Can Planning meet Data Cleansing?

Roberto Boselli and **Mirko Cesarini** and **Fabio Mercorio** and **Mario Mezzanzanica**
Dept. of Statistics and Quantitative Methods - CRISP Research Centre, University of Milano-Bicocca, Milan, Italy

Abstract

One of the motivations for research in data quality is to automatically identify cleansing activities, namely a sequence of actions able to cleanse a dirty dataset, which today are often developed manually by domain-experts. Here we explore the idea that AI Planning can contribute to identify data inconsistencies and automatically fix them. To this end, we formalise the concept of cost-optimal Universal Cleanser - an object summarising the best cleansing actions for each feasible data inconsistency - as a planning problem, then we present a motivating government application in which it has be used.

Keywords: Data Quality, Data Cleansing, Government Application

1 Introduction and Related Work

Today, most of the personal, business, and administrative data are collected, managed and distributed electronically, thanks to the wide diffusion of the Information Systems (IS). Today, most researchers agree that the quality of data is frequently poor (Fisher et al. 2012) and, according to the "garbage in, garbage out" principle, dirty data may affect the effectiveness of decision making processes. In such a scenario, the *data cleansing* (or cleaning) research area focuses on the identification of a set of domain-dependent activities able to cleanse a dirty database (wrt quality requirements), which usually have been realised in the industry by means of *business rules*. However, the design of such business rules relies on the experience of domain-experts, thus, there is no guarantee that the cleansing activities identified are suitable for the analysis purposes. Furthermore, exploring cleansing alternatives is a very time-consuming task: each business rule has to be developed and analysed separately, and resulting solutions need to be compared and evaluated manually. In this regard, expressing data cleansing problems via planning may contribute in addressing the following open issues.

(i) Modelling the behaviour of longitudinal data. Usually *longitudinal data* (aka *panel* or *historical data*) extracted by ISs provide knowledge about a given subject, object or phenomena observed at multiple sampled time points. In this regard, planning languages like PDDL can help domain experts formalising how data should evolve according to an expected behaviour. Indeed, as argued by McDermott et al. (1998) "the PDDL language is intended to express the physics of a domain, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are". In our context, a planning domain describes how data arriving from the external world - the database - may change the subject status[1], while a planning instance is initiated with subject's data. The goal of such a planning problem is to evaluate if data evolve according to the domain model.

(ii) Expressing data quality requirements. Data quality is a domain-dependent concept, usually defined as "fitness for use", thus quality considered appropriate for one use may not be sufficient for another use. Here we shall focus on *consistency*, which refers to "the violation of semantic rules defined over (a set of) data items, where items can be tuples of relational tables or records in a file" (Batini and Scannapieco 2006). In reference to relational models, such "semantic rules" have usually been expressed through functional dependencies (FDs), conditional dependencies, join dependencies, and inclusion dependencies, useful for specifying integrity constraints. Indeed, as argued by Chomicki (1995), FDs are expressive enough to model static constraints, which evaluate the current state of the database, but they do not take into account the past, that is how the current state of the database has evolved over time. Furthermore, even though FDs enable the detection of errors, they have limited usefulness since they fall short of acting as a guide in correcting them (Fan et al. 2010). Finally, FDs are only a fragment of first-order logic and this motivates the usefulness of formal systems in databases, as studied by Vardi (1987). In this regard, planning formalisms are expressive enough to model complex temporal constraints, e.g. by using temporal logics in expressing goal conditions. A cleansing approach based on AI planning might allow domain ex-

[1] "Status" here is considered in terms of a value assignment to a set of finite-domain state variables.

perts to concentrate on *what* quality constraints need to be modelled rather than on *how* to verify them.

