Inside the Class of REGEX Languages

Markus L. Schmid,
Loughborough University, UK

DLT 2012

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$(\mathtt{a} \mid \mathtt{b})^* \quad \mathtt{c}\,\mathtt{b}^*$

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$$(\mathtt{a} \mid \mathtt{b})^* \quad \mathtt{c}\,\mathtt{b}^*$$

$$\{w\mathtt{c}\mathtt{b}^n \quad \mid w \in \{\mathtt{a}, \mathtt{b}\}^*, n \geq 0\}$$

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$(_1\ (\mathtt{a} \mid \mathtt{b})^*\ )_1\, \mathtt{c}\, \mathtt{b}^*$
$\{w\mathtt{c}\mathtt{b}^n \quad \mid w \in \{\mathtt{a}, \mathtt{b}\}^*, n \geq 0\}$

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$(_1\ (\mathtt{a} \mid \mathtt{b})^*\ )_1\, \mathtt{c}\, \mathtt{b}^* \backslash 1$
$\{w \mathtt{c} \mathtt{b}^n \quad \mid w \in \{\mathtt{a}, \mathtt{b}\}^*, n \geq 0\}$

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$(_1\ (\mathtt{a} \mid \mathtt{b})^*\ )_1\, \mathtt{c}\, \mathtt{b}^* \backslash 1$
$\{w\mathtt{c}\mathtt{b}^n w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*, n \geq 0\}$

Extended Regular Expressions with Backreferences (REGEX)

REGEX are a combination of *regular expressions* and the concept of *homomorphic replacement*.

$(_1\, (\mathtt{a} \mid \mathtt{b})^*\,)_1\, \mathtt{c}\, \mathtt{b}^* \backslash 1$
$\{w\mathtt{c}\mathtt{b}^n w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*, n \geq 0\}$

$(_1\, (_2\, \mathtt{a}^*\,)_2\, \mathtt{b}\, \backslash 2\,)_1\, \mathtt{c} \backslash 1$
$\{\mathtt{a}^n\mathtt{b}\mathtt{a}^n\mathtt{c}\mathtt{a}^n\mathtt{b}\mathtt{a}^n \mid n \geq 0\}$

Practical Relevance of REGEX

- REGEX are intensely applied in practice...
  - Traditional and Modern `grep`
  - `vi`
  - Modern `sed`
  - GNU `Emacs`
  - `Perl`
  - `Python`
  - `Java`
  - `.Net`

Practical Relevance of REGEX

- REGEX are intensely applied in practice...
  - Traditional and Modern `grep`
  - `vi`
  - Modern `sed`
  - GNU `Emacs`
  - `Perl`
  - `Python`
  - `Java`
  - `.Net`
- ...even though their membership problem is NP-complete.

An Easy Example

$(_1\ (\mathtt{a} \mid \mathtt{b})^*\mathtt{c}^*\ )_1\ (_2\ (\mathtt{b} \mid \mathtt{d})\ )_2\ (\backslash 1 \mid \backslash 2)$

An Easy Example

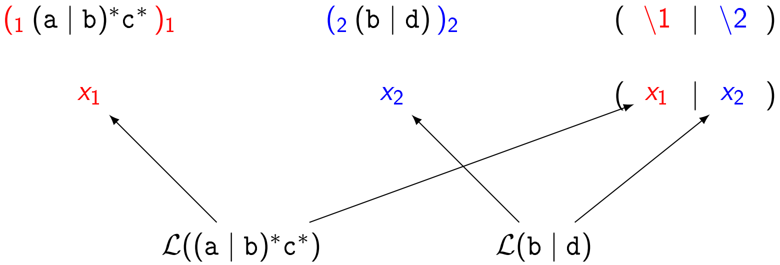
$(_1\ (\mathtt{a} \mid \mathtt{b})^* \mathtt{c}^*\ )_1$ $\quad$ $(_2\ (\mathtt{b} \mid \mathtt{d})\ )_2$ $\quad$ $(\ \backslash 1 \mid \backslash 2\ )$

An Easy Example

$({}_1\ (\mathtt{a} \mid \mathtt{b})^* \mathtt{c}^*\ )_1$ $\quad$ $({}_2\ (\mathtt{b} \mid \mathtt{d})\ )_2$ $\quad$ $(\ \backslash 1 \mid \backslash 2\ )$

x_1 $\quad$ x_2 $\quad$ $(\ x_1 \mid x_2\ )$

An Easy Example



A More Complex Example

$(_1\ (_2\ (_3\ (\mathtt{a} \mid \mathtt{b})^*\ )_3\ \mathtt{c} \backslash 3\ )_2\ \mathtt{d}\ (\ \backslash 2\ \mathtt{c})^*\ )_1\ \mathtt{e}\ \backslash 1$

A More Complex Example

$(_1\ (_2\ (_3\ (\mathtt{a} \mid \mathtt{b})^*\ )_3\ \mathtt{c} \backslash 3\ )_2\ \mathtt{d}\ (\ \backslash 2\ \mathtt{c})^*\ )_1 \quad \mathtt{e} \quad \backslash 1$

A More Complex Example

$({}_1\ ({}_2\ ({}_3\ (\mathtt{a} \mid \mathtt{b})^*\ )_3\ \mathtt{c}\backslash 3\ )_2\ \mathtt{d}\ (\ \backslash 2\ \mathtt{c})^*\ )_1 \quad \mathtt{e} \quad \backslash 1$

x_1

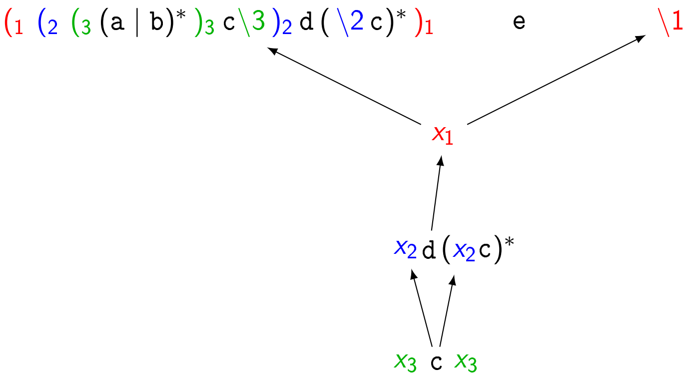
A More Complex Example

$$(_1\ (_2\ (_3\ (\mathtt{a} \mid \mathtt{b})^*\ )_3\ \mathtt{c}\backslash 3\ )_2\ \mathtt{d}\ (\ \backslash 2\ \mathtt{c})^*\ )_1 \qquad \mathtt{e} \qquad \backslash 1$$

