Part Attribute Discovery for On-Demand Manufacturing

Terrance Swift
Department of Computer Science
SUNY Stony Brook
Stony Brook, N.Y.
tswift@cs.umd.edu

June 1, 1999

1 Introduction

The Defense Logistics Agency (DLA) purchases over $\$nnn$ of supplies a year for the various branches of the armed forces. Standard commodities such as food and fuel constitute much of these purchases, but a large amount of these purchases are comprised of spare parts needed to maintain various weapons systems. To support these purchases DLA databases must maintain information on over *nnn* spare parts. Maintaining this information is, by any standards, a difficult task. First, storing information about such a large number of parts can be a problem by itself. Second, it is not usually obvious how information about a spare part or part assembly should be structured. And finally, information about parts is continually changing, as procurement channels change or as parts are redesigned. DLA has responded to all of these problems. It maintains several large hierarchical databases, most noticeably the FLIS catalog of parts, and the CTDF procurement database[1]. However, there are problems with retrieving part information from these databases. Data is often entered textually in these databases, partly because integrity constraints have not been available in the underlying hierarchical systems, but more importantly because it is difficult to structure information about parts within the constraints of a hierarchical (or relational) database. In addition, there are often inconsistencies between the FLIS and CTDF databases. As an example, changes stemming from, say, re-engineering, are often made to the CTDF database, but are not propagated to the FLIS database. While FLIS provides means for such propagation, the complex structure of both databases is often not understood by users, who as a result may not make use of mechanisms for propagating updates from one database to the other.

The situation can be summarized by saying that DLA databases contain a remarkable amount of information, but that the information is poorly structured and prone to inconsistencies between databases. The premise of part attribute discovery is that the DLA databases have a rich informational content that is poorly exploited. This document is intended as a technical guide to understanding how attributes of parts can be *discovered* from DLA databases to aid on-demand manufacturing. In particular, we describe Version 1.5.1 of the part attribute extractor which is implemented using the XSB system [5, 2]. It is important to note that the rules for technical

[1] Many DLA product centers also support access to this data from Oracle which is a relational system. The structure of the Oracle database, though, is derived from that of the hierarchical databases.

attribute extraction are being continuously modified and refined, so that the statements made in this paper for Version 1.5.1 may not be true about future versions of the attribute extractor.

We define attribute discovery as *the process of inferring a structured, coherent view of the attributes of a part or part assembly from a collection of relational or textual databases.* In order to explain this definition, we consider its application to DLA databases. A DLA part is defined by a *National Stock Number*, or NSN. The NSN in turn is made up of a *National Item Identification Number* or NIIN along with a *Federal Supply Code* identifier or FSC. The NIIN forms a primary key to information both in the CTDF database, and in the FLIS database. Information in each of these databases may be unstructured, in the sense that it is embedded within textual fields. For instance, consider the following data fact[2]:

```
tc('000044958','GENERAL CHARACTERISTICS ITEM DESCRIPTION',
                ' STEEL,DADMIUM PLATED QQ-P-416').
```

This fact, which is in Prolog syntax, shows that the material type, and plating are embedded into what is essentially free text in which a misspelling has occurred. The fact can be read in the following manner. The fact begins with the table name, `tc`, and the set of fields of the fact (also called *arguments*) is delimited by parentheses, and ended with a period. Each single field in the fact is separated by a comma, and, in this case, delimited by single quotation marks. The Prolog syntax may appear somewhat redundant for simple facts such as this — for instance by requiring both parentheses and a period for delimiting facts. However, Prolog syntax is based on first-order logic which allows representation of data and complex rules in the same manner. This representation can lead ro powerful mechanisms to reason about data some of which will be described below.

