REGULAR PROGRAMMING OVER DATA STREAMS

Mukund Raghothaman (Joint work with Rajeev Alur, Loris D'Antoni, Dana Fisman, Adam Freilich and Arjun Radhakrishna)

ExCAPE PI Meeting, 2015

abc@gmail.com $\rightarrow$ gmail.com

- $\mathit{swallowUserName} = (\texttt{ALNUM}^*) \cdot \texttt{@}\,? \cdot \epsilon$
- $\mathit{echoDomain} =$ Identity function over strings
- $\mathit{getDomain} = \mathit{split}(\mathit{swallowUserName}, \mathit{echoDomain})$

"What was the highest winter-time temperature?"

```
...;
Temp 32°C;
AutumnEquinox;
Temp 27°C;
LeonidMeteorShowers;
Temp 26°C;
Temp 21°C;
WinterSolstice;
Temp 15°C;
CalendarYearEnd;
Temp 12°C;
SpringEquinox;
...
```

- $allWinterHi = iter\text{-}max(annualWinterHi)$
- The function *annualWinterHi* returns the highest winter-time temperature of one summmer-winter cycle

"What was the highest winter-time temperature?"

```
...;
Temp 32°C;
AutumnEquinox;
Temp 27°C;
LeonidMeteorShowers;
Temp 26°C;
Temp 21°C;
WinterSolstice;
Temp 15°C;
CalendarYearEnd;
Temp 12°C;
SpringEquinox;
...
```

- $\mathit{allWinterHi} = \mathit{iter\text{-}max}(\mathit{annualWinterHi})$
- The function *annualWinterHi* returns the highest winter-time temperature of one summmer-winter cycle
- $\mathit{annualWinterHi} = \mathit{split\text{-}plus}(\mathit{winterHi}, r_{\mathit{summer}} \,?\, 0)$

The customer and the bank

```
...; Deposit $32; Deposit $7; Deposit $9;
...; Withdraw $2; ...; EndOfMonth; ...
```

"What was the customer reward?"

$$reward = \mathit{if}\ (goodMonth)\ \mathit{then}\ (hiReward)\ \mathit{else}\ (loReward)$$

The functions *hiReward* and *loReward* compute the customer reward in qualifying and non-qualifying months respectively.

INTRODUCING DREX

DReX is a simple, expressive programming model for string transformations, with:

1. robust expressiveness,
2. fast evaluation algorithms, and
3. tools for static analysis

