Planning Meets Data Cleansing*

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

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 - a collection of cleansing actions for each data inconsistency - as a planning problem. We present then a motivating government application in which it has be used.

Keywords: Data Quality, Data Cleansing, Government Application

Introduction and Related Work

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 focusing on *business rules* relying on the experience of domain-experts. Furthermore, exploring cleansing alternatives is a very time-consuming task as each business rule has to be analysed and coded separately, then the overall solution still needs to be manually evaluated. This paper aims at expressing data cleansing problems via planning to contribute in addressing the following issues in the data cleansing area.

(i) Modelling the behaviour of longitudinal data. Usually *longitudinal data* (aka *panel* or *historical data*) extracted by Information Systems (ISs) provide knowledge about a given subject, object or phenomena observed at multiple sampled time points (see (Singer and Willett 2003; Bartolucci, Farcomeni, and Pennoni 2012). In this regard, planning languages like PDDL can help domain experts formalising how data should evolve according to an expected behaviour. Specifically, a planning domain describes how data arriving from the external world - and stored into 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". Here we shall focus on *consistency*, which refers to "the violation of semantic rules defined over (a set of) data items." (Batini and Scannapieco 2006). In reference to relational models, such "semantic rules" have usually been expressed through functional dependencies (FDs) and their variants useful for specifying integrity constraints. 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 how the the database state has evolved over time. Furthermore, even though FDs enable the detection of errors, 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). Planning formalisms are expressive enough to model complex temporal constraints then a cleansing approach based on AI planning might allow domain experts to concentrate on *what* quality constraints have 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*(see. e.g. (Chomicki and Marcinkowski 2005)). As a drawback 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 promising heuristics approaches have been defined (Yakout, Berti-Équille, and Elmagarmid 2013;

*This work is partially supported within a Research Project granted by the CRISP Research Centre (Interuniversity Research Centre on Public Services - www.crisp-org.it) and Arifl (Regional Agency for Education and Labour - www.arifl.regione.lombardia.it)

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

Kolahi and Lakshmanan 2009), whose effectiveness still has to be evaluated on real-life problems.

Recently the NADEEF (Dallachiesa et al. 2013) and LLUNATIC (Geerts et al. 2013) tools have been developed for unifying the most used cleansing solutions by both academy and industry through variants of FDs. As Dallachiesa et al. argue, the consistency requirements are usually defined on either a single tuple, two tuples or a set of tuples. While the first two classes can be modelled through FDs and their variants, the latter class of quality constraints requires reasoning with a (finite but not bounded) set of data items over time as the case of longitudinal data, and this makes the exploration-based technique (as the AI Planning) a good candidate for that task.

In this regard, planning would contribute to the generation 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.

The Labour Market Dataset

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), 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 respectively. **e_date** is the event occurrence date whilst **e_type** describes the event type occurring to the worker's career. The event types can be 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 has 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 be changed (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$_Y$

Let us consider a worker's career as in Tab. 1. A worker started a limited-term part-time contract with Company$_X$ then converting it twice: to unlimited-term in November 2010 and to full-time in January 2012. 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 as it violates the constraint c1. Clearly, there are several alternatives that domain experts may define to fix the inconsistency (as shown in Tab. 2) many of which are often inspired by common practice. Probably, in the example above a communication was lost. Thus it is reasonable to assume that the full-time contract has been closed in a period between 12^{th} January 2012 and 28^{th} July 2013.

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)

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 take into account such hypothesis.

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 are more interesting for the analysis purposes. 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 and an ordered set of records forms an *event sequence*. We can formalise this concept as follows.

Definition 1 (Events Sequence) *Let $\mathcal{R} = (R_1, \ldots, R_l)$ 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 $m \leq l | r_1 \in R_1, \ldots, r_m \in R_m$;*

(ii) Let $\sim$ be a total order *relation over events, an* event sequence *is a $\sim$-ordered sequence of events $\epsilon = e_1, \ldots, e_k$ 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 represents the object or subject we are observing, while an event is an action able to modify the system state.

