μ-calculus over data words

Thomas Colcombet, Amaldev Manuel

LIAFA, Université Paris-Diderot

Data words and languages

A data word $w = (a_1, d_1) \dots (a_n, d_n)$, $a_i \in \Sigma$, $d_i \in \mathcal{D}$ where,

- Σ is a finite alphabet.
- $\mathcal{D}$ is an infinite domain, eg. $\mathbb{N}$

A data language $L \subseteq (\Sigma \times \mathcal{D})^*$ is invariant under permutations of $\mathcal{D}$.

Data words and languages

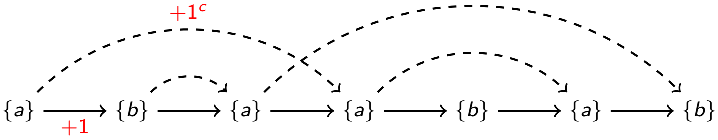
A data word $w = (a_1, d_1) \dots (a_n, d_n)$, $a_i \in \Sigma$, $d_i \in \mathcal{D}$ where,

- Σ is a finite alphabet.
- $\mathcal{D}$ is an infinite domain, eg. $\mathbb{N}$

A data language $L \subseteq (\Sigma \times \mathcal{D})^*$ is invariant under permutations of $\mathcal{D}$.

Example

$$w = \begin{array}{ccccccc} a & b & a & a & b & a & b \\ 1 & 2 & 2 & 1 & 3 & 1 & 2 \end{array}$$



Data word as a graph $([n], \Sigma, +1, +1^c)$

μ-calculus on data words

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathtt{X}^{g}\varphi \mid \mathtt{Y}^{g}\varphi \mid \mathtt{X}^{c}\varphi \mid \mathtt{Y}^{c}\varphi \mid \mu p.\varphi,\ \ p \text{ occurs positively in } \varphi$$

μ-calculus on data words

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathtt{X}^g\varphi \mid \mathtt{Y}^g\varphi \mid \mathtt{X}^c\varphi \mid \mathtt{Y}^c\varphi \mid \mu p.\varphi, \ \ p \text{ occurs positively in } \varphi$$

Semantics

$$\begin{array}{lcl} [\![\varphi]\!]_w & = & \text{“All positions in } w \text{ where } \varphi \text{ holds.”} \\ \\ [\![\mathtt{X}^g\varphi]\!]_w & = & [\![\varphi]\!]_w - 1 \\ [\![\mathtt{X}^c\varphi]\!]_w & = & [\![\varphi]\!]_w - 1^c \\ [\![\mathtt{Y}^g\varphi]\!]_w & = & [\![\varphi]\!]_w + 1 \\ [\![\mathtt{Y}^c\varphi]\!]_w & = & [\![\varphi]\!]_w + 1^c \\ [\![\mu p.\varphi]\!]_w & = & \text{L.f.p of } \varphi(p) \end{array}$$

μ-calculus on data words

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathtt{X}^g\varphi \mid \mathtt{Y}^g\varphi \mid \mathtt{X}^c\varphi \mid \mathtt{Y}^c\varphi \mid \mu p.\varphi, \ \ p \text{ occurs positively in } \varphi$$

Semantics

$$\llbracket\varphi\rrbracket_w = \text{“All positions in } w \text{ where } \varphi \text{ holds.”}$$

$$\begin{aligned}
\llbracket \mathtt{X}^g\varphi\rrbracket_w &= \llbracket\varphi\rrbracket_w - 1 \\
\llbracket \mathtt{X}^c\varphi\rrbracket_w &= \llbracket\varphi\rrbracket_w - 1^c \\
\llbracket \mathtt{Y}^g\varphi\rrbracket_w &= \llbracket\varphi\rrbracket_w + 1 \\
\llbracket \mathtt{Y}^c\varphi\rrbracket_w &= \llbracket\varphi\rrbracket_w + 1^c \\
\llbracket \mu p.\varphi\rrbracket_w &= \text{L.f.p of } \varphi(p)
\end{aligned}$$

