable x :

$$\begin{aligned}
\Theta(P_k) &= x \in P_k \\
\Theta(F_1 \wedge F_2) &= \Theta(F_1) \wedge \Theta(F_2) \\
\Theta(\neg F) &= \neg\Theta(F) \\
\Theta(\textcircled{i}\, F) &= (\Theta(F))[xi/x] \\
\Theta(z) &= Y_z \\
\Theta(\nu z.F) &= \exists S.x \in S \wedge (\forall y.y \in S \Rightarrow \\
&\quad (\Theta(F))[y/x][S/Y_z]) \wedge \forall T.(\forall z.z \in T \Rightarrow (\Theta(F))[z/x][T/Y_z] \Rightarrow T \subseteq S
\end{aligned}$$

($[xi/x]$ denotes a uniform substitution of xi for x.) Note that atomic predicates are carried over and that recursion variables become second order variables. The translation of fixed-points is just a formulation of Tarski's fixed-point induction principle. Induction in the structure of CML -formulae shows that this translation always gives a formula with one variable and that $\|F\| = \|\Theta(F)\|$. □

Since CML by the above lemma can be embedded in SnS , and since the latter is decidable [8] and since the translation is effectively constructible we also see that CML is *decidable* (cf. [9]).

4 CML is as expressive as SnS modulo $\simeq_A$

We now establish that CML is as expressive as SnS up to an 'observational' equivalence of tree languages. We here use the tree theorem of [8].

Definition 3 *A* Rabin automaton *on n-ary A-labelled trees is a quadruple $\mathcal{A} = (Q, q_0, \Delta, \Omega)$, where Q is the finite set of* states, *q_0 is the* start state, *$\Delta \subseteq Q \times A \times Q^n$ is the finite* transition relation *and $\Omega \subseteq \mathbf{2}^Q \times \mathbf{2}^Q$ is a finite collection of finite* acceptance pairs.

Definition 4 *A* run *of the Rabin automaton $\mathcal{A} = (Q, q_0, \Delta, \Omega)$ on an A-labelled tree t is any Q-labelled tree r such that*

$$r(\epsilon) = q_0$$

$$r(s) = q \text{ and } r(si) = q_i, 1 \leq i \leq n \text{ with } (q, t(s), q_1, \ldots, q_n) \in \Delta$$

Definition 5 *A run r of $\mathcal{A}$ is* accepting *if for all paths π there is an acceptance pair $(L_i, U_i) \in \Omega_{\mathcal{A}}$ such that* [3]

$$In(r \mid \pi) \cap L_i = \emptyset \text{ and } In(r \mid \pi) \cap U_i \neq \emptyset$$

[3] $In(r \mid \pi)$ denotes the set of states occurring infinitely often along π.

Definition 6 *A tree language L is* Rabin-recognizable *if there is a Rabin automaton $\mathcal{A}$ such that $t \in L$ iff t admits an accepting run of $\mathcal{A}$.*

Theorem 1 ([8]) *A tree language is SnS -definable iff it is Rabin recognizable in that*

- *For any $\phi \in \mathrm{FORM}_{SnS_{P_1,\dots,P_m}}$ there exists a Rabin automaton $\mathcal{A}_\phi$ over the alphabet $\mathbf{2}^{P_1,\dots,P_m}$ such that $t_\mathcal{M}$ admits an accepting run on A_ϕ iff $\mathcal{M} \models \phi$.*
- *For any Rabin automaton $\mathcal{A}$ over the alphabet $A = \{\alpha_1, \dots, \alpha_m\}$ there exists a $\phi_\mathcal{A} \in \mathrm{FORM}_{SnS P_{\alpha_1},\dots,P_{\alpha_m}}$ such that $\mathcal{M}_t \models \phi_\mathcal{A}$ iff t admits an accepting run on $\mathcal{A}$.*

