UNIVERSITÄT STUTTGART
FAKULTÄT INFORMATIK

Proceedings ASMICS Workshop
Infinite Traces
Tübingen, January 23 - 25, 1992

Volker Diekert, Werner Ebinger
(Editors)

Bericht 4/92

On logical definability of ω-trace languages †

Werner Ebinger
Universität Stuttgart
Institut für Informatik
Breitwiesenstr. 20–22
D 7000 Stuttgart 80
ebinger@informatik.uni-stuttgart.de

April 1992

Abstract

Our main result is the equivalence of monadic second order logic and recognizability for languages of infinite traces. This is a generalization of the work of Thomas [Tho90b]. We propose a logical characterization that is independent of any special sort of trace automata. For another approach we use standard constructions and Büchi asynchronous cellular automata defined by Gastin and Petit [GP91].

1 Introduction

The theory of ω-trace languages is a generalization of the theory of ω-word languages [Tho90b, for an overview] and of the theory of trace languages [Maz77]. Both theories are well studied. ω-trace languages need some extra investigation, because there are new difficulties. The basic definitions of trace languages are explained well in other papers in

†This research has been supported by the EBRA working group No. 3166 ASMICS.

these proceedings. So we only mention some important notions in this section.

In the second section we will give a short introduction to monadic second order logic. In the third section we will prove equivalence of monadic second order logic and recognizability for ω-trace languages. This is a generalization of the corresponding results for ω-words [Tho90a, for an overview] and finite traces [Tho90b]. We will provide a new proof that does not depend on any special sort of automata for infinite traces. In the fourth section we will recall the definition of the Büchi asynchronous automata of Gastin and Petit [GP91] and give a more classical proof of the equivalence of monadic second order logic and automata for ω-trace languages. In the fifth section we want to give some remarks on first order logic on ω-traces and star-free sets. Similar ideas have been proposed independently by Hoogeboom and Thomas. However, they never were worked out extensively.

We assume that the reader is familiar with the following notions of trace languages that can be found in other papers in this proceedings. In order to extend the free partially commutative monoid $\mathbb{M}(\Sigma, D)$ [Maz77, Die90] over a dependence alphabet (Σ, D) to the infinite case, it is convenient to consider dependence graphs. In the monoid $\mathbb{G}(\Sigma, D)$ of infinite dependence graphs we consider the subset $\mathbb{R}(\Sigma, D)$ of real traces where every node has a finite past, and the submonoid $\mathbb{C}(\Sigma, D)$ which is the quotient of $\mathbb{G}(\Sigma, D)$ by the coarsest congruence which preserves real traces [Die91]. We will use $\mathbb{M}$ and $\mathbb{R}$ as abbreviations for $\mathbb{M}(\Sigma, D)$ and $\mathbb{R}(\Sigma, D)$.

A language of real traces is called *recognizable* if it is recognized by some morphism $\eta : \mathbb{M} \to S$ into a finite monoid. A real trace language $R \in \mathbb{R}$ is recognized by η if for any sequence (r_i), $r_i \in \mathbb{M}$ we have

$$r_1 r_2 r_3 \ldots \in R \Rightarrow \eta^{-1}(\eta(r_1))\, \eta^{-1}(\eta(r_2))\, \eta^{-1}(\eta(r_3)) \ldots \subseteq R$$

The connection between recognizability for infinite traces and infinite words is given by [Gas91]

$$R \in \mathrm{Rec}(\mathbb{R}) \iff \varphi^{-1}(R) \in \mathrm{Rec}(\Sigma^\infty),$$

where $\varphi : \Sigma^\infty \to \mathbb{R}(\Sigma, D)$ is the extension of the canonical mapping to the infinite case.

2 Monadic second order logic over ω-traces

We can represent a real trace $t \in \mathbb{R}(\Sigma, D)$ as a labeled partial order $(V, <, \ell)$. Logical *formulae* can be defined over a *structure* with *signature* $(V, <, (P_a)_{a\in\Sigma})$. We use first order variables x, y, z, ... ranging over V and monadic second order variables X, Y, Z, ... ranging over $\mathcal{P}(V)$. Formulae are defined inductively:

- *Predicates:* First order predicates of the form $x < y$, $P_a(x)$ and monadic second order predicates of the form $X(x)$ are formulae for arbitrary variables x, y, X and all $a \in \Sigma$.

- *Logical operations:* If φ and ψ are formulae, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, and $(\neg\varphi)$ are formulae, too.

- *Quantifiers:* If φ is a formula, then $\exists x\varphi$, $\forall x\varphi$, $\exists X\varphi$, and $\forall X\varphi$ are formulae, too.