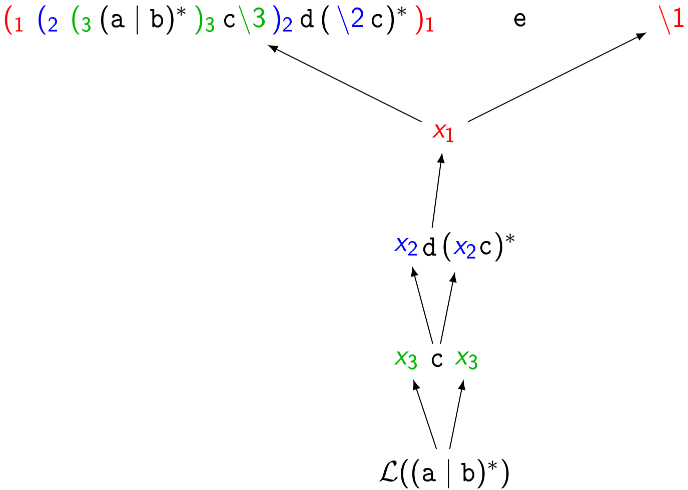
$$x_1$$

$$x_2\, \mathtt{d}\, (x_2\, \mathtt{c})^*$$

A More Complex Example



A More Complex Example



A Fairly Involved Example

$(_1\, \texttt{a}^*\,)_1\, \texttt{c}\, (_2\, \backslash 1\, (\texttt{b} \mid \texttt{c})\, \backslash 1\,)_2\, \texttt{c}\, (_3\, (\backslash 1 \mid \texttt{d})^*\, \backslash 2\,)_3\, \backslash 3$

A Fairly Involved Example

$(_1\ \mathtt{a}^*\ )_1 \quad \mathtt{c} \quad (_2\ \backslash 1\ (\mathtt{b} \mid \mathtt{c})\ \backslash 1\ )_2 \quad \mathtt{c} \quad (_3\ (\backslash 1 \mid \mathtt{d})^*\ \backslash 2\ )_3 \quad \backslash 3$

A Fairly Involved Example

$$(_1\, \mathtt{a}^*\,)_1 \quad \mathtt{c} \quad (_2\, \backslash 1\, (\mathtt{b} \mid \mathtt{c})\, \backslash 1\,)_2 \quad \mathtt{c} \quad (_3\, (\backslash 1 \mid \mathtt{d})^*\, \backslash 2\,)_3 \quad \backslash 3$$

x_1 x_2 x_3

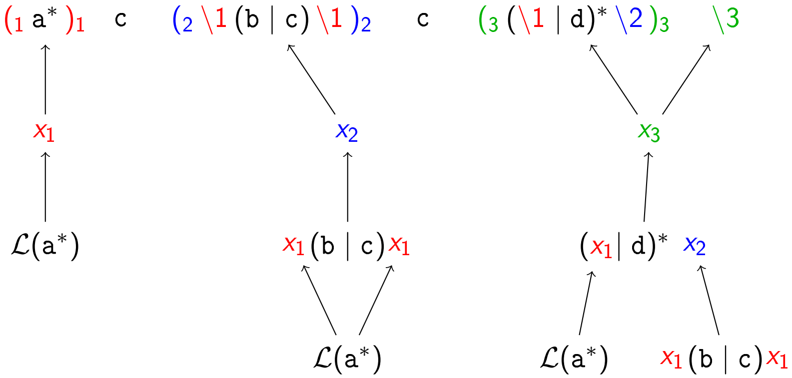
A Fairly Involved Example

$$(_1\, \mathtt{a}^*\,)_1 \quad \mathtt{c} \quad (_2\, \backslash 1\, (\mathtt{b} \mid \mathtt{c})\, \backslash 1\,)_2 \quad \mathtt{c} \quad (_3\, (\backslash 1 \mid \mathtt{d})^*\, \backslash 2\,)_3 \quad \backslash 3$$

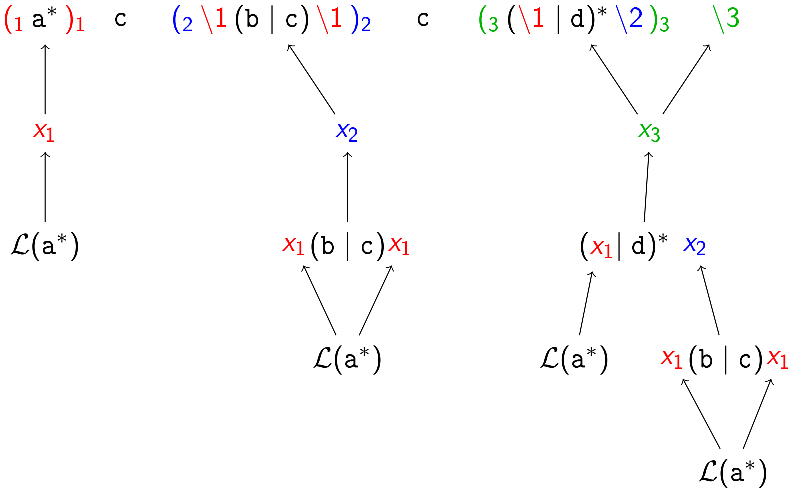
$$x_1 \qquad x_2 \qquad x_3$$

$$\mathcal{L}(\mathtt{a}^*) \qquad x_1\, (\mathtt{b} \mid \mathtt{c})\, x_1 \qquad (x_1 \mid \mathtt{d})^*\ x_2$$

A Fairly Involved Example



A Fairly Involved Example



Aim of This Paper

- Regular expressions on the one hand and homomorphic replacement on the other a well understood concepts in language theory.

Aim of This Paper

- Regular expressions on the one hand and homomorphic replacement on the other a well understood concepts in language theory.
- In REGEX, these two concepts seem inherently entangled and it seems difficulty to treat them separately.

Aim of This Paper

- Regular expressions on the one hand and homomorphic replacement on the other a well understood concepts in language theory.
- In REGEX, these two concepts seem inherently entangled and it seems difficulty to treat them separately.
- Our approach: Study REGEX by investigating alternative ways to combine regular expressions and homomorphic replacement...