In what follows, the first half of Theorem 1 will be essential. For a given $\phi \in \mathrm{FORM}_{SnS_{P_1,\dots,P_m}}$ we express the Rabin acceptance condition of the corresponding automaton $\mathcal{A}_\phi$ in CML . The acceptance condition assumes a knowledge of the states assigned to a node. However, we do not have the state predicates available in our structures, only labelling predicates for the automaton alphabet A, so we must find a way of overcoming this. We do so by coding the product of a tree and its run on $\mathcal{A}_\phi$, $t\hat{}r \in T^\omega_{A\times Q}$, as a tree where the states assumed in the run can be recovered from the position of the nodes in the encoding, using the $ⓘ$-operator. In what follows we assume wlog that $n = 2$.

Definition 7 *A* computation history *(ρ, S) for the Rabin automaton $\mathcal{A} = (Q, q_0, \Delta, \Omega)$ is any function $\rho : S \to A$ and its domain $S \subseteq (\{0,1\}^*Q)^+$ which satisfy*

$$q_0 \in S \text{ and } \rho(q_0) = a \text{ for some } a \in \{a \mid \exists q', q'' : (q_0, a, q', q'') \in \Delta\}$$

$$sq_k \in S \Rightarrow \exists (q_k, a, q_1, q_2) \in \Delta : \begin{cases} sq_k 000 q_k 0 q_1 & \in S \\ sq_k 000 q_k 1 q_2 & \in S \end{cases} \text{ and } \rho(sq_k) = a$$

There is an obvious isomorphism H between $T^\omega_{A\times Q}$ and the set of computation histories, and we shall feel free to speak of the computation history associated with a given tree. Also, given a history (ρ, S) it is easy to find the unique run $r_{(\rho,S)} : \{0,1\}^* \to A$ to which it corresponds.

We now code all computation histories of $\mathcal{A}$ as full binary trees labelled by $A\cup\{\tau\}$, where τ is a dummy letter which signifies that the node in the encoding does not correspond to a node in the original computation history. The coding consists in taking the homomorphic extension K^* of the node-coding

$$
\begin{array}{rcll}
K(q_i) & = & 10^i 1 & 1 \leq i \leq |Q| \\
K(w) & = & w & w \in \{0,1\}^*
\end{array}
$$

and defining the associated labelling $K^*(\rho, S) : \{0,1\}^* \to A \cup \{\tau\}$ by

$$
K^*(\rho, S)(w) = \begin{cases} a & \text{if } w = K^*(\rho, S)(s) \text{ for some } s \in S \text{ with } \rho(s) = a \\ \tau & \text{otherwise} \end{cases}
$$

These encoded computation histories correspond to those A-labelled trees admitting a run on $\mathcal{A}$ via the $\simeq_A$ equivalence to be defined below.

The state assumed in a node is thus reflected in the path to its descendants in the coding. Using this fact we can now define the state predicates in CML, assuming a predicate P_a for every $a \in A \cup \{\tau\}$. We describe the possible paths in the coding from one node labelled by τ to the next; the state information is contained in the shape of the path. We must also describe that all nodes on the path between two non-dummy nodes and all subtrees thereof are labelled by τ. Letting $\langle i \rangle F$ denote the formula $\textcircled{i}\ (F \wedge P_\tau) \wedge \bigwedge_{j \neq i} \textcircled{j}\ \forall G.P_\tau$ and letting $\langle i \rangle^j$ denote this iterated j times, we define

$$
\mathbf{Q_{q_j}} \stackrel{\text{def}}{=} \neg P_\tau \wedge \bigvee_{q_i \in Q} \wedge_{k=0}^{1} \langle 0 \rangle \langle 0 \rangle \langle 0 \rangle \langle 1 \rangle \langle 0 \rangle^j \langle 1 \rangle \textcircled{k}\ \langle 1 \rangle \langle 0 \rangle^i \langle 1 \rangle (\neg P_\tau)
$$

A node in a *coded* computation history satisfies the derived state predicate exactly if the run corresponding to the history is labelled by q_j at the corresponding node:

Lemma 2 *For any tree t and associated computation history (ρ, S) any node s in the run $r_{(\rho,S)}$ with associated Q-labelling predicates satisfies $\{Q_{q_j}\}$ $s \in \|Q_{q_j}\|^{\mathcal{M}_r}$ iff $K^*(s) \in \|\mathbf{Q_{q_j}}\|^{\mathcal{M}_{K^*(\rho,S)}}$*

(That the coded computation history contains the same ancestral information as the original tree w.r.t. non-τ-labels will be made precise by the definition of $\simeq_A$ (Definition 9).)