This monadic second order logic is called MSOL. We will also write $z \in X$ instead of $X(z)$ and freely use abbreviations like $X \subseteq Y$. Of course, we can also express $x = y$ and $X = Y$ in MSOL.

Example 1 We can express the property "t begins with b" by the formula

$$\exists x(P_b(x) \wedge \forall y(x = y \vee x < y)).$$

$\square$

In monadic second order the power of the logic does not depend on the difference between the $<$-relation and the edge relation of the Hasse diagram of $(V, <)$, which constitutes some kind of successor relation. The edge relation E of the Hasse diagram of $(V, <)$ is expressible by $<$ (even in first order logic):

$$x\ E\ y \quad \text{iff} \quad x < y \wedge \neg\exists z(x < z \wedge z < y),$$

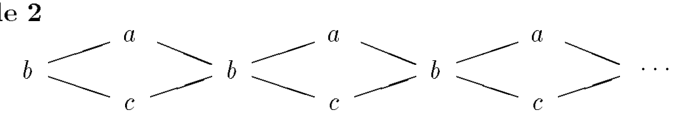
and vice versa (in monadic second order logic):

$$x < y \quad \text{iff} \quad \neg x = y \wedge$$

$$\forall X(x \in X \wedge \underbrace{\forall z \forall z'(z \in X \wedge z\ E\ z' \rightarrow z' \in X)}_{\substack{\text{If a node is in } X\text{, then} \\ \text{all } <\text{-greater nodes are in } X\text{, too.}}} \rightarrow y \in X)$$

Now we are free to use both, the $<$-relation and the edge relation E in monadic second order formulae.

Example 2



Traces of the language $T = (bac)^\omega$ over the dependence alphabet D : $a — b — c$ are described by the following formula. The parts in quotes can be easily replaced by a proper formula (even in first order).

"There is exactly one minimal node, labeled b."
$\wedge$ "Every b-node has two successors, labeled a and c"
$\wedge$ "Every b-node except the minimal one has two predecessors, labeled a and c"
$\wedge$ "Every a-node has exactly one predecessor and one successor, both labeled b"
$\wedge$ "Every c-node has exactly one predecessor and one successor, both labeled b"

□

In the fifth section we will go from formula to automata. These constructions can be simplified if we first reduce the logic MSOL to the *restricted monadic second order logic* MSOL_0. The logic MSOL_0 is the logic MSOL with the restriction, that there are no first order variables. Instead of first order predicates we use only second order predicates of the form $X_i \subseteq X_j$, $X_i < X_j$, and $X_i \subseteq P_a$. Here $X_i < X_j$ means X and Y are singleton sets and the element of X is smaller than the element of

Y. As formulae grow really huge using only these basic predicates, we give the following abbreviations:

$$\begin{array}{rcl} X = Y & \text{for} & X \subseteq Y \wedge Y \subseteq X \\ X \neq Y & \text{for} & \neg X = Y \\ Sing(X) & \text{for} & \exists Y(\overbrace{Y \subseteq X \wedge Y \neq X}^{\text{"proper subset"}} \wedge \\ & & \underbrace{\neg \exists Z(Z \subseteq X \wedge Z \neq X \wedge Z \neq Y)}_{\text{"exactly one"}}) \end{array}$$

We now describe how to eliminate first order variables by giving an example. A formula like

$$\underline{\forall x}\ \exists Y(x \in Y \wedge \underline{\exists z}(x < z \wedge P_a(z))$$

is expressed as

$$\underline{\forall X(Sing(X) \rightarrow}\ \exists Y(X \subseteq Y \wedge \underline{\exists Z(Sing(Z) \wedge}\ X < Z \wedge Z \subseteq P_a)).$$

3 Equivalence of recognizability and logic

In this section we prove the equivalence of recognizability and logic. Our proof is independent of any special automata model for (infinite) traces. In the proof we will use *c-rational trace languages* [Och85, GPZ91] (also called co-rational). c-rational languages are defined inductively:

- All *finite trace languages* $A \subseteq \mathbb{M}$ of finite traces are c-rational.
- If $A \subseteq \mathbb{M}$, $B \subseteq \mathbb{M}$, and $C \subseteq \mathbb{R}$ are c-rational, then the *concatenation* $A\,B$ and $A\,C$ is c-rational.
- If $A \subseteq \mathbb{M}$ is c-rational, then the concurrent iteration A^{c*} (called coiteration by Ochmansky) and the concurrent ω-iteration $A^{c\omega}$ are c-rational. For the concurrent iteration independent (connected) parts are iterated concurrently. This means A^{c*} ($A^{c\omega}$) in the usual star (omega) iteration over $C(A) = \{\ t \in \mathbb{M} \mid t$ is a connected component of some $s \in A\ \}$.