$$w \models \varphi \text{ if } \min \in \llbracket\varphi\rrbracket_w$$

$$L(\varphi) = \{w \in (\Sigma \times \mathcal{D})^* \mid w \models \varphi\}$$

Example

$$w \models \mu x.\,(\mathtt{X}^g\mathtt{X}^c x \vee \max) \text{ if } \min + 1 + 1^c \ldots + 1 + 1^c = \max$$

$$\llbracket \nu x.\mathtt{X}^g\mathtt{Y}^c x\rrbracket_w = \{i \mid i + 1^c = i + 1\} = \llbracket\mathcal{S}\rrbracket_w$$

μ-calculus on data words

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathtt{X}^g\varphi \mid \mathtt{Y}^g\varphi \mid \mathtt{X}^c\varphi \mid \mathtt{Y}^c\varphi \mid \mu p.\varphi,\ \ p \text{ occurs positively in } \varphi$$

Semantics

$$\begin{array}{lcl}
[\![\varphi]\!]_w & = & \text{“All positions in } w \text{ where } \varphi \text{ holds.”} \\
\\
[\![\mathtt{X}^g\varphi]\!]_w & = & [\![\varphi]\!]_w - 1 \\
[\![\mathtt{X}^c\varphi]\!]_w & = & [\![\varphi]\!]_w - 1^c \\
[\![\mathtt{Y}^g\varphi]\!]_w & = & [\![\varphi]\!]_w + 1 \\
[\![\mathtt{Y}^c\varphi]\!]_w & = & [\![\varphi]\!]_w + 1^c \\
[\![\mu p.\varphi]\!]_w & = & \text{L.f.p of } \varphi(p)
\end{array}$$

$$w \models \varphi \text{ if } \min \in [\![\varphi]\!]_w$$

$$L(\varphi) = \{w \in (\Sigma \times \mathcal{D})^* \mid w \models \varphi\}$$

Example

$$w \models \mu x.\,(\mathtt{X}^g\mathtt{X}^c x \vee \max) \text{ if } \min + 1 + 1^c \ldots + 1 + 1^c = \max$$

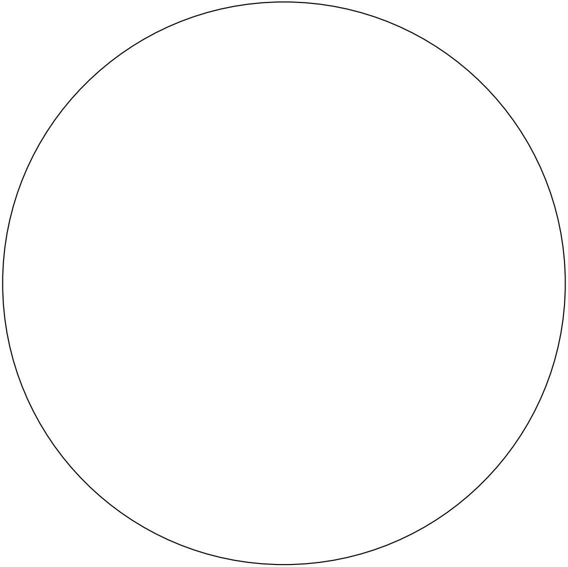
$$[\![\nu x.\mathtt{X}^g\mathtt{Y}^c x]\!]_w = \{i \mid i + 1^c = i + 1\} = [\![\mathcal{S}]\!]_w$$

Equivalent syntax

$$\varphi := p \mid \neg p \mid \mathcal{S} \mid \neg\mathcal{S} \mid \mathcal{P} \mid \neg\mathcal{P} \mid x \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \{\mathtt{X}^g, \mathtt{Y}^g, \mathtt{X}^c, \mathtt{Y}^c, \tilde{\mathtt{X}}^g, \tilde{\mathtt{Y}}^g, \tilde{\mathtt{X}}^c, \tilde{\mathtt{Y}}^c\}\varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

