REGULAR PROGRAMMING FOR QUANTITATIVE PROPERTIES OF DATA STREAMS

Rajeev Alur Dana Fisman Mukund Raghothaman

ESOP 2016

University of Pennsylvania

DATA STREAMS

WEB-PAGE REQUESTS

- P2
- EOD
- P2
- P1
- P1
- EOM
- P2
- P2

- P2
- P1
- EOD
- P2
- P2
- P1
- P2
- P1

- EOM
- P2
- P2
- P2
- P1
- P2
- EOM
- P1

- P2
- EOD
- P2
- P2
- P2
- P2
- EOM
- P1
- ...

- What is the maximum number of daily requests for P2?

$$\textit{iter-max}(\textit{day-count}_{\text{P2}})$$

- What is the maximum number of daily requests for P2?

$$\textit{iter-max}(\textit{day-count}_{\text{P2}})$$

- What is the number of requests for P1 in an average month?

$$\textit{iter-avg}(\textit{month-count}_{\text{P1}})$$

- What is the maximum number of daily requests for P2?

$$\textit{iter-max}(\textit{day-count}_{\text{P2}})$$

- What is the number of requests for P1 in an average month?

$$\textit{iter-avg}(\textit{month-count}_{\text{P1}})$$

- What is the total number of requests for P1?

$$\textit{count}_{\text{P1}} = \textit{iter-sum}(\textit{day-count}_{\text{P1}})$$

- What is the maximum number of daily requests for P2?

$$\textit{iter-max}(\textit{day-count}_{\text{P2}})$$

- What is the number of requests for P1 in an average month?

$$\textit{iter-avg}(\textit{month-count}_{\text{P1}})$$

- What is the total number of requests for P1?

$$\textit{count}_{\text{P1}} = \textit{iter-sum}(\textit{day-count}_{\text{P1}})$$

- What is the maximum of the total number of requests for P1 and P2?

$$\max(\textit{count}_{\text{P1}}, \textit{count}_{\text{P2}})$$

QUANTITATIVE REGULAR EXPRESSIONS

Languages	$\Sigma^* \to \mathtt{bool}$	$\equiv$	Regular expressions
Cost functions	$\Sigma^* \to \mathbb{R}$	$\equiv$	QREs

FUNCTION COMBINATORS

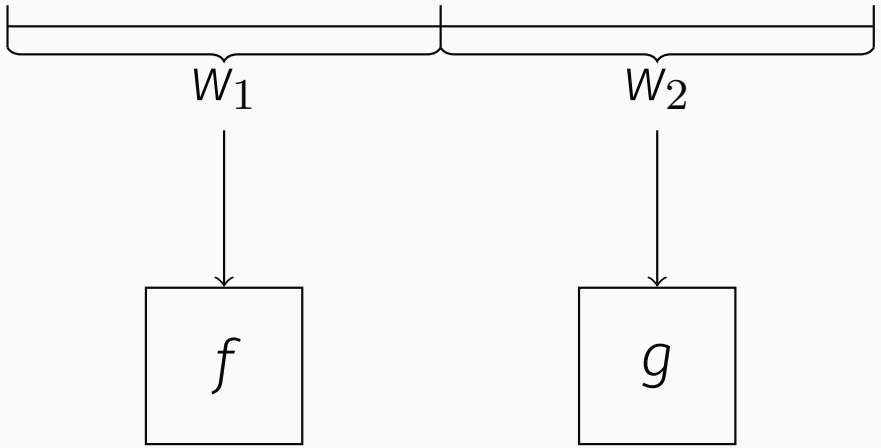
$a \mapsto d$

- If $w = a$, then output d, otherwise undefined
- P1 $\mapsto 0$, P1 $\mapsto 1$, EOD $\mapsto 5$, ...
- Analogue of basic regular expressions

f else g

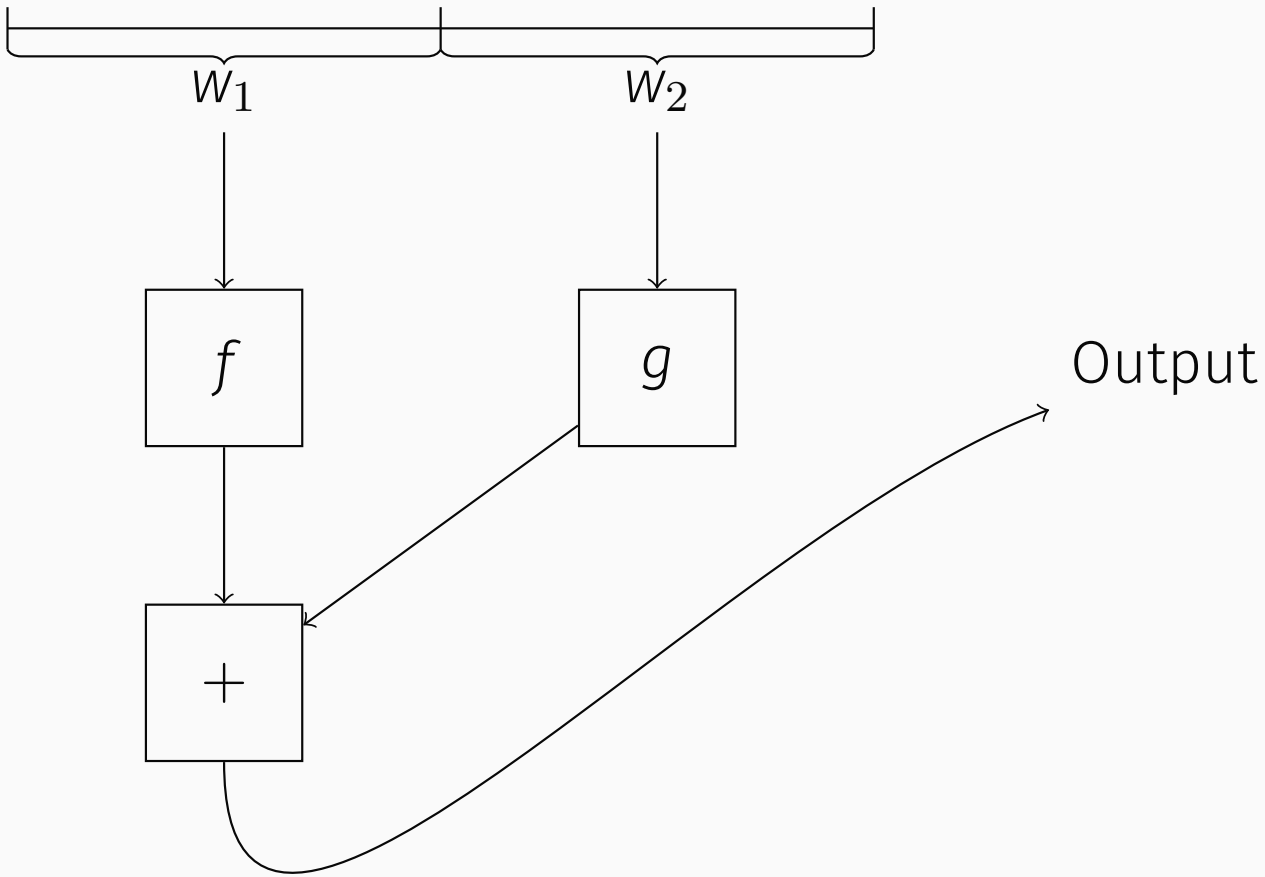
- If $f(w)$ is defined, then output $f(w)$, otherwise output $g(w)$
- Analogue of regular expression union