(iii) Automatic *identification of cleaning activities.* A gap between practice-oriented approaches and academic research contributions still exists in the data quality field. From an academic point of view, two very effective approaches based on FDs are *database repair* and *consistent query answering* (Chomicki and Marcinkowski 2005). The former aims at finding a repair, i.e. a database instance that satisfies integrity constraints and minimally differs from the original one while the latter tries to compute consistent query answers in response to a query, namely answers that are true in every repair of the given database, but without fixing the source data. The main drawback of these approaches is that finding consistent answers to aggregate queries becomes NP-complete already using two (or more) FDs, as observed by Bertossi(2006). To mitigate this problem, a number of works have recently exploited heuristics to find a database repair (Yakout, Berti-Équille, and Elmagarmid 2013) (Kolahi and Lakshmanan 2009). These approaches seem to be very promising, even though their effectiveness has not been evaluated on real-life domains.

From an industry perspective, a lot of off-the-shelf tools are available and well-supported, but they often lack of formality in addressing domain independent problems, as the case of several ETL tools[2]. In such tools a quite relevant amount of the data analysis and cleaning work has still to be done manually or by ad-hoc routines, that may be difficult to write and maintain (Rahm and Do 2000).

In this regard, planning can contribute to the synthesis of optimal cleansing activities (wrt a domain-dependent objective function) by enabling domain experts to express complex quality requirements and effortlessly identify the best suited cleansing actions for a particular data quality context.

2 The Labour Market Dataset

The scenario we are presenting focuses on the Italian labour market domain studied by statisticians, economists and computer scientists at the CRISP Research Centre[3]. According to the Italian law, every time an employer hires or dismisses an employee, or an employment contract is modified (e.g. from part-time to full-time, or from fixed-term to unlimited-term), a *Compulsory Communication* - an event - is sent to a job registry. The public administration has developed an ICT infrastructure (The Italian Ministry of Labour and Welfare 2012) generating an administrative archive useful for studying the labour market dynamics (see, e.g.(Lovaglio and Mezzanzanica 2013)). Each mandatory communication is stored into a record which presents several relevant attributes. **e_id** and **w_id** are ids identifying the communication and the person involved. **e_date** is the event occurrence date whilst **e_type** describes the event type occurring to the worker's career. Events types are the *start*, *cessation* and *extension* of a working contract, and the *conversion* from a contract type to a different one. **c_flag** states whether the event is related to a full-time or a part-time contract while **c_type** describes the contract type with respect to the Italian law. Here we consider the *Limited* (fixed-term) and *unlimited* (unlimited-term) contracts. Finally, **empr_id** uniquely identifies the employer involved. A *communication* represents an event arriving from the external world (ordered with respect to *e_date* and grouped by *w_id*), whilst a career is a longitudinal data sequence whose consistency have to be evaluated. To this end, the *consistency* semantics has been derived from the Italian labour law and from the domain knowledge as follows.

c1: an employee cannot have further contracts if a full-time is active;
c2: an employee cannot have more than K part-time contracts (signed by different employers), in our context we shall assume $K = 2$;
c3: an *unlimited term* contract cannot be extended;
c4: a contract extension can change neither the contract type (*c_type*) nor the modality (*c_flag*), for instance a part-time and fixed-term contract cannot be turned into a full-time contract by an extension;
c5: a conversion requires either the *c_type* or the *c_flag* to change (or both).

Table 1: Example of a worker career

e_date	e_type	c_flag	c_type	empr_id
1^{st} May 2010	start	PT	limited	Company_X
1^{st} Nov 2010	convert	PT	unlimited	Company_X
12^{th} Jan 2012	convert	FT	unlimited	Company_X
28^{th} July 2013	start	PT	limited	Company_Y

For simplicity, we omit to describe some trivial constraints e.g., an employee cannot have a *cessation* event for a company for which she/he does not work, an event cannot be recorded twice, etc.

To clarify the matter, let us consider a worker's career as in Tab. 1. A worker started a limited-term part-time contract with Company_X in May 2010. After six month the contract was converted to unlimited-term. Then, in January 2012 the worker converted its contract from part-time to full-time. Finally, in July 2013 a communication arrived from Company_Y reporting that the worker had started a new part-time contract, but no communication concerning the cessation of the previous active contract had ever been notified. The last communication makes the career inconsistent with

[2] In the ETL approach (Extract, Transform and Load) data extracted from a source system pass through a sequence of transformations, that analyse, manipulate and then cleanse the data before loading them into a Datawarehouse