The Fig. 1 should clarify the matter by showing some PDDL components modelling the Labour Market domain. The `CO` predicate represents a communication as described before while the `is_active` predicate is used for modelling an ongoing contract. Then, the actions are mapped on the e_type domain, as the case of the `start_contract` action. Specifically, the action requires that a `START` communication is arrived with the expected `event_seq_number` to be fired. The action's effect is to add the `is_active` predicate (initiated with the communication parameters) and to enable the receiving of the next event (if any).

```
(:types
  worker c_type c_flag  company_t e_type nat_number - object)

(:constants
 START END CONVERT EXTEND NULL_E - e_type
 LIMITED UNLIMITED NULL_C - c_type
 PT FT NULL_T - c_flag
 CX CY CZ NULL_CM - company_t)

(:predicates
(CO ?w - worker ?ct - c_type ?f - c_flag ?c - company_t
  ?e - e_type  ?n - nat_number)
(event_seq_number ?n1 - nat_number)
(is_active ?c1 - company_t ?f1 - c_flag ?ct1 - c_type)
(next ?n - nat_number ?m - nat_number))

(:action start_contract
 :parameters (?w - worker ?ct - c_type ?f  - c_flag ?c -
 company_t ?e - e_type ?n ?m - nat_number)
 :precondition (and
  (event_seq_number ?n)
  (next ?n ?m)
  (mn ?w ?ct ?f ?c ?e ?n) ;; the CO exists
  (= ?e START)) ;; it is a "start" event
 :effect (and
  (not (event_seq_number ?n)) ;; disable the n-th CO
  (event_seq_number ?m) ;; be prepared for receiving the m-th
   CO
  (is_active ?c ?f ?ct))) ;; add the contract
```

Figure 1: Main PDDL domain components

Once a model describing the evolution of an event sequence has been defined, a planner works in two steps for generating the cleansing activities.

Step 1. It simulates the execution of *all* the feasible (bounded) event sequences, summarising all the inconsistencies into an object, the so-called *Universal Checker* (UCK), that would represent a topology of the inconsistencies that may affect a data source.

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. This task can be accomplished by enabling a planner to continue the search when a goal has been found. The Fig. 2 shows an example of goal statement for catching simulated career paths that violate the constraint C1. We formalise such a planning problem on FSSs as follows.

```
(:goal (and
(exists (?c1 ?c2 ?c3 - company_t ?ct1 ?ct2 ?ct3 - c_type)
  (or (and
        (is_active ?c1 FT ?ct1)
        (is_active ?c2 FT ?ct2)
        (or (not (= ?c1 ?c2)) (not (= ?ct1 ?ct2))))
      (and
        (is_active ?c1 FT ?ct1)
        (is_active ?c2 PT ?ct2))))))
```

Figure 2: An example of PDDL goal statement

Definition 2 (Planning Problem on FSS) *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 $F : S \times A \to S$ 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) $\pi = s_0 a_0 s_1 a_1 \ldots s_{n-1} a_{n-1} s_n$ 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 $s_n \in G \subseteq Reach(S)$.*

Thus a collection of all the inconsistencies identified can be synthesised as follows.

Definition 3 (Universal Checker (UCK)) *Let $PP = (\mathcal{S}, G, T)$ be a Planning Problem on a FSS $\mathcal{S}$ = (S,I,A,F). Moreover, let Π be the set of all paths able to reach a goal for PP, then a* Universal Checker *$\mathcal{C}$ is a collection of state-action pairs that for each $\pi \in \Pi$ summarises all the pairs (s_{n-1}, a_{n-1}) s.t. $F(s_{n-1}, a_{n-1}) \in G$.*

Step 2. *For each* pair $(s_i, a_i) \in UCK$ denoting an inconsistency classified with a unique identifier - the *error-code*, 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; (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 criterion.