Basic results †

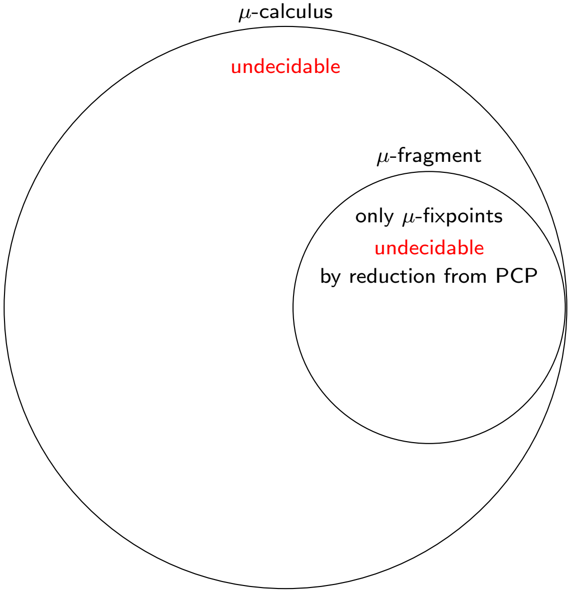
μ-calculus



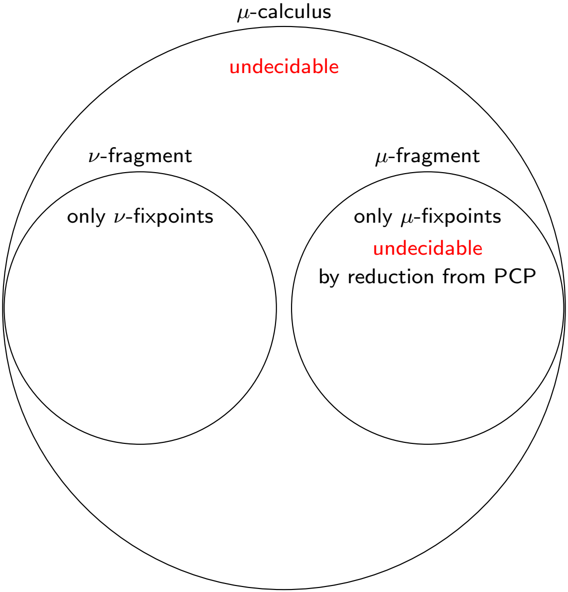
Basic results †

μ-calculus

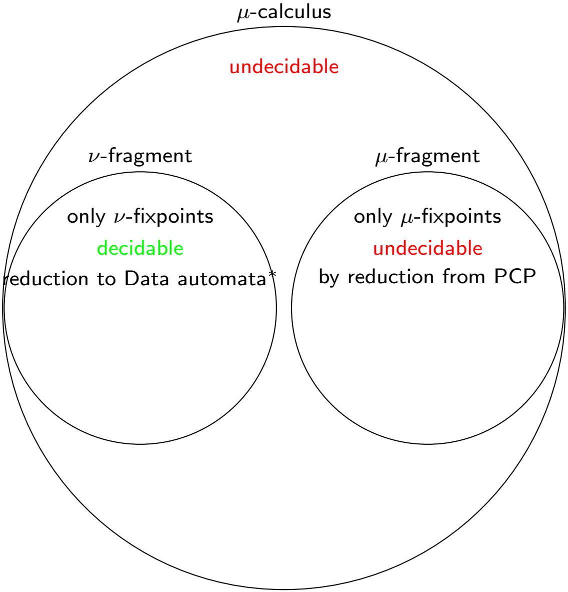
Basic results †



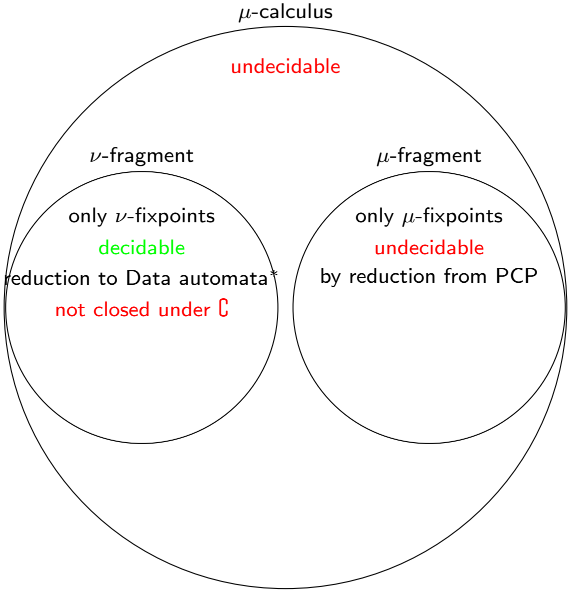
Basic results †



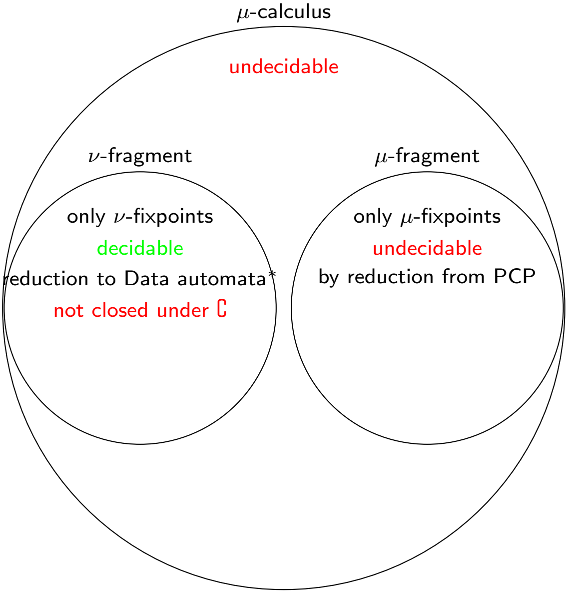
Basic results †



Basic results †



Basic results [†]



---

[*] Mikolaj Bojanczyk et al. "Two-variable logic on data words". In: *ACM Trans. Comput. Log.* 12.4 (2011), p. 27

[†] Mikolaj Bojanczyk and Thomas Schwentick. Private communication.

Fragments Bounded Mode-Alternation and Bounded Reversal

Modalities have direction and mode.