Small collection of core combinators

Basic functions: $data\ ?\ cost$

Conditional choice: $f\ else\ g$

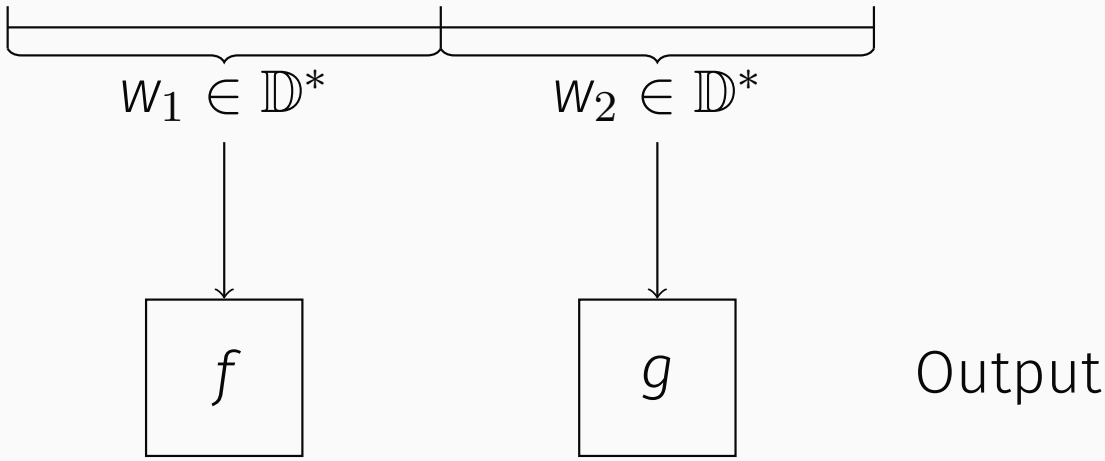
Small collection of core combinators

Basic functions: $data\ ?\ cost$

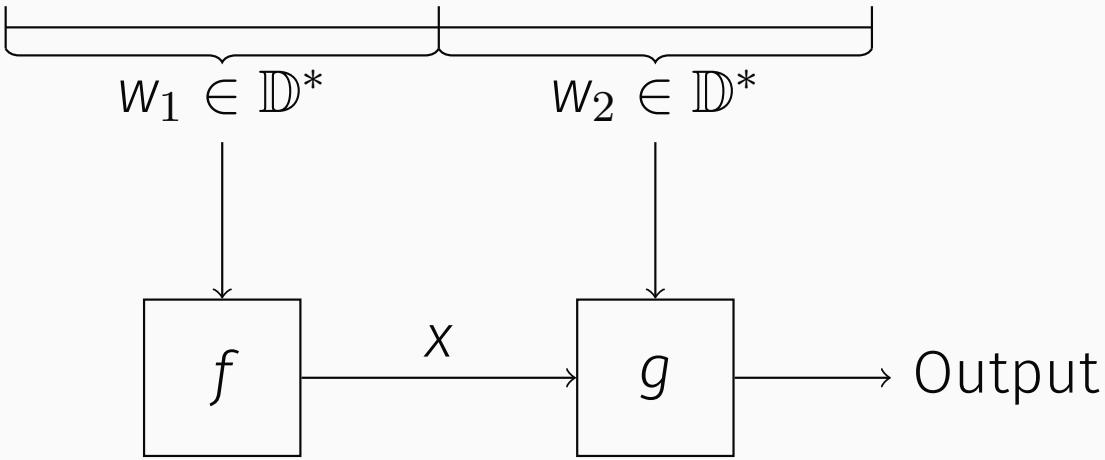
Conditional choice: $f\ else\ g$

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

What about concatenation?



What about concatenation?