[3] The Interuniversity Research Centre on Public Services. `http://www.crisp-org.it/english`

respect to the labour law as it violates the constraint c1. Being able to catch and fix such an inconsistency is quite important as it may strongly affect a statistical indicator based on "working days", and this makes the data cleansing process necessary to guarantee the believability of the overall decision making process. Clearly, there are several alternatives that domain experts may define to fix an inconsistency (as shown in Tab. 2) many of which are often inspired by common practice. Probably, in the example above the communication was lost, thus it is reasonable to assume that the full-time contract has been closed in a period between 1st January 2012 and 28th July 2013. However, one might argue that the communication might have been a *conversion* to part-time rather than a cessation of the previous contract and, in such a case, the career does not violate any constraints as two part-time contracts are allowed. Although this last scenario seems unusual, a domain expert should consider such hypothesis.

3 Data Cleansing as Planning

Modelling data cleansing as a planning problem can be used to (i) confirm if data evolution follows or not an expected behaviour (wrt quality requirements) and (ii) to support domain experts in the identification of *all* cleansing alternatives, summarising those that minimize/maximize an indicator. To this aim, a planner is well-suited to *explore* all the feasible actions able to cleanse the dataset and to *select* the best ones with respect to given criteria, which are domain-dependent. Notice that an IS recording longitudinal data can be seen as an event-driven system where a database record is an *event* modifying the system state whereas an ordered set of records forms an *event sequence*. We can formalise this concept as follows.

Definition 1. *Let $\mathcal{R} = (R_1,\ldots,R_n)$ be a schema relation of a database. Then,*
(i) An event $e = (r_1,\ldots,r_m)$ *is a* record *of the projection $(R_1,\ldots,R_m)$ over $\mathcal{R}$ with $m \leq n$, s.t. $r_1 \in R_1,\ldots,r_m \in R_m$;*
(ii) Let $\sim$ be a total order *relation over events, an* event sequence *is a $\sim$-ordered sequence of events $\epsilon = e_1,\ldots,e_n$ concerning the same object or subject.*

In classical planning, a model describes how the system evolves in reaction to input actions. A planner fires actions to explore the domain dynamics, in search of a (optimal) path to the goal. Similarly, when dealing with longitudinal data the system is represented by the object or subject we are observing, while an event represents an action able to modify the state of the system. Once a model describing the evolution of an event sequence has been defined, a planner works in two distinct and separated steps.

First, it simulates the execution of *all* the feasible (bounded) event sequences, summarising all the inconsistencies into an object, the so-called *Universal Checker* (UCK). This represents a topology of all the feasible inconsistencies that may affect a data source. Second, for each inconsistency classified into the UCK with a unique identifier - the *error-code* - the planner looks for the best correction by exploring all the cleansing alternatives. The result is a *Universal Cleanser* (UC) that enhances the UCK with a sequence of actions able to fix the inconsistency.

The first step can be easily accomplished by enabling a planner to continue the search when a goal (an inconsistency) has been found. Indeed, in this first phase, the goal of the planning problem is to identify an *inconsistency*, that is a violation of one or more semantic rules. We formalise our planning problem on FSSs.

Definition 2. *A* Finite State System *(*FSS*) $\mathcal{S}$ is a 4-tuple (S,I,A,F), where: S is a finite set of* states, *$I \subseteq S$ is a finite set of* initial states, *A is a finite set of* actions *and $F : S \times A \rightarrow S$ is the* transition function, *i.e. $F(s,a) = s'$ iff the system from state s can reach state s' via action a.*

Then, a planning problem *on FSS is a triple $PP = (\mathcal{S},G,T)$ where $s_0 \in S$, $G \subseteq S$ is the set of the goal states, and T is the finite temporal horizon. A solution for PP is a* path on the FSS *(*plan*) $\pi = s_0a_0s_1a_1\ldots s_{n-1}a_{n-1}s_n$ where, $\forall i = 0,\ldots,n-1$, $s_i \in Reach(S)$ is a state reachable from the initial ones, $a_i \in A$ is an action, $F(s_i,a_i)$ is defined, $|\pi| \leq T$, and $s_n \in G \subseteq Reach(S)$.*