- $f + g$: Map w to $f(w) + g(w)$
- $f - g$: Map w to $f(w) - g(w)$
- $\max(f, g)$: Map w to $\max(f(w), g(w))$
- ...
- $op(f_1, f_2, \ldots, f_k)$: Map w to $op(f_1(w), f_2(w), \ldots, f_k(w))$
- Analogue of regular expression intersection

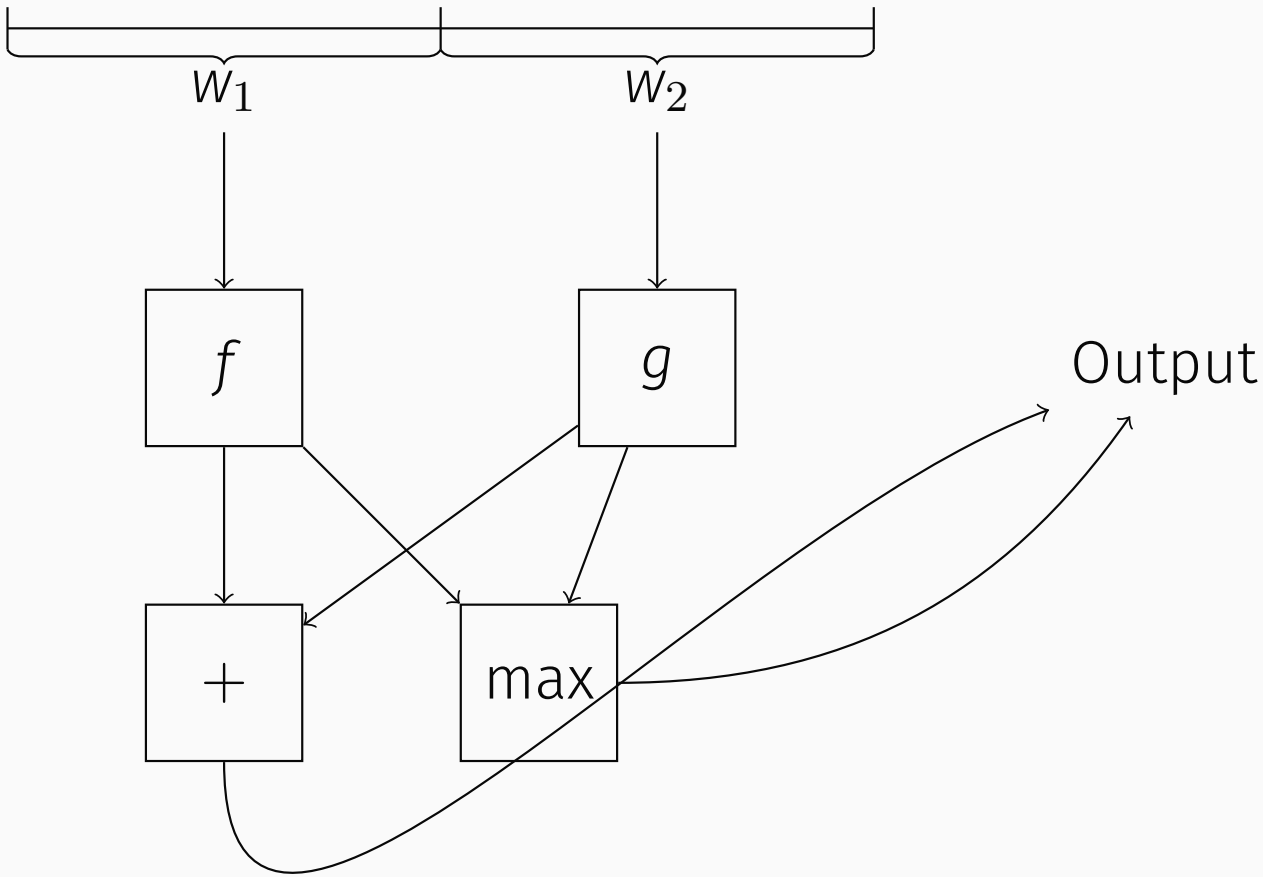


Output

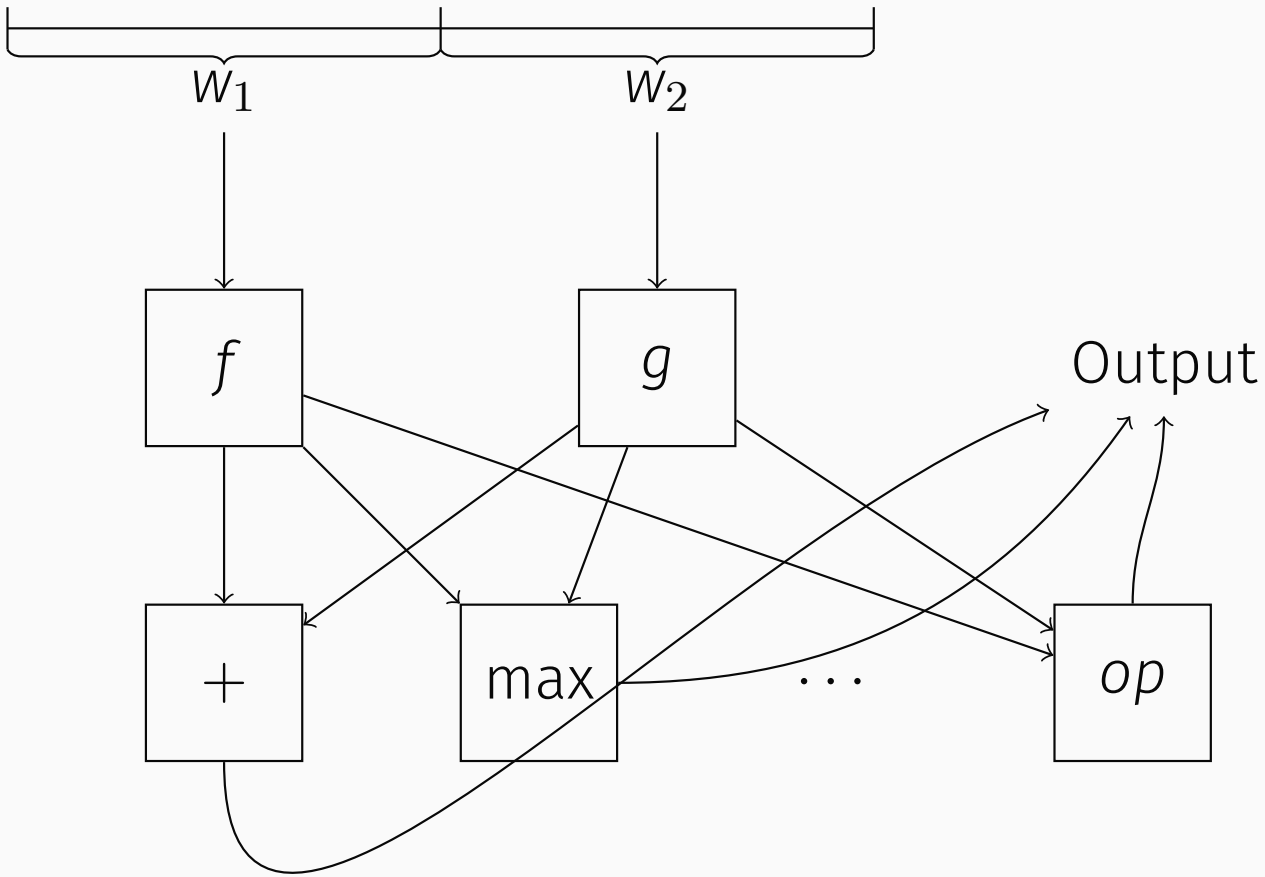
split-plus(f, g)

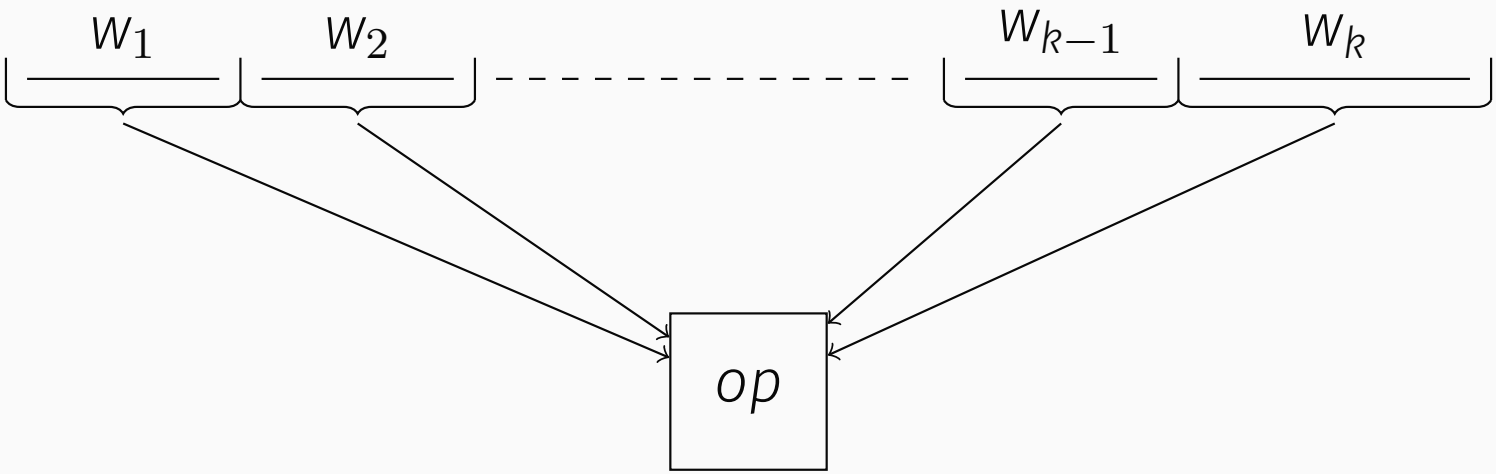


split-plus(f, g) , *split-max*(f, g)



split-plus(f, g) , *split-max*(f, g) , ..., *split-op*(f, g)




w_1
w_2
w_{k-1}
w_k
op

Basic functions: $a \mapsto d$

Conditional choice: f *else* g

Concatenation: *split-op*(f, g)

Function iteration: *iter-op*(f)

Cost operations: $op(f_1, f_2, \ldots, f_k)$

Basic functions: $a \mapsto d$

Conditional choice: f *else* g

Concatenation: *split-op*(f, g)

Function iteration: *iter-op*(f)

Cost operations: $op(f_1, f_2, \ldots, f_k)$

- What about non-binary, non-commutative or non-associative operators?