We can also use a constraint version of c-iteration were only connected parts may be iterated. This gives the same class of c-rational languages.

Theorem 3 *A trace language $T \subseteq \mathbb{R}(\Sigma, D)$ is recognizable if and only if T is definable in monadic second order logic.*

Proof: The following figure shows what we have to prove. $\bigcup T$ denotes the union of all traces $t \in T$, that is the word language that contains all members of traces in T. For the canonical mapping $\varphi : \Sigma^\infty \to \mathbb{R}$ we have $\bigcup T = \varphi^{-1}(T)$.

$$\begin{tikzcd}
 & \bigcup T \text{ is definable in MSOL} \arrow[r, Leftrightarrow, "\text{(Büchi)}"] & \bigcup T \text{ is recognizable} \arrow[dr, Leftrightarrow, "\text{[Gas91]}"] & \\
T \text{ is definable in MSOL} \arrow[ur, Rightarrow, "(1)"] & & & T \text{ is recognizable} \arrow[dl, Leftrightarrow, "\text{[GPZ91]}"] \\
 & & T \text{ is c-rational} \arrow[ull, Rightarrow, "(2)"] &
\end{tikzcd}$$

(1): We can not use the same formula for both, traces and words, as the underlying interpretation of the order is different. For example the formula

$$\exists x, y, z\ P_a(x) \wedge P_c(y) \wedge P_b(z) \wedge x < z \wedge y < z \wedge (x < y \vee y < x)$$

over the dependence alphabet $D:\quad a — b — c$ is not true for the trace acb, but it is true for both words in this trace. But it suffices to express the partial order on the traces by the linear order on the words. This can be done by one single first order formula, but in order to render it easier to follow, we proceed in two steps. We can first express the partial order by the edge relation E of the Hasse diagram of the dependence graph. This was described in the section on monadic second order logic. Then we express the edge relation E by the linear order on the words: $x\ E\ y$ if and only if

$$\bigvee_{(a,b)\in D} (Q_a(x) \wedge Q_b(y) \wedge x < y$$

$$\wedge\ \forall z\ (x < z \wedge z < y \to \bigwedge_{\substack{c \text{ with} \\ (a,c)\in D \text{ or} \\ (b,c)\in D}} \neg Q_c(z))).$$

Now if a trace t satisfies a formula φ, then all words w in the trace satisfy the new formula φ', which we obtained by replacing the predicates $x\ E\ y$ by the formula above. And vice versa if a word $w \in t$ satisfies the new formula φ', then all words $w' \in t$ in the same trace satisfy this new formula φ', and the trace t containing this word w satisfies the original formula φ. We can formalize this using $\models$ as abbreviation of "is a model for"

$$t \models \varphi \iff \exists w \in t\ w \models \varphi' \iff \forall w \in t\ w \models \varphi'.$$

(2): For this implication of the diagram we perform an induction over the construction of concurrent rational (c-rational) expressions.

- A is a finite set of traces: We give a formula φ_t for every single word in A and combine them in a disjunction
$$\varphi_A = \bigvee_{t\in A} \varphi_t.$$
- $A \cup B$ for c-rational sets A and B: Combine the formulae φ_A and φ_B for A and B to
$$\varphi_{A\cup B} = \varphi_A \vee \varphi_B.$$
- $A \cdot B$ for c-rational sets A and B: For the formulae φ_A and φ_B we define formulae with restricted quantification. $\varphi_A|_X$ for some set X is the formula φ_A where we replace every subformula $\exists x\psi$ by $\exists x\ (x \in X \wedge \psi)$, $\forall x\psi$ by $\forall x\ (x \in X \to \psi)$, $\exists Y\psi$ by $\exists Y\ (Y \subseteq X \wedge \psi)$, and $\forall Y\psi$ by $\exists Y\ (Y \subseteq X \to \psi)$. The formula $\varphi_{A\cdot B}$ is defined as
$$\begin{array}{ll}
\begin{array}{l}\text{possible}\\ \text{max./min.}\\ \text{elements}\\ \text{of } A/B\end{array} & \displaystyle\bigvee_{\substack{\{a_1,\ldots,a_k\}\\ \subseteq \Sigma}} \bigvee_{\substack{\{b_1,\ldots,b_l\}\\ \subseteq \Sigma}} \exists x_{a_1}\ldots\exists x_{a_k}\exists x_{b_1}\ldots\exists x_{b_l} \\
\text{incomparable} & \displaystyle\Bigg(\bigwedge_{\substack{1\le i\le k\\ 1\le j\le k\\ i\ne j}} (\neg\, x_{a_i} < x_{a_j} \wedge \neg\, x_{b_i} < x_{b_j}) \wedge
\end{array}$$

