Domain Specific Induction for

Data Wrangling Automation

Lidia Contreras-Ochando
liconoc@upv.es
@liconoc

Joint work with: **Cèsar Ferri, José Hernández-Orallo, Susumu Katayama, Fernando Martínez-Plumed and María José Ramírez-Quintana**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

宮崎大学 University of Miyazaki

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

ELP GROUP

dmip

Contents

- Motivation
- Data wrangling
- Approach
- Experiments
- Results
- Open research questions

Motivation

Real example of a dataset from a bike sharing system

Row No.	Station	Date
1	001	03/10/2016 00:18:36
2	001	03/10/2016 00:25:45
…	…	…
69852	001	6-10-16 20:35

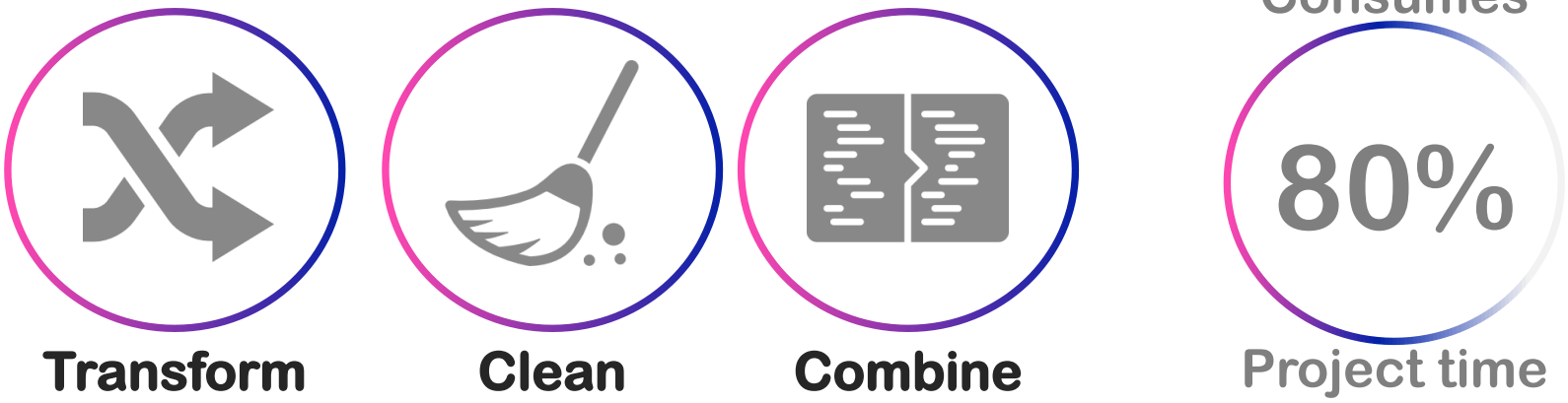
What if we want to use…

- Only dates, not hours
- Only the day
- Only the month
- Only the year

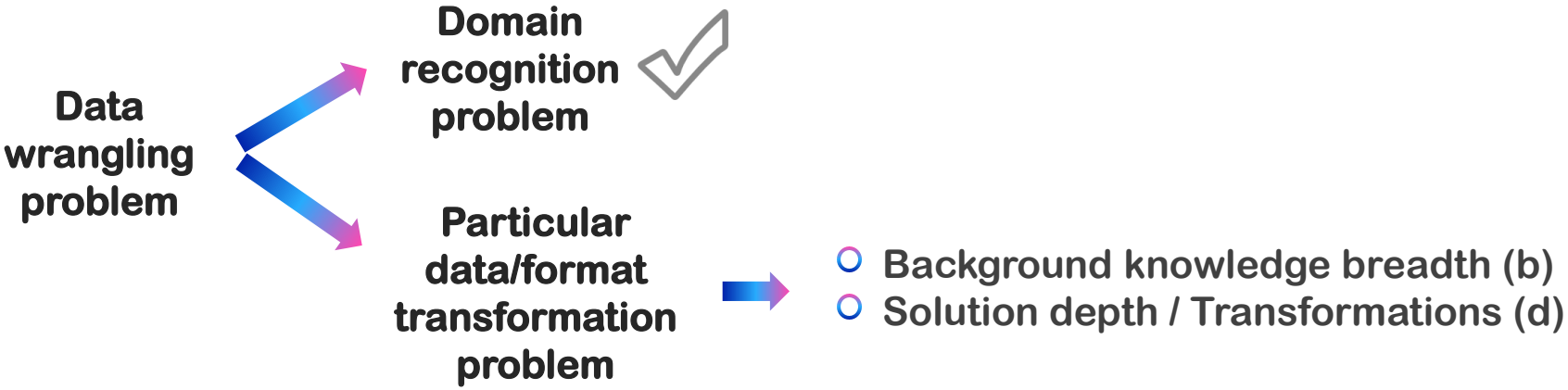
Problem? → Dates are in **different formats**