Small collection of core combinators

Basic functions: $data\ ?\ cost$

Conditional choice: $f\ else\ g$

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

Small collection of core combinators

Basic functions: *data ? cost*, *regex ? term*

Conditional choice: *f else g*

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

Small collection of core combinators

Basic functions: $data\ ?\ cost$, $regex\ ?\ term$

Conditional choice: $f\ else\ g$

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

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

Small collection of core combinators

Basic functions: $data\ ?\ cost$, $regex\ ?\ term$

Conditional choice: $f\ else\ g$

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

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

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

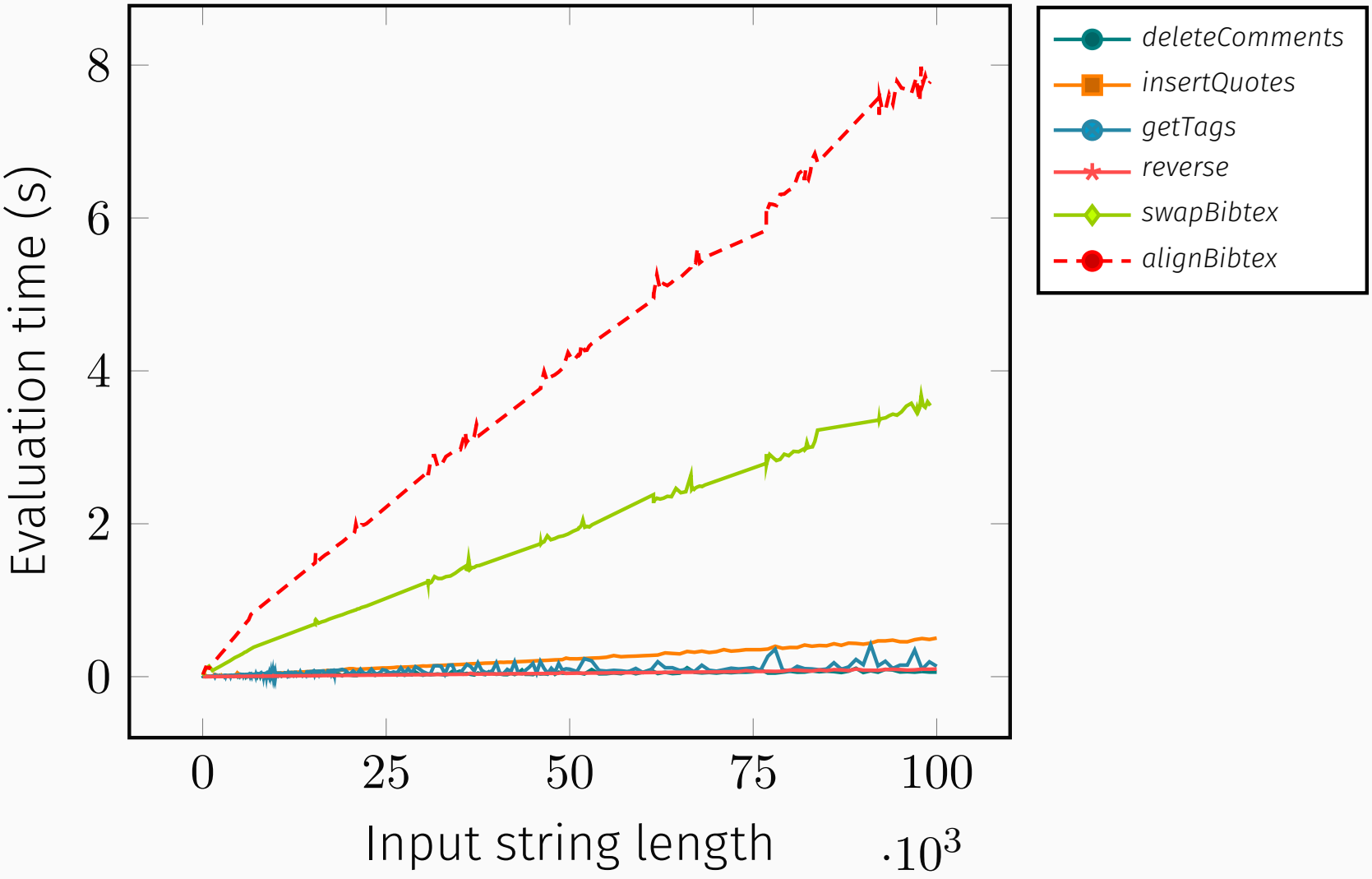
Function descriptions are modular

```
isWinter: bool, currHi: int
while exists next event e:
  if e is AutumnEquinox:
    isWinter := true
  elif e is SpringEquinox:
    isWinter := false
  elif isWinter:
    currHi := max(currHi, e.temp)
```

DReX allows regular parsing of the input data stream

- Unrestricted global choice: *f else g*
- Iteration patterns part of query, not of the data: *iter-max*(*weeklyAverageTemp*)

- Expressively equivalent to regular string-to-term transformations
- Multiple characterizations: two-way finite state transducers, MSO-definable graph transformations, streaming string transducers
- Closed under various operations: function composition, regular look-ahead, etc.


Evaluation time (s)
0
2
4
6
8
0
25
50
75
100
Input string length
$\cdot 10^3$
deleteComments
insertQuotes
getTags
reverse
swapBibtex
alignBibtex

Over arbitrary cost domains $(\mathbb{D}, G)$

- $\mathbb{D} = \mathbb{N}$ or $\mathbb{Z}$ or ...
- Cost operations, $G = \{+, \min, \max, \mathit{avg}, \mathit{median}, \ldots\}$
- DReX expression maps the input data word w to a term t
- When t admits a succinct representation, we can evaluate quickly
  - Γ^* with concatenation: ✓
  - $\mathbb{Z}$ with $\{\min, +\}$, $\min(x + a, b)$: ✓
  - $\{\mathit{avg}\}$ (set total, set size): ✓
  - ...
  - $\mathbb{Z}$ with *median*: ✗?

- Is the transformation defined for all inputs?
- In the case of string-to-string transformations:
  - Does the output always have some "nice" property?
    $\forall w$, is it the case that $f(w) \in L$?
  - Are two transformations equivalent?

Can we synthesize DReX expressions equivalent to given sed / Bash scripts and audit requirements?

```
#!/usr/bin/env bash
./run-experiments
for f in ‘ls *.tmp‘
do
  BASE=‘echo $f | sed s/\.[^\.]*$//‘
  ./process-log ”$BASE.log” >> outfile
  rm ”$BASE”*
done
```

STATUS

Done: $\Sigma^* \to \Gamma^*$ (LICS 2014, POPL 2015)

Ongoing: $\Sigma^* \to \mathbb{Z}$ (and $\mathbb{N}$ and $\mathbb{R}$ and ...)

THANK YOU!

TRY IT OUT AT HTTP://DREXONLINE.COM