nothing between A and B

$$\bigwedge_{\substack{1 \leq i \leq k \\ 1 \leq j \leq l}} \neg \exists z \, (x_{a_i} < z \wedge z < x_{b_j}) \wedge$$

every node belongs to a trace

$$\forall z \bigvee_{\substack{1 \leq i \leq k \\ 1 \leq j \leq l}} \Big(z < x_{a_i} \vee z = x_{a_i} \vee x_{b_j} < z \vee x_{b_j} = z \Big) \wedge$$

$$\varphi_A|_{\{x | \bigvee_{1\leq i\leq k} x \leq x_{a_i}\}} \wedge \varphi_B|_{\{x | \bigvee_{1\leq j\leq l} x_{b_j} \leq x\}} \Big)$$

- A^{c*}, $A^{c\omega}$ for a c-rational set A, where only connected parts are iterated: The logical definability of A^{c*} was already claimed by Thomas [Tho90b]. We define $\varphi_{A^{c*}}$ and $\varphi_{A^{c\omega}}$ as

$$\exists X_{min} \exists X_{max}$$

For every node x find borders of subtrace

$$\forall x \Bigg(\bigvee_{\substack{\{a_1,\ldots,a_k\} \\ \subseteq \Sigma}} \bigvee_{\substack{\{b_1,\ldots,b_l\} \\ \subseteq \Sigma}} \exists x_{a_1} \ldots \exists x_{a_k} \exists x_{b_1} \ldots \exists x_{b_l}$$

$$\Big(x_{a_1} \in X_{min} \wedge \ldots \wedge x_{a_k} \in X_{min} \wedge$$

starting/ending nodes of the subtrace are incomparable and maximal such sets

$$x_{b_1} \in X_{max} \wedge \ldots \wedge x_{b_l} \in X_{max} \wedge$$

"$x_{a_1} \ldots x_{a_k}$ are a maximal antichain in X_{min} ($x_{a_1} \ldots x_{a_k}$ are incomparable and there's no node in X_{min}, that is incomparable to $x_{a_1} \ldots x_{a_k}$)" $\wedge$

"$x_{b_1} \ldots x_{b_l}$ are a maximal antichain in X_{max}" $\wedge$

subtrace around x well bounded

$$\bigwedge_{1\leq i\leq k} x_{a_i} \leq x \wedge \bigwedge_{1\leq j\leq l} x \leq x_{b_j} \wedge \neg \exists z$$

$$\Big(\big((z < x \wedge \bigvee_{1\leq i\leq k} x_{a_i} < z) \vee (x < z \wedge \bigvee_{1\leq j\leq l} z < x_{b_j}) \big) \wedge$$

$$(z \in X_{min} \vee z \in X_{max}) \Big) \wedge$$

Apply φ_A to subtrace

$$\varphi_A|_{\geq\{x_{a_i}\}, \leq\{x_{b_j}\}} \Big) \Bigg) \wedge$$

for A^{c*}:

$$\forall x (\text{“}x \text{ minimal in the whole trace”}) \rightarrow x \in X_{min}) \wedge$$

$$\forall x (\text{“}x \text{ maximal in the whole trace”}) \rightarrow x \in X_{max}) \wedge$$

$$\bigvee_{1\leq i\leq|\Sigma|} (\exists x_1 \ldots \exists x_i \, \forall x \bigvee_{1\leq j\leq i} x \leq x_j)$$

for $A^{c\omega}$:

$$\forall x \exists y \; x < y$$

□

4 Büchi Asynchronous Cellular Automata and equivalence to logic

Büchi asynchronous automata and Büchi asynchronous cellular automata were defined by Gastin and Petit [GP91]. These are generalizations of Büchi automata on infinite words and asynchronous (cellular) automata defined by Zielonka [Zie87]. The recognition power of Büchi asynchronous automata is equal to the recognition power of Büchi asynchronous cellular automata, like it is equal in the finite case for asynchronous automata and asynchronous cellular automata. In this paper we restrict ourselves to Büchi asynchronous cellular automata. The proofs are almost the same for Büchi asynchronous automata, and some details might be even easier.