Basic functions: $a \mapsto d$

Conditional choice: f *else* g

Concatenation: *split-op*(f, g)

Function iteration: *iter-op*(f)

Cost operations: $op(f_1, f_2, \ldots, f_k)$

- What about non-binary, non-commutative or non-associative operators?
- Are these operators "sufficient"? If we add more operators?

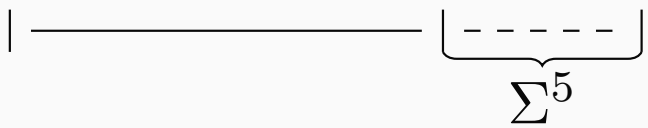
$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$

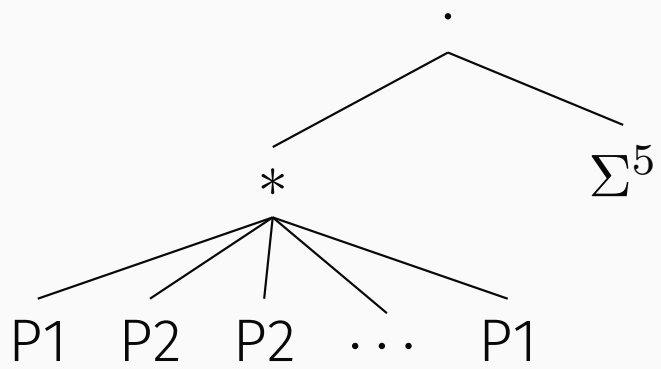
PARSE TREES AND COMPUTATION TREES

$$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$$

PARSE TREES AND COMPUTATION TREES

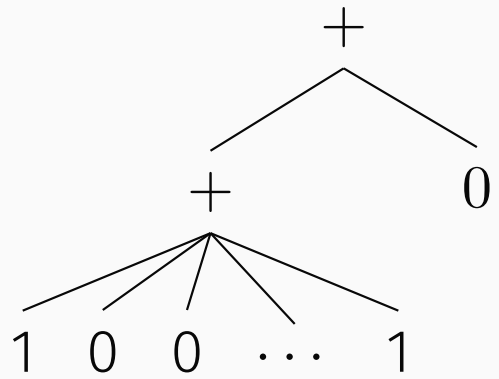
$$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$$





PARSE TREES AND COMPUTATION TREES

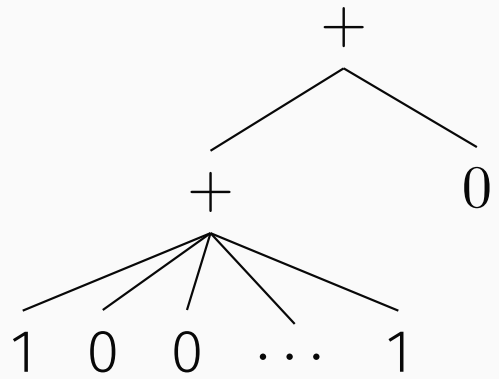
$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$



- Attempt 1 annotates the parse tree with cost operations to obtain the computation tree

PARSE TREES AND COMPUTATION TREES

$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$



- Attempt 1 annotates the parse tree with cost operations to obtain the computation tree
- What if QREs map input streams to computation trees?

Key Insight

QREs map input streams to terms over the cost domain

- Terms contain parameters
- $f(aaab) = 5 + p$, where p is a parameter

Key Insight

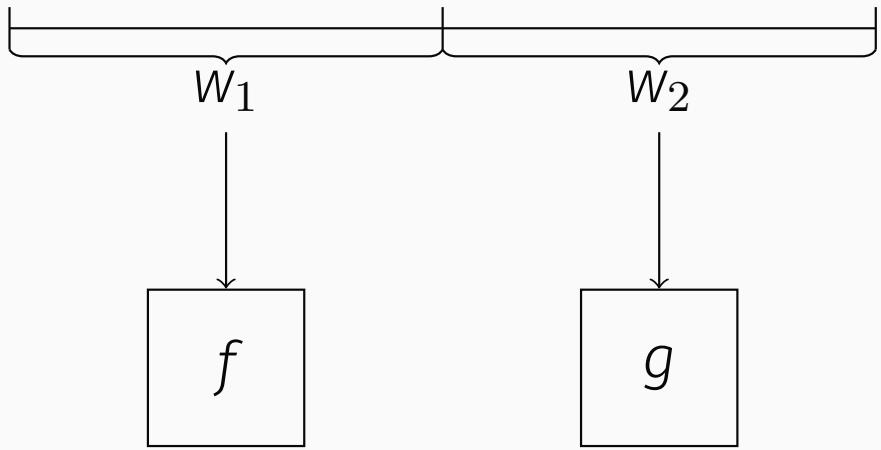
QREs map input streams to terms over the cost domain

- Terms contain parameters
- $f(aaab) = 5 + p$, where p is a parameter
- Parameter substitution is also an operation