Finally, let Π be the set of all plans, a Universal Checker *C is a set of state-action pairs that for each $\pi \in \Pi$ summarises all the pairs (s_{n-1},a_{n-1}) s.t. $F(s_{n-1},a_{n-1}) \in G$.*

Second, for each pair $(s_i,a_i) \in C$ denoting an inconsistency, we shall construct a new planning problem which differs from the previous one as follows: (i) the new initial state is $I = \{s_i\}$, where s_i is the state before the inconsistent one, that is $F(s_i,a_i) = s_{i+1}$ where s_{i+1} violates the rules, and (ii) the new goal is to "execute action a_i". Intuitively, a corrective action sequence represents an alternative route leading the system from a state s_i to a state s_j where the action a_i can be applied (without violating the consistency rules). To this aim, in this phase the planner *explores* the search space and *selects* the best corrections according to a given policy.

Definition 3. *Let $PP = (\mathcal{S},G,T)$ be a planning problem and let C be a Universal Checker of PP. Then, a T-*cleansing action sequence *for the pair $(s_i,a_i) \in C$ is a non-empty sequence of actions $\epsilon^c = c_0,\ldots,c_n$, with $|\epsilon^c| \leq T$ s.t. exists a path $\pi_c = s_ic_0 \ldots s_{i+n}c_n\ s_ka_is_{k+1}$ on $\mathcal{S}$, where all the states $s_i,\ldots,s_k \notin G$ whilst $s_{k+1} \in G$.*

Finally, let $C : S \times A \rightarrow \mathbb{R}^+$ be a cost function, a cost-optimal *cleansing sequence is a sequence s.t. for all other sequences π'_c the following holds: $C(\pi_c) \leq C(\pi'_c)$.*

Table 2: Some corrective action sequences

state	employed [FT,Limited,CompanyX]
Inconsistent event	(start,PT,Limited,CompanyY)
Alternative 1	(cessation,FT,Limited,CompanyX)
Alternative 2	(conversion,PT,Limited,CompanyX)
Alternative 3	(conversion,PT,Unlimited,CompanyX)
Alternative 4	(conversion,FT,Unlimited,CompanyX) (cessation,FT,Unlimited,CompanyX)

Then, a UC is a collection of cleansing action sequences synthesised for each inconsistency identified.

Definition 4. *Let* C *be a universal checker. A* Universal Cleanser *is a* map $\mathcal{K} : Reach(S) \times A \rightarrow 2^A$ *which assigns to each pair* $(s_i, a_i) \in C$ *a T-cleansing action sequence* ϵ^c.

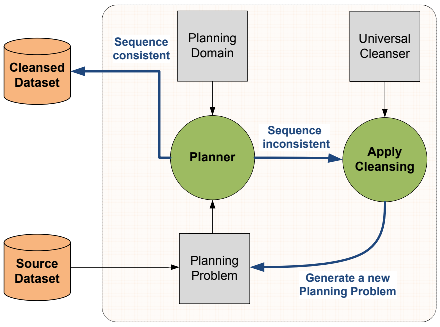

Figure 1: Overview of the cleansing process

The UC is synthesised off-line and it contains a *single* cost-optimal action sequence for each entry. Clearly, the cost function is domain-dependent and usually driven by the purposes of the analysis.To give a few examples, one could fix the data by minimising/maximising either the number of interventions or an indicator computed on the overall cleansed sequence. We remark that the UC generated is *domain-dependent* as it can deal only with event sequences conforming to the model used during its generation. On the other hand, the UC is also *data-independent* since it has been computed by taking into account all the feasible (bounded) event sequences, and this makes the UC able to cleanse *any* data source, as shown in Fig. 1.

Preliminary Results and Expected Outlook. Actually, we used the UPMurphi temporal planner (Della Penna et al. 2012; Della Penna et al. 2009) to synthesise a UC for our domain, by exploiting the planning as model checking paradigm, as shown in (Mezzanzanica et al. 2013; Boselli et al. 2013). The UC has been synthesised with no optimisation requirements, thus the result is an *exhaustive* repository of all the (bounded) feasible cleansing activities. The UC contains 342 different *error-codes*, i.e. *all* the possible 3-steps (state,action) pairs leading to an inconsistent state of the model. We have performed a data consistency evaluation on an online dataset provided in (Mezzanzanica et al. 2013) [4] composed by 1,248,814 anonymized mandatory communications describing the careers of 214,429 people observed in ten years. UPMurphi has recognised 92,598 careers as inconsistent (43% of total) assigning an error-code to each of them.

Such a kind of result is actually quite relevant for domain-experts at CRISP as it provides a bird's eye view of the inconsistency distribution affecting the source dataset. For instance, we discovered that about 30% of total inconsistencies arose due to an extension, cessation or conversion event received when the worker was in the *unemployed* status, showing that the identification of cleansing activities for these careers is essential to guarantee the quality of the cleansed data.

As a further step, we intend to model the labour market domain through PDDL3 (that would enable the use of temporal logics) and to realise the process described in Fig. 1 by connecting a PDDL planner to our DBMS, so that the archive can be cleansed at real-time.

[4] Dataset available at `http://goo.gl/zrbrR`

References

[Batini and Scannapieco 2006] Batini, C., and Scannapieco, M. 2006. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer.

[Bertossi 2006] Bertossi, L. 2006. Consistent query answering in databases. *ACM Sigmod Record* 35(2):68–76.

[Boselli et al. 2013] Boselli, R.; Cesarini, M.; Mercorio, F.; and Mezzanzanica, M. 2013. Inconsistency knowledge discovery for longitudinal data management: A model-based approach. In *CHI-KDD*, volume 7947 of *LNCS*, 183–194. Springer.

[Chomicki and Marcinkowski 2005] Chomicki, J., and Marcinkowski, J. 2005. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*. Springer. 119–150.

[Chomicki 1995] Chomicki, J. 1995. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems (TODS)* 20(2):149–186.

[Della Penna et al. 2009] Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009. UPMurphi: a tool for universal planning on PDDL+ problems. In *ICAPS 2009*, 106–113. AAAI Press.

[Della Penna et al. 2012] Della Penna, G.; ; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Applied Intelligence* 36(4):932–959.

[Fan et al. 2010] Fan, W.; Li, J.; Ma, S.; Tang, N.; and Yu, W. 2010. Towards certain fixes with editing rules and master data. *Proceedings of the VLDB Endowment* 3(1-2):173–184.

[Fisher et al. 2012] Fisher, C.; Lauría, E.; Chengalur-Smith, S.; and Wang, R. 2012. *Introduction to information quality*.

[Kolahi and Lakshmanan 2009] Kolahi, S., and Lakshmanan, L. V. 2009. On approximating optimum repairs for functional dependency violations. In *ICDT*, 53–62. ACM.

[Lovaglio and Mezzanzanica 2013] Lovaglio, P. G., and Mezzanzanica, M. 2013. Classification of longitudinal career paths. *Quality & Quantity* 47(2):989–1008.

[McDermott et al. 1998] McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. Pddl-the planning domain definition language.

[Mezzanzanica et al. 2013] Mezzanzanica, M.; Boselli, R.; Cesarini, M.; and Mercorio, F. 2013. Automatic synthesis of data cleansing activities. In *DATA*, 138 – 149. Scitepress.

[Rahm and Do 2000] Rahm, E., and Do, H. 2000. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin* 23(4):3–13.

[The Italian Ministry of Labour and Welfare 2012] The Italian Ministry of Labour and Welfare. 2012. Annual report about the CO system, available at `www.cliclavoro.gov.it/Barometro-Del-Lavoro/Documents/Rapporto_CO/Executive_summary.pdf` last accessed: November 2013.

[Vardi 1987] Vardi, M. 1987. Fundamentals of dependency theory. *Trends in Theoretical Computer Science* 171–224.

[Yakout, Berti-Équille, and Elmagarmid 2013] Yakout, M.; Berti-Équille, L.; and Elmagarmid, A. K. 2013. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *International conference on Management of data*, 553–564. ACM.