Definition 4 *A Büchi asynchronous cellular automaton $\mathcal{A}$ is a tuple $\mathcal{A} = ((Q_a)_{a\in\Sigma}, (\delta_a)_{a\in\Sigma}, S, F, \mathcal{R})$ where Q_a is the local state set for process a, $\delta_a : \prod_{i\in D(A)} Q_i \rightarrow \mathcal{P}(Q_a)$ is the local transition function for process a, $S \subseteq \prod_{a\in\Sigma} Q_a$ is the set of starting states, $F \subseteq \prod_{a\in\Sigma} Q_a$ is the set of accepting states for finite traces and $\mathcal{R} \subseteq \prod_{a\in\Sigma} \mathcal{P}(Q_a)$ is the acceptance table for infinite traces.*

In order to define runs of such an automaton we consider a trace $t \in \mathbb{R}(\Sigma, D)$ as a *labeled partial order* $(V, <, \ell)$ with $\ell : V \rightarrow A$. A *run*

r of $\mathcal{A}$ on t is a function $r : V \cup \{v_0\} \to (\bigcup_{a\in\Sigma} Q_a) \cup S$, where v_0 is an additional $<$-minimal node v_0, that is labeled with a starting state $r(v_0) \in S$, and for all $v \in V$:

$$r(v) \in \delta_{\ell(v)}((r(v^i_{max}))_{i\in D(\ell(v))}),$$

where v^i_{max} is the $<$-maximal node labeled i before v. The set of *locally repeated states* is defined as

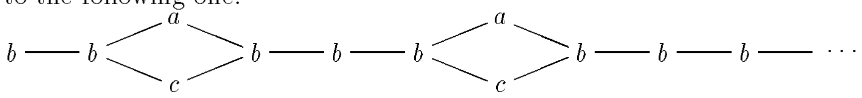
$$\begin{aligned} f_a(r) \quad &:= \quad \{\, q \in Q_a \mid \exists^\omega v : (\ell(v) = a \wedge r(v) = q) \\ &\qquad \vee\, \exists v(\neg\exists w(v < w \wedge \ell(w) = a) \wedge r(v) = q)\,\}. \qquad (1) \\ f(r) \quad &:= \quad \prod_{a\in\Sigma} f_a(r) \qquad (2) \end{aligned}$$

is the tuple of *repeated states*. The run r *accepts* t if $f(r(t)) \supseteq R$ for some $R \in \mathcal{R}$ or if the maximal nodes of a finite run form a state in F. The accepted trace language of $\mathcal{A}$ is

$$T(\mathcal{A}) = \{\, t \mid \text{there exists an accepting run } r\,\}.$$

Gastin and Petit [GP91] proved, that Büchi asynchronous cellular automata accept exactly the recognizable ω-trace languages. These languages are closed under intersection $\cap$, union $\cup$, complement $\bar{\ }$ and the concurrent star operation ^{c*}.

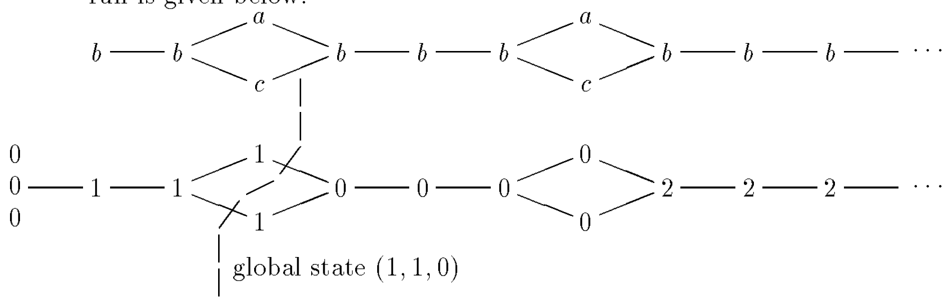
Example 5 The dependence graphs of traces in the language $T = (b^*bac)^*b^\omega$ with the dependence alphabet $D: \quad a - b - c \quad$ look similar to the following one.



