M. Baclet

Logical Characterization of Aperiodic Data Languages

Research Report LSV–03–12, Sep. 2003

Laboratoire
Spécification
et
Vérification

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Logical characterization of aperiodic data languages

Manuel BACLET

LSV - CNRS UMR 8643 - ENS de Cachan
61, Av. du Président Wilson - 94235 Cachan Cedex - France
e-mail: baclet@lsv.ens-cachan.fr

Abstract. Data languages are an extension of timed languages, as defined by R. Alur and D. L. Dill, proposed by P. Bouyer, A. Petit and D. Thérien to provide a suitable theoretical framework, comparable to the formal languages theory. A class of data languages, naturally called recognizable data languages, can be defined in an equivalent way by automata, finite monoids and logic formulas.
The aim of this paper is to study the subclass of recognizable data languages which are defined by *aperiodic* monoids. Note that the corresponding class for formal languages plays a central role in the verification of untimed systems. For instance, a recognizable formal language is defined by a first order logic formula (or by a linear temporal logic formula) if and only if it is recognized by an aperiodic monoid. We propose extensions of this deep result to data languages framework.

Keywords: Real-time systems, Timed languages, Formal verification

Introduction

Timed languages Timed automata, introduced by R. Alur and D. L. Dill (see, for example, [AD94]), are a suitable model for describing and verifying timed systems. They have been widely studied and several verification tools, such as Uppaal ([BL96] and [LPY97]) or Kronos ([DOTY96] and [Yov97]), based on them have been developped. Those tools have been successfully used to verify (or to find bugs) in real protocols ([BBP03],[TY98],[HSLL97],[BF99]). However, the theory of timed languages is not as satisfying as the theory of formal languages: it lacks an algebraic characterization of recognizable timed languages and the Kleene theorem has no simple counterpart, even if several extensions have been proposed ([ACM02],[BP02]).

Data languages In order to fill those gaps, P. Bouyer, A. Petit and D. Thérien proposed a generalization of timed languages: the *data languages* [BPT03]. A data word is a finite sequence of pairs (a, d) where a belongs to a finite alphabet and d to a set, the data set, which can be a time domain or anything else. They proposed both notions of automata and monoid mechanisms that lead to the definition of recognizable data languages similar to the definition of regular languages.
In [Bou02], P. Bouyer proposed a characterization of recognizable data languages by a second-order monadic logic with data predicates. Hence, like in the case of formal languages, recognizable data languages can be equivalently defined by means of automata, monoids and logics. Figure 1 summarizes the equivalences obtained for data languages, compared with those existing for formal languages.

Aperiodic monoids In the framework of formal languages, important subclasses of regular languages can be equivalently characterized in terms of monoids, logics, automata and/or rational expressions. It is possible to characterize the first order fragment: a formal language can be defined by a first order formula if and only if it can be recognized by an aperiodic monoid ([Sch65] and [NP71]). This deep result combined with a result of H. Kamp [Kam68] gives an algorithm to decide whether a regular language can be defined by a Linear Temporal Logic (LTL, [Pnu77]) formula. This algorithm is fundamental in the theory of formal verification.
The aim of the paper is to prove a result on data languages similar to the equivalence between first-order definable regular languages and languages recognized by aperiodic monoids. Figure 2 summarizes the results concerning the «first-order» fragments of the theories of formal and data languages.

Formal languages	Data languages
Finite automata	Data automata [BPT03]
Rational expressions	???
MSO	«second order logic» [Bou02]
Monoid morphisms	Monoid mechanisms [BPT03]

Fig. 1.

Outline We start by recalling the basic definitions of data languages and we describe data automata, monoid mechanisms and logics that allow to define equivalently the class of recognizable data languages. The rest of the paper is divided into two parts. The first one is a proof of a logical characterization of a fragment of the data languages that can be recognized by an aperiodic monoid and the second one, which is essentially a technical corollary of the first result, is devoted to a characterization of the class of data languages that can be recognized by an aperiodic monoid.

Formal languages	Data languages
Counterfree automata	???
Star free languages	???
FO/LTL	«first order logic»
Aperiodic monoids	Aperiodic monoids

Fig. 2.

Due to a lack of place, two proofs are written in the appendix.

1 Recognizable data languages

In the following, Σ is a finite alphabet and $\mathcal{D}$ is a possibly infinite set, the *data* set. Among the data, we distinguish a special initial value, denoted by $\perp$. A *data word* is an element of $(\Sigma \times \mathcal{D})^*$ and a *data language* is a set of data words.

Remark 1. Choosing a time domain for $\mathcal{D}$, such as $\mathbb{Q}_+$ or $\mathbb{R}_+$, and setting $\perp$ to 0 bring us back to the case of timed languages: data languages extend timed languages.

Example 1. The language $\{(a,\tau)(a,2\tau)...(a,n\tau) \mid \tau > 0, n \in \mathbb{N}\}$ is a data language over the data set $\mathcal{D} = \mathbb{R}$.

If k is an integer, an *update* up over $\mathcal{D}^k$ is a subset of $\{1..k\}$. If $\theta = (\theta_i)_{1 \leqslant i \leqslant k} \in \mathcal{D}^k$ and $d \in \mathcal{D}$, we denote $up(\theta, d)$ the tuple θ' defined by:

$$\theta'_i = \begin{cases} d \text{ if } i \in up \\ \theta_i \text{ otherwise.} \end{cases}$$

The set of all updates is denoted by $\mathcal{U}$.
Recognizable data languages can be equivalently defined in (at least) three ways([BPT03] and [Bou02]). This is detailed in the following subsections.

1.1 Data automata

In the framework of formal languages, regular languages are defined by using a notion of automata. It is possible to define a notion of data automata that recognize a class of data languages: the *recognizable* data languages. This can be found in [BPT03].

Definition 1. *A data automaton over Σ and $\mathcal{D}$ is a tuple $\mathcal{A} = (Q, q_0, F, k, \sim, T)$ where:*
- *Q is a finite set of states,*
- *$q_0 \in Q$ is the initial state,*
- *$F \subseteq Q$ is the set of final states,*
- *$k \in \mathbb{N}$ is the number of registers,*
- *$\sim$ is an equivalence relation of finite index defined on $\mathcal{D}^k$, and*
- *$T \subset (Q \times \mathcal{D}^k/\sim \times \Sigma \times \mathcal{U} \times \mathcal{D}^k/\sim \times Q)$ is a finite set of transitions ($\mathcal{U}$ is a set of updates).*

A data word $(a_1, d_1) \ldots (a_n, d_n)$ is accepted by the data automaton $\mathcal{A}$ if there exists a path in $\mathcal{A}$

$$q_0 \xrightarrow[d_1]{g_1, a_1, up_1, g'_1} q_1 \xrightarrow[d_2]{g_2, a_2, up_2, g'_2} q_2 \quad \ldots \quad q_{n-1} \xrightarrow[d_n]{g_n, a_n, up_n, g'_n} q_n$$

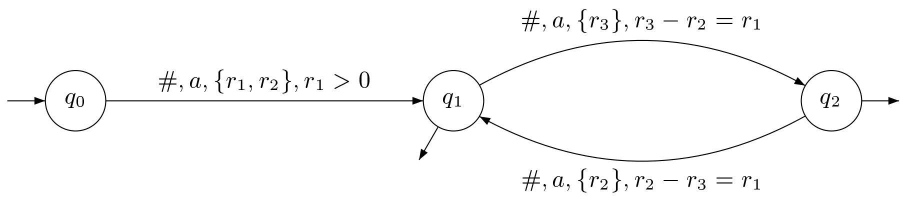
such that the sequence $(\theta_i)_{i=0\ldots n}$ defined by

$$\theta_0 = \bot^k \quad \text{and} \quad \theta_{i+1} = up_{i+1}(\theta_i, d_{i+1})$$

satisfies $\overline{\theta_{i-1}} = g_i$ for $1 \leq i \leq n$, $\overline{\theta_i} = g'_i$ for $1 \leq i \leq n$ and $q_n \in F$.
The set of data words that are accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

Definition 2. *A data languages is recognizable if it is the set of words accepted by a data automaton.*

Example 2. The data language $\{(a,\tau)(a,2\tau)...(a,n\tau) \mid \tau > 0, n \in \mathbb{N}\}$ is recognized by no classical timed automaton [DZ98] but is recognized by the following data automaton (where $\#$ is a shortcut to represent any equivalence class):



We now describe a run of the automaton over the data word $(a,\pi)(a,2\pi)(a,\sqrt{2})$:

State	q_0	$\xrightarrow[\pi]{a}$	q_1	$\xrightarrow[2\pi]{a}$	q_2	$\xrightarrow[\sqrt{2}]{a}$	_
Values of the registers	$\begin{pmatrix}0\\0\\0\end{pmatrix}$		$\begin{pmatrix}\pi\\\pi\\0\end{pmatrix}$		$\begin{pmatrix}\pi\\\pi\\2\pi\end{pmatrix}$		$\begin{pmatrix}\pi\\\sqrt{2}\\2\pi\end{pmatrix}$
Guard	_		$r_1 > 0$		$r_3 - r_2 = r_1$		$r_2 - r_3 \neq r_1$

Remark 2. Any timed language recognizable by a classical timed automaton can be recognized by a data automaton when viewed as a data language ([BPT03]). This result combined with the result stated in example 2 shows that the class of timed languages recognized by a classical timed automaton is a strict subset of the class of timed languages recognized by a data automaton.

1.2 Monoid mechanisms

In the framework of formal languages, a language $L \subset \Sigma^*$ is regular if and only if there exists a finite monoid M, a subset F of M and a monoid morphism $\Phi : \Sigma^* \to M$ such that $L = \Phi^{-1}(F)$. We now describe a simular notion that has been proposed, in [BPT03], for data languages.

Definition 3. *A k-register (non deterministic) mechanism over a finite monoid M is a tuple $\rho = [(Up_{m,a})_{m\in M, a\in\Sigma}, \sim, \phi, F]$ where:*
- *for all $(m,a) \in M \times \Sigma$, $Up_{m,a}$ is a set of updates,*
- *$\sim$ is an equivalence relation of finite index over $\mathcal{D}^k$,*
- *ϕ is a monoid morphism from $(\Sigma \times \mathcal{D}^k/\sim)^*$ into M, and*
- *F is a subset of M, the set of the accepting elements of M.*

Computation in a mechanism A computation of $\rho = [(Up_{m,a})_{m \in M, a \in \Sigma}, \sim, \phi, F]$ over the data word $u = (a_1, d_1)...(a_n, d_n)$ is done in the following way, where $\overline{\theta}$ refers to the class of θ modulo $\sim$:

% Initialisation
    $m := 1$
    $\forall 1 \leqslant j \leqslant k, \theta[j] := \perp$
% Computation
    For $i := 1$ to n do
        Choose $up \in Up_{m,a_i}$
        $\theta := up(\theta, d_i)$
        $m := m.\phi(a_i, \overline{\theta})$
    Endfor
% Result
    If $m \in F$ return yes
    Else, return no

We say that a word u is *recognized* by the mechanism if and only if there exists a computation over u that returns «yes».

Definition 4. *Let L be a data language over Σ and $\mathcal{D}$ and let M be a finite monoid. We say that M recognizes L if there exists a k-register mechanism $\rho = [(up_{m,a})_{m \in M, a \in \Sigma}, \sim, \phi, F]$ such that the set of words recognized by ρ is L.*

Definition 5. *A data language is monoid recognizable if there exists a monoid mechanism that recognizes it.*

Example 3. The language $\{(a, \tau)(a, 2\tau)...(a, n\tau) \mid \tau > 0, n \in \mathbb{N}\}$ is recognized by the monoid $M = \{1, x, x^2, y\}$ where $x^3 = x, y^2 = y$ and $xy = yx = y$, using $\{1, x, x^2\}$ for the accepting subset. The mechanism ρ is defined in the following way:

- ρ has three registers r_1, r_2 and r_3, with initial value 0,
- The updates are defined in the following way:
  $Up_{1,a} = \{r_1, r_2\}$,
  $Up_{x,a} = \{r_3\}$,
  $Up_{x^2,a} = \{r_2\}$,
  $Up_{y,a} = \emptyset$.
- The relation $\sim$ is defined by its equivalences classes:
  $g_1 = \{(d_1, d_2, d_3) \mid d_1 = 0 \text{ and } d_2 = d_3\}$
  $g_2 = \{(d_1, d_2, d_3) \mid d_2 - d_3 = d_1 \text{ and } d_1 \neq 0\}$
  $g_3 = \{(d_1, d_2, d_3) \mid d_3 - d_2 = d_1 \text{ and } d_1 \neq 0\}$
  $g_4 = \{(d_1, d_2, d_3) \mid d_2 - d_3 \neq d_1 \text{ and } d_3 - d_2 \neq d_1\}$
- ϕ is defined in the following way:
  $\phi(a, g_1) = y$,
  $\phi(a, g_2) = x$,
  $\phi(a, g_3) = x$,
  $\phi(a, g_4) = y$.

We now describe the computation of ρ over the data word $(a, \pi)(a, 2\pi)(a, \sqrt{2})$:

$$\begin{array}{lccccccc}
\text{Value in } M & 1 & \xrightarrow[\pi]{a} & x & \xrightarrow[2\pi]{a} & x^2 & \xrightarrow[\sqrt{2}]{a} & y \\
\text{Values of the registers} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & & \begin{pmatrix} \pi \\ \pi \\ 0 \end{pmatrix} & & \begin{pmatrix} \pi \\ \pi \\ 2\pi \end{pmatrix} & & \begin{pmatrix} \pi \\ \sqrt{2} \\ 2\pi \end{pmatrix} \\
\text{Equivalence class} & g_1 & & g_2 & & g_3 & & g_4
\end{array}$$

Remark 3. If the data set consists of one element, namely $\perp$, everything defined above reduces to the case of regular languages. Indeed, a data language $L \subset (\Sigma \times \{\perp\})^*$ is monoid recognizable if and only if $\pi(L)$ is a regular language, where

$$\begin{aligned}\pi : (\Sigma \times \{\perp\})^* &\to \Sigma^* \\ (a_i, \perp)_{i=1..n} &\mapsto (a_i)_{i=1..n}\end{aligned}$$

We now state the equivalence between languages recognized by data automata and languages recognized by monoid mechanisms:

Theorem 1 ([BPT03]). *A data language is monoid recognizable if and only if it is the set of words recognized by a data automaton.*

1.3 Logics for data languages

P. Bouyer has proposed, in [Bou02], a logical language that describes the recognizable data languages. This logical language $\mathcal{L}$ is defined as the set of closed formulae $\exists X_1 ... \exists X_k.\psi$ where the $(X_i)_{1 \leqslant i \leqslant k}$ are second-order variables (which are interpreted over $\mathbb{N}$) and ψ is a formula of the following grammar $\mathcal{G}$:

$$\mathcal{G}\colon \psi ::= x < y \,|\, x = y \,|\, \exists x.\psi \,|\, \forall x.\psi \,|\, R_a x \,|\, Xx \,|\, g(X_1, .., X_l, x) \,|\, \neg\psi \,|\, \psi \wedge \psi \,|\, \psi \vee \psi$$

where g is a subset of $\mathcal{D}^l$ and $a \in \Sigma$. The variables x and y are first-order variables (which are interpreted over $\mathbb{N}$). This logical language is a second order logic completed with *data predicates* $g(X_1, .., X_l, x)$.

Those formulae are interpreted over data words and assignments for the free variables appearing in the formula. In the following μ and ν are respectively assignments for the first-order and second-order variables. We assume that $u = (a_1, d_1)...(a_n, d_n)$ is a data word and that μ and ν are assignments for the free variables of the considered formula. $\mu[x/i]$ means that the assignment μ is extended to the first-order variable x so that $\mu(x) = i$. We define $\nu[X/\alpha]$ in a similar way.

The satisfaction relation is defined inductively in the following way:

$$\begin{array}{lcl}
(u;\mu.\nu) \models x < y & \iff & \mu(x) < \mu(y) \\
(u;\mu.\nu) \models x = y & \iff & \mu(x) = \mu(y) \\
(u;\mu.\nu) \models \exists x.\psi & \iff & \text{there exists } 1 \leqslant i \leqslant n, (u;\mu[x/i].\nu) \models \psi \\
(u;\mu.\nu) \models \forall x.\psi & \iff & \text{for all } 1 \leqslant i \leqslant n, (u;\mu[x/i].\nu) \models \psi \\
(u;\mu.\nu) \models R_a x & \iff & a_{\mu(x)} = a \\
(u;\mu.\nu) \models Xx & \iff & \mu(x) \in \nu(X) \\
(u;\mu.\nu) \models \neg\psi & \iff & (u;\mu.\nu) \not\models \psi \\
(u;\mu.\nu) \models \psi_1 \wedge \psi_2 & \iff & (u;\mu.\nu) \models \psi_1 \text{ and } (u;\mu.\nu) \models \psi_2 \\
(u;\mu.\nu) \models \psi_1 \vee \psi_2 & \iff & (u;\mu.\nu) \models \psi_1 \text{ or } (u;\mu.\nu) \models \psi_2 \\
(u;\mu.\nu) \models \exists X.\psi & \iff & \text{there exists } \alpha \subset \{1, .., n\} \text{ such that } (u;\mu.\nu[X/\alpha]) \models \psi
\end{array}$$

and

$$(u,\mu.\nu) \models g(X_1, ..., X_l, x) \iff (d^{(\mu,\nu)}_{max(X_1 \leqslant x)}, ..., d^{(\mu,\nu)}_{max(X_l \leqslant x)}) \in g$$

where for all $1 \leqslant i \leqslant l$,

$$d^{(\mu,\nu)}_{max(X_i \leqslant x)} = \begin{cases} d_\lambda \text{ if } \lambda \in \nu(X_i) \text{ and } \forall j, (j \in \nu(X_i) \text{ and } j \leqslant \mu(x) \text{ implies } j \leqslant \lambda) \\ \perp \text{ otherwise.} \end{cases}$$

Intuitively, the second-order variable X_i represents the set of dates where the register r_i is updated. $max(X_i \leqslant x)$ is therefore the last date before x where r_i has been updated. If we follow the sequence of updates defined by the X_is, $g(X_1, ..., X_l, x)$ thus checks whether the registers satisfy the constraint g at the date x.

If ϕ is a closed formula (i.e without free variable), the satisfaction relation is defined for the empty assignment. In this case, we write $u \models \phi$ for $(u, \emptyset.\emptyset) \models \phi$.

A data language L is the set of *models* of ϕ if:

$$L = \{u \in (\Sigma \times \mathcal{D})^* \,|\, u \models \phi\}.$$

which we write $L = L(\phi)$.

Example 4. The language $\{(a,\tau)(a,2\tau)...(a,n\tau)\,|\,\tau > 0, n \in \mathbb{N}\}$ is the set of models of the formula:

$$\exists X_0.\exists X_1.\exists X_2.\Big[\exists x_0.\Big(X_0 x_0 \wedge \forall y.\big((X_0 y \Rightarrow y = x_0) \wedge y \geqslant x_0\big) \wedge X_2 x_0 \wedge g_0(X_0, x_0)\Big)\Big]$$

$$\wedge \forall y.\forall x.\left[\begin{array}{c}(X_1 x \wedge y = x+1) \Rightarrow \big(X_2 y \wedge \neg X_1 y \wedge g(X_0, X_1, X_2, y)\big)\\ \wedge (X_2 x \wedge y = x+1) \Rightarrow \big(X_1 y \wedge \neg X_2 y \wedge g(X_0, X_2, X_1, y)\big)\end{array}\right]$$

where $y = x+1$ is written for $\forall z.[(x < z \Rightarrow y \leqslant z) \wedge (z < y \Rightarrow z \leqslant x)]$, $g_0 = \{d\,|\,d > 0\}$ and $g = \{(d_1, d_2, d_3)\,|\,d_2 - d_3 = d_1\}$.

Intuitively, the first line means that the first data (τ) is stored in the register r_1. The last lines mean that each of the remaining data is stored alternatively in the registers r_2 and r_3 and the data predicates test whether $r_2 - r_3 = r_1$ or $r_3 - r_2 = r_1$.

We come now to the fundamental result due to P. Bouyer in [Bou02]:

Theorem 2 (Logical characterization). *A data language is monoid recognizable if and only if it is the set of models of a closed formula of $\mathcal{L}$.*

Remark 4. When the data set is reduced to $\{\perp\}$, the set of data predicates $g(X_1, .., X_k, x)$ is semantically reduced to the logical predicates $true$ or $false$ and the logical language $\mathcal{L}$ has exactly the same expressiveness as MSO. Indeed, a language $L \subset (\Sigma \times \{\perp\})^*$ can be described by a formula of $\mathcal{L}$ if and only if $\pi(L)$ (where π is defined in the remark 3) can be described by a formula of MSO. This can be seen by following the proof of theorem 2 in [Bou02].

Remark 5. The logical language $\mathcal{L}$ is close to the monadic logic of distance proposed by Th. Wilke in [Wil94] to characterize recognizable timed languages. There is a simple condition under which we can decide the emptiness of the set of models of a given formula of $\mathcal{L}$. Moreover, the decidable fragment of the logic, when the data set is a time domain, is strictly more expressive than the monadic logic of distance.

2 Logical characterization of aperiodic data languages

We recall that a finite monoid M is aperiodic if for all elements $x \in M$, there exists $n \in \mathbb{N}$, such that $x^n = x^{n+1}$.

In the framework of formal languages, the class of regular languages that can be recognized by an aperiodic monoid can be logically characterized, using two important results of M. P. Schützenberger [Sch65] and R. Mc Naughton and S. Papert [NP71]:

Theorem 3. *A language $L \subset \Sigma^*$ can be recognized by a finite aperiodic monoid if and only if it can be defined by a first-order logical formula.*

A proof of the theorem and a description of the logic can be found in [Str94].

Definition 6. *A data language is* **aperiodic** *if it recognized by an aperiodic monoid.*

In the remainder of the paper we present a proof of a similar result concerning data languages. We start by showing that we can use some simplified form for the formulae of $\mathcal{L}$. We then characterize logically the data languages that can be recognized by an aperiodic monoid mechanism whose updates are of a particular form. Finally, we describe a logical characterization of the class of aperiodic data languages.

2.1 A "simplified form" for the formulae of $\mathcal{L}$

We describe here a simplified form for the formulae of $\mathcal{L}$ introduced by P. Bouyer in [Bou02]. An *atomic formula* is a formula of the form $x < y, x = y, R_a x, Xx$ or $g(Y_1, ..., Y_l, x)$. An atomic formula $g(Y_1, ..., Y_l, x)$ is an *atomic data predicate*. Every closed formula of $\mathcal{L}$ can be transformed into a prenex form formula:

$$\phi = \exists X_1 ... \exists X_k . Q_1 x_1 ... Q_p x_p . \left(\bigvee_i \bigwedge_j p_{i,j} \right)$$

where for all i, $Q_i \in \{\exists, \forall\}$ and for all $i, j, p_{i,j}$ is an atomic formula or its negation. Let ϕ be a prenex form formula. For all second-order variables of ϕ, there exists an integer $1 \leqslant i_Y \leqslant k$ such that $Y = X_{i_Y}$. We define an equivalence relation $\sim_\phi$ on $\mathcal{D}^k$:

$$(\theta_i)_{i=1..k} \sim_\phi (\theta'_i)_{i=1..k} \iff$$

$$\Big[\text{for every atomic data predicate } g(Y_1, .., Y_l, x) \text{ of } \phi,$$

$$(\theta_i)_{i=i_{Y_1}..i_{Y_l}} \in g \iff (\theta'_i)_{i=i_{Y_1}..i_{Y_l}} \in g\Big]$$

This equivalence relation is of finite index as there is only a finite number of atomic data predicates in a given formula.

If g is an equivalence class for $\sim_\phi$, an atomic data predicate of the form $g(X_1, .., X_k, x)$ is ϕ*−primitive*. It is easy to check that an atomic data predicate can be transformed into a disjunction of ϕ−primitive atomic data predicates.

Example 5. Coming back to our favorite example, we have four equivalence classes:
$g_1 = \{(d_1, d_2, d_3) \,|\, d_1 = 0 \text{ and } d_2 = d_3\}$
$g_2 = \{(d_1, d_2, d_3) \,|\, d_2 - d_3 = d_1 \text{ and } d_1 \neq 0\}$
$g_3 = \{(d_1, d_2, d_3) \,|\, d_3 - d_2 = d_1 \text{ and } d_1 \neq 0\}$
$g_4 = \{(d_1, d_2, d_3) \,|\, d_2 - d_3 \neq d_1 \text{ and } d_3 - d_2 \neq d_1\}$
and we have the following equivalences:

$$g(X_0, X_1, X_2, y) \equiv g_1(X_0, X_1, X_2, y) \vee g_2(X_0, X_1, X_2, y)$$

and

$$g(X_0, X_2, X_1, y) \equiv g_1(X_0, X_1, X_2, y) \vee g_3(X_0, X_1, X_2, y).$$

Lemma 1. *Every closed formula ϕ of $\mathcal{L}$ can be transformed into an equivalent formula of the following form:*

$$\exists X_1 ... \exists X_k . Q_1 x_1 ... Q_p x_p . \left(\bigvee_i \bigwedge_j p_{i,j} \right)$$

where each $p_{i,j}$ is an atomic formula or its negation and where **any atomic data predicate is ϕ-primitive.**

2.2 A first attempt

We logically characterize the data languages recognized by an aperiodic monoid mechanism whose updates are of a particular form.

Let $\rho = [(Up_{m,a})_{m \in M, a \in \Sigma}, \sim, \phi, F]$ be a monoid mechanism over the monoid M. We define a condition over ρ:

$$\text{Condition } (\dagger) : Up_{m,a} = \mathcal{P}(\{1..k\}), \forall (m, a) \in M \times \Sigma$$

Theorem 4. *Let L be a data language, the following sentences are equivalent:*
i. L can be defined by a formula of $\mathcal{L}$ without atomic formula of the form Xx.
ii. L can be recognized by an aperiodic monoid mechanism which satisfies condition $(\dagger)$.

In the proof of theorem 4, we will use notions introduced by P. Bouyer in [Bou02]. The idea behind the proof is to get down to a language over a free monoid, namely $(\Sigma \times \mathcal{D}^k / \sim_\phi)^*$, where ϕ is a formula defining the language we consider, and to apply the classical result presented as theorem 3, concerning regular languages. The fact that we consider monoid mechanisms satisfying condition (†) is fundamental here. The proof is given in the appendix.

Remark 6. There exist data languages that are recognized by an aperiodic monoid mechanism but that are not recognized by any aperiodic monoid mechanism satisfying condition (†). This is true for the language $L = \{u = (a,d)(a,d') \mid d = d' \neq \perp\}$ over the alphabet $\Sigma = \{a\}$ and a given infinite data set $\mathcal{D}$.
Indeed, it is easy to see that L can be recognized by the aperiodic monoid $M = \{0,1\}$ where $0 * x = 0$ and $1 * x = x$, using the updates $up_{x,a} = \{\{1\},\{2\},\{\emptyset\}\}$ and two registers.
Let us now suppose that an aperiodic monoid mechanism ρ, satisfying condition (†), recognizes L. Then for each $d \in \mathcal{D} \setminus \{\perp\}$, the data word $u_d = (a,d)(a,d)$ is recognized by our mechanism. Hence, there exist two updates $up_1(d)$ and $up_2(d)$ such that the computation of the mechanism over the word u_d, using the updates $up_1(d)$ and $up_2(d)$, ends in an accepting state:

$$\begin{array}{llll} \text{Value in } M: & 1_M \xrightarrow[up_1(d)]{(a,d)} & m_1(d) \xrightarrow[up_2(d)]{(a,d)} & m_2(d) \\ \text{Value of the registers:} & \perp^k & \theta_1(d) & \theta_2(d) \end{array}$$

We now define an equivalence relation on $\mathcal{D} \setminus \{\perp\}$:

$$d \sim d' \Leftrightarrow up_1(d) = up_1(d'), up_2(d) = up_2(d'), \overline{\theta_1(d)} = \overline{\theta_1(d')} \text{ and } \overline{\theta_2(d)} = \overline{\theta_2(d')}$$

where $\overline{\theta}$ is the equivalence class of θ for the equivalence relation of the mechanism.

As $\mathcal{D}$ is infinite, there exists d and d' in $\mathcal{D} \setminus \{\perp\}$, $d \neq d'$ such that $d \sim d'$.

We now show that the data word $u = (a,d)(a,d')$ is accepted by ρ.
A possible computation of ρ over u is:

$$\begin{array}{llll} \text{Value in } M: & 1_M \xrightarrow[up_1(d)]{(a,d)} & m_1(d) = m_1(d') \xrightarrow[up_1(d) \cup up_2(d)]{(a,d')} & m_2(d) = m_2(d') \\ \text{Value of the registers:} & \perp^k & \theta_1(d) & \theta_2(d') \end{array}$$

Hence u is accepted by the mechanism whereas u is not in L and no mechanism satisfying (†) recognizes L.

Remark 7. There also exist data languages that are recognized by a monoid mechanism but that are not recognized by any aperiodic monoid mechanism.
For example, $L = \{(a,\perp)^n \mid n \text{ is even}\}$, over the alphabet $\{a\}$ and the data set $\{\perp\}$ is not recognized by any aperiodic monoid, as the regular language $\tilde{L} = \{a^n \mid n \text{ is even}\}$ cannot be recognized by an aperiodic monoid.

We have logically characterized a fragment of the class of recognizable data languages. Moreover the examples above show that the class of recognizable data languages, the class of data languages that can be recognized by an aperiodic monoid and the class of data languages that can be recognized by an aperiodic monoid mechanism satisfying condition (†) are all distinct.
We now prove a logical characterization of the data languages that are recognized by an aperiodic monoid without restriction on the updates. More precisely, we show that adding a bit of second order formulae to the logical language $\mathcal{L}$ is enough to define exactly the languages recognized by an aperiodic monoid.

2.3 Logical characterization of aperiodic data languages

We first extend a classical result on formulae of MSO which can be found in [Str94] to our logical language $\mathcal{L}$, for formulae without terms of the form Xx.

Proposition 1. *Let $\phi \in \mathcal{G}$ be a logical formula without term of the form Xx and x be a first-order variable that doesn't appear in ϕ. Then there exists a formula $\phi_x \in \mathcal{G}$ such that if $(a_1, d_1)...(a_n, d_n)$ is a data word and $i \in [1, n]$, we have the following equivalence:*

$$((a_1, d_1)...(a_i, d_i); \mu.\nu) \models \phi \iff ((a_1, d_1)...(a_n, d_n); \mu[x/i].\nu) \models \phi_x$$

Proof:
We define ϕ_x by induction on the structure of ϕ:
(i.) if $\phi = (y < z)$, we define $\phi_x = (y < z) \wedge (z \leqslant x)$,
(ii.) if $\phi = (y = z)$, we define $\phi_x = (y = z) \wedge (z \leqslant x)$,
(iii.) if $\phi = \exists y.\psi$, we define $\phi_x = \exists y.(\psi_x \wedge (y \leqslant x))$,
(iv.) if $\phi = R_a y$, we define $\phi_x = R_a y \wedge (y \leqslant x)$,
(v.) if $\phi = g(X_1, ..., X_l, y)$, we define $\phi_x = g(X_1, ..., X_l, y) \wedge (y \leqslant x)$,
(vi.) if $\phi = \neg\psi$, we define $\phi_x = \neg\psi_x$,
(vii.) if $\phi = \psi \wedge \psi'$, we define $\phi = \psi_x \wedge \psi'_x$ and
(viii.) if $\phi = \psi \vee \psi'$, we define $\phi = \psi_x \vee \psi'_x$.
It remains to show that the claimed equivalence is true for each of these transformations. The equivalence is clear for i., ii., iv.,vi. and vii. and viii..

We proceed by induction on the structure of the formula.
We write in details the proof for the third case.
Let $u = (a_1, d_1)...(a_n, d_n)$ be a data word and i in $[1..n]$.

If $((a_1, d_1)...(a_i, d_i); \mu.\nu) \models \exists y.\psi$, there exists an assignment for y, $\mu(y) = k$, where $k \leqslant i$, such that $((a_1, d_1)...(a_i, d_i); \mu[y/k].\nu) \models \psi$. The induction hypothesis shows that:

$$((a_1, d_1)...(a_n, d_n); \mu[y/k][x/i].\nu) \models \psi_x$$

and hence $(u; \mu[x/i].\nu) \models \exists y.(\psi_x \wedge (y \leqslant x))$.

Conversely, if $(u; \mu[x/i].\nu) \models \exists y.(\psi_x \wedge (y \leqslant x))$, it exists an assignment for y, $\mu(y) = k$, where $k \leqslant i$, such that:

$$(u; \mu[x/i][y/k].\nu) \models \psi_x.$$

The induction hypothesis shows that $((a_1, d_1)...(a_i, d_i); \mu[y/k].\nu) \models \psi$.
Hence, $((a_1, d_1)...(a_i, d_i); \mu.\nu) \models \exists y.\psi$ and the claimed equivalence is true for iii. The proof for v. is similar and this ends the proof of proposition 1. □

Let $\mathcal{L}_a$ denote the set of formulae of the following form:

$$\exists X_1...\exists X_k.\psi \wedge \bigwedge_i \Big(\forall x.\psi_{i,x} \Rightarrow \big(\bigwedge_{a \in \Sigma} R_a(x+1) \Rightarrow f_{a,i}(X_1(x+1), ..., X_k(x+1))\big)\Big)$$

where $f_{a,i}$ are boolean formulae, ψ, the ψ_i are first order formulae (i.e. without atomic formula of the form Xx) and the index i ranges over a finite domain.
We now state a logical characterization of aperiodic data languages:

Theorem 5. *Let L be a data language. The following sentences are equivalent:*
i. L can be defined by a formula from $\mathcal{L}_a$.
ii. L can be recognized by an aperiodic monoid mechanism.

The proof can be found at the end of the appendix.

Conclusion

We have presented a logical characterization of the class of data languages recognized by aperiodic monoids. In the first part of the proof, we characterized the languages that can be recognized by an aperiodic monoid mechanism which verifies condition (†). This proof is of particular interest because it is equivalent to the similar result concerning regular languages. This emphasizes the interest of data languages. The last proof is a rather technical corollary of

the previous one, which shows that a logical characterization of aperiodic monoid recognized languages is possible, even if it lacks the elegance of the theory of regular languages. We think that the similarities of both theories highlight the interest of the data language model.
Further studies will consist in finding some extension of the linear temporal logic to the data languages framework and defining some notion of syntactic monoid, like in the case of regular languages.

Acknowledgments: I would like to thank Antoine Petit and Patricia Bouyer for their careful readings of drafts of this paper.

References

[ACM02] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *Journal of the Association for Computing Machinery*, 49(2):172–206, 2002.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BBP03] B. Bérard, P. Bouyer, and A. Petit. Une analyse du protocole PGM avec UPPAAL. In *Actes 4ième Colloque sur la Modélisation des Systèmes réactifs (MSR'2003)*, pages 415–430. Hermès, 2003. To appear.

[BF99] B. Bérard and L. Fribourg. Automatic verification of a parametric real-time program: the ABR conformance protocol. In *Proc. 11th Int. Conf. Computer Aided Verification (CAV'99)*, volume 1633 of *Lecture Notes in Computer Science*, pages 96–107. Springer-Verlag, 1999.

[BL96] J. Bengtsson and F. Larsson. UPPAAL, a tool for automatic verification of real-time systems. Master's thesis, Department of Computer Science, Uppsala University (Sweden), 1996. ISSN 0283-0574.

[Bou02] P. Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002.

[BP02] P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2):167–186, 2002.

[BPT03] P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data and timed languages. *Information and Computation*, 182(2):137–162, 2003.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Proc. Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer-Verlag, 1996.

[DZ98] F. Demichelis and W. Zielonka. Controlled timed automata. In *Proc. 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1998.

[HSLL97] K. Havelund, A. Skou, K. G. Larsen, and K. Lund. Formal modelling and analysis of an audio/video protocol: An industrial case study using UPPAAL. In *Proc. 18th IEEE Real-Time Systems Symposium (RTSS'97)*, pages 2–13. IEEE Computer Society Press, 1997.

[Kam68] H. Kamp. *On tense logic and the theory of linear order*. PhD thesis, UCLA, 1968.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[NP71] R. Mc Naughton and S. Papert. *Counter-Free Automata*. M.I.T. Press, 1971.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57. IEEE Computer Society Press, 1977.

[SBB+01] Ph. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie, and A. Petit. *Model-checking techniques and tools*. Springer, 2001.

[Sch65] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, vol.8:190–194, 1965.

[Str94] H. Straubing. *Finite automata, formal logic, and circuit complexity*. Birkhäuser, 1994.

[TY98] S. Tripakis and S. Yovine. Verification of the fast reservation protocol with delayed transmission using the tool KRONOS. In *Proc. 4th IEEE Real-Time Technology and Applications Symposium (RTAS'98)*. IEEE Computer Society Press, 1998.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 694–715. Springer, 1994.

[Yov97] S. Yovine. KRONOS: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1–2):123–133, 1997.

Appendix

Proof of theorem 4

i. $\Rightarrow$ ***ii.*** Let L be a data language defined by a formula ϕ of $\mathcal{L}$. Thanks to lemma 1, we can assume that ϕ is of the form:

$$\exists X_1 ... \exists X_k . Q_1 x_1 ... Q_p x_p . \bigvee_i \bigwedge_j p_{i,j}$$

where $p_{i,j}$ is an atomic formula or its negation, and such that any atomic data predicate is ϕ-primitive. We suppose that none of them is of the form Xx.

Let $\psi = \bigvee_i \bigwedge_j p_{i,j}$. We transform ψ into $\overline{\psi}$ using the rules of figure 3:

$$\overline{x < y} \equiv x < y \qquad \overline{g(X_1..X_k, x)} \equiv R_g x$$

$$\overline{x = y} \equiv x = y \qquad \overline{\psi_1 \wedge \psi_2} \equiv \overline{\psi_1} \wedge \overline{\psi_2}$$

$$\overline{R_a x} \equiv R_a x \qquad \overline{\neg \psi} \equiv \neg \overline{\psi}$$

$$\overline{\psi_1 \vee \psi_2} \equiv \overline{\psi_1} \vee \overline{\psi_2}$$

Fig. 3.

The overlined formulae are interpreted over words of $(\Sigma \times \mathcal{D}^k / \sim_\phi)^*$ and assignments μ and ν for the free first-order and second order variables, respectively:

$$((a_1, g_1)...(a_n, g_n); \mu.\nu) \vDash R_g x \Longleftrightarrow g_{\mu(x)} = g$$

$$((a_1, g_1)...(a_n, g_n); \mu.\nu) \vDash R_a x \Longleftrightarrow a_{\mu(x)} = a$$

Let $u = (a_1, d_1)...(a_n, d_n) \in (\Sigma \times \mathcal{D})^*$ be a data word and ν an assignment for the second-order variables $X_1, .., X_k$. We associate with u the words u_ν and u'_ν defined in the following way:

$$u'_\nu = (a_1, d_1, \theta_1, \chi_1)...(a_n, d_n, \theta_n, \chi_n) \in \Big(\Sigma \times \mathcal{D} \times \mathcal{D}^k \times \mathcal{P}(\{1..k\})\Big)^*$$

where $\theta_0 = \perp^k, \theta_{i+1}[j] = \begin{cases} \theta_i[j] & \text{if } i \notin \nu(X_j) \\ d_i & \text{if } i \in \nu(X_j) \end{cases}$ and $\chi_i = \{j \mid i \in \nu(X_j)\}$,

$$u_\nu = (a_1, \overline{\theta_1})...(a_n, \overline{\theta_n}) \in (\Sigma \times D^k / \sim_\phi)^*$$

We now show that:

$$(u; \nu) \vDash Q_1 x_1 ... Q_p x_p \psi \Longleftrightarrow (u_\nu; \nu) \vDash Q_1 x_1 .. Q_p x_p \overline{\psi} \tag{1}$$

Let μ be an assignment for the first-order variables. We will show that for every atomic formula ρ, $(u, \mu.\nu) \vDash \rho$ if and only if $(u_\nu, \mu.\nu) \vDash \overline{\rho}$. The proof of equivalence (1) will follow because ψ is a boolean combination of atomic formulae.

- if $\rho = R_a x$, we have that $\overline{\rho} = R_a x$ and

$$(u; \mu.\nu) \vDash R_a x \Longleftrightarrow a_{\mu(x)} = a \Longleftrightarrow (u_\nu; \mu.\nu) \vDash R_a x.$$

- if $\rho = g(X_1, ..., X_k, x)$, then $\overline{\rho} = R_g x$ and

$$(u; \mu.\nu) \vDash g(X_1, ..., X_k, x) \Longleftrightarrow (d^{(\mu,\nu)}_{max(X_1 \leqslant x)}, ..., d^{(\mu,\nu)}_{max(X_k \leqslant x)}) \in g.$$

But $\theta_{\mu(x)} = (d^{(\mu,\nu)}_{max(X_1 \leqslant x)}, ..., d^{(\mu,\nu)}_{max(X_k \leqslant x)})$. Hence,

$$(u_\nu; \mu.\nu) \vDash R_g x \Longleftrightarrow \overline{\theta_{\mu(x)}} = g.$$

This shows equivalence (1).

Because $Q_1x_1..Q_px_p\overline{\psi}$ is a first-order formula, there exists an aperiodic monoid M, a subset F of M and a monoid morphism $f : (\Sigma \times \mathcal{D}^k/\sim_\phi)^* \to M$ such that $\tilde{L} = f^{-1}(F)$, if $\tilde{L}$ is the set of models of $Q_1x_1..Q_px_p.\overline{\psi}$, thanks to Theorem 3.

We now show that the data language L, defined by the formula ϕ, is recognized by the monoid M and the $k-$register monoid mechanism:

$$\rho = [(Up_{m,a})_{m\in M, a\in\Sigma}, \sim_\phi, f, F]$$

where $Up_{m,a} = \mathcal{P}(\{1..k\}), \forall (m,a) \in M \times \Sigma$.

Let $u = (a_1,d_1)...(a_n,d_n) \in (\Sigma\times\mathcal{D})^*$ be a data word.
If $u \in L$, there exists an assignment ν for the second-order variables such that $(u;\nu) \vDash Q_1x_1...Q_px_p.\psi$. We know that:

$$u_\nu = (a_1,\overline{\theta_1})...(a_n,\overline{\theta_n}) \vDash Q_1x_1...Q_px_p\overline{\psi}.$$

Hence, $f(a_1,\overline{\theta_1})...f(a_n,\overline{\theta_n}) \in F$.
We define a sequence of updates $(up_i)_{1\leqslant i\leqslant n}$:

$$up_i = \{j \mid i \in \nu(X_j)\}$$

Then u is recognized by the mechanism, using the sequence of updates $(up_i)_{1\leqslant i\leqslant n}$.

Conversely, if u is recognized by the mechanism, there exists a sequence of updates $(up_i)_{1\leqslant i\leqslant n}$ such that the sequence:

$$\theta_0 = \perp^k, \theta_{i+1} = up_i(\theta_i, d_i), i \leqslant n$$

satisfies:

$$f(a_1,\overline{\theta_1})...f(a_n,\overline{\theta_n}) \in F.$$

Let ν be the assignment for the second-order variables defined by:

$$\nu(X_i) = \{j \mid i \in up_j\}$$

We have now: $u_\nu = (a_1,\overline{\theta_1})...(a_n,\overline{\theta_n})$ and as $u_\nu \vDash Q_1x_1...Q_px_p\overline{\psi}$,
$(u;\nu) \vDash Q_1x_1...Q_px_p\psi$ and $u \vDash \phi$.

Hence, L is recognized by an aperiodic monoid mechanism satisfying condition (†).

ii.* $\Rightarrow$ *i. Conversely, let us suppose that L is recognized by a mechanism ρ over an aperiodic monoid M satisfying condition (†):

$$\rho = [(Up_{m,a})_{m\in M, a\in\Sigma}, \sim, f, F]$$

The morphism f with the subset F recognizes a language $\tilde{L} \subset (\Sigma\times\mathcal{D}^k/\sim)^*$ which can be defined by a first-order formula in prenex form $Q_1x_1..Q_px_p\overline{\psi}$. We transform $\overline{\psi}$ into ψ, reversing the rules of figure 3.

We now show that:

$$L = L(\exists X_1...\exists X_k.Q_1x_1...Q_px_p\psi).$$

Let $u = (a_1,d_1)...(a_n,d_n) \in (\Sigma\times\mathcal{D})^*$ be a data word.
If $u \vDash \exists X_1...\exists X_k.Q_1x_1...Q_px_p.\psi$, there exists an assignment ν for $X_1,...,X_k$ such that:

$$(u;\nu) \vDash Q_1x_1...Q_px_p.\psi.$$

Then, it follows that:

$$u_\nu = (a_1,\overline{\theta_1})...(a_n,\overline{\theta_n}) \vDash Q_1x_1..Q_px_p.\overline{\psi}.$$

We define a sequence of updates $(up_i)_{1\leqslant i\leqslant n}$:

$$up_i = \{j \mid i \in \nu(X_j)\}$$

The mechanism, using the sequence up_i, recognizes the word u.

Conversely, if the mechanism recognizes the data word u, there exists a sequence of updates $(up_i)_{1 \leqslant i \leqslant n}$, such that the sequence:

$$\theta_0 = \perp^k, \theta_{i+1} = up_i(\theta_i, d_i)$$

is such that $f(a_1, \overline{\theta_1})...f(a_n, \overline{\theta_n}) \in F$.
If ν is the assignment defined by: $\mu(X_i) = \{j \mid i \in up_j\}$, $u_\nu = (a_1, \overline{\theta_1})...(a_n, \overline{\theta_n})$
and $u_\nu \vDash Q_1 x_1..Q_p x_p.\overline{\psi}$.
It follows that: $u \vDash \exists X_1..\exists X_k.Q_1 x_1...Q_p x_p.\psi$ and this ends our proof. □

Proof of theorem 5

i.* $\Rightarrow$ *ii. Let L be a data language defined by a formula ϕ of $\mathcal{L}_a$:

$$\phi = \exists X_1...\exists X_k.\psi \wedge \bigwedge_{i=1..r} \Big(\forall x.\psi_{i,x} \Rightarrow \big(\bigwedge_{a \in \Sigma} R_a(x+1) \Rightarrow f_{a,i}\big(X_1(x+1), ..., X_k(x+1)\big)\big)\Big)$$

where the $f_{a,i}$ are boolean formulae, ψ and the ψ_i are first-order formulae.
The formulae ψ and $(\psi_i)_{1 \leqslant i \leqslant r}$ define data languages recognized by k-register aperiodic monoid mechanisms, denoted by

$$\rho = [(Up_{n,a})_{n \in N, a \in \Sigma}, \sim, f, G],$$

and

$$\rho_i = [(Up_{i_{n,a}})_{n \in N_i, a \in \Sigma}, \sim_i, f_i, G_i].$$

We define $M = N \times N_1 \times ... \times N_r$, which is an aperiodic monoid, as a finite product of aperiodic monoids, and $F = G \times N_1 \times ... \times N_r$.
The updates will be defined using the boolean formulae $f_{a,i}$. Indeed, a boolean formula f with k variables represents a set of subsets, that we denote by P_f, of $[1..k]$: a subset $I \subset [1..k]$ is in P_f if and only if the assignment v, defined by $v(x_i) = 1$ if and only if $i \in I$, satisfies f.
Hence, the boolean formulae $f_{a,i}$ define update sets. If $m = (n, n_1, ..., n_r) \in M$, we define

$$\overline{Up_{m,a}} = \{I \subset [1..k] \mid \forall i \in [1..r], n_i \in G_i \Rightarrow I \vDash f_{a,i}\}.$$

We define the k-register mechanism $\overline{\rho} = [\overline{(Up_{m,a})}_{m \in M, a \in \Sigma}, \approx, \overline{f}, F]$ over the monoid M:
- $\approx$ is the equivalence relation generated by $\sim$ and the family $(\sim_i)_{i=1..r}$:

$$R \approx R' \text{ iff } R \sim R' \text{ and } R \sim_i R', \forall i \in [1..r],$$

- $\overline{f}$ is defined by:

$$\overline{f}(a, \overline{\theta}^{\approx}) = (f(a, \overline{\theta}^{\sim}), f_1(a, \overline{\theta}^1), ..., f_r(a, \overline{\theta}^r))$$

where $\overline{\theta}^{\approx}$ is the class of θ modulo $\approx$, $\overline{\theta}^{\sim}$ is the class of θ modulo $\sim$ and $\overline{\theta}^i$ is the class of θ modulo $\sim_i$.

We now show that $\overline{\rho}$ recognizes L.
Let $u = ((a_1, d_1)...(a_n, d_n))$ be a data word.
If u is recognized by $\overline{\rho}$, there exists a sequence of updates $(up_i)_{1 \leqslant i \leqslant n}$ such that:
- the sequence $\theta_0 = \perp^k, \theta_{i+1} = up_{i+1}(\theta_i, d_{i+1})_{i=0..(n-1)}$ verifies:

$$\overline{f}(a_1, \overline{\theta_1}^{\approx})...\overline{f}(a_n, \overline{\theta_n}^{\approx}) \in F$$

- $\forall i \in [0..n-1]$, if $m_i = \overline{f}(a_1, \overline{\theta_1}^{\approx})...\overline{f}(a_i, \overline{\theta_i}^{\approx})$, then $up_{i+1} \in \overline{Up_{m_i, a_{i+1}}}$.

We define a second-order variables assignment $(X_i)_{1 \leqslant i \leqslant k}$:

$$\nu(X_i) = \{j \mid i \in up_j\}.$$

Let us show that:

$$(u; \emptyset.\nu) \vDash \psi \wedge \bigwedge_{i=1..r} \Big(\forall x.\psi_{i,x} \Rightarrow \big(\bigwedge_{a \in \Sigma} R_a(x+1) \Rightarrow f_{a,i}\big(X_1(x+1), ..., X_k(x+1)\big)\big)\Big).$$

It is clear that $(u;\emptyset.\nu) \vDash \psi$. Let $i \in [1..r]$ and μ be an assignment for the first-order variable x: $\mu(x) = l$ such that $(u;\mu.\nu) \vDash \psi_{i,x}$.
We will show that $(u;\mu.\nu) \vDash \bigwedge_{a\in\Sigma} R_a(x+1) \Rightarrow f_{a,i}((x+1)X_1,...,(x+1)X_k)$, which is equivalent to $up_{l+1} \vDash f_{a_{l+1},i}$. The proposition 1 shows that $((a_1,d_1)...(a_l,d_l);\emptyset.\nu) \vDash \psi_i$ and hence, by definition of the updates, $up_{l+1} \vDash f_{a_{l+1},i}$.
Conversely, if $u \vDash \phi$, there exists an assignment ν for the second-order variables such that:

$$(u;\emptyset.\nu) \vDash \psi \wedge \bigwedge_{i=1..r}\Big(\forall x.\psi_{i,x} \Rightarrow \Big(\bigwedge_{a\in\Sigma} R_a(x+1) \Rightarrow f_{a,i}\big(X_1(x+1),...,X_k(x+1)\big)\Big)\Big).$$

We define a sequence of updates $(up_i)_{1\leqslant i\leqslant n}$:

$$up_i = \{j \mid i \in \nu(X_j)\}$$

The sequence: $\theta_0 = \perp^k, \theta_{i+1} = up_{i+1}(\theta_i, d_{i+1}), \forall i = 0..(n-1)$ verifies:

$$\overline{f}(a_1, \overline{\theta_1}^{\approx})...\overline{f}(a_n, \overline{\theta_n}^{\approx}) \in F$$

and, if $m_i = \overline{f}(a_1, \overline{\theta_1}^{\approx})...\overline{f}(a_{i-1}, \overline{\theta_i}^{\approx})$, then $up_{i+1} \in \overline{Up}_{m_i,a_{i+1}}$ by definition of the sets $\overline{Up}_{m,a}$.
It follows that the word u is recognized by the mechanism $\overline{\rho}$.
Hence, L is recognized by the aperiodic monoid mechanism $\overline{\rho}$.

ii.* $\Rightarrow$ *i. Conversely, let us suppose that the language L is recognized by the k-register mechanism

$$\rho = [(Up_{m,a})_{m\in M, a\in\Sigma}, \sim, f, F]$$

where M is aperiodic.
For each element m of the monoid, we denote by L_m the language recognized by the k-register mechanism

$$\rho' = [(Up'_{m,a})_{m\in M, a\in\Sigma}, \sim, f, \{m\}]$$

where ρ verifies condition (†). As the monoid M is aperiodic, there exists, for all $m \in M$, a formula $\exists X_1...\exists X_k.\psi_m$ of $\mathcal{L}$ without atomic formulae of the form Xx such that ψ_m defines L_m, thanks to theorem 4. We set $\psi = \bigvee_{m\in F} \psi_m$. Each update of the mechanism $Up_{m,a}$ can be described by a boolean formula $f_{m,a}$, as we have seen above. We define:

$$\phi = \exists X_1...\exists X_k.\psi \wedge \bigwedge_{m\in M}\Big(\forall x.\psi_{m,x} \Rightarrow \Big(\bigwedge_{a\in\Sigma} R_a(x+1) \Rightarrow f_{m,a}\big(X_1(x+1),...,X_k(x+1)\big)\Big)\Big)$$

We now show that $L = L(\phi)$.
Let $u = ((a_1,d_1)...(a_n,d_n))$ be a data word.
If u is recognized by ρ, there exists a sequence of updates $(up_i)_{1\leqslant i\leqslant n}$ such that:

- the sequence $\theta_0 = \perp^k, \theta_{i+1} = up_{i+1}(\theta_i, d_{i+1}), \forall i = 0..(n-1)$ verifies:

$$f(a_1,\theta_1)...f(a_n,\theta_n) \in F,$$

- if $m_i = (a_1,\theta_1)...f(a_i,\theta_i), \forall i = 0..n-1$, $up_{i+1} \in Up_{m_i,a_{i+1}}$.

We then define a second-order variables assignment ν:

$$\nu(X_i) = \{j \mid i \in up_j\}.$$

We have that $(u;\emptyset.\nu) \vDash \psi$, as u is recognized by ρ. Let $m \in M$ and μ be an assignment for the first-order variable x such that $(u;\mu.\nu) \vDash \psi_{m,x}$. It remains to show that if $a \in \Sigma$ and if $a_{\mu(x)+1} = a$, we have then

$$(u;\mu.\nu) \vDash f_{m,a}(X_1(x+1),...,X_k(x+1)).$$

The proposition 1 allows us to write:

$$((a_1,d_1)...(a_{\mu(x)},d_{\mu(x)});\emptyset.\nu) \vDash \psi_m.$$

Hence, $((a_1, d_1)...(a_\mu(x), d_{\mu(x)})) \in L_m$ and $up_{\mu(x)+1} \in Up_{m,a}$. The definition of the boolean formulae $f_{m,a}$ shows then that $(u; \mu.\nu) \vDash f_{m,a}(X_1(x+1), ..., X_k(x+1))$.
Conversely, if $u \vDash \phi$, there exists a second-order variables assignment ν such that $(u; \emptyset.\nu) \vDash \psi$ and

$$(u; \emptyset.\nu) \vDash \bigwedge_{m \in M} \Big(\forall x.\psi_{m,x} \Rightarrow \big(\bigwedge_{a \in \Sigma} R_a(x+1) \Rightarrow f_{m,a}\big(X_1(x+1), ..., X_k(x+1)\big)\big)\Big).$$

We define a sequence of updates $(up_i)_{1 \leqslant i \leqslant n}$:

$$up_i = \{j \mid i \in \nu(X_j)\}.$$

We show now that the mechanism $\overline{\rho}$, using the sequence up_i recognizes the word u.
The sequence $\theta_0 = \perp^k, \theta_{i+1} = up_{i+1}(\theta_i, d_{i+1})_{i=0..(n-1)}$ verifies

$$f(a_1, \theta_1)...f(a_n, \theta_n) \in F$$

because $(u; \nu.\emptyset) \vDash \psi$.
All that remains to prove is that if $m_i = f(a_1, \theta_1)...f(a_i, \theta_i)$, then we have that $up_{i+1} \in Up_{m_i, a_{i+1}}$, for all $i \in [0..n-1]$. Let $i \in [0..n-1]$.
The proposition 1 shows that

$$((a_1, d_1)...(a_n, d_n); \mu[x/i].\nu) \vDash \psi_{m_i,x}$$

and then, $up_{i+1} \in Up_{m_i, a_{i+1}}$. The mechanism $\overline{\rho}$ recognizes u. Hence, the language L can be defined by a formula of $\mathcal{L}_a$. $\square$