Parameter subtitution, $f[p/g]$

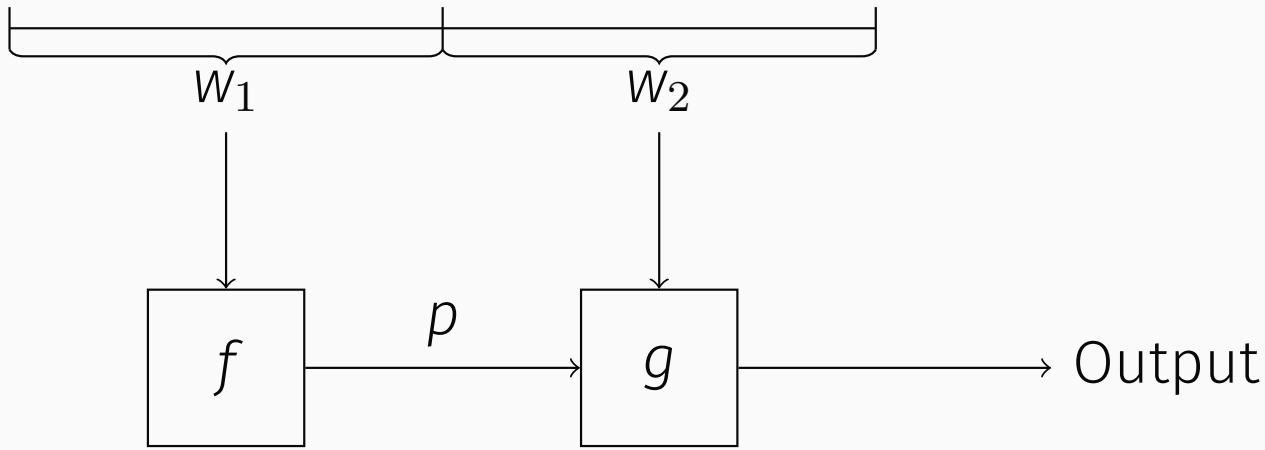
- Say $f(w) = t_f$,
- and $g(w) = t_g$
- $f[p/g](w) = t_f[p/t_g]$

$split(f \rightarrow^{p} g)$

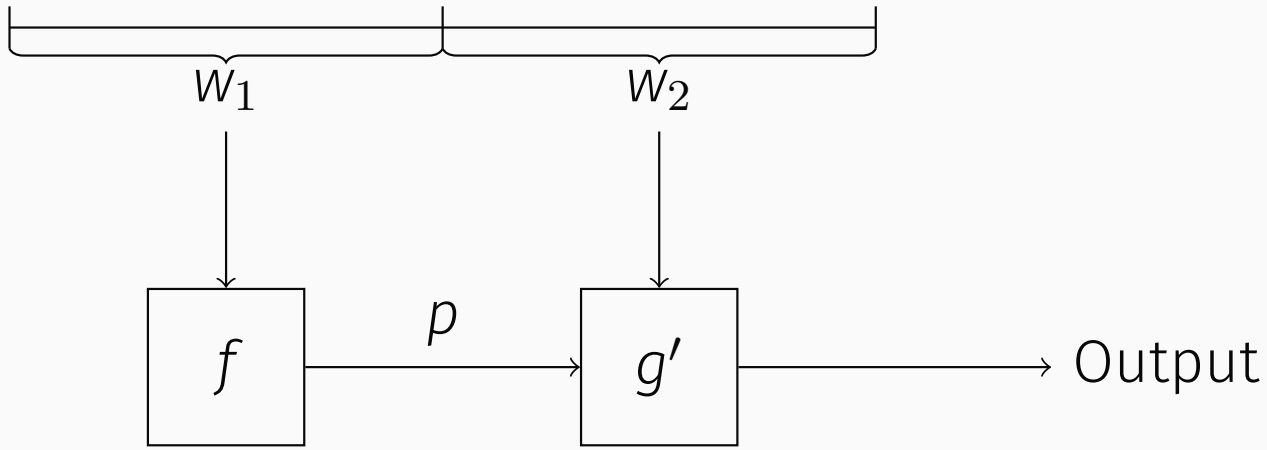


Output

$split(f \rightarrow^{p} g)$

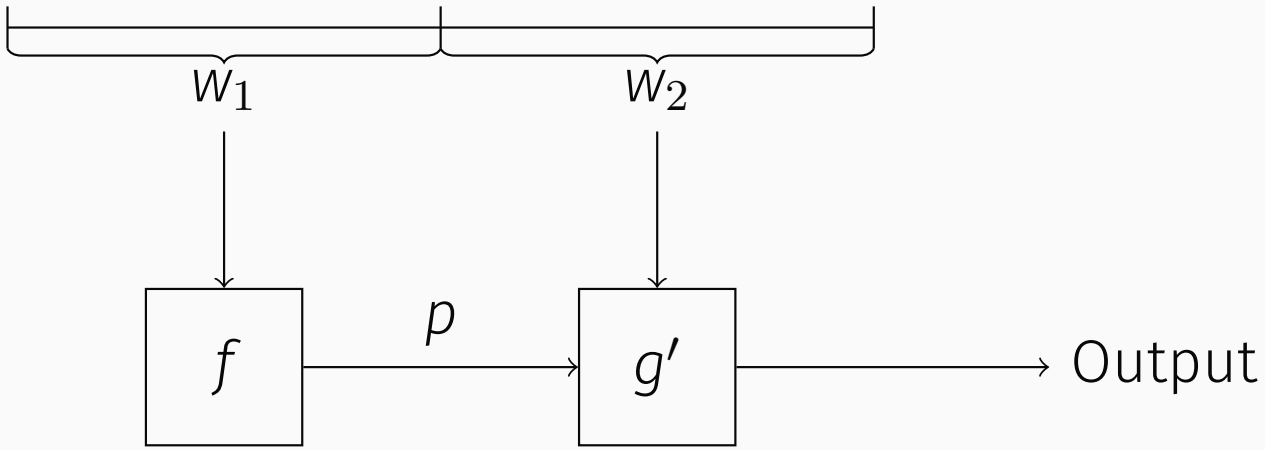


Simulating *split-plus*(f, g)