An automaton that accepts T is $\mathcal{A} = ((Q_a)_{a\in\Sigma}, (\delta_a)_{a\in\Sigma}, S, F, \mathcal{R})$ with $Q_a = \{0,1\}$, $Q_b = \{0,1,2\}$, $Q_c = \{0,1\}$, and the transition function

$$\begin{array}{llll@{\qquad}llll} \delta_a: & (0,1) & \mapsto & \{1\} & \delta_b: & (0,0,0) & \mapsto & \{1,2\} \\ & (1,0) & \mapsto & \{0\} & & (0,1,0) & \mapsto & \{1\} \\ \delta_c: & (1,0) & \mapsto & \{1\} & & (1,1,1) & \mapsto & \{0,2\} \\ & (0,1) & \mapsto & \{0\} & & (1,0,1) & \mapsto & \{0\} \\ & & & & & (0,2,0) & \mapsto & \{2\} \\ & & & & & (1,2,1) & \mapsto & \{2\} \end{array}$$

There are two possibilities to define the acceptance table: $\mathcal{R} = \{(\{0\},\{2\},\{0\}),(\{1\},\{2\},\{1\})\}$ or $\mathcal{R}' = \{(\emptyset,\{2\},\emptyset)\}$. An example for a run is given below.



□

The equivalence of automata and logic is a consequence of the equivalence of recognizability and logic, since Büchi asynchronous cellular automata accept exactly the recognizable languages. In this section we will use the classical approach and provide an independent proof. We use standard techniques and some special constructions for Büchi asynchronous cellular automata. Since the case of finite traces has already been solved [Tho90b] we restrict ourselves to the case of infinite traces.

Theorem 6 *A trace language $T \subseteq \mathbb{R}(\Sigma, D)$ is accepted by some Büchi asynchronous cellular automaton if and only if T is definable in monadic second order logic.*

Proof: "⇒": Let $\mathcal{A} = ((Q_a)_{a\in\Sigma}, (\delta_a)_{a\in\Sigma}, S, F = \emptyset, \mathcal{R})$ with $Q_a = \{0,\ldots,m_a\}$ be a Büchi asynchronous cellular automaton which accepts T. Every trace t is considered as a structure $(V, <, \ell)$. A computation r corresponds to a starting state $s \in S$ and sets $Y_{a0}, Y_{a1}, \ldots, Y_{am_a}$ (for every $a \in \Sigma$), where $Y_{ai} = \{\, v \in V \mid \ell(v) = a \wedge r(v) = i \,\}$ contains all nodes that are labeled a and have the state i in the run r. The formula is given as follows.

$$\begin{array}{l} \text{existence} \\ \text{of a} \\ \text{computation} \end{array} \quad \bigvee_{s\in S} \ \exists Y_{a_1 0} \ldots \exists Y_{a_1 m_{a_1}}$$

$$\vdots$$

$$\exists Y_{a_n 0} \ldots \exists Y_{a_n m_{a_n}}$$

every node gets at most one state

$$\Big(\text{“disjunct sets”} \wedge$$

labeling according to transition table

$$\forall v \bigvee_{\substack{(p,\,a,\,q)\\ q\,\in\,\delta_a(p)}} \Big(P_a(v) \wedge v \in Y_{aq} \wedge$$

find preceding state

$$\bigwedge_{b \in D(a)} \Big(\exists w \; (P_b(w) \wedge w < v \wedge$$

$$\neg \exists u \; (w < u \wedge u < v \wedge P_b(u)) \wedge$$

$$Y_{b\pi_b(p)}(w)) \vee$$

or the starting state

$$(\neg \exists w \; (P_a(w) \wedge w < v) \wedge$$

$$\text{“}\pi_b(s) = \pi_b(p)\text{”}) \Big) \Big) \wedge$$

choose an element of the acceptance table

$$\bigvee_{R = \prod_{a \in \Sigma} R_a \in \mathcal{R}} \; \bigwedge_{a \in \Sigma} \; \bigwedge_{\substack{i=1\\ R_a = \{q_1, \ldots, q_l\}}}^{l}$$

infinite repetition

$$\Big(\exists X (\text{“}X \text{ is infinite.”} \wedge X \subseteq Y_{aq_i}) \vee$$

only finitely many a

$$\exists v \; v \in Y_{aq_i} \wedge \neg \exists w \; (v < w \wedge P_a(w)) \Big) \Big)$$

“$\Leftarrow$”: Let φ be a second order formula that is in the restricted form of MSOL_0 . Without loss of generality we have to consider predicates $X \subseteq Y$, $X < Y$, and $X \subseteq P_a$. We are going to perform induction over the construction of formulae. As there appear formulae with free variables in this induction we have to consider formulae $\varphi(X_1, \ldots, X_n)$ with free variables $X_1, \ldots, X_n$ over the dependence alphabet $(\Sigma \times \{0,1\}^n, D_n)$ with

$$(a, \cdot) \; D_n \; (b, \cdot) \quad \text{iff} \quad a \; D \; b.$$

Induction foundation: We are working with formulae $\varphi(X, Y)$ with two free variables X, Y. In this case we use an alphabet $\Sigma \times \{0,1\}^2$. The first additional digit indicates membership in X and the second additional digit indicates membership in Y. The generalization to arbitrary many free variables is easy. For all $x \in \Sigma$ we consider processes $x00$, $x01$, $x10$, and $x11$. That gives a number of $4|\Sigma|$ processes.