$$\begin{array}{lcl lcl} M_X & = & \{\mathtt{X}^c, \mathtt{X}^g, \tilde{\mathtt{X}}^c, \tilde{\mathtt{X}}^g\} & M_Y & = & \{\mathtt{Y}^c, \mathtt{Y}^g, \tilde{\mathtt{Y}}^c, \tilde{\mathtt{Y}}^g\} \\ M^g & = & \{\mathtt{X}^g, \mathtt{Y}^g, \tilde{\mathtt{X}}^g, \tilde{\mathtt{Y}}^g\} & M^c & = & \{\mathtt{X}^c, \mathtt{Y}^c, \tilde{\mathtt{X}}^c, \tilde{\mathtt{Y}}^c\} \end{array}$$

Fragments Bounded Mode-Alternation and Bounded Reversal

Modalities have direction and mode.

$$\begin{array}{lllllll} M_X & = & \{\mathtt{X}^c, \mathtt{X}^g, \tilde{\mathtt{X}}^c, \tilde{\mathtt{X}}^g\} & & M_Y & = & \{\mathtt{Y}^c, \mathtt{Y}^g, \tilde{\mathtt{Y}}^c, \tilde{\mathtt{Y}}^g\} \\ M^g & = & \{\mathtt{X}^g, \mathtt{Y}^g, \tilde{\mathtt{X}}^g, \tilde{\mathtt{Y}}^g\} & & M^c & = & \{\mathtt{X}^c, \mathtt{Y}^c, \tilde{\mathtt{X}}^c, \tilde{\mathtt{Y}}^c\} \end{array}$$