Aim of This Paper

- Regular expressions on the one hand and homomorphic replacement on the other a well understood concepts in language theory.
- In REGEX, these two concepts seem inherently entangled and it seems difficulty to treat them separately.
- Our approach: Study REGEX by investigating alternative ways to combine regular expressions and homomorphic replacement...
- ...without exceeding the expressive power of REGEX languages.

Aim of This Paper

- Regular expressions on the one hand and homomorphic replacement on the other a well understood concepts in language theory.
- In REGEX, these two concepts seem inherently entangled and it seems difficulty to treat them separately.
- Our approach: Study REGEX by investigating alternative ways to combine regular expressions and homomorphic replacement...
- ...without exceeding the expressive power of REGEX languages.
- Informally: Take regular expressions, take some mechanism of homomorphic replacement, combine them and see how much of the class of REGEX languages we actually get.

(Typed) Pattern languages

Pattern: A word containing terminals (e. g. $\Sigma = \{a, b, c\}$) and variables ($X := \{x_1, x_2, x_3, \ldots\}$).

(Typed) Pattern languages

Pattern: A word containing terminals (e. g. $\Sigma = \{a, b, c\}$) and variables ($X := \{x_1, x_2, x_3, \ldots\}$).

$\alpha = x_1 \; x_2 \; \mathsf{b} \; x_1 \; x_2 \; x_1$

(Typed) Pattern languages

Pattern: A word containing terminals (e. g. $\Sigma = \{a, b, c\}$) and variables ($X := \{x_1, x_2, x_3, \ldots\}$).

$\alpha = x_1 \ x_2 \ \mathrm{b} \ x_1 \ x_2 \ x_1$

$\mathcal{L}_\Sigma(\alpha) = \{w \mid w = u \ v \ \mathrm{b} \ u \ v \ u, u, v \in \Sigma^*\}.$

(Typed) Pattern languages

Pattern: A word containing terminals (e. g. $\Sigma = \{a, b, c\}$) and variables ($X := \{x_1, x_2, x_3, \ldots\}$).

$\alpha = x_1 \; x_2 \; \mathtt{b} \; x_1 \; x_2 \; x_1$

$\mathcal{L}_\Sigma(\alpha) = \{w \mid w = u \; v \; \mathtt{b} \; u \; v \; u, u, v \in \Sigma^*\}$.

A *type* for α: $\mathcal{T} := (T_{x_1}, T_{x_2})$

(Typed) Pattern languages

Pattern: A word containing terminals (e. g. $\Sigma = \{a, b, c\}$) and variables ($X := \{x_1, x_2, x_3, \ldots\}$).

$\alpha = x_1 \; x_2 \; \mathtt{b} \; x_1 \; x_2 \; x_1$

$\mathcal{L}_\Sigma(\alpha) = \{w \mid w = u \, v \, \mathtt{b} \, u \, v \, u, u, v \in \Sigma^*\}$.

A *type* for α: $\mathcal{T} := (T_{x_1}, T_{x_2})$

$\mathcal{L}_\mathcal{T}(\alpha) = \{w \mid w = u \, v \, \mathtt{b} \, u \, v \, u, u \in T_{x_1}, v \in T_{x_2}\}$.

(Typed) Pattern languages

$\mathrm{PAT} := (\Sigma \cup X)^+$.

(Typed) Pattern languages

$\text{PAT} := (\Sigma \cup X)^+$.

var(α): Set of variables occurring in α.
E. g. $\text{var}(x_1 ab x_2 ba x_1 x_2 c x_3) = \{x_1, x_2, x_3\}$.

(Typed) Pattern languages

$\mathrm{PAT} := (\Sigma \cup X)^+$.

$\mathrm{var}(\alpha)$: Set of variables occurring in α.
E. g. $\mathrm{var}(x_1 ab x_2 ba x_1 x_2 c x_3) = \{x_1, x_2, x_3\}$.

For any language class $\mathfrak{L}$,
$\mathcal{L}_{\mathfrak{L}}(\mathrm{PAT}) := \{\mathcal{L}_{\mathcal{T}}(\alpha) \mid \alpha \in \mathrm{PAT}, \mathcal{T} \in \mathfrak{L}^{|\mathrm{var}(\alpha)|}\}$.

(Typed) Pattern languages

$\mathsf{PAT} := (\Sigma \cup X)^+$.

$\mathsf{var}(\alpha)$: Set of variables occurring in α.
E. g. $\mathsf{var}(x_1 ab x_2 ba x_1 x_2 c x_3) = \{x_1, x_2, x_3\}$.

For any language class $\mathfrak{L}$,
$\mathcal{L}_{\mathfrak{L}}(\mathsf{PAT}) := \{\mathcal{L}_{\mathcal{T}}(\alpha) \mid \alpha \in \mathsf{PAT}, \mathcal{T} \in \mathfrak{L}^{|\mathsf{var}(\alpha)|}\}$.

Proposition

$\mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}) \subseteq \mathcal{L}(\mathsf{REGEX})$.

(Typed) Pattern languages

Idea:

$\mathfrak{L}_1 := \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT})$,

(Typed) Pattern languages

Idea:

$\mathfrak{L}_1 := \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT})$,

$\mathfrak{L}_2 := \mathcal{L}_{\mathfrak{L}_1}(\mathsf{PAT})$,

(Typed) Pattern languages

Idea:

$$\mathfrak{L}_1 := \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}),$$

$$\mathfrak{L}_2 := \mathcal{L}_{\mathfrak{L}_1}(\mathsf{PAT}),$$

$$\mathfrak{L}_3 := \mathcal{L}_{\mathfrak{L}_2}(\mathsf{PAT}),$$

$$\vdots$$

(Typed) Pattern languages

Idea:

$$\mathfrak{L}_1 := \mathcal{L}_{\mathrm{REG}}(\mathrm{PAT}),$$

$$\mathfrak{L}_2 := \mathcal{L}_{\mathfrak{L}_1}(\mathrm{PAT}),$$

$$\mathfrak{L}_3 := \mathcal{L}_{\mathfrak{L}_2}(\mathrm{PAT}),$$

$$\vdots$$

Proposition

For any class of languages $\mathfrak{L}$, $\mathcal{L}_{\mathfrak{L}}(\mathrm{PAT}) = \mathcal{L}_{\mathcal{L}_{\mathfrak{L}}(\mathrm{PAT})}(\mathrm{PAT})$.

(Typed) Pattern languages

Idea:

$$\mathfrak{L}_1 := \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}),$$

$$\mathfrak{L}_2 := \mathcal{L}_{\mathfrak{L}_1}(\mathsf{PAT}),$$

$$\mathfrak{L}_3 := \mathcal{L}_{\mathfrak{L}_2}(\mathsf{PAT}),$$

$$\vdots$$

Proposition

For any class of languages $\mathfrak{L}$, $\mathcal{L}_{\mathfrak{L}}(\mathsf{PAT}) = \mathcal{L}_{\mathcal{L}_{\mathfrak{L}}(\mathsf{PAT})}(\mathsf{PAT})$.

Hence, the aspect of regular expressions cannot be limited to the type languages.

Patterns with Regular Operators

$\mathsf{PAT}_{\mathsf{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathsf{PAT}_{\mathsf{ro}}$ is a *pattern with regular operators*.

Patterns with Regular Operators

$\mathsf{PAT}_{\mathsf{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathsf{PAT}_{\mathsf{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

Patterns with Regular Operators

$\mathsf{PAT}_{\mathsf{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathsf{PAT}_{\mathsf{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

- $\mathcal{T}$ is a type for α and

Patterns with Regular Operators

$\mathsf{PAT}_{\mathsf{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathsf{PAT}_{\mathsf{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

- $\mathcal{T}$ is a type for α and
- $\mathcal{L}(\alpha) = \{\beta_1, \beta_2, \beta_3, \ldots\}$.

Patterns with Regular Operators

$\mathrm{PAT_{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathrm{PAT_{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

- $\mathcal{T}$ is a type for α and
- $\mathcal{L}(\alpha) = \{\beta_1, \beta_2, \beta_3, \ldots\}$.

Example: $\mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}((x_1\,\mathtt{c})^+) =$

Patterns with Regular Operators

$\text{PAT}_{\text{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \text{PAT}_{\text{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

- $\mathcal{T}$ is a type for α and
- $\mathcal{L}(\alpha) = \{\beta_1, \beta_2, \beta_3, \ldots\}$.

Example: $\mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}((x_1\,\mathtt{c})^+) =$
$\mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}) \cup \mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}\,x_1\,\mathtt{c}) \cup \mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}\,x_1\,\mathtt{c}\,x_1\,\mathtt{c}) \cup \ldots =$

Patterns with Regular Operators

$\mathrm{PAT}_{\mathrm{ro}} := \{\alpha \mid \alpha$ is a regular expression over $(\Sigma \cup X)\}$. Every $\alpha \in \mathrm{PAT}_{\mathrm{ro}}$ is a *pattern with regular operators*.

$\mathcal{L}_{\mathcal{T}}(\alpha) := \mathcal{L}_{\mathcal{T}}(\beta_1) \cup \mathcal{L}_{\mathcal{T}}(\beta_2) \cup \mathcal{L}_{\mathcal{T}}(\beta_3) \cup \ldots$, where

- $\mathcal{T}$ is a type for α and
- $\mathcal{L}(\alpha) = \{\beta_1, \beta_2, \beta_3, \ldots\}$.

Example: $\mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}((x_1\,\mathtt{c})^+) =$
$\mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}) \cup \mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}\,x_1\,\mathtt{c}) \cup \mathcal{L}_{(\mathcal{L}(\mathtt{b}^*))}(x_1\,\mathtt{c}\,x_1\,\mathtt{c}\,x_1\,\mathtt{c}) \cup \ldots =$
$\{(\mathtt{b}^n\mathtt{c})^m \mid n \geq 0, m \geq 1\}$.

Expressive Power

Theorem

$\mathcal{L}_{\{\Sigma^*\}}(\mathsf{PAT}) \subset \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}) \subset \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}_{\mathsf{ro}})$.

Iteratively Typing Patterns with Regular Operators

$\mathfrak{L}_{\text{ro},0} := \text{REG},$

Iterative Typing Patterns with Regular Operators

$\mathfrak{L}_{\mathsf{ro},0} := \mathsf{REG},$

$\mathfrak{L}_{\mathsf{ro},1} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},0}}(\mathsf{PAT}_{\mathsf{ro}}) = \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}_{\mathsf{ro}}),$

Iteratively Typing Patterns with Regular Operators

$$\mathfrak{L}_{\mathsf{ro},0} := \mathsf{REG},$$

$$\mathfrak{L}_{\mathsf{ro},1} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},0}}(\mathsf{PAT}_{\mathsf{ro}}) = \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}_{\mathsf{ro}}),$$

$$\mathfrak{L}_{\mathsf{ro},2} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},1}}(\mathsf{PAT}_{\mathsf{ro}}),$$

Iteratively Typing Patterns with Regular Operators

$$\mathfrak{L}_{\text{ro},0} := \text{REG},$$

$$\mathfrak{L}_{\text{ro},1} := \mathcal{L}_{\mathfrak{L}_{\text{ro},0}}(\text{PAT}_{\text{ro}}) = \mathcal{L}_{\text{REG}}(\text{PAT}_{\text{ro}}),$$

$$\mathfrak{L}_{\text{ro},2} := \mathcal{L}_{\mathfrak{L}_{\text{ro},1}}(\text{PAT}_{\text{ro}}),$$

$$\mathfrak{L}_{\text{ro},3} := \mathcal{L}_{\mathfrak{L}_{\text{ro},2}}(\text{PAT}_{\text{ro}}),$$

$$\vdots$$

Iteratively Typing Patterns with Regular Operators

$$\mathfrak{L}_{\mathsf{ro},0} := \mathsf{REG},$$

$$\mathfrak{L}_{\mathsf{ro},1} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},0}}(\mathsf{PAT}_{\mathsf{ro}}) = \mathcal{L}_{\mathsf{REG}}(\mathsf{PAT}_{\mathsf{ro}}),$$

$$\mathfrak{L}_{\mathsf{ro},2} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},1}}(\mathsf{PAT}_{\mathsf{ro}}),$$

$$\mathfrak{L}_{\mathsf{ro},3} := \mathcal{L}_{\mathfrak{L}_{\mathsf{ro},2}}(\mathsf{PAT}_{\mathsf{ro}}),$$

$$\vdots$$

$$\mathfrak{L}_{\mathsf{ro},\infty} := \bigcup_{i=0}^{\infty} \mathfrak{L}_{\mathsf{ro},i}\,.$$