- $X \subseteq Y$: This is a very simple automaton with only one global state $0^{4|\Sigma|} \in S$. The processes $x00$, $x01$, and $x11$ continue and stay in the local state 0, if they read a node labeled $x00$, $x01$, and $x11$. These are nodes that are in Y if they are in X. The processes $x10$ stop, because they read nodes that are in X, but not in Y. The acceptance table is $\mathcal{R} = (\{0\}, \ldots, \{0\})$.

$$\begin{array}{rcll} \delta_{x00}(0^{D(x00)}) & = & \{0\} & \\ \delta_{x01}(0^{D(x00)}) & = & \{0\} & \\ \delta_{x11}(0^{D(x00)}) & = & \{0\} & \\ \delta_{x10}(0^{D(x00)}) & = & \emptyset & \quad \text{for all } x \in \Sigma \end{array}$$

- $X < Y$: Again we start out with the global starting state $0^{4|\Sigma|}$. The local state 0 indicates a process not having any information. 1 indicates a process suggesting that it has read the first node in X, 2 means that there was a node in X somewhere, and 3 stands for reading a node in Y after there has been a node somewhere in X. Every process $x00$ remains in state 0, if all dependent states are still in state 0. Every process $x10$ waits until it is active and then changes its local state to 1, if all dependent processes are still in state 0. Now the automaton has read a node that is in X but not in Y. Every process $x00$ that sees a 1 or 2 in a dependent state and is still in state 0 changes its own state to 2. In doing this they spread the news that there has been a node in X. A process $x01$ that sees a 1 or 2 in a dependent state changes its own state to 3, to indicate that it has read a node in Y that is $<$-greater than the first node that was in X. $\mathcal{R} = \{ \prod_{a \in \Sigma} \mid |R_a| = 1$ for all $a \in \Sigma$ and exactly once $R_a = \{1\}$ and exactly once $R_a = \{3\}$ $\}$.

$$\begin{aligned}
\delta_{x00}(0^{D(x00)}) &= \{0\} \\
\delta_{x00}(\underbrace{\{0,1,2\}^{D(x00)}}_{\text{some 1 or 2, } q_{x00}=0}) &= \{2\} \\
\delta_{x10}(0^{D(x00)}) &= \{1\} \\
\delta_{x01}(\underbrace{\{0,1,2\}^{D(x00)}}_{\text{some 1 or 2, } q_{x01}=0}) &= \{3\} \\
\delta_{x00}(\underbrace{\{0,1,2,3\}^{D(x00)}}_{q_{x00}=q}) &= \{q\} \quad q \in \{1,2,3\}
\end{aligned}$$

- $X \subseteq P_a$: Now we have only one free variable and we start from $0^{2|\Sigma|} \in S$. All processes $x0$ (for all $x \in \Sigma$) and the process $a1$ continue and stay in the local state 0. All other processes stop. The acceptance table is $\mathcal{R} = \{\{0\}, \ldots, \{0\}\}$.
$$\begin{aligned}
\delta_{x0}(0,\ldots,0) &= \{0\} \quad \text{for all } x \in \Sigma \\
\delta_{a1}(0,\ldots,0) &= \{0\}
\end{aligned}$$

Induction step:

- Closure under conjunction $\wedge$ and disjunction $\vee$: Cartesian product of automata [GP91].
- Closure under complement $\neg$: Büchi asynchronous automata recognize exactly the recognizable languages [GP91] and these are closed under complement.
- Closure under quantification (projection) of variables $\exists$ ($\forall$ may be expressed by $\neg$ and $\exists$): The automaton for a formula $\exists X \varphi(X)$ (we can easily add more free variables, that are not quantified in this step) over the alphabet $\Sigma \times \{0,1\}$ is built taking the automaton for $\varphi(X)$. We melt every pair of processes $x0, x1$. The new process x has state set $Q_x = Q_{x0} \times Q_{x1}$ and guesses nondeterministically whether a node is in X or not and then performs a transition either in the component Q_{x0} or in the component Q_{x1}. □

If we had a construction of a complement automaton, our proof would be completely constructive. A complementation construction similar to the standard complementation construction for Büchi automata does work. But we are looking for a more constructive complementation.