We also need to express a next-time operator w.r.t. coded histories, $X_i.F$, which is to denote that the ith proper descendant has property F. (By a *proper descendant* of a node s we mean any descendant not labelled by τ such that all nodes between it and s are labelled by τ, see Definition 8). This only makes sense in non-dummy nodes, so we get

$$
X_i.F \stackrel{\text{def}}{=} \neg P_\tau \quad \wedge \bigvee_{q_{k_1}, q_{k_2} \in Q} \textcircled{0}\ \textcircled{0}\ \textcircled{0}\ \textcircled{1}\ \textcircled{0}^{\,k_1} \textcircled{1}\ \textcircled{i}\ \textcircled{1}\ \textcircled{0}^{\,k_2} \textcircled{1}\ F
$$

The CTL branching time operators can now be redefined w.r.t. $X_i.F$ so we get e.g. $\forall \mathbf{G} P \stackrel{\text{def}}{=} \nu z.P \wedge \bigwedge_{i=0}^{1} X_i.z$.

Rabin acceptance is now formulated as a conjunction of two CML formulae interpreted over $A \cup \{\tau\}$-labelled trees, namely

$$Acc_{\mathcal{A}} \stackrel{\text{def}}{=} Acc_1 \wedge Acc_2$$

where Acc_1 describes that the $A \cup \{\tau\}$-labelled tree indeed is a coded computation history and Acc_2 describes the Rabin acceptance condition itself.

Acc_1 includes a description of the transition function of the automaton:

$$\begin{aligned} Acc_1 \;\stackrel{\text{def}}{=}\;& P_\tau \wedge ⓪\, (\forall G.P_\tau) \wedge ①\, ⓪\, (\forall G.P_\tau) \wedge \\ & ①\, ①\, (\mathbf{Q_{q_0}} \wedge \nu z.(\bigoplus_{q \in Q} \mathbf{Q_q} \wedge \bigoplus_{(q,a,q_1,q_2) \in \Delta} (\mathbf{Q_q} \wedge P_a \supset \bigwedge_{i=0}^{1} X_i(\mathbf{Q_{q_i}} \wedge z))) \end{aligned}$$

Acc_2 is given as follows:

$$\begin{aligned} Acc_2 \;\stackrel{\text{def}}{=}\;& (\forall \mathbf{F}.(\nu z. \bigvee_{q \in \bigcup U_i} \mathbf{Q_q} \wedge \forall \mathbf{F}.z)) \wedge \\ & (\bigwedge_{U_i} \bigwedge_{\mathbf{Q_q} \in U_i} \neg(\exists \mathbf{F}.\mu z.(\bigvee_{p \in L_i} \mathbf{Q_p} \wedge \exists \mathbf{F}.(\mathbf{Q_q} \wedge z)))) \end{aligned}$$

The first conjunct in Acc_2 states that on every path, some state in the U-component of an acceptance pair occurs infinitely often. The second conjunct states that for no acceptance pair is there a path such that both a state in the L-component and a state in the U-component occur infinitely often.