Definition 4 (Cleansing Actions Sequence) *Let $PP = (\mathcal{S}, G, T)$ be a planning problem and let $\mathcal{C}$ be a Universal Checker of PP.*

Then, a T-cleansing actions sequence *for the pair $(s_i, a_i) \in \mathcal{C}$ is a non-empty sequence of actions $\epsilon^c = c_1, \ldots, c_j$, with $|\epsilon^c| \leq T$ s.t. exists a path $\pi_c = s_i c_1 \ldots s_{i+j} c_j \; s_k a_i s_{k+1}$ on $\mathcal{S}$, where all the states $s_i, \ldots, s_k \notin G$ whilst $s_{k+1} \in G$. Finally, let $\mathcal{W} : S \times A \to \mathbb{R}^+$ be a cost function, a* cost-optimal *cleansing sequence is a sequence s.t. for all other sequences π'_c the following holds: $\mathcal{W}(\pi_c) \leq \mathcal{W}(\pi'_c)$ by denoting the cost of the path π_c as $\mathcal{W}(\pi'_c) = \sum_{i=0}^{k-1} \mathcal{W}(s_i, a_i)$.*

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

Definition 5 (Universal Cleanser (UC)) *Let $\mathcal{C}$ be a universal checker. A* Universal Cleanser *is a* map $\mathcal{K} : Reach(S) \times A \rightarrow 2^A$ *which assigns to each pair* $(s_i, a_i) \in \mathcal{C}$ *a T-cleansing action sequence* ϵ^c.

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: one could fix the data by minimising/-maximising either the number of interventions or an indicator computed on the overall cleansed sequence.

Furthermore the UC 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 (bounded) event sequences, and this makes the UC able to cleanse *any* data source, according to the process shown in Fig. 3.

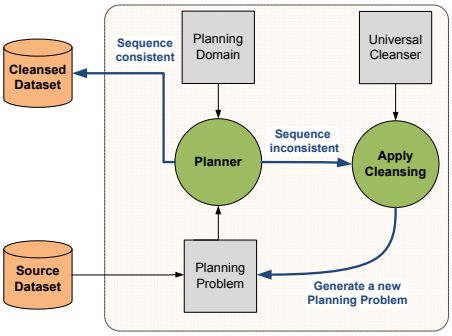

Figure 3: Overview of the cleansing process

Preliminary Results and Future Outlook

We used the UPMurphi temporal planner (Della Penna et al. 2009; Mercorio 2013) to synthesise a UC for the domain presented, by exploiting the planning as model checking paradigm. Furthermore, it is the basis of the (Multidimensional) Robust Data Quality Analysis (Mezzanzanica et al. 2011; Boselli et al. 2013). Notice that the UC has been synthesised with no optimisation criteria, thus it actually represents 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.

An important metric for measuring data quality is the initial quality level of the source archive. Hence, as a first step we used the UPMurphi planner to identify careers presenting at least one inconsistency on $1,248,814$ anonymized COs describing the careers of $214,429$ people. UPMurphi has recognised $92,598$ careers as inconsistent (43% of total). The blue triangles in Fig. 4 represent error-codes found on the dataset whilst red crosses identify error-codes not found. Such a kind of result is actually quite relevant for domain-experts at CRISP institute as it provides a bird's eye view of the inconsistency distribution affecting the source dataset. The refinement of cleansing activities for such careers plays a crucial role in guaranteeing the quality of