5 Star-free ω-trace languages and first order logic

In the case of finite traces [GRS91], finite words, and ω-words [Tho90a, for an overview] star-free languages are exactly the first order definable ones. In order to investigate star-free ω-trace languages we first have to give a definition of star-free ω-trace languages. One possibility is to use star-free expressions with only boolean operations and concatenation, where the complement can be taken with respect to A^* in order to get star-free sets of finite traces and with respect to A^* and A^ω in order to get star-free sets of infinite traces. We can perform a part of the proof of the equivalence of recognizability and logic in the case of star-free languages and first order logic:

$$\begin{tikzcd}
 & \bigcup T \text{ is definable in FO}[<] \arrow[r, Leftrightarrow, "\text{Ladner, Thomas}"] & \bigcup T \text{ is star-free} \arrow[d, Rightarrow, "(3')"] \\
T \text{ is definable in FO}[<] \arrow[ru, Rightarrow, "(1')"] & T \text{ is star-free} \arrow[l, Rightarrow, "(2')"] &
\end{tikzcd}$$

For the implication $(1')$ we have to express the $<$-relation on the traces directly by a first order formula using the $<$-relation on the words: $x < y$ in the partial order on the traces if and only if

$$\bigvee_{\substack{(a_1,a_2)\in D \\ (a_2,a_3)\in D \\ \vdots \\ (a_{l-1},a_l)\in D \\ \{a_1,\ldots,a_l\}\subseteq A \\ |\{a_2,\ldots,a_l\}|=l-1}} \exists x_2 \ldots \exists x_{l-1} \big(Q_{a_1}(x) \wedge Q_{a_2}(x_2) \wedge \ldots \ldots \wedge Q_{a_{l-1}}(x_{l-1}) \wedge Q_{a_l}(y) \wedge x < x_2 < \ldots < x_{l-1} < y\big)$$

For the implication ($2'$) we replace the boolean operations by the corresponding logical operations. For concatenation we can use the first order formula that we gave in the proof of equivalence of logic and recognizability. The remaining direction ($3'$) will be shown in a forthcoming joint work with V. Diekert and A. Muscholl.

Acknowledgements

The author wants to thank Volker Diekert, Paul Gastin, Hendrik Jan Hoogeboom, Anca Muscholl, Antoine Petit, and Wolfgang Thomas for many helpful discussions, corrections, and suggestions.

References

[Die90] V. Diekert. *Combinatorics on Traces.* Number 454 in Lecture Notes in Computer Science. Springer, Berlin-Heidelberg-New York, 1990.

[Die91] V. Diekert. On the concatenation of infinite traces. In Choffrut C. et al., editors, *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91), Hamburg 1991*, number 480 in Lecture Notes in Computer Science, pages 105–117. Springer, Berlin-Heidelberg-New York, 1991.

[Gas91] P. Gastin. Recognizable and rational trace languages of finite and infinite traces. In Choffrut C. et al., editors, *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91), Hamburg 1991*, number 480 in Lecture Notes in Computer Science, pages 89–104. Springer, Berlin-Heidelberg-New York, 1991.

[GP91] Paul Gastin and Antoine Petit. Asynchronous automata for infinite traces. Rapport de Recherche 707, Université de Paris-Sud, 1991. To appear in *ICALP 1992*, also this volume, pages 29–45.

[GPZ91] P. Gastin, A. Petit, and W. Zielonka. A Kleene theorem for infinite trace languages. In J. Leach Albert et al., editors, *Proceedings of the 18th International Colloquium on Automata Languages and Programming (ICALP'91), Madrid (Spain) 1991*, number 510 in Lecture Notes in Computer Science, pages 254–266. Springer, Berlin-Heidelberg-New York, 1991.

[GRS91] Giovanna Guaiana, Antonio Restivo, and Sergio Salemi. On aperiodic trace languages. In Choffrut C. et al., editors, *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91), Hamburg 1991*, number 480 in Lecture Notes in Computer Science, pages 76–88. Springer, Berlin-Heidelberg-New York, 1991.

[Maz77] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Och85] E. Ochmanski. Regular behaviour of concurrent systems. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 27:56–67, Oct 1985.

[Tho90a] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

[Tho90b] Wolfgang Thomas. On logical definability of trace languages. In V. Diekert, editor, *Proceedings of a workshop of the ESPRIT Basic Research Action No 3166: Algebraic and Syntactic Methods in Computer Science (ASMICS), Kochel am See, Bavaria, FRG (1989)*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990.

[Zie87] W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O.-Informatique Théorique et Applications*, 21:99–135, 1987.