Iteratively Typing Patterns with Regular Operators

$$\mathfrak{L}_{\text{ro},0} := \text{REG},$$

$$\mathfrak{L}_{\text{ro},1} := \mathcal{L}_{\mathfrak{L}_{\text{ro},0}}(\text{PAT}_{\text{ro}}) = \mathcal{L}_{\text{REG}}(\text{PAT}_{\text{ro}}),$$

$$\mathfrak{L}_{\text{ro},2} := \mathcal{L}_{\mathfrak{L}_{\text{ro},1}}(\text{PAT}_{\text{ro}}),$$

$$\mathfrak{L}_{\text{ro},3} := \mathcal{L}_{\mathfrak{L}_{\text{ro},2}}(\text{PAT}_{\text{ro}}),$$

$$\vdots$$

$$\mathfrak{L}_{\text{ro},\infty} := \bigcup_{i=0}^{\infty} \mathfrak{L}_{\text{ro},i}.$$

Theorem

$\mathfrak{L}_{\text{ro},0} \subset \mathfrak{L}_{\text{ro},1} \subset \mathfrak{L}_{\text{ro},2} \subseteq \mathfrak{L}_{\text{ro},3} \subseteq \mathfrak{L}_{\text{ro},4} \subseteq \ldots.$

Pattern Expressions

Introduced by Câmpeanu and Yu, 2004.

A *pattern expression* is a tuple

$$(x_1 \rightarrow r_1, x_2 \rightarrow r_2, \ldots, x_n \rightarrow r_n)\,,$$

Pattern Expressions

Introduced by Câmpeanu and Yu, 2004.

A *pattern expression* is a tuple

$$(x_1 \to r_1, x_2 \to r_2, \ldots, x_n \to r_n)\,,$$

where

- $\text{var}(r_1) = \emptyset$,
- $\text{var}(r_2) \subseteq \{x_1\}$,
- $\text{var}(r_3) \subseteq \{x_1, x_2\}$,
- $\text{var}(r_4) \subseteq \{x_1, x_2, x_3\}$,

  $\vdots$

Pattern Expressions

Introduced by Câmpeanu and Yu, 2004.

A *pattern expression* is a tuple

$$(x_1 \to r_1, x_2 \to r_2, \ldots, x_n \to r_n),$$

where

- $\text{var}(r_1) = \emptyset$,
- $\text{var}(r_2) \subseteq \{x_1\}$,
- $\text{var}(r_3) \subseteq \{x_1, x_2\}$,
- $\text{var}(r_4) \subseteq \{x_1, x_2, x_3\}$,

$\vdots$

The set of all pattern expressions is denoted by PE.

Pattern Expressions

Introduced by Câmpeanu and Yu, 2004.

A *pattern expression* is a tuple

$$(x_1 \to r_1, x_2 \to r_2, \dots, x_n \to r_n)\,,$$

where

- $\mathsf{var}(r_1) = \emptyset$,
- $\mathsf{var}(r_2) \subseteq \{x_1\}$,
- $\mathsf{var}(r_3) \subseteq \{x_1, x_2\}$,
- $\mathsf{var}(r_4) \subseteq \{x_1, x_2, x_3\}$,

$$\vdots$$

The set of all pattern expressions is denoted by PE.

Example: $q := (x_1 \to \mathtt{a}^*, x_2 \to x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \to x_1 \mathtt{c} x_2)$.

Pattern Expression Languages

$$q := (x_1 \to \mathtt{a}^*, x_2 \to x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \to x_1\mathtt{c}x_2),$$

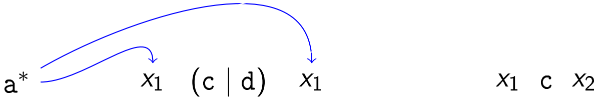
Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$

$\mathtt{a}^*$ x_1 $(\mathtt{c} \mid \mathtt{d})$ x_1 x_1 $\mathtt{c}$ x_2

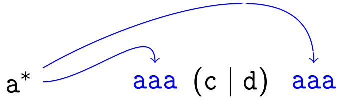
Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1\mathtt{c}x_2),$$



Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$



x_1 c x_2

Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1\mathtt{c}x_2),$$

$\mathtt{a}^*$ $\quad$ $\mathtt{aaa}\ (\mathtt{c} \mid \mathtt{d})\ \mathtt{aaa}$ $\quad$ $x_1\ \mathtt{c}\ x_2$

Pattern Expression Languages

$$q := (x_1 \to \mathtt{a}^*, x_2 \to x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \to x_1\mathtt{c}x_2),$$

$\mathtt{a}^*$ $\quad$ $\mathtt{aaa}\ (\mathtt{c} \mid \mathtt{d})\ \mathtt{aaa}$ $\quad$ $x_1\ \mathtt{c}\ \mathtt{aaacaaa}$

Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1\mathtt{c}x_2),$$

$\mathtt{a}^*$ $\mathtt{aaa}$ $(\mathtt{c} \mid \mathtt{d})$ $\mathtt{aaa}$ x_1 $\mathtt{c}$ $\mathtt{aaacaaa}$

Pattern Expression Languages

$$q := (x_1 \to \mathtt{a}^*, x_2 \to x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \to x_1\mathtt{c}x_2),$$

$\mathtt{a}^*$ $\quad$ $\mathtt{aaa}\ (\mathtt{c} \mid \mathtt{d})\ \mathtt{aaa}$ $\quad$ $\mathtt{a}\ \ \mathtt{c}\ \ \mathtt{aaacaaa}$

Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$

$\mathtt{a}^*$ $\mathtt{aaa}$ $(\mathtt{c} \mid \mathtt{d})$ $\mathtt{aaa}$ $\mathtt{a}$ $\mathtt{c}$ $\mathtt{aaacaaa}$

Pattern Expression Languages

$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2)$,

$\mathtt{a}^*$ $\mathtt{aaa}$ $(\mathtt{c} \mid \mathtt{d})$ $\mathtt{aaa}$ $\mathtt{a}$ $\mathtt{c}$ $\mathtt{aaacaaa}$

$\mathcal{L}_{\text{it}}(q) = \{\mathtt{a}^k \mathtt{c} \mathtt{a}^m u \mathtt{a}^m \mid k, m \in \mathbb{N}_0, u \in \{\mathtt{c}, \mathtt{d}\}\}$ is the *language generated by q with respect to iterated substitution*.

Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1\mathtt{c}x_2),$$

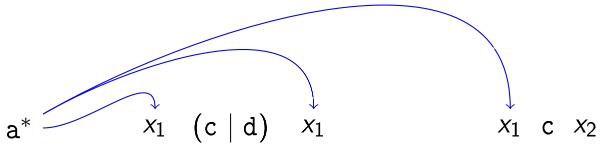
Pattern Expression Languages

$$q := (x_1 \rightarrow \texttt{a}^*, x_2 \rightarrow x_1(\texttt{c} \mid \texttt{d})x_1, x_3 \rightarrow x_1 \texttt{c} x_2),$$

$\texttt{a}^*$ x_1 $(\texttt{c} \mid \texttt{d})$ x_1 x_1 $\texttt{c}$ x_2

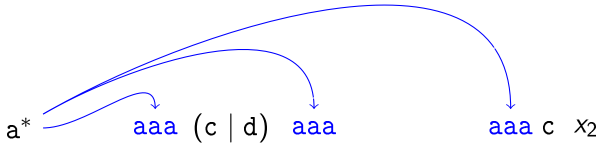
Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$



Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$



Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$

$\mathtt{a}^*$ $\quad$ $\mathtt{aaa}\ (\mathtt{c} \mid \mathtt{d})\ \mathtt{aaa}$ $\quad$ $\mathtt{aaa}\ \mathtt{c}\ x_2$

Pattern Expression Languages

$$q := (x_1 \rightarrow \texttt{a}^*, x_2 \rightarrow x_1(\texttt{c} \mid \texttt{d})x_1, x_3 \rightarrow x_1\texttt{c}x_2),$$

$\texttt{a}^*$ aaa (c | d) aaa aaa c aaadaaa

Pattern Expression Languages

$$q := (x_1 \rightarrow \mathtt{a}^*, x_2 \rightarrow x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \rightarrow x_1 \mathtt{c} x_2),$$

$\mathtt{a}^*$ $\quad$ $\mathtt{aaa}\ (\mathtt{c} \mid \mathtt{d})\ \mathtt{aaa}$ $\quad$ $\mathtt{aaa}\ \mathtt{c}\ \mathtt{aaadaaa}$

Pattern Expression Languages

$q := (x_1 \to \mathtt{a}^*, x_2 \to x_1(\mathtt{c} \mid \mathtt{d})x_1, x_3 \to x_1 \mathtt{c} x_2)$,

$\mathtt{a}^*$ $\mathtt{aaa}$ $(\mathtt{c} \mid \mathtt{d})$ $\mathtt{aaa}$ $\mathtt{aaa}$ $\mathtt{c}$ $\mathtt{aaadaaa}$

$\mathcal{L}_{\mathsf{uni}}(q) = \{\mathtt{a}^m \mathtt{c} \mathtt{a}^m u \mathtt{a}^m \mid m \in \mathbb{N}_0, u \in \{\mathtt{c}, \mathtt{d}\}\}$ is the *language generated by q with respect to uniform substitution.*

Pattern Expression Languages

Proposition

[Campeanu and Yu] For every $p \in \mathrm{PE}$, $\mathcal{L}_{\mathrm{it}}(p)$ is a REGEX language.

Theorem

$\mathfrak{L}_{\mathrm{ro},\infty} = \mathcal{L}_{\mathrm{it}}(\mathrm{PE})$.

Iterated vs. Uniform Substitution

Proposition

Let $p := (x_1 \rightarrow r_1, \ldots, x_m \rightarrow r_m) \in \mathsf{PE}$.

- $\mathcal{L}_{\text{uni}}(p) \subseteq \mathcal{L}_{\text{it}}(p)$,
- if, for every i, j, $1 \leq i < j \leq m$, $\text{var}(r_i) \cap \text{var}(r_j) = \emptyset$, then $\mathcal{L}_{\text{it}}(p) \subseteq \mathcal{L}_{\text{uni}}(p)$.

Iterated vs. Uniform Substitution

Proposition

Let $p := (x_1 \to r_1, \ldots, x_m \to r_m) \in \mathrm{PE}$.

- $\mathcal{L}_{\mathrm{uni}}(p) \subseteq \mathcal{L}_{\mathrm{it}}(p)$,
- if, for every i, j, $1 \leq i < j \leq m$, $\mathrm{var}(r_i) \cap \mathrm{var}(r_j) = \emptyset$, then $\mathcal{L}_{\mathrm{it}}(p) \subseteq \mathcal{L}_{\mathrm{uni}}(p)$.

Theorem

$\mathcal{L}_{\mathrm{it}}(\mathrm{PE}) \subset \mathcal{L}_{\mathrm{uni}}(\mathrm{PE})$.

Notation

A REGEX r is *star-free initialised* iff every referenced subexpression does not occur under a star.

Notation

A REGEX r is *star-free initialised* iff every referenced subexpression does not occur under a star.

- $((_1(\mathtt{a} \mid \mathtt{b})^*)_1\mathtt{b}\backslash 1)^*\mathtt{b}\backslash 1$
- $(_1(\mathtt{a} \mid \mathtt{b})^*)_1\backslash 1(_2\mathtt{c}^*)_2(\mathtt{d}\backslash 1\backslash 2)^*$

PE w. r. t. uniform subst. vs. star-free initialised REGEX

Lemma

For every pattern expression p, there exists a star-free initialised REGEX r *with* $\mathcal{L}_{\mathsf{uni}}(p) = \mathcal{L}(r)$.

PE w. r. t. uniform subst. vs. star-free initialised REGEX

Lemma

For every pattern expression p, there exists a star-free initialised REGEX r *with* $\mathcal{L}_{\text{uni}}(p) = \mathcal{L}(r)$.

Lemma

For every star-free initialised REGEX r*, there exists a pattern expression* p *with* $\mathcal{L}(r) = \mathcal{L}_{\text{uni}}(p)$.

PE w. r. t. uniform subst. vs. star-free initialised REGEX

Lemma

For every pattern expression p, there exists a star-free initialised REGEX r *with* $\mathcal{L}_{\text{uni}}(p) = \mathcal{L}(r)$.

Lemma

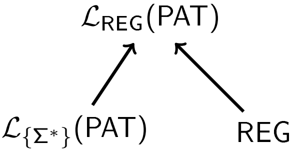
For every star-free initialised REGEX r*, there exists a pattern expression* p *with* $\mathcal{L}(r) = \mathcal{L}_{\text{uni}}(p)$.

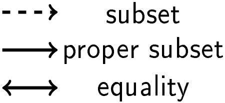
Theorem

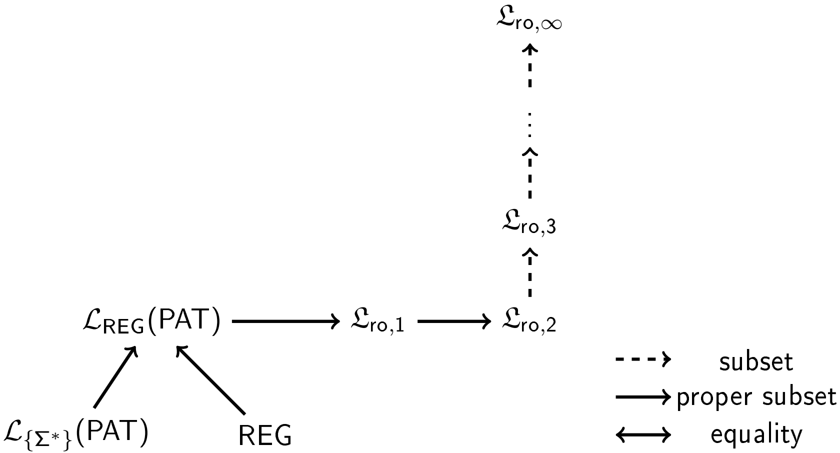
$\mathcal{L}(\text{REGEX}_{\text{sfi}}) = \mathcal{L}_{\text{uni}}(\text{PE})$.

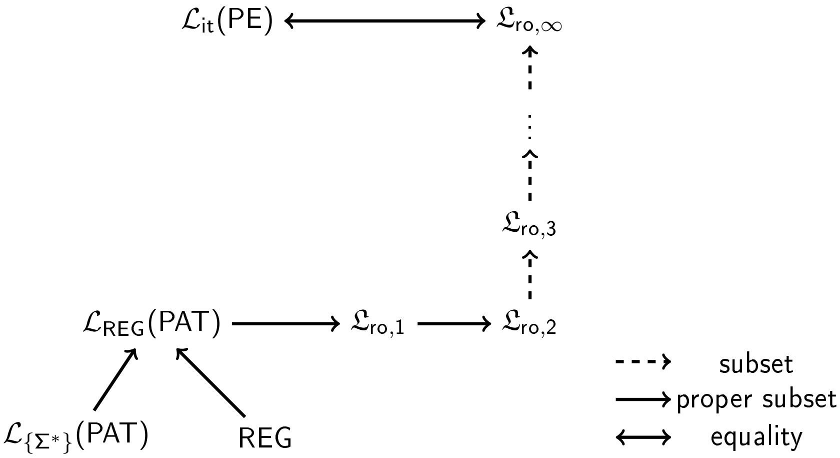
$\mathcal{L}_{\{\Sigma^*\}}(\text{PAT})$ REG

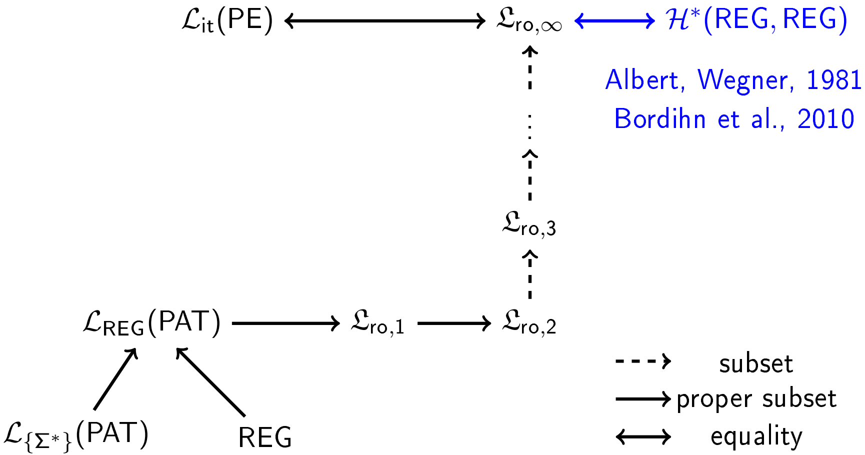
- - -> subset
⟶ proper subset
⟷ equality

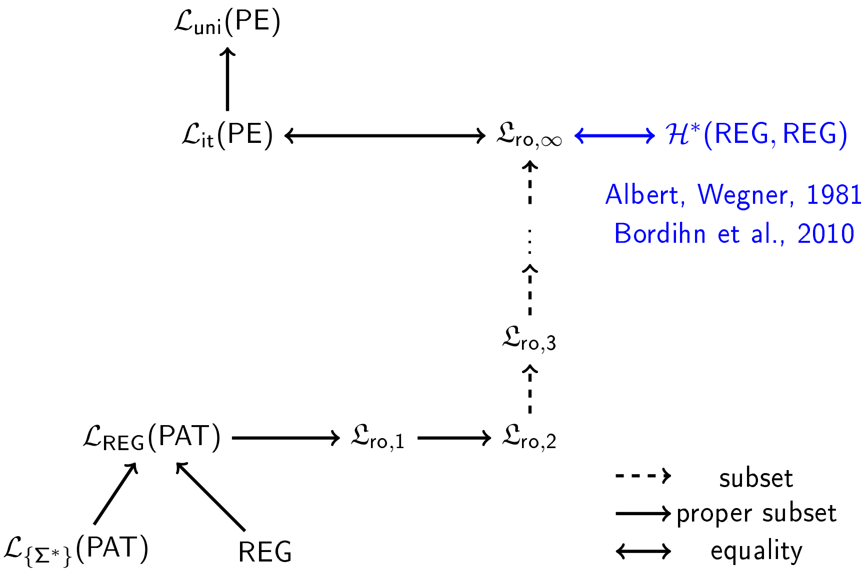

$\mathcal{L}_{\mathsf{REG}}(\mathsf{PAT})$
$\mathcal{L}_{\{\Sigma^*\}}(\mathsf{PAT})$
REG