BR– "every fix-point formula changes direction a bounded number of times"
BMA – "every fix-point formula changes mode a bounded number of times"

Fragments Bounded Mode-Alternation and Bounded Reversal

Modalities have direction and mode.

$$\begin{array}{lcl lcl} M_X & = & \{\mathtt{X}^c, \mathtt{X}^g, \tilde{\mathtt{X}}^c, \tilde{\mathtt{X}}^g\} & M_Y & = & \{\mathtt{Y}^c, \mathtt{Y}^g, \tilde{\mathtt{Y}}^c, \tilde{\mathtt{Y}}^g\} \\ M^g & = & \{\mathtt{X}^g, \mathtt{Y}^g, \tilde{\mathtt{X}}^g, \tilde{\mathtt{Y}}^g\} & M^c & = & \{\mathtt{X}^c, \mathtt{Y}^c, \tilde{\mathtt{X}}^c, \tilde{\mathtt{Y}}^c\} \end{array}$$

BR– "every fix-point formula changes direction a bounded number of times"
BMA – "every fix-point formula changes mode a bounded number of times"

Example

$$\varphi_1 = \nu x.(\tilde{\mathtt{X}}^c x \vee \mathtt{X}^g \mu y.(q \wedge \tilde{\mathtt{Y}}^c y)) \qquad \varphi_2 = \nu x.(\mathtt{X}^c \max \vee \mathtt{X}^c \mathtt{Y}^g x)$$

$$\varphi_3 = \mu x.((\nu y.\, q \vee \mathtt{X}^c y) \vee \mathtt{X}^g x \vee \mathtt{Y}^g x) \qquad \varphi_4 = \mu x.(\mathtt{X}^c \mathtt{X}^g x \vee p)$$

- $\varphi_1 \in BR, \in BMA,$
- $\varphi_2 \notin BR, \notin BMA,$
- $\varphi_3 \notin BR, \in BMA,$
- $\varphi_4 \in BR, \notin BMA.$

Fragments Bounded Mode-Alternation and Bounded Reversal

Formally,

- $\mu\nu(M)$:- formulas using only modalities from M,
- $\mathrm{Comp}(\Psi)$:- smallest set containing Ψ and closed under substitution.

‡ Henrik Björklund and Thomas Schwentick. "On notions of regularity for data languages". In: *Theor. Comput. Sci.* 411.4-5 (2010), pp. 702–715

Fragments Bounded Mode-Alternation and Bounded Reversal

Formally,

- $\mu\nu(M)$:- formulas using only modalities from M,
- $\mathrm{Comp}(\Psi)$:- smallest set containing Ψ and closed under substitution.

$$BR = \mathrm{Comp}\left(\mu\nu\left(M_X\right) \cup \mu\nu\left(M_Y\right)\right)$$

$$BMA = \mathrm{Comp}\left(\mu\nu\left(M^g\right) \cup \mu\nu\left(M^c\right)\right)$$

‡ Henrik Björklund and Thomas Schwentick. "On notions of regularity for data languages". In: *Theor. Comput. Sci.* 411.4-5 (2010), pp. 702–715

Fragments Bounded Mode-Alternation and Bounded Reversal

Formally,

- $\mu\nu(M)$:- formulas using only modalities from M,
- Comp(Ψ) :- smallest set containing Ψ and closed under substitution.

$$BR = \mathrm{Comp}\left(\mu\nu\left(M_X\right) \cup \mu\nu\left(M_Y\right)\right)$$

$$BMA = \mathrm{Comp}\left(\mu\nu\left(M^g\right) \cup \mu\nu\left(M^c\right)\right)$$

Automata Characterization

- BR = cascade of det. class-memory automata‡
- BMA = cascade of finite state automata

‡ Henrik Björklund and Thomas Schwentick. "On notions of regularity for data languages". In: *Theor. Comput. Sci.* 411.4-5 (2010), pp. 702–715