Thus we have

Theorem 2 *For any Rabin automaton $\mathcal{A}$ over $A = \{P_{\alpha_1}, \ldots, P_{\alpha_m}\}$ there is a $CML_{P_{\alpha_1},\ldots,P_{\alpha_m},P_\tau}$-formula $Acc_{\mathcal{A}}$ such that for any tree t there is a corresponding computation history (ρ, S) with $K^*(\rho, S) \in \|Acc_{\mathcal{A}}\|^{\mathcal{M}_t}$ iff $\mathcal{A}$ accepts t.*

Proof: Apparent from the above discussion. □

Definition 8 *For any coded computation history $K^*(\rho, S)$, $\delta \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ is defined by*

$$\delta(s, s_1, s_2) \quad \textit{iff} \quad K^*(\rho, S)(s) \neq \tau \wedge \exists w : s_1 = sw0 \ \wedge s_2 = sw1$$
$$\wedge \forall w' : (\neg(sw < w')) \supset K^*(\rho, S)(w') = \tau$$

s_1 and s_2 are called the proper left and right descendants of s.

For A-labelled trees δ clearly reduces to the normal ancestral relation. Our equivalence of nodes in coded computation history/A-labelled trees is essentially the equivalence on trees of [3]:

Definition 9 *The relation $\simeq_A \subseteq \{0,1\}^* \times \{0,1\}^*$ is the maximal relation such that $s' \simeq_A s''$ whenever $K^*(\rho, S)(s') = K^*(\rho, S)(s'')$ and*

1. $\delta(s', s'_1, s'_2) \rightarrow \exists s''_1, s''_2 : \delta(s'', s''_1, s''_2)$ *with* $s'_1 \simeq_A s''_1$ *and* $s'_2 \simeq_A s''_2$
2. $\delta(s'', s''_1, s''_2) \rightarrow \exists s'_1, s'_2 : \delta(s', s'_1, s'_2)$ *with* $s''_1 \simeq_A s'_1$ *and* $s''_2 \simeq_A s'_2$

Thus, $\simeq_A$ identifies two trees labelled by alphabets containing A if their ancestral information w.r.t. A is the same.

Inherent in the definition of $\simeq_A$ is a functional $\mathcal{E}_{\mathcal{F}}$ on binary relations on $\{0,1\}^*$. This functional is seen to be monotonic on the lattice $(\mathbf{2}^{\{0,1\}^* \times \{0,1\}^*}, \subseteq)$; $\simeq_A$ is its maximal fixed point and therefore the union of all post-fixed points of $\mathcal{E}_{\mathcal{F}}$. Thus the proof that this is a well-defined equivalence relation is analogous to the proof for the observational equivalence of [6].

We now formally state the correspondence between trees and coded computation histories. The root of a tree is equivalent to the first node not labelled by τ in the coded computation history, namely the node 11:

Theorem 3 *For any A-labelled tree t and any of its corresponding coded computation histories $K^*(H(t\hat{\ }r))$ where r is some run of t we have that*

$$\epsilon_t \simeq_A 11_{K^*(H(t\hat{\ }r))}$$

Proof: It is enough to show that the set of pairs of corresponding proper descendants of ϵ_t and $11_{K^*(H(t\hat{\ }r))}$, i.e. the least set D such that

$$\begin{aligned} D \quad = \quad & \{\epsilon_t, 11_{K^*(H(t\hat{\ }r))}\} \cup \\ & \{(s'_1, s''_1) \mid \exists (s', s'') \in D. \exists (s'_2, s''_2). \delta(s', s'_1, s'_2) \wedge \delta(s'', s''_1, s''_2)\} \cup \\ & \{(s'_2, s''_2) \mid \exists (s', s'') \in D. \exists (s'_1, s''_1). \delta(s', s'_1, s'_2) \wedge \delta(s'', s''_1, s''_2)\} \end{aligned}$$

is a post-fixed point of the above-mentioned $\mathcal{F}$. So take any $({s'}_i, {s''}_i) \in D$. By the isomorphism H and the definition of K^* we have that $t(s) = K^*(H(t\hat{}r))(K^*(H(s)))$; it is immediately seen that the pairs of proper descendants also belong to D. $\Box$