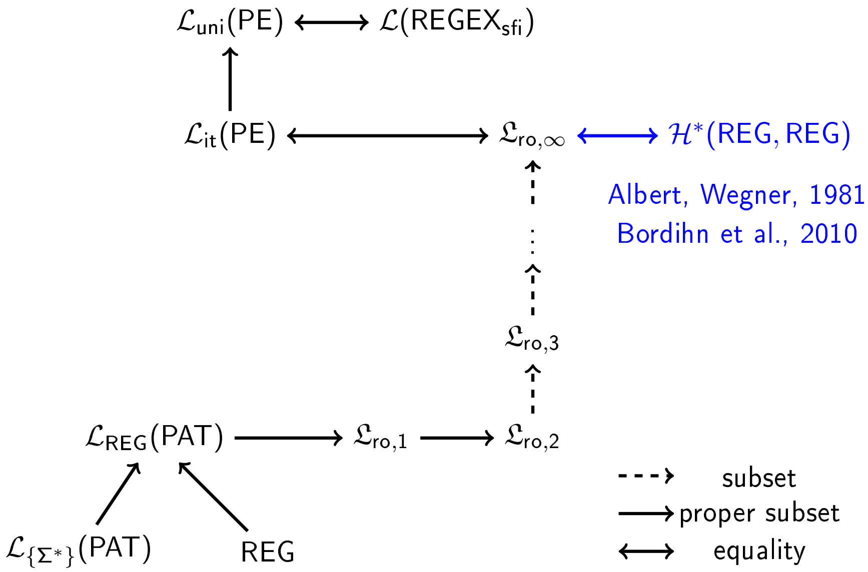

subset
proper subset
equality

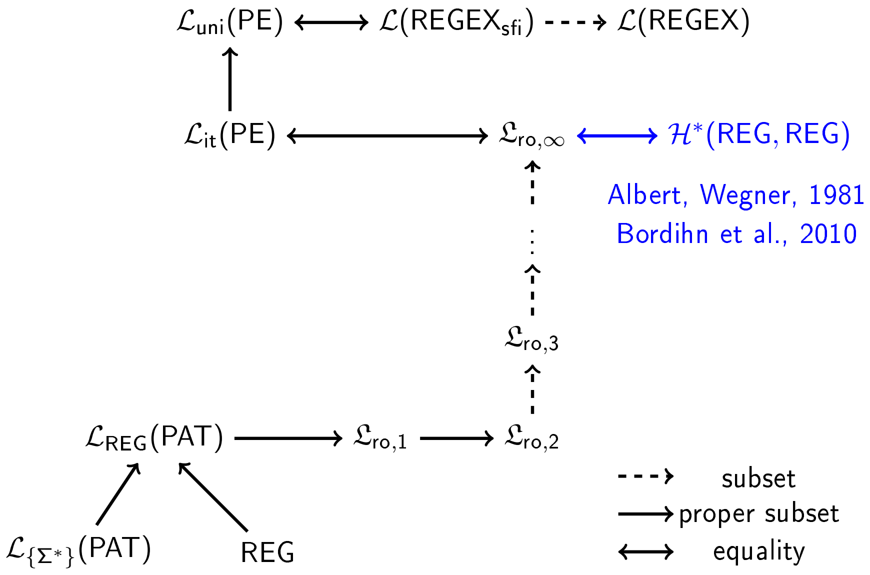

$\mathfrak{L}_{\text{ro},\infty}$
$\mathfrak{L}_{\text{ro},3}$
$\mathcal{L}_{\text{REG}}(\text{PAT})$
$\mathfrak{L}_{\text{ro},1}$
$\mathfrak{L}_{\text{ro},2}$
$\mathcal{L}_{\{\Sigma^*\}}(\text{PAT})$
REG
subset
proper subset
equality


$\mathcal{L}_{it}(PE)$
$\mathfrak{L}_{ro,\infty}$
$\mathfrak{L}_{ro,3}$
$\mathcal{L}_{REG}(PAT)$
$\mathfrak{L}_{ro,1}$
$\mathfrak{L}_{ro,2}$
$\mathcal{L}_{\{\Sigma^*\}}(PAT)$
REG
subset
proper subset
equality


$\mathcal{L}_{it}(PE)$
$\mathfrak{L}_{ro,\infty}$
$\mathcal{H}^*(REG, REG)$
Albert, Wegner, 1981
Bordihn et al., 2010
$\mathfrak{L}_{ro,3}$
$\mathcal{L}_{REG}(PAT)$
$\mathfrak{L}_{ro,1}$
$\mathfrak{L}_{ro,2}$
$\mathcal{L}_{\{\Sigma^*\}}(PAT)$
REG
subset
proper subset
equality


$\mathcal{L}_{\mathrm{uni}}(\mathrm{PE})$
$\mathcal{L}_{\mathrm{it}}(\mathrm{PE})$
$\mathfrak{L}_{\mathrm{ro},\infty}$
$\mathcal{H}^*(\mathrm{REG},\mathrm{REG})$
Albert, Wegner, 1981
Bordihn et al., 2010
$\mathfrak{L}_{\mathrm{ro},3}$
$\mathcal{L}_{\mathrm{REG}}(\mathrm{PAT})$
$\mathfrak{L}_{\mathrm{ro},1}$
$\mathfrak{L}_{\mathrm{ro},2}$
$\mathcal{L}_{\{\Sigma^*\}}(\mathrm{PAT})$
REG
subset
proper subset
equality


$\mathcal{L}_{\text{uni}}(\text{PE})$
$\mathcal{L}(\text{REGEX}_{\text{sfi}})$
$\mathcal{L}_{\text{it}}(\text{PE})$
$\mathfrak{L}_{\text{ro},\infty}$
$\mathcal{H}^*(\text{REG}, \text{REG})$
Albert, Wegner, 1981
Bordihn et al., 2010
$\mathfrak{L}_{\text{ro},3}$
$\mathcal{L}_{\text{REG}}(\text{PAT})$
$\mathfrak{L}_{\text{ro},1}$
$\mathfrak{L}_{\text{ro},2}$
$\mathcal{L}_{\{\Sigma^*\}}(\text{PAT})$
REG
subset
proper subset
equality


$\mathcal{L}_{\text{uni}}(\text{PE})$
$\mathcal{L}(\text{REGEX}_{\text{sfi}})$
$\mathcal{L}(\text{REGEX})$
$\mathcal{L}_{\text{it}}(\text{PE})$
$\mathfrak{L}_{\text{ro},\infty}$
$\mathcal{H}^*(\text{REG}, \text{REG})$
Albert, Wegner, 1981
Bordihn et al., 2010
$\mathfrak{L}_{\text{ro},3}$
$\mathcal{L}_{\text{REG}}(\text{PAT})$
$\mathfrak{L}_{\text{ro},1}$
$\mathfrak{L}_{\text{ro},2}$
$\mathcal{L}_{\{\Sigma^*\}}(\text{PAT})$
REG
subset
proper subset
equality