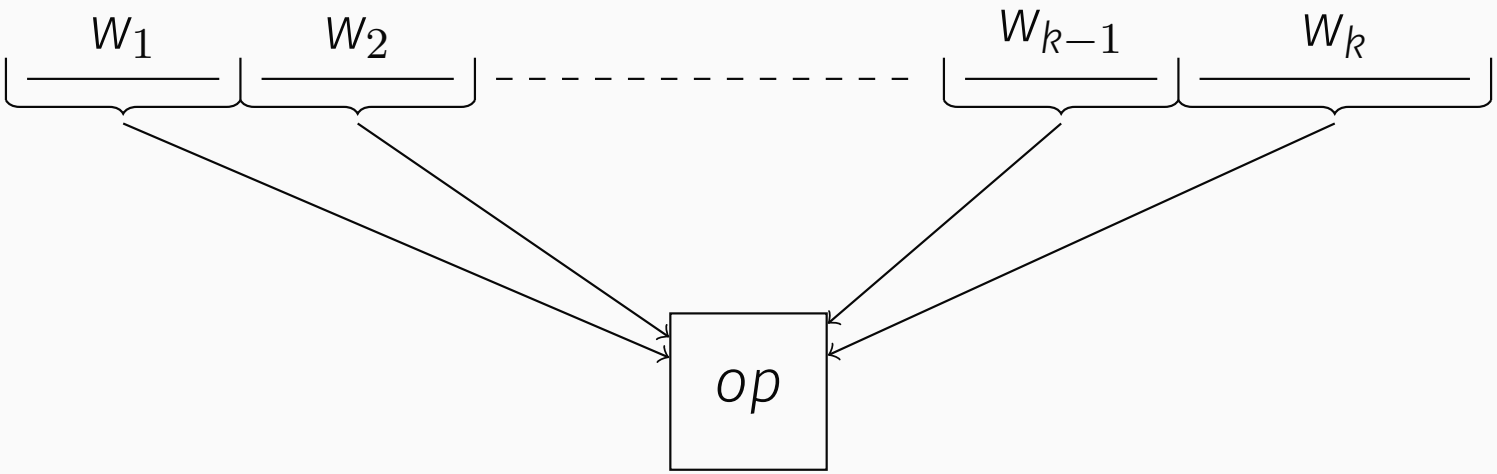


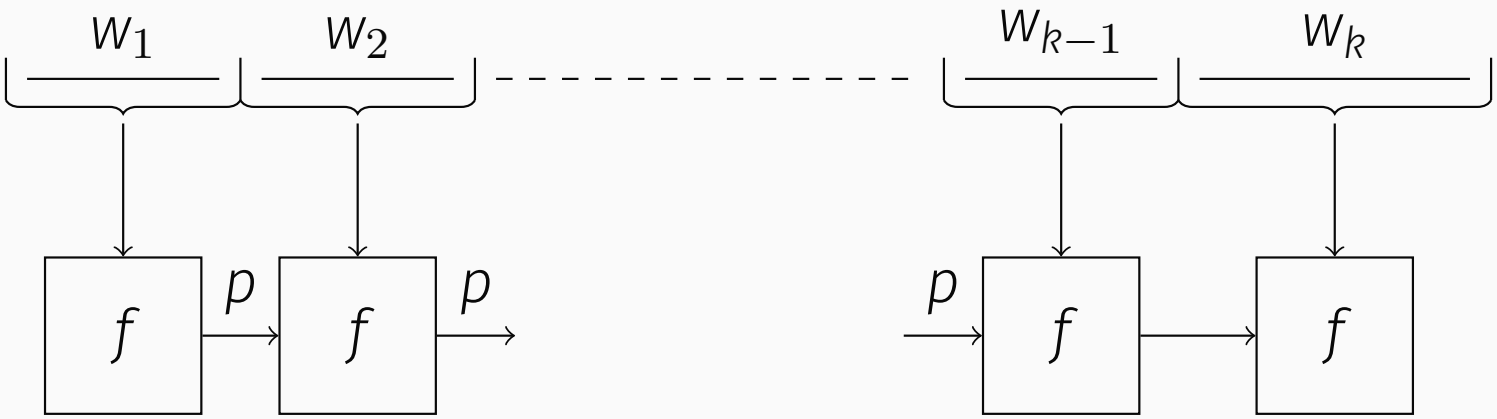
- *split-plus*$(f, g) =$ *split*$(f \to^p g')$

Simulating *split-plus*(f, g)



- *split-plus*$(f, g) =$ *split*$(f \to^p g')$
- $g'(w) = p + g(w)$


w_1
w_2
w_{k-1}
w_k
op


w_1
w_2
w_{k-1}
w_k
f
p
f
p
p
f
f

QUANTITATIVE REGULAR EXPRESSIONS

Basic functions: $a \mapsto d$, $\mathit{regex} \mapsto \mathit{term}$

Conditional choice: $f \mathit{else}\, g$

Concatenation: $\mathit{split}(f \rightarrow^p g), \mathit{split}(f \leftarrow^p g)$

Function iteration: $\mathit{iter}^{\rightarrow}(f), \mathit{iter}^{\leftarrow}(f)$

Cost operations: $op(f_1, f_2, \ldots, f_k), f[p/g]$

QUANTITATIVE REGULAR EXPRESSIONS

Basic functions: $a \mapsto d$, $\mathit{regex} \mapsto \mathit{term}$

Conditional choice: $f \; \mathit{else} \; g$

Concatenation: $\mathit{split}(f \to^{p} g), \mathit{split}(f \leftarrow^{p} g)$

Function iteration: $\mathit{iter}^{\rightarrow}(f), \mathit{iter}^{\leftarrow}(f)$

Cost operations: $op(f_1, f_2, \ldots, f_k), f[p/g]$

- QREs map string to terms
- Structural operators decoupled from cost operators

- Was our choice of combinators ad-hoc? What functions can the formalism express?
- Given f and an input stream w, can we efficiently compute $f(w)$?

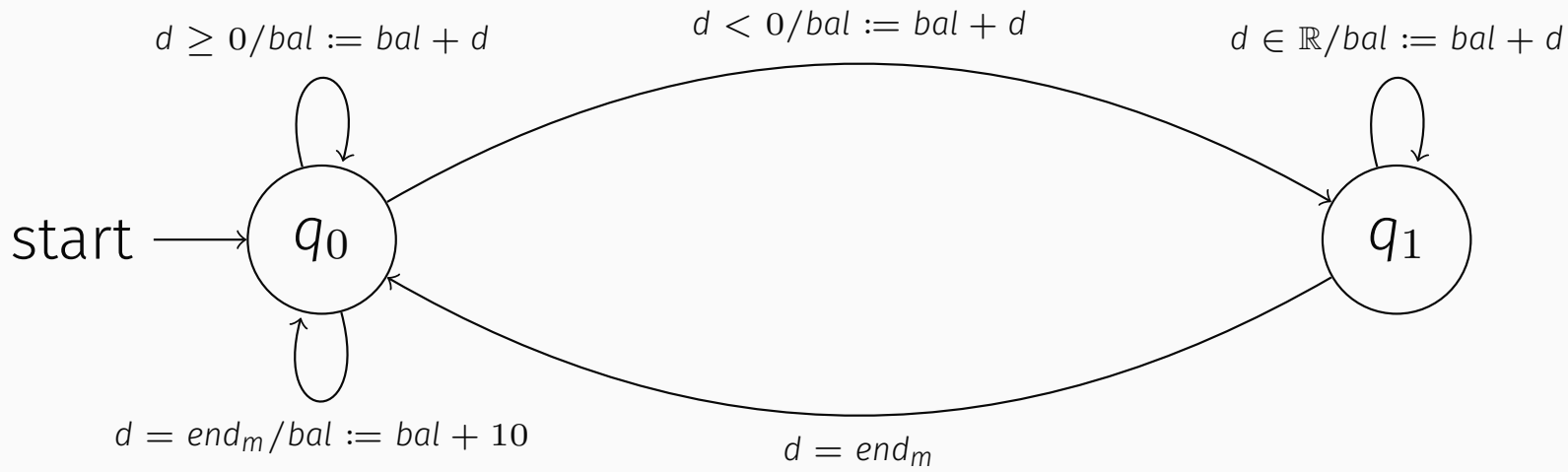
EXPRESSIVENESS

REGULAR COST FUNCTIONS [LICS 2013]

Languages	$\Sigma^* \to \texttt{bool}$	$\equiv$	Finite automata
Cost functions	$\Sigma^* \to \mathbb{R}$	$\equiv$	?

Regular languages have many appealing properties: many natural equi-expressive characterizations, robust closure properties, decidable analysis problems, practical utility

COST REGISTER AUTOMATA [LICS 2013]



- Finite state space, finitely many registers
- Registers hold terms: Parameterized by set of operations
- Equivalent to MSO definable string-to-term transducers
- Closed under regular lookahead, input reversal, etc

Theorem

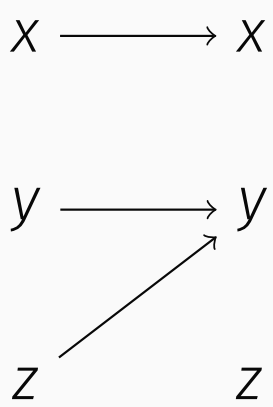
QREs can express exactly the same functions as CRAs

Taste of the completeness proof

- Piggy-back on DFA to regular expression translation
  Construct $R^i(q, q')$: all strings from q to q' while only traversing states less than q_i
- Construct $f^i(q, q', x)$: express final value of register x as a DReX expression

Taste of the completeness proof

- Capture data flows using “shapes”
- Construct a partial order over shapes, and use as basis for induction



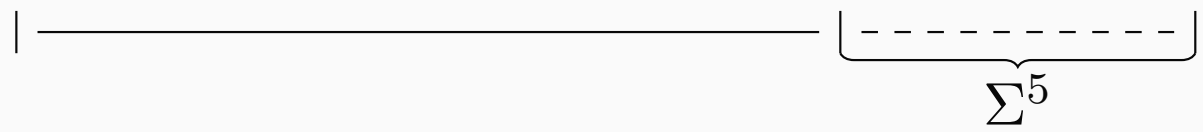
FAST EVALUATION ALGORITHMS

Given a QRE f and an input stream w, find $f(w)$

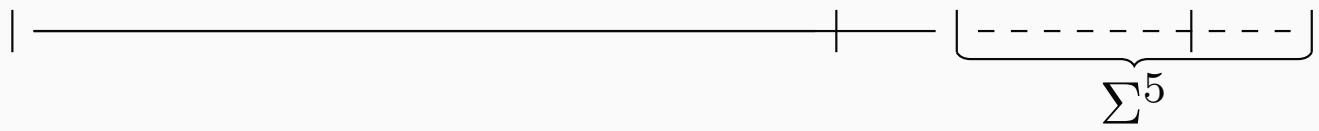
- QREs separate intent from evaluation
- If QREs are unambiguous, then $f(w)$ can be computed with a single pass over w in time $O(|w| \cdot poly(|f|))$

$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$

$$\mathit{split\text{-}plus}(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$$



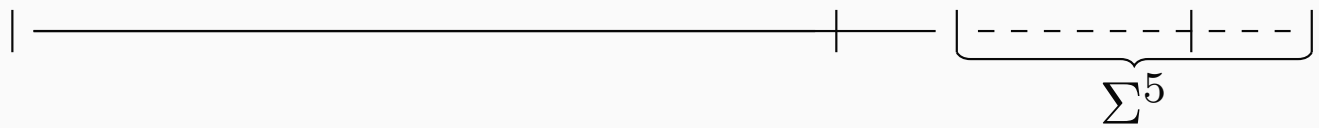
split-plus$(\mathit{count}_{\mathsf{P1}}, \Sigma^5 \mapsto 0)$



split-plus($count_{\mathsf{P1}}, \Sigma^5 \mapsto 0$)

5 potential parse trees needed at each step

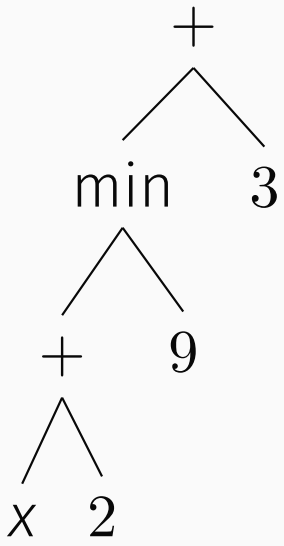
split-plus($count_{\mathsf{P1}}, \Sigma^5 \mapsto 0$)

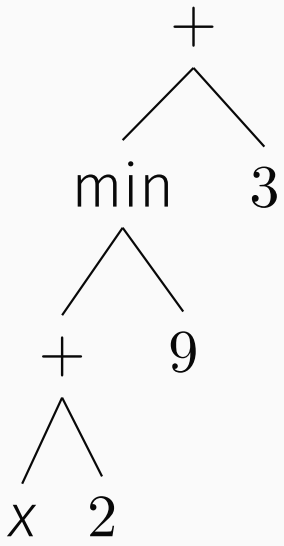


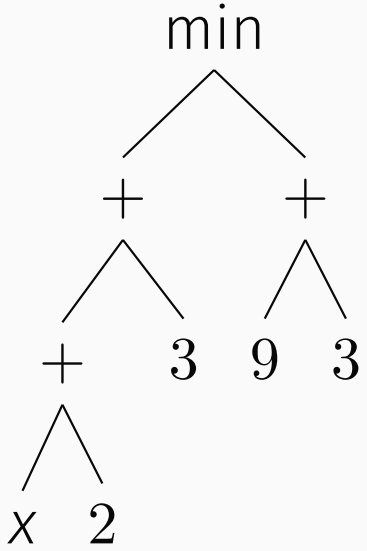
5 potential parse trees needed at each step

- Idea 1: Statically bound number of potential parse trees: $O(poly(|f|))$, irrespective of $|w|$

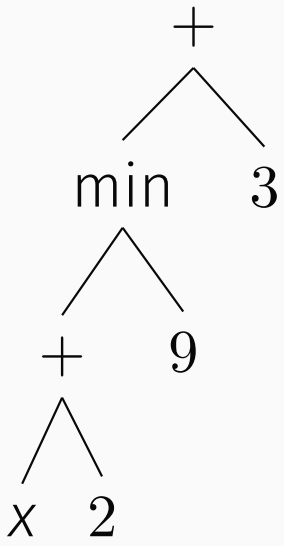
EVALUATION ALGORITHMS

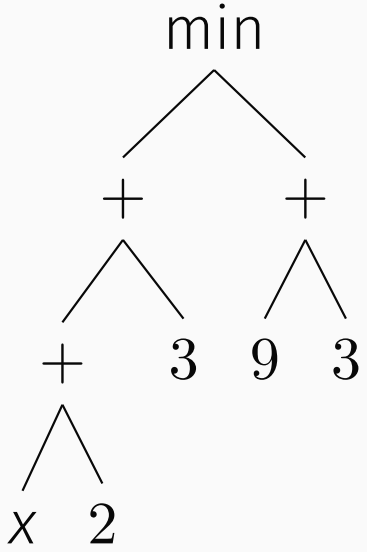


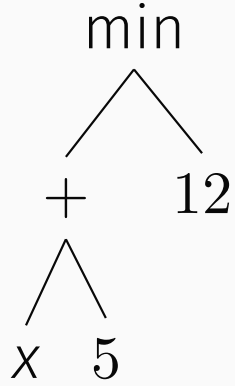

+
min
3
+
9
x
2

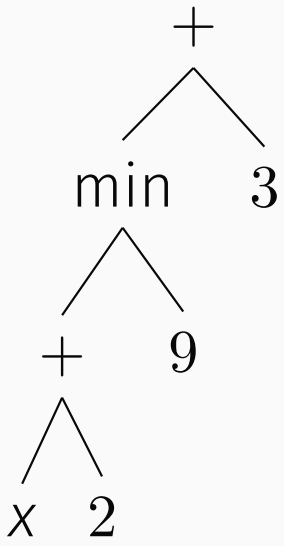

min
+
+
+
3
9
3
x
2

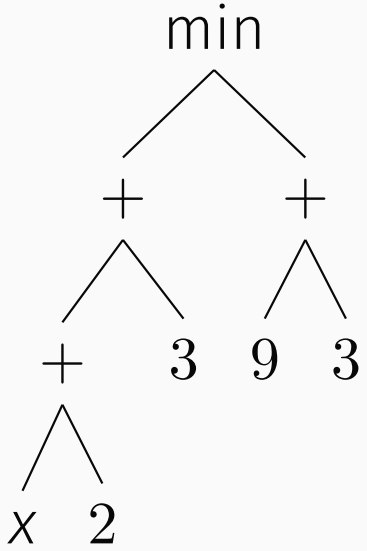
EVALUATION ALGORITHMS

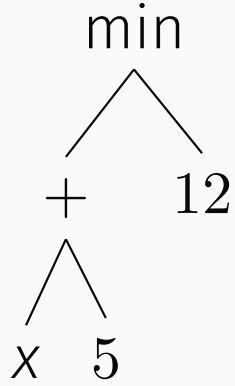












- Term compression ensures intermediate terms of bounded size
- Idea 2: If only operators are $*$, $+$, $-$, min, max, *avg*, then space usage is polynomially bounded too!

CONCLUSION

CONCLUSION

- Introduced Quantitative Regular Expressions (QRE)
- Idea of function combinators
  - Function descriptions are modular
  - Regular parsing of the input data stream
- Simple, expressive programming model for stream processing, with strong theoretical foundations and fast evaluation algorithms

QUANTITATIVE REGULAR EXPRESSIONS

Languages	$\Sigma^* \to \mathtt{bool}$	$\equiv$	Regular expressions
Cost functions	$\Sigma^* \to \mathbb{R}$	$\equiv$	QREs

- Also works for $\Sigma^* \to \mathbb{N}$, $\Sigma^* \to \mathbb{Q}$, ...
- $\Sigma^* \to \mathbb{D}$, for arbitrary cost domain $\mathbb{D}$
- Key insight: Generalizing to string-to-term transformations

CONCLUSION

- Expressively equivalent to regular cost functions / cost register automata
- Fast one-pass evaluation algorithms for unambiguous expressions
- Low space usage if only operations used are $*$, $+$, $-$, min, max, *avg*

- Approximate evaluation algorithms for certain operations such as *iter-median* (Jointly with Sanjeev Khanna and Kostas Mamouras)
- Exploring connections to streaming databases

FIN!

QUESTIONS, COMMENTS, BRICKBATS?