If we extend the definition of $\simeq_A$ to tree languages over alphabets including A as

$$L_1 \simeq_A L_2 \overset{\text{def}}{\Longleftrightarrow} \begin{cases} \forall t \in L_1 \exists t' \in L_2 . t \simeq_A t' \\ \forall t' \in L_2 \exists t \in L_1 . t' \simeq_A t \end{cases}$$

we get from Theorems 2 and 3 that

Corollary 1 *If L is $SnS_{P_{\alpha_1},\ldots,P_{\alpha_m}}$-definable, there exists a $CML_{P_{\alpha_1},\ldots,P_{\alpha_m}}$-definable language L' such that $L \simeq_A L'$.*

5 Concluding remarks

We have in this note established that a modal mu-calculus CML defines tree languages up to an equivalence, $\simeq_A$. Readers feeling uneasy about the extra label τ can think of K^* as defining a *partial* labelling function; the labelling predicate P_τ should then be seen as a definedness predicate, and $\simeq_A$ should therefore be seen as a 'Kleene equality' on such partially labelled trees.

The above results show that the modal mu-calculus CML defines the Rabin tree languages up to the equivalence $\simeq_A$. Readers feeling uneasy about the extra label τ can think of K^* as defining a *partial* labelling function; the labelling predicate P_τ should then be seen as a definedness predicate, and $\simeq_A$ should therefore be seen as a 'Kleene equality' on such partially labelled trees.

Is CML then more useful for describing program properties than SnS ? We can think of CML as a logic describing the behaviour of a process system with a fixed number of asynchronous components, the idea of an equivalence identifying only a number of labels makes even more sense. For we can label nodes (= process states) according to which actions they can perform; then $\simeq_A$ describes equivalence w.r.t. to the action capabilities that we are interested in, namely those found in A. Moreover, given the modality of CML and the fixed-point construct it is straightforward to describe properties as e.g. the repeated enabling of one and the same component or a particular desired sequence of component activations.

6 Acknowledgments

I would like to thank Colin Stirling for many illuminating discussions on the contents of this paper. I would also like to thank Kees Goossens for a careful proofreading of the paper.

References

[1] Amir, A., Gabbay, D. *Preservation of Expressive Completeness in Temporal Models*, Inf. Contr. 72(1987), 66-83.

[2] Ben-Ari, M., Manna, Z., Pnueli, A., *The Temporal Logic of Branching Time*, Acta Informatica (20)1983, pp. 207-226.

[3] Browne, M.C., Clarke, E.M., Grümberg, O., *Characterizing Kripke Structures In Temporal Logic*, Tech. Rep. CMU-CS-87-104, Carnegie-Mellon University 1987.

[4] Hafer, T., Thomas, W., *Computation Tree Logic CTL^* and Path Quantifiers In The Monadic Theory Of The Binary Tree*, Proc. 11th Int. Coll. on Autom. Lang. and Prog., Springer LNCS 267, pp. 269-279.

[5] Kozen, D., *Results On The Propositional μ-Calculus*, Theoret. Comp. Sci., 27(1983), pp. 333-354.

[6] Milner, R., *Communication and Concurrency*, Prentice-Hall 1989.

[7] Niwinski, D., *Fixed Points vs. Infinite Generation*, Proc. 3rd Symp. on Logic in Comp. Sci., Edinburgh 1988.

[8] Rabin, M.O., *Decidability of Second-order Theories and Automata on Infinite Trees*, Trans. Amer. Math. Soc. 141(1969), pp. 1-35.

[9] Street, R.S., Emerson, E.A., *The Propositional Mu-calculus Is Elementary*, Proc. 11th Int. Coll. on Autom. Lang. and Prog., Springer LNCS 172, pp. 465-472.

[10] Wolper, P. *Temporal Logic Can Be More Expressive*, Inform. Control, 56(1983), pp. 72-99.

[11] Wolper, P., Vardi, M.Y., Sistla, A.P., *reasoning About Infinite Computation Paths*, Proc. 24th IEEE Symp. on Found. of Comput. Sci., pp. 185-194.