The first task of attribute discovery then is to extract relevant data from a given field, and if that field is textual, extraction in itself may be complicated (see [8, 4, 1] for examples of this). Once data is extracted, inferences may need to be made in order to get the most out of the extracted data. To illustrate this process, we consider another example of a sleeve bushing, whose technical characteristics information is as follows:

```
tc('011179479','ITEM NAME','BUSHING,SLEEVE').
tc('011179479','OVERALL LENGTH','0.2400INCHES MINIMUM AND 0.2450 INCHES MAXIMUM').
tc('011179479','BODY OUTSIDE DIAMETER','0.6882 INCHES MINIMUM AND 0.6891 INCHES MAXIMUM').
tc('011179479','BODY INSIDE DIAMETER','0.4930 INCHES MINIMUM AND 0.4950 INCHES MAXIMUM').
tc('011179479','CORNER SHAPE',' SQUARE ALL CORNERS').
tc('011179479','HARDNESS RATING',
                '33.0 ROCKWELL C MINIMUM OVERALL AND 37.0 ROCKWELL C MAXIMUM').
tc('011179479','SURFACE TREATMENT',' CADMIUM OVERALL').
tc('011179479','SURFACE TREATMENT DOCUMENT AND CLASSIFICATION',
                'QQ-P-416,TY 1,CL 3 FED SPEC SINGLE TREATMENT RESPONSE OVERALL').
tc('011179479','STYLE DESIGNATOR','17C SOLID').
```

It turns out that there is no CTDF Product Information Data (PID) on the part. Note that the material is missing, but there is information that the part is cadmium plated arising from its surface treatment and its surface treatment document and classification. Using this information, an engineering knowledge base is used to infer that the part is made of steel. All inferred information about the part is maintained in a *coherent database* that structures information gathered from the technical characteristics, PID, information derived from decoding the commercial part number, and

[2]Actual DLA data is used for all examples in this paper. While we try to make the content of this paper self-contained, we rely on a slight reading knowledge of XSB (Prolog) syntax, although this syntax should be easily understandable to anyone familiar with relational databases.

so on. For the previous part, the relevant portion of the coherent database is represented by the following facts:

```
process('011179479','PLATING','CADMIUM',tc,'QQ P 416').
dimension('011179479','LENGTH',0.2400,tc,ss('OVERALL LENGTH')).
material('011179479','STEEL',null,tc,proc_mat_const('PLATING')).
```

For this part, information about process, dimension, and material are all extracted or inferred from the technical characteristics. While specifics of the coherent database are discussed in Section 6, we mention for now that the second to last field denotes the *source* of the information: in this case `tc` denotes technical characteristics. The last field is termed the *reason* field. For the process this indicates that the reason for inferring cadmium plating is a federal specification, for dimension the reason is a textual search of the `OVERALL LENGTH` field, and for the material, the reason is a constraint that the cadmium plating process puts on the material.

The process of part attribute discovery then, can be quite complex, with inferences based on data that itself may be inferred. Users however will not need to understand this complexity. The coherent database is designed to be viewed in a database such as Microsoft Access (or directly from XSB). Furthermore, if a user did not want to see information from a particular database, or information based on certain classes of inferences, a database view could be designed so that the user was not presented with the information.

We discuss all aspects of part attribute extraction in detail in later sections of this paper. For now, we note that we divide this information into three types: *technical attributes*, *engineering attributes* and *business attributes*. Technical attributes reflect the materials of which the part is made, of the manufacturing processes required for the part, the dimensions of a part, and so on. Engineering attributes reflect information not about the part itself, but about the knowledge of the part: examples include information about whether a technical drawing is available for the part, about whether government mylar is required for making the part, whether inspection of the part is to be done by the manufacturer, and so on. Finally, the business attributes of a part include, for instance, the average amount ordered each year, the average cost, the procurement history. The current part attribute extractor concerns itself only with technical and engineering attributes.

1.1 Overall Structure of Part Attribute Discovery

The process of part attribute discovery occurs in several steps, each of which will be explained in detail in the sections that follow.

1. Military, federal, and commercial specifications are extracted from technical characteristics, pid, and pid control data. Using a knowledge base of specifications, information about technical attributes such as materials, material shapes, processes is derived (Section 3).

2. Other textual data in the technical characteristics, pid and pid control data is then analyzed to extract technical, engineering and business attributes, such as past procurement history (Section 4).

3. Using what are termed *ncp* tables, consisting of tuples of NIINs, Cage identifiers, and Commercial Part Numbers, the commercial part number itself is parsed to determine information such as the weapon subsystem of a part, the material, and process (Section 5).

4. Finally, a set of inferences are performed to infer, say the part material from the manufacturing process used to make the part, or to infer the process or material from the nomenclature of the part (Section 6)

Before explaining this process we discuss the engineering knowledge which is used throughout part attribute discovery.