Solution? → Spending time in finding the **different formats** and **transforming** each of them into a proper and unique format

Data wrangling



Automating data wrangling process is essential to reduce time and cost



Systems for Data Wrangling

State-of-the-art systems for data wrangling are usually based on **Domain Specific Languages (DSL)**

- IP Systems
  - FlashFill
  - FlashExtract
  - FlashRelate
  - …
- Other type of systems
  - Trifacta Wrangler
  - …

Some difficulties…

- New domain **transformations**
- New **domains**

It requires redesigning the system?

Approach: Domain-Specific Induction (DSI)

Our approach

General-purpose Inductive Programming Learning Systems + **General-purpose Declarative Languages (GPDL)** + **Domain Specific Background Knowledge (DSBK)**

- Languages such as: Prolog, Haskell, Mod, Erlang…
- Systems such as: Progol, IGOR2, MagicHaskeller, Metagol, gErl, FLIP…

Advantages

- Specialization at the **DSBK**
- One tool, **many domains**
- General purpose and **well known** declarative language
- **Easier** to add transformations in the **different domains**
- **Increase** the range of applications

Can they be as powerful as DSL systems?

Approach: MagicHaskeller

MagicHaskeller

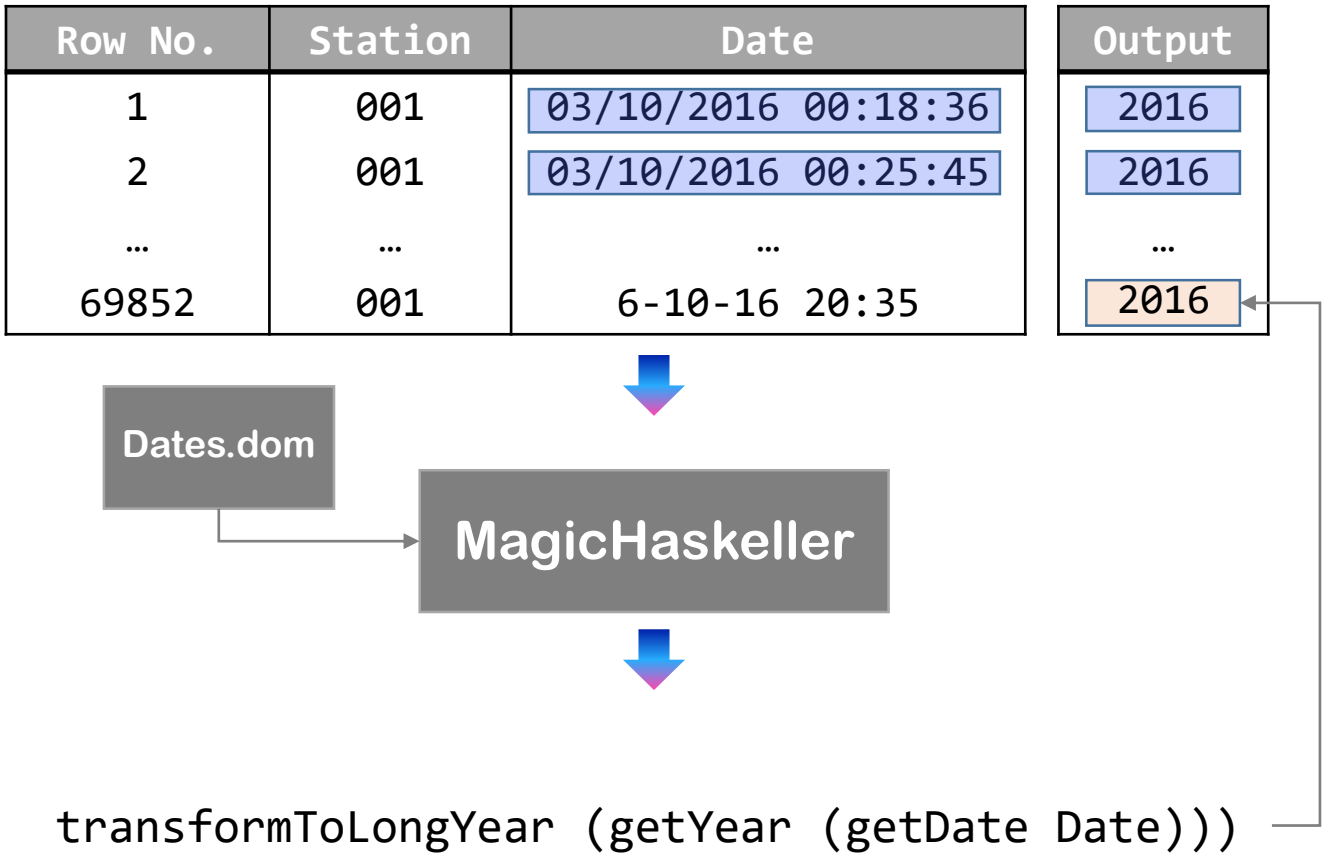
- **Inductive functional programing system**
- It **learns Haskell programs** from pairs of **input-output examples**
- How it works?
  1. It **receives an input** example (**x**) and **the expected result** (**y**)
  2. Making combinations with all the functions at the **BK** by brute force, it **finds different possible solutions** that makes the values of the expressions fx and y be equal (**f x == y**)
  3. Finally, it **returns a list of functions** (**f**)

