An Algebraic Characterization of Data and Timed Languages

Patricia Bouyer*, Antoine Petit*, Denis Thérien**

* LSV – CNRS UMR 8643 & ENS de Cachan

** McGill University – Montréal

Overview

Formal languages	Timed languages
Finite automata	Timed automata [AD94]
Rational expressions	Some attempts [ACM97/02,Asa98,BP99/02]
Logical characterization MSO(<), LTL	Some ad hoc logics [Wilke94,HRS98]
Algebraic characterization (using monoids)	?

A more general framework

- We consider a set of data $\mathcal{D}$ that can be a time domain
- We are interested in the languages from $(\Sigma \times \mathcal{D})^*$
- $\mathcal{D}$ has an initial data, $\perp$.

  For example, if $\mathcal{D}$ is a time domain, then $\perp$ is zero. Otherwise, $\perp$ can be the empty data.

Our definition

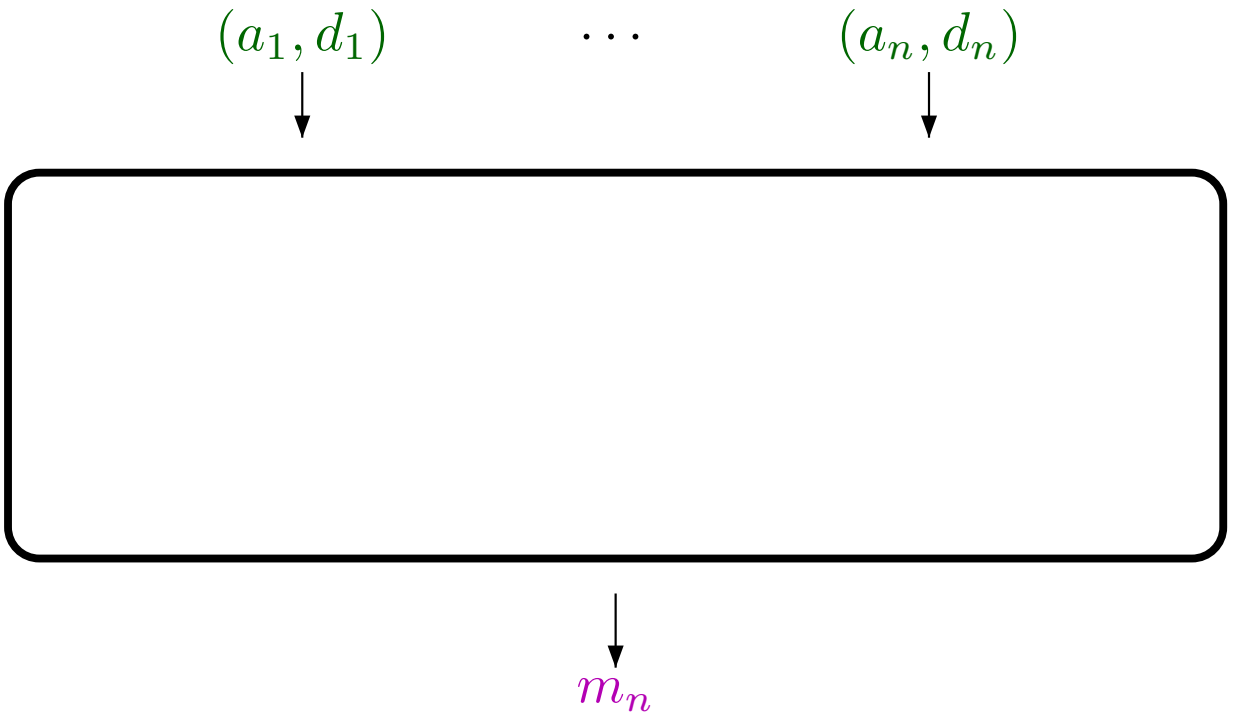
We are given:

- a finite monoid M
- a finite "memory" consisting in k registers

Our definition

We are given:

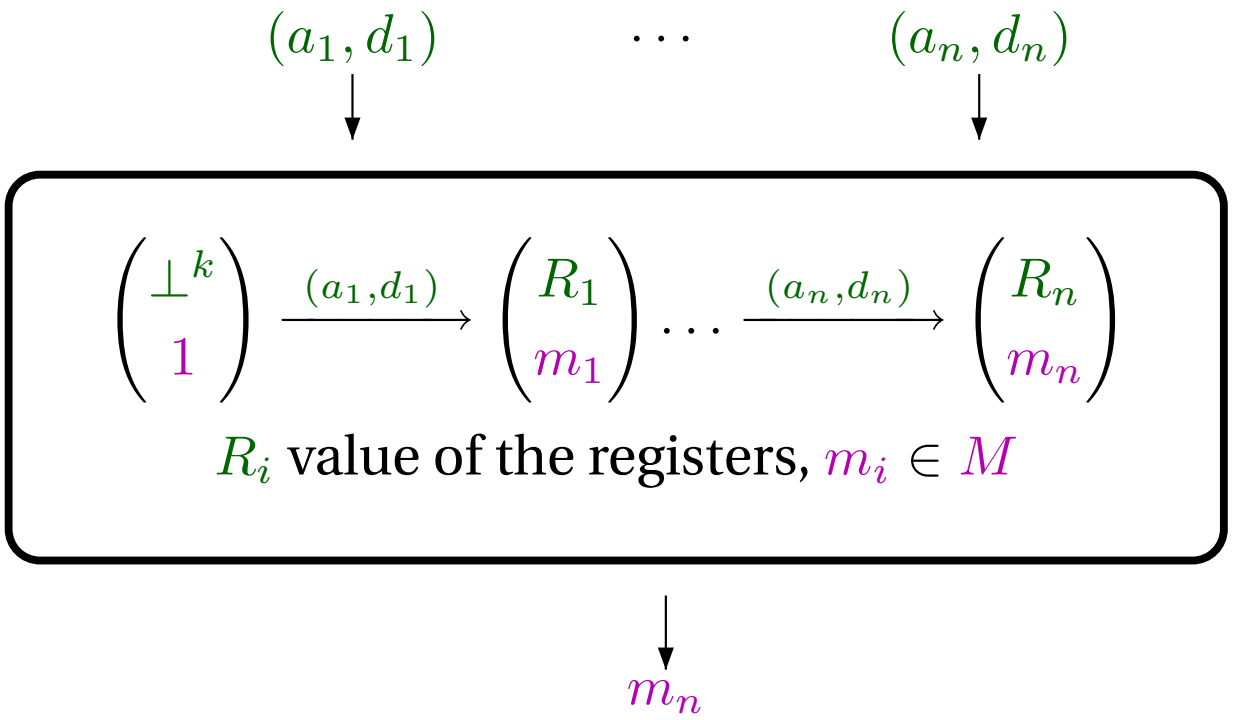
- a finite monoid M
- a finite "memory" consisting in k registers



Our definition

We are given:

- a finite monoid M
- a finite "memory" consisting in k registers



Our definition

We are given:

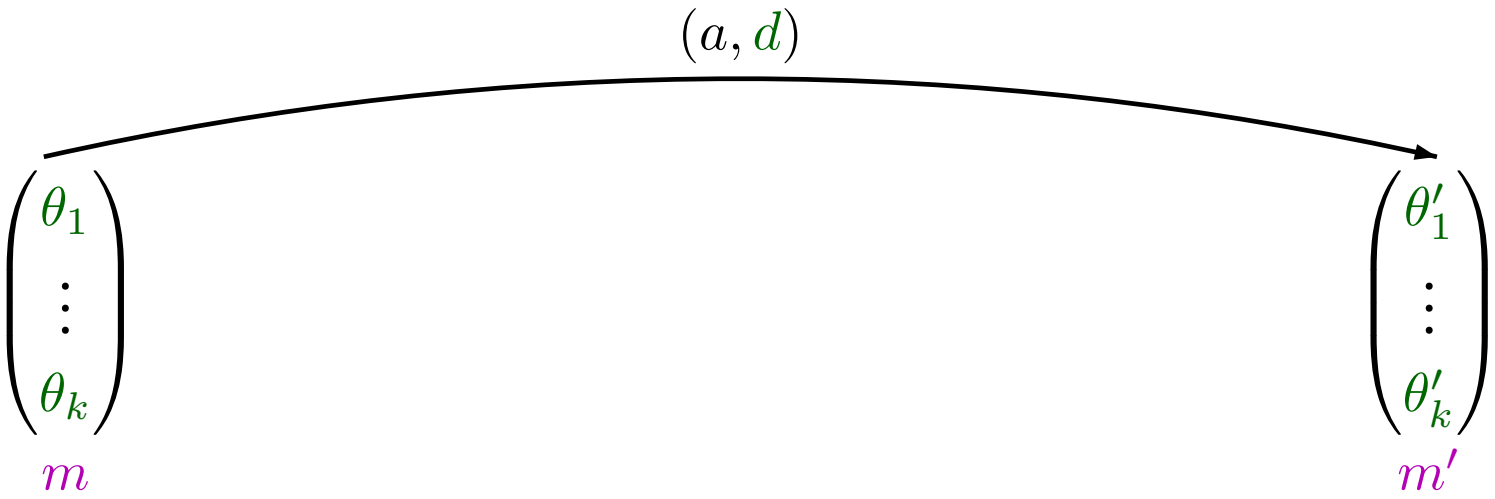
- a finite monoid M
- a finite "memory" consisting in k registers

$$\begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix}$$
$$m$$

Our definition

We are given:

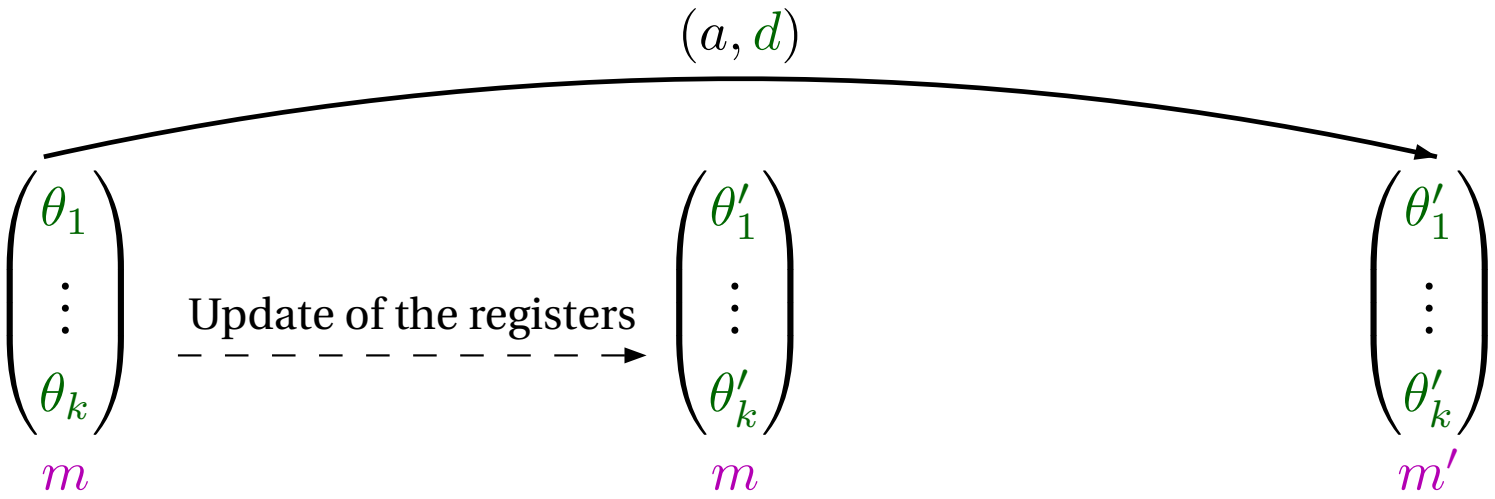
- a finite monoid M
- a finite "memory" consisting in k registers



Our definition

We are given:

- a finite monoid M
- a finite "memory" consisting in k registers

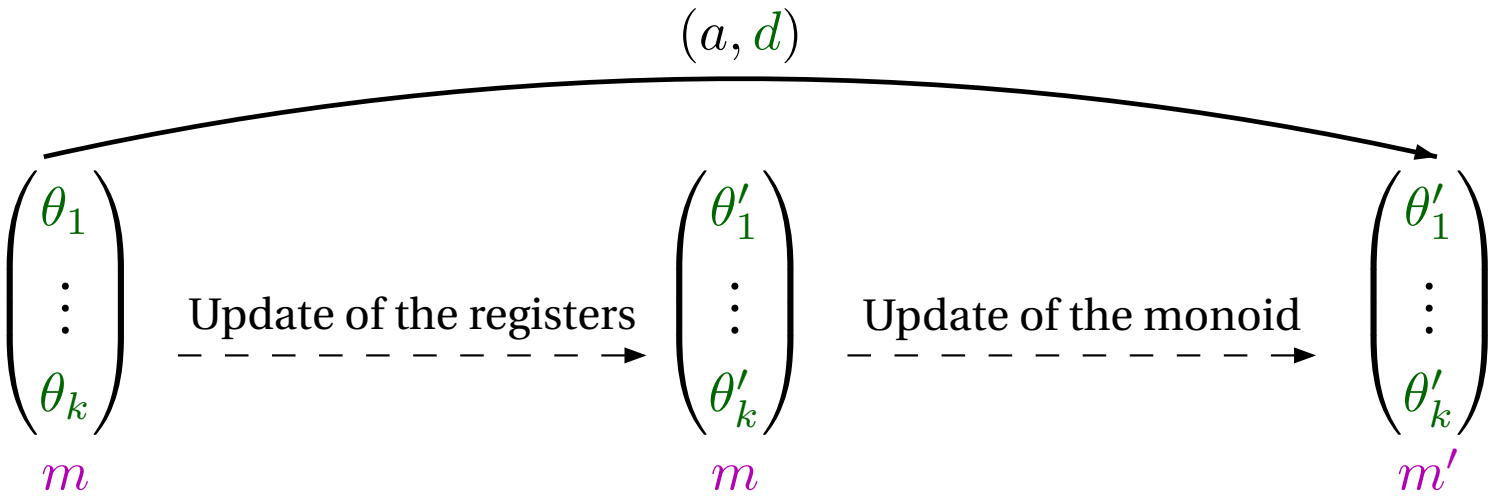


- $\theta'_i = \theta_i$ or d depending on m and a

Our definition

We are given:

- a finite monoid M
- a finite "memory" consisting in k registers



- $\theta'_i = \theta_i$ or d depending on m and a
- m' depends on m, on a and finitely on $(\theta'_i)_{i=1\ldots k}$

Indeed, $m' = m.\varphi(a, \overline{(\theta'_i)_{i=1\ldots k}})$

Example

Example: the language $\{(a,d)(a,d') \mid d \neq \perp,\ d' \neq d\}$

- $M = \{1, 0, y, y^2\}$ where $y^3 = 0$
- two registers
- $up_1 = \{1\}$, $up_y = \{2\}$ and $up_0 = up_{y^2} = \emptyset$
- $$\begin{pmatrix} d \\ d' \end{pmatrix} \mapsto \begin{cases} y & \text{if } d \neq d' \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{pmatrix} \perp \\ \perp \end{pmatrix} \xrightarrow{(a,d)} \begin{pmatrix} d \\ \perp \end{pmatrix} \xrightarrow{(a,d')} \begin{pmatrix} d \\ d' \end{pmatrix}$$

$$1 \qquad 1.y = y \qquad y.y = y^2$$

Remarks

- **Remark:** if $\mathcal{D} = \{\perp\}$

  recognizable formal language $\equiv$ recognizable data language

  + same monoid

- **Property:** if $\mathcal{D}$ is finite, $L \subseteq (\Sigma \times \mathcal{D})^*$ is a regular formal language on the finite alphabet $\Sigma \times \mathcal{D}$ iff it is a recognizable data language.

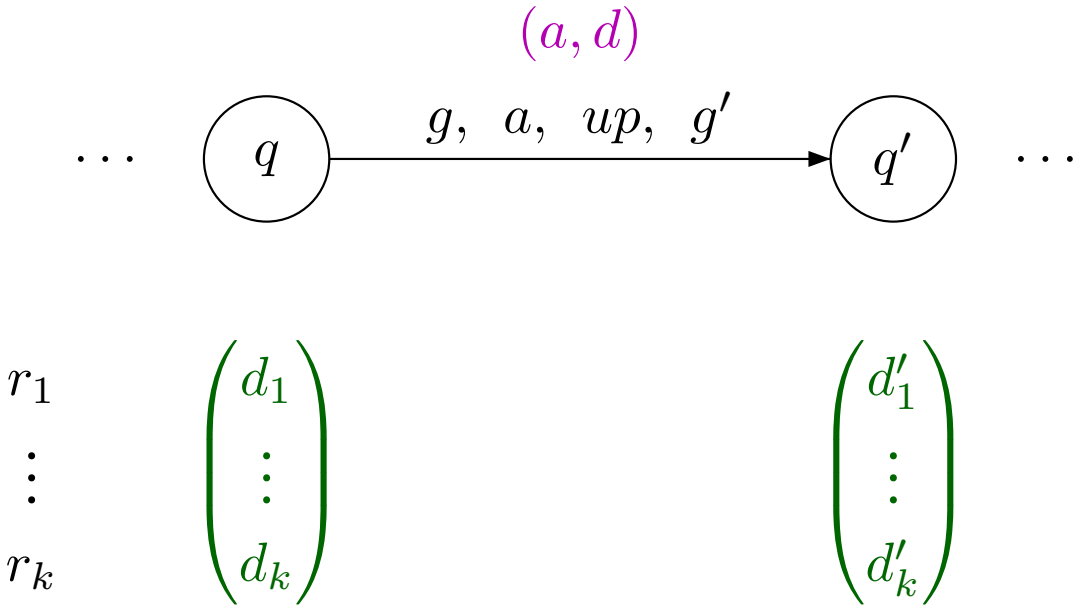
  But different monoids

New overview

Formal languages	Timed languages	Data languages
Finite automata	Timed automata	?
Rational expressions	Some attempts	?
Logic MSO(<), LTL	Some ad hoc logics	?
Algebraic characterization	?	✔

Data automata

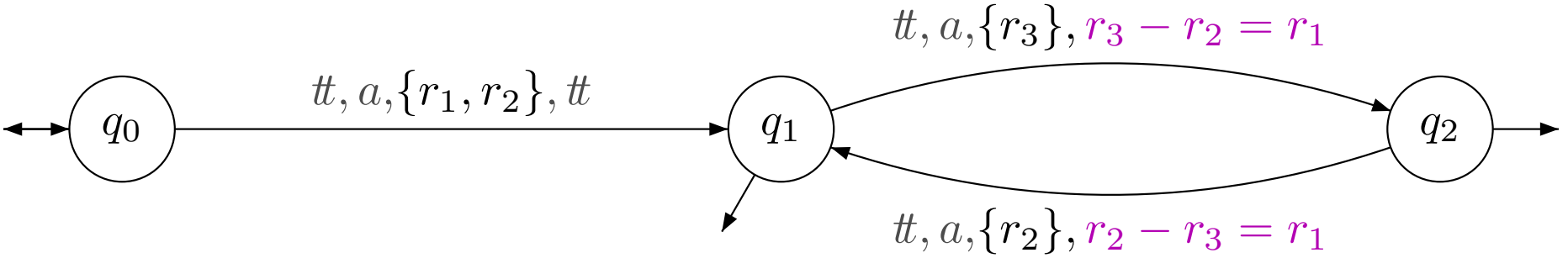
Data automaton with k registers:



with $(d_i)_{i=1\ldots n} \in g$, $(d'_i)_{i=1\ldots n} \in g'$ and $\begin{cases} d'_i = d_i & \text{if } r_i \notin up \\ d'_i = d & \text{if } r_i \in up \end{cases}$

An example

The data language $\{(a,\tau)(a,2\tau)\dots(a,n\tau) \mid n \in \mathbf{N},\ \tau > 0\}$ is accepted by:



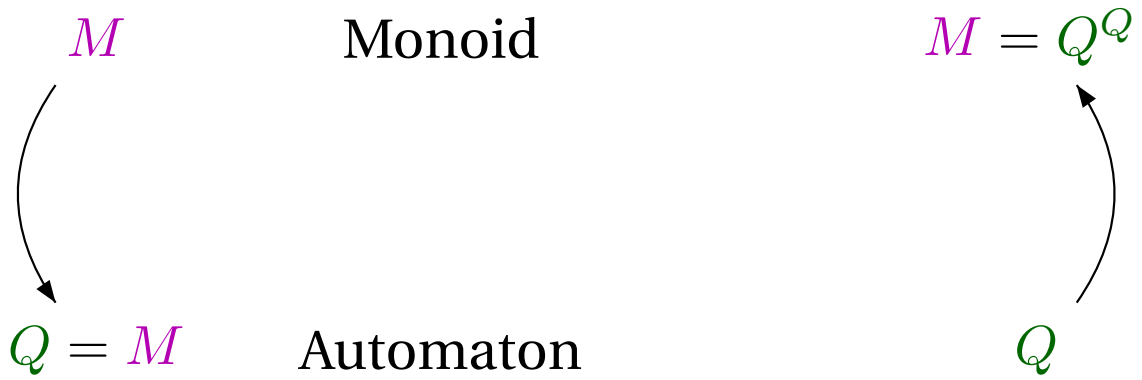
Example of computation:

$$q_0, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{(a,\tau)} q_1, \begin{pmatrix} \tau \\ \tau \\ 0 \end{pmatrix} \xrightarrow{(a,2\tau)} q_2, \begin{pmatrix} \tau \\ \tau \\ 2\tau \end{pmatrix} \xrightarrow{(a,3\tau)} q_1, \begin{pmatrix} \tau \\ 3\tau \\ 2\tau \end{pmatrix} \dots$$

Equivalence

Theorem: equivalence between monoid recognizability and data automata acceptance.

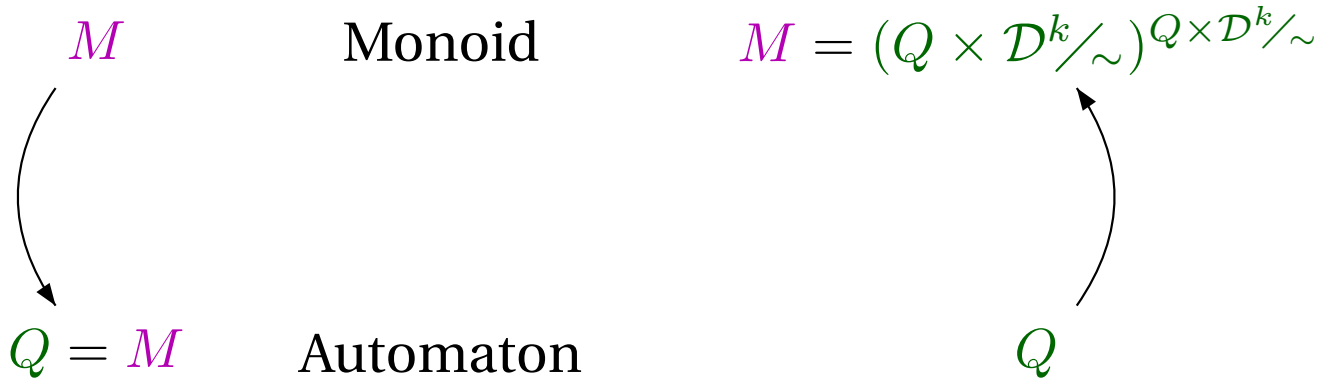
- Proof close to the proof for the formal regular languages.



Equivalence

Theorem: equivalence between monoid recognizability and data automata acceptance.

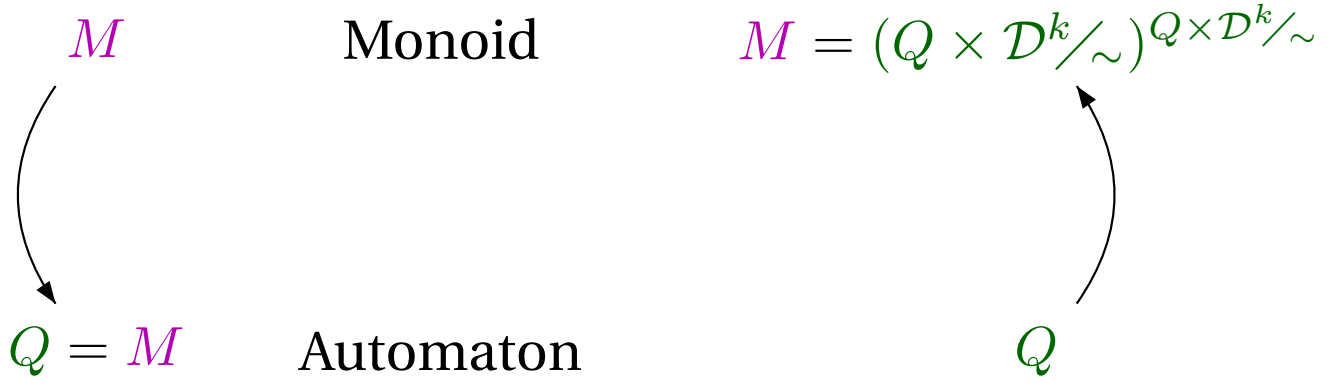
- Proof close to the proof for the formal regular languages.



Equivalence

Theorem: equivalence between monoid recognizability and data automata acceptance.

- Proof close to the proof for the formal regular languages.



- The same number of registers and equivalence relation.

Properties

- The monoid plays a fundamental role.

  "Two distinct varieties of monoids generate two different classes of data languages"

- Closure properties

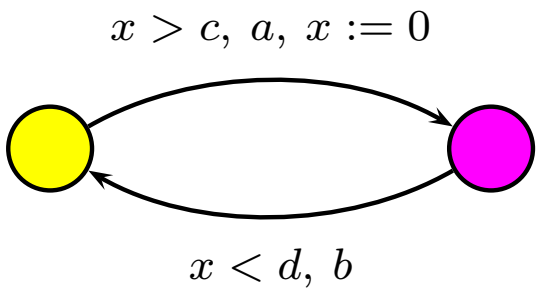
- Relative hierarchies registers/monoids:
  - The hierarchy on the registers is strictly monotonic

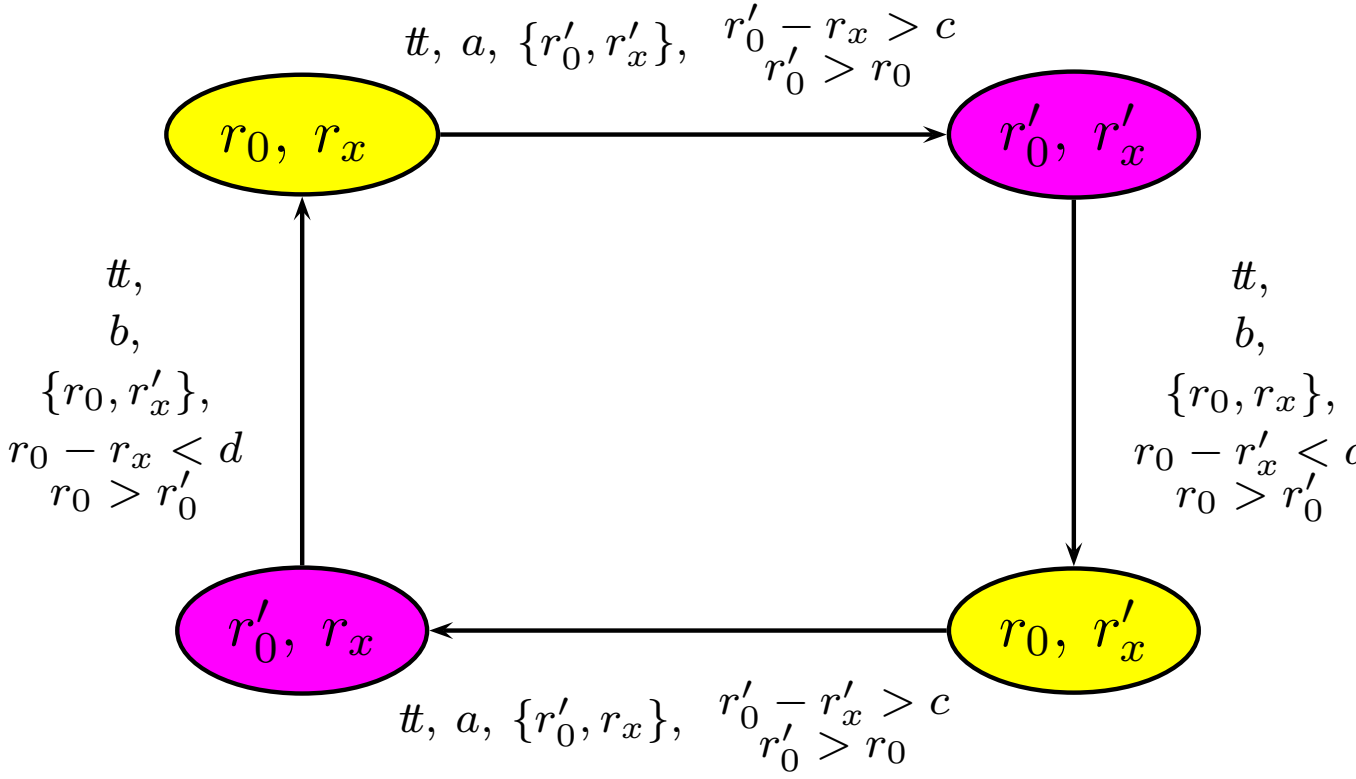
$$L_k = \{(a, d_1) \dots (a, d_n) \mid i \equiv j \mod (k-1) \implies d_i = d_j\}$$

  - Fixing a finite monoid, the hierarchy collapses.

Link with timed languages ?

- Given a deterministic timed automaton, there exists a data automaton that recognizes the same timed/data language.





Link with timed languages ?

- Given a deterministic timed automaton, there exists a data automaton that recognizes the same timed/data language.
- Data automata are more expressive than timed automata

$$L = \{(a,\tau)(a,2\tau)\dots(a,n\tau) \mid n \in \mathbf{N},\ \tau > 0\}$$

➥ data automata = generalization of timed automata

Some extensions

- Erasing and swapping registers: does not increase the expressiveness
- Extending the operations on the registers: the monoid is no more relevant
- Adding non-determinism: extends the expressiveness of the model

$$\{(a, d_1) \dots (a, d_n) \mid n \in \mathbb{N} \text{ and } \exists i \neq j \text{ s.t. } d_i = d_j\}$$

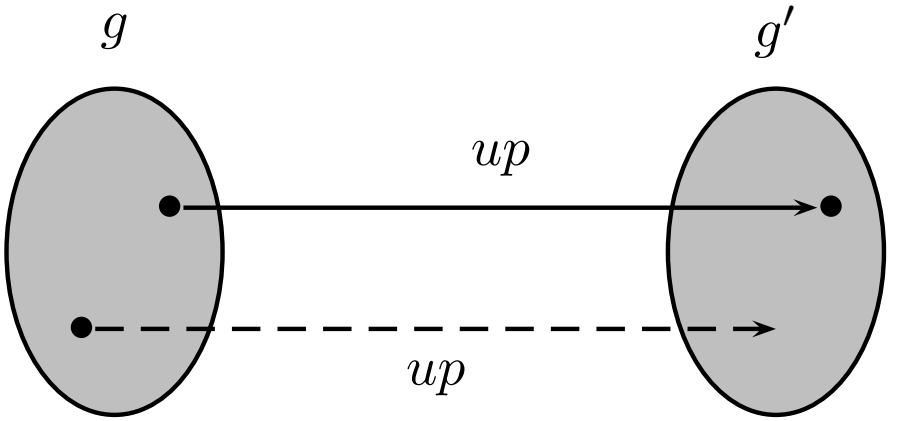
is "non-deterministically" recognized, but not "determiniscally" recognized.

  - same equivalence property
  - closed by concatenation

Decidability

- General model = undecidable
- A decidability condition:

$$\exists v \in g,\ \exists d \in \mathcal{D},\ up(v,d) \in g' \iff \forall v \in g,\ \exists d \in \mathcal{D},\ up(v,d) \in g'$$



Remark: a data automaton obtained from a timed automaton satisfies the decidability condition

Conclusion and further work

- a notion of monoid recognizability for data languages
- an equivalent automaton model, more expressive than timed automata and with a decidability condition
- numerous algebraic properties have to be studied like
  - aperiodic data languages ? *cf* Manuel
  - and if $\mathcal{D}$ is finite ? What is the exact relation with the formal languages case ?
  - power of the monoid *vs* power of the updates
  - ...
- logical characterization? *cf* Manuel
  rational expressions?