2 An Engineering Knowledge Base

A critical aspect of part attribute discovery concerns how to represent data about a part, once that data has been extracted and inferred. To take an example, one data source may represent the material of a part as `STEEL`, another as `STAINLESS STEEL`, and yet another as `CRES`. It is the function of the knowledge base to represent that `STAINLESS STEEL` is a synonym for `CRES`, and that both are subtypes of `STEEL`. The rules for representing part attribute knowledge are in the *knowledge base* of the attribute extractor.

One principle of knowledge representation is that knowledge should be represented in as declarative a manner as possible — in statements or simple logical rules, rather than in embedded in procedures. The disadvantage of embedding knowledge (or assumptions) within procedures can be seen from the difficulties caused by so-called millenial bugs, in which assumptions about how dates are stored an manipulated are buried deeply within procedural code. Logic programming in general and XSB in particular offer advantages for knowledge representation since they treat rules and data in the same manner, and allow representation and manipulation of structured data.

The knowledge base of the technical attribute extractor is largely declarative, but does have some knowledge represented procedurally. We discuss aspects of the knowledge base.

Attribute Hierarchy The attribute hierarchy provides a standard vocabulary for representing part attributes. Consider the following fragment of code:

```
isa_link('BONDING','PROCESS').
   isa_link('ADHESION','BONDING').
      isa_link('LAMINATING','ADHESION').

   isa_link('METAL METAL BONDING','BONDING').
      isa_link('SOLDERING','METAL METAL BONDING').
      isa_link('BRAZING','METAL METAL BONDING').
      isa_link('WELDING','METAL METAL BONDING').

         isa_link('FRICTION WELDING','WELDING').
         isa_link('FUSION WELDING','WELDING').
         isa_link('PRESSURE WELDING','WELDING').
```

This code, which is in standard Prolog syntax, can be read as specifying the subtype relationships between various processes. For instance, bonding is a subtype of process, while metal-metal bonding is a subtype of bonding. In terminology common to knowledge bases, this is depicted in an through an *isa* relation, (pronounced "is-a"). The relation name `isa_link` is used to denote that one process

is an *immediate* subtype of another process, while the relation name *isa* is used to represent that one term is a subtype of another, but not necessarily an immediate subtype. For instance, the isa relation holds between pressure welding and process, but the isa_link relation does not hold. Rules for creating the isa relation from the isa_link relation are also contained as part of the knowledge base.

One purpose of the hierarchies is to guide the translation of specifications. Consider the following Boeing process specification:

```
BAC5901: Certification of Weld Settings for Machine Fusion Welding.
```

Translated into a knowledge base fact, this spec would be represented as

```
spec_material_attr('BAC 5901','FUSION WELDING',process).
```

The first argument of this fact is the standardized form of the specification, the second is the value field, that the specification has something to do with fusion welding, and the third the type of information in the specification. The exact form of the specification knowledge base is presented in Section 3.

In Version 1.5.1 of the attribute extractor, there is a hierarchy for materials and processes, as well as a hierarchy for aircraft subsystems. However, sometimes information is derived from a specification, but there is not a hierarchy available to categorize the information. In this case, the information is termed *unintegrated*. In Version 1.5.1 information about electrical attributes of parts is one type of information that is unintegrated. In the case of unintegrated data, knowledge can be maintained in simple tables:

```
electrical_spec('BAC 5105').
```

A second purpose of the hierarchies is to allow resolution of data from different sources, as in the example about `STEEL` and `CRES` above. The usual strategy in this case is to take the more specific information, and conclude that a given part was made from corrosion-resistant steel. Finally, hierarchies also aid in *keyword searches* as discussed in Section 4.

3 Deriving Part Attribute Information From Specifications

Military, commercial and industrial specifications are some of the best sources of information of part attributes. In order to use this information specifications are first extracted from textual data and afterwards a knowledge base is used to reason about the extracted specifications.

3.1 Extracting Specifications from Textual Data

Given the technology of standardization, extracting specifications from technical data is not particularly difficult, although care must be taken to address variations in the presentation of specs, such as whether dashes are used or not and other variations in punctuation. Currently the following classes of specifications are extracted.