Advantages

- **General-purpose** learning system
- **Haskell** as pure functional programming language
- **Hypothesis-oriented**, not data-oriented
- It learns from **one** or few input-output examples
- We can use **different BK**, specialising them for each domain

Approach: Overall idea

Automate the transforming process, depending on the domain and using MagicHaskeller with few examples



Approach: Experiments

Domains

Dates

Emails

Names

Words

Transformations

Dates	Emails	Names	Words
117 primitives: • Get the day in ordinal format • Month to numeric format • Two-digit year to four-digit year • etc.	58 primitives: • Get words before "@" • Append "@" • Join two strings with "@" • etc.	72 primitives: • Determine if a string is a courtesy title • Reduce surname • Get the initials • etc.	102 primitives: • Change punctuation marks • Split a string • Concatenate two strings • etc.

Data wrangling dataset repository

- **16 datasets** covering the different domains and data wrangling problems
- Datasets **collected** from other tools and literature & **new generated** datasets
- Now **published, open and available** at: http://users.dsic.upv.es/~flip/datawrangling/

Experiments: Preliminary Results (I)

Real domain	Dataset id	Background knowledge (domain) used					
		default	dates	emails	names	words	all
Dates	addPunctuation	0.00	**1.00**	0.00	0.00	0.00	0.00
	changePunctuation	**1.00**	**1.00**	**1.00**	0.00	**1.00**	0.00
	getDay	0.00	**1.00**	0.00	0.17	0.00	0.00
	getWeekDay	0.00	**1.00**	0.00	0.00	0.00	0.00
Emails	addAt	0.00	0.00	**1.00**	0.00	0.00	0.00
	getBetweenAtAndDot	0.00	0.00	**1.00**	0.00	0.00	0.00
	getAfterAt	0.00	0.13	**1.00**	0.00	0.00	0.00
	joinWithAt	0.00	0.00	**1.00**	0.00	0.00	0.00
Names	getGender	0.00	0.00	0.00	**1.00**	0.00	0.00
	getTitle	0.00	0.00	0.33	**1.00**	0.33	0.00
	reduceName1	0.00	0.00	0.00	**0.50**	0.00	0.00
	reduceName2	0.00	0.00	0.00	**1.00**	0.00	0.00
Words	deletePunctuation	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.00
	getInitials	0.00	0.00	0.00	0.50	**1.00**	0.00
	getSubset1	**1.00**	**1.00**	0.00	**1.00**	**1.00**	0.00
	getSubset2	0.00	0.00	0.00	0.00	**1.00**	0.00

Accuracy obtained by the DSI approach using one example in each dataset, depending on the set of primitives (DSBK) used.

Experiments: Preliminary Results (II)

input	Expected output	FlashFill	Trifacta Wrangler	Our Approach (DSI)
03/29/86 74-03-31 05 30 85	**29** 31 30	03 30	03 30	31 30
Sunday, 9 November 2014 2 July 2010, Monday 2003-Nov-9, Sunday	**Sunday** Monday Sunday	2 July 2010 2003-Nov-9	2 July 2010 2003-Nov-9	Monday Sunday
Nancy@coffee.com Andrew@traders.com Laura@add-works.com	**coffee.com** traders.com add-works.com	traders.com add-works.com	traders.com works.com	traders.com add-works.com
Dr. Mark Sipser Louis Jonhson, PhD Prof. Edward David	**Dr.** PhD Prof.	Lou Prof.	Louis Prof.	PhD Prof.

Example of the results of our approach compared with **FlashFill** and **Trifacta Wrangler**.
The **first row** of each dataset **is the example** given to the system to learn.
Red color means **incorrect** result. Green color means **correct** result.

Open Research Questions

- How can we adapt general IP learning systems for data wrangling?
- Should we start from scratch?
- Incremental Knowledge adquisition?
- More and different domains? Which ones?
- Automate the detection of the domain?
- Create new benchmarks/datasets?