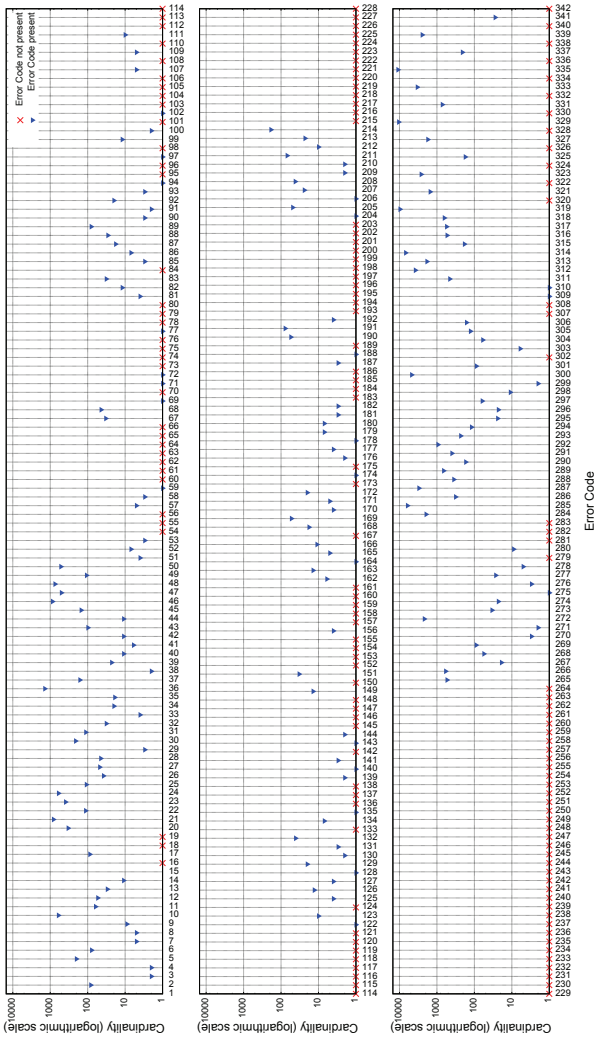

Figure 4: A graphical visualisation of the distribution of the error-codes found on the dataset. The x-axis reports the error-codes while the y-axis summarises the number of careers affected by that error.

the cleansed data. For instance, the three most numerous error codes are related to an extension, cessation or conversion event received when the worker was in the *unemployed* status (error codes 335, 329 and 319 represent about 30% of total inconsistencies). For the sake of completeness, the dataset and the results have been made available online at http://goo.gl/DBsKTp. The traditional development of cleansing routines is a resource consuming and error prone activity as the huge set of data to be cleansed, the complexity of the domain, and the continuous business rules evolution make the cleansing process a challenging task.

As a further step, we intend to model the labour market domain through PDDL that would represent the first planning benchmark problem of a data quality scenario. Then, a lot of off-the-shelf planners such as METRIC-FF (Hoffmann 2001) might be used and evaluated on a real-life dataset.

References

Bartolucci, F.; Farcomeni, A.; and Pennoni, F. 2012. *Latent Markov models for longitudinal data*. Boca Raton, FL: Chapman & Hall/CRC Press.

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

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

Boselli, R.; Cesarini, M.; Mercorio, F.; and Mezzanzanica, M. 2013. Inconsistency knowledge discovery for longitudinal data management: A model-based approach. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data - Third International Workshop, HCI-KDD*, volume 7947 of *LNCS*, 183–194. Springer.

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

Dallachiesa, M.; Ebaid, A.; Eldawy, A.; Elmagarmid, A. K.; Ilyas, I. F.; Ouzzani, M.; and Tang, N. 2013. Nadeef: a commodity data cleaning system. In Ross, K. A.; Srivastava, D.; and Papadias, D., eds., *SIGMOD Conference*, 541–552. ACM.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009. UPMurphi: a tool for universal planning on PDDL+ problems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 106–113. AAAI Press.

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, C.; Lauría, E.; Chengalur-Smith, S.; and Wang, R. 2012. *Introduction to information quality*.

Geerts, F.; Mecca, G.; Papotti, P.; and Santoro, D. 2013. The llunatic data-cleaning framework. *PVLDB* 6(9):625–636.

Hoffmann, J. 2001. FF: The fast-forward planning system. *AI magazine* 22(3):57.

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

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

Mercorio, F. 2013. Model checking for universal planning in deterministic and non-deterministic domains. *AI Commun.* 26(2):257–259.

Mezzanzanica, M.; Boselli, R.; Cesarini, M.; and Mercorio, F. 2011. Data quality through model checking techniques. In *Proceedings of the 10th International Conference on Intelligent Data Analysis (IDA)*, volume 7014 of *Lecture Notes in Computer Science*, 270–281. Springer.

Singer, J., and Willett, J. 2003. *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford University Press, USA.

The Italian Ministry of Labour and Welfare. 2012. Annual report about the CO system, available at http://goo.gl/XdALYd last accessed 6 november 2013.

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

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.