Inside the ν-fragment

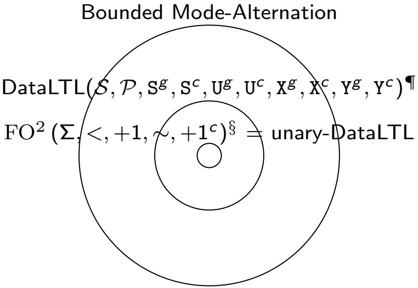
$$\mathrm{FO}^2\,(\boldsymbol{\Sigma}, <, +1, \sim, +1^c)^{\S} = \textsf{unary-DataLTL}$$

Inside the ν-fragment

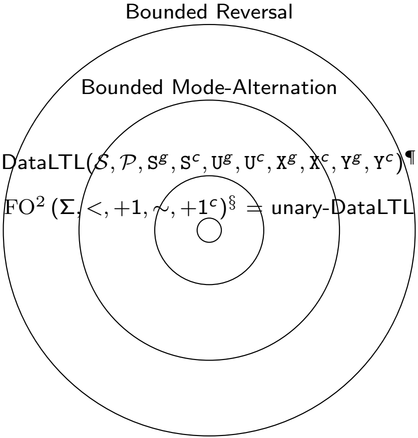
$\mathsf{DataLTL}(\mathcal{S}, \mathcal{P}, \mathtt{S}^g, \mathtt{S}^c, \mathtt{U}^g, \mathtt{U}^c, \mathtt{X}^g, \mathtt{X}^c, \mathtt{Y}^g, \mathtt{Y}^c)^\P$

$\mathrm{FO}^2\,(\mathbf{\Sigma}, <, +1, \sim, +1^c)^\S = \mathsf{unary\text{-}DataLTL}$

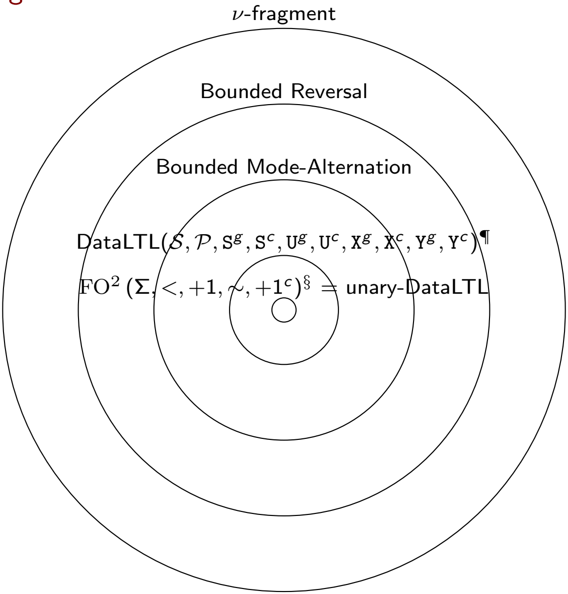
Inside the ν-fragment



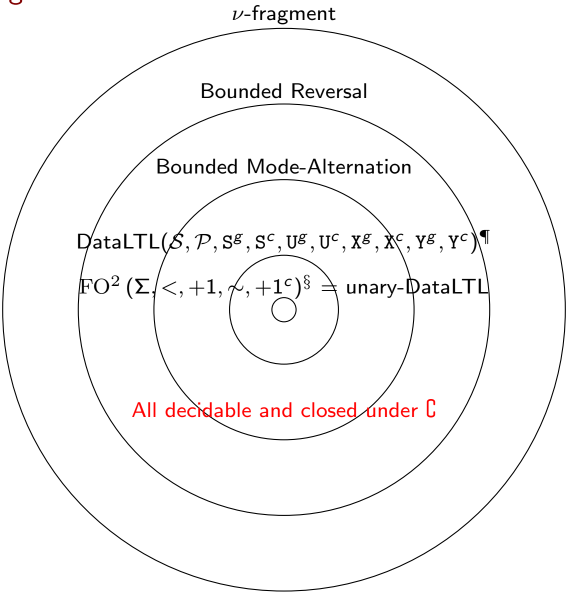
Inside the ν-fragment



Inside the ν-fragment



Inside the ν-fragment



---

§ Mikolaj Bojanczyk et al. "Two-variable logic on data words". In: *ACM Trans. Comput. Log.* 12.4 (2011), p. 27