- Military Specifications, such as `MIL-P-15035`.
- Military Standards, such as `MIL STD 130J`.

- Federal Specifications, such as `QQ P 35`, `WW-T-700`, `TT-V-109`, or `VV-L-800`.
- Industrial Specifications. These comprise several types:
  - Those from the American Metallurgical Society (e.g. `AMS 2406`);
  - Those from the American National Standards Institute (e.g. `ANSI B 46.1`);
  - ASTM specifications (e.g. `ASTM B221`);
  - Those from the American Iron and Steel Institute: AISI specifications (e.g. `AISI 301`).
- Commercial Specifications. Currently these include
  - Boeing Material Specifications (e.g. `BMS 1 17`);
  - Boeing Process Specifications (e.g. `BAC 5982`);
  - Grumman specifications (e.g. `GSS 8050`);
  - General Motors specifications (e.g. `GM 3011`).

Specification extraction is a straightforward standardization problem in that, in most cases no account need be taken of linguistic contexts. However in the PID, there are occurrences of text such as

```
... ITEM PROCESSES AND FINISHES ARE IN ACCORDANCE WITH FOLLOWING
SPECIFICATIONS: MIL-H-6875 IN LIEU OF GSS5100
```

The Grumman specification `GSS 5100` is outdated. The specification extraction routines recognize variations of 'IN LIEU OF', along with variations of 'DELETE REFERENCE' and ALTERNATE MATERIAL.

Extracted specifications create tables of the form

$$contains\_spec(NSN, Spec, Source, Reason).$$

Here, *Spec* is a standardized specification of one of the types mentioned above. *Source* has the form:

- `tc` if the specification was extracted from the technical characteristics relation.
- `pid` if the specification was extracted from the PID relation.
- `ctrl` if the specification was extracted from the PID control relation.

Reason has the form:

- `ilo(`*Spec*`)` if the specification was extracted from an in-lieu-of context;
- `del(`*Spec*`)` if the specification was extracted from a delete-reference context;
- `alt(`*Spec*`)` if the specification was extracted from an alternate-material context;
- `base` otherwise.

Ordinarily, only those specifications whose reasons were `base` or `alt/1` would be useful for further inferencing.

3.2 Translating Technical Specifications

Once the technical specifications have been extracted, they must be translated into useful information about the material of a part, the process used to make the part, or other information. As mentioned in Section 2, the specification knowledge base is used for this purpose. This knowledge base consists of several relations:

- Attributes for which a classification hierarchy exists — currently materials and processes; and
- *Unintegrated* attributes, for which a classification hierarchy has not yet been made.

We discuss each in turn.

Two relations are used to translate materials and processes: `spec_material(Specification,Material)`, and `spec_material_attr(Specification,Attribute,Attribute_type)`. In these relations, `Specification` is a standardized specification of the types outlined in Section 3. `Material` is a material in the material hierarchy. `Attribute_type` is either process or shape. If `Attribute_type` is `process`, then `Attribute` is an element of the process hierarchy. Otherwise, if `Attribute_type` is `shape`, then the specification indicates either a shape or a set of possible shapes. Currently, the shapes for metals are

```
'BAR', 'CASTING', 'EXTRUSION', 'INGOT', 'PLATE', 'RING', 'SHEET', 'STRIP',
                              'TUBE', 'WIRE',
```

Shapes for non-metals are currently experimental and are subject to review. As of September, 1998 the knowledge base contains material or material attribute information for 786 specifications.

The hierarchy of parts and materials is geared towards aircraft structural parts. Other types of information may be available from specifications that is considered to be *unintegrated* with a particular knowledge hierarchy. Currently, specification information is used to infer whether a part is in one of the following broad classes:

```
magnetic_spec(Spec).
fiber_optic_spec(Spec).
honeycomb_spec(Spec).
hydraulic_spec(Spec).
electrical_spec(Spec).
```

These classes are expected to change as new knowledge hierarchies are created (e.g. for electrical parts) and as new unintegrated classes are identified. As of September, 1998 there are 194 specifications thus partially classified.

4 Textual Searches Beyond Specifications

Standardization techniques [8, 4, 1] are used to extract information about the overall dimensions of a part, the immediate constituents or next higher assembly of a part, the designated platