Tabular Constraint Learning

Name1 Surname1 and **Name2 Surname2** and **Name3 Surname3** [1]

Abstract. abstract

1 Introduction

SERGEY: bullet points for luc to start introduction

Key question:
Can we discover or reconstruct structural relations in flat tabular spreadsheet data? [in a general way that allows declarative specification of constraints to discover]

Motivation:

- File generated from model, model got lost, need to reconstruct
- Constraint programming is hard - is Excel hard?
- Avoid manual analysis, provide selection of constraints
- Error checking
- Completion, gain speed and insights (Complicated constraints, also complicated to verify, too much output)

Novelty:

- Unsupervised setting (contrary to flashfill, etc)
- Numeric, different constraints (contrary to single textual function solution in flashfill, etc)
- Data format (2D) – data is no longer in rows like a classic ML or DM settings
- Declarative, general / modular, stacking of constraint problems

SERGEY: to himself we need structure here

2 Formalization

Algorithm 1 Tabular constraint learning

Input: G – set of groups
Output: S – learned constraints with their satisfaction assignment
if G is not defined **then**
  $G \leftarrow \mathit{extractGroups}(D)$
$S \leftarrow \emptyset$
for $c \in \mathcal{C}$ **do** ▷ $\mathcal{C}$– the set of predefined Excel constraints
  $v_1, ..., v_n$ = variables of c
  for $v_1\colon G_1, \ldots, v_n\colon G_n \in \mathit{generateAssignments}(c, G, S)$ **do**
    $S \leftarrow S \cup \mathit{findSolutions}(c, v_1\colon G_1, \ldots, v_n\colon G_n, S)$
return S

SAMUEL: Provide information on how functions work

3 Case Study aka Experiments

Approach

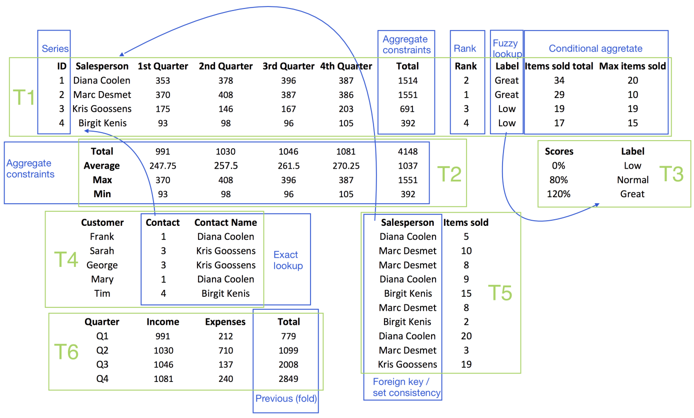

Figure 1. An example of constraint reconstruction (in blue) with indicated groups (in green)

Algorithm 2 Workflow

Input: D – dataset, (optional: tables T, groups G)
Output: S – learned constraints with their satisfaction assignment
if T is **not** provided **then**
  $T \leftarrow \mathit{extractTables}(D)$
if G is **not** provided **then**
  $G \leftarrow \mathit{extractGroups}(D, T)$
$S \leftarrow \mathit{learnConstraints}(G)$
$S \leftarrow \mathit{postProcess}(S)$
return S

- Notation
- Algorithm (select constraints, find assignments, find solutions)

Experimental questions

- How accurate are we? (Accuracy / recall)
- How fast are we and which factors affect the runtime (how)?
- How general is our approach, what limitations are there?

4 Related Work

SERGEY: key bullet points for Luc and possibly Samuel and me to make related work section

SERGEY: ECAI reference style file ignores their guideline and their guideline ignores what is written in the guidelines! flashfill, flashextract, flashmeta [3, 4, 5]

- their supervised vs our unsupervised approach

[1] KU Leuven, Belgium, email: firstname.lastname@kuleuven

- they look for a single “smallest” solution, we enumerate them all
- they are looking for a function, we solve constraint satisfaction problems
- we do not assume classic row based data layout, we work in the tabular setting

sketch [7]

- look for a constant that would fill in the gap in a program
- tailored for programming languages
- similar to model checking
- looks for a single solution
- similar to constraint satisfaction and sat, where one is interested in a single assignment that works for any potential input

tabular [2]

- language based on the excel tables that specify probabilistic models
- a system for probabilistic inference and similarity mostly in the usage of excel
- probabilistic constraint satisfaction (?) and graphical models
- single solution again

modelseeker [1] SERGEY: Samuel, Luc, probably you would need elaborate here more in details

- not designed for excel-like data representation (type consistency, groups, etc)
- not designed for excel-like constraints (lookups, conditional ifs, etc)
- does not support user extensions (?)

claudien [6] SERGEY: Samuel, Luc, you would need to help with this one

REFERENCES

[1] Nicolas Beldiceanu and Helmut Simonis, *Principles and Practice of Constraint Programming: 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, chapter A Model Seeker: Extracting Global Constraint Models from Positive Examples, 141–157, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[2] Andrew D. Gordon, Thore Graepel, Nicolas Rolland, Claudio Russo, Johannes Borgstrom, and John Guiver, ‘Tabular: A schema-driven probabilistic programming language’, Technical Report MSR-TR-2013-118, (December 2013).

[3] Sumit Gulwani, ‘Automating string processing in spreadsheets using input-output examples’, in *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL ’11, pp. 317–330, New York, NY, USA, (2011). ACM.

[4] Vu Le and Sumit Gulwani, ‘Flashextract: a framework for data extraction by examples’, in *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI ’14, Edinburgh, United Kingdom - June 09 - 11, 2014*, p. 55, (2014).

[5] Oleksandr Polozov and Sumit Gulwani, ‘Flashmeta: a framework for inductive program synthesis’, in *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2015, part of SLASH 2015, Pittsburgh, PA, USA, October 25-30, 2015*, pp. 107–126, (2015).

[6] Luc De Raedt and Luc Dehaspe, ‘Clausal discovery’, *Machine Learning*, **26**(2-3), 99–146, (1997).

[7] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat, ‘Combinatorial sketching for finite programs’, *SIGOPS Oper. Syst. Rev.*, **40**(5), 404–415, (October 2006).