¶ Ahmet Kara, Thomas Schwentick, and Thomas Zeume. "Temporal Logics on Words with Multiple Data Values". In: *FSTTCS*. Vol. 8. LIPIcs. 2010, pp. 481–492

Separating BMA and BR – via circuits

Combinatorial circuits

- circuits taking sequences of integers as input, defining functions of the form $f : \mathbb{N}^* \to \{0, 1\}$,
- made up of gates of the form $g : E^k \to F$ where $E, F \subseteq \mathbb{N}$ such that either,
  - finitary gates E and F are finite, or
  - binary gates $k \leq 2$.

Separating BMA and BR – via circuits

Combinatorial circuits

- circuits taking sequences of integers as input, defining functions of the form $f : \mathbb{N}^* \to \{0, 1\}$,
- made up of gates of the form $g : E^k \to F$ where $E, F \subseteq \mathbb{N}$ such that either,
  - finitary gates E and F are finite, or
  - binary gates $k \leq 2$.

Example

- binary functions : $+, \times, \text{Prime} : \mathbb{N} \to \{0, 1\}, \ldots$
- finitary functions : $M^k \to M$ (for a monoid M), $f : \{0, 1\}^k \to \{0, 1\}$, ...
- $C_n = \bigwedge_n (zero(x_1), \ldots, zero(x_n))$ is a circuit checking all numbers are non-zero.

Separating BMA and BR – via circuits

Combinatorial circuits

- circuits taking sequences of integers as input, defining functions of the form $f : \mathbb{N}^* \to \{0, 1\}$,
- made up of gates of the form $g : E^k \to F$ where $E, F \subseteq \mathbb{N}$ such that either,
  - finitary gates E and F are finite, or
  - binary gates $k \leq 2$.

Example

- binary functions : $+, \times, \text{Prime} : \mathbb{N} \to \{0, 1\}, \ldots$
- finitary functions : $M^k \to M$ (for a monoid M), $f : \{0, 1\}^k \to \{0, 1\}$, ...
- $C_n = \bigwedge_n(zero(x_1), \ldots, zero(x_n))$ is a circuit checking all numbers are non-zero.

Theorem

There does not exist a family of circuits of constant depth which takes as input $x_1, \ldots, x_k, x_{k+1}$ *and checks if*

$$\sum_{i=1}^{k} x_i = 0 \mod x_{k+1}$$

Separating BR and BMA

Proved using,

Theorem (Gallai-Witt)

For every finite set of colors C and every finite subset $F \subseteq \mathbb{N}^k$,
there is an $n \in \mathbb{N}$ such that all colourings of $[n]^k$ using C has a "translated scaled copy" of F which is monochromatic.
"translated scaled copy of F" :- $\vec{a} + \lambda F$ for some $\vec{a} \in \mathbb{N}^k$ and some positive integer λ.

It follows that,

- BMA $\subsetneq$ BR.

Separating BR and BMA

Proved using,

Theorem (Gallai-Witt)

For every finite set of colors C and every finite subset $F \subseteq \mathbb{N}^k$,
there is an $n \in \mathbb{N}$ such that all colourings of $[n]^k$ using C has a "translated scaled copy" of F which is monochromatic.
"translated scaled copy of F" :- $\vec{a} + \lambda F$ for some $\vec{a} \in \mathbb{N}^k$ and some positive integer λ.

It follows that,

- BMA $\subsetneq$ BR.
- BMA forms a hierarchy under Comp-height (analogous hierarchies for FO^2 and Data-LTL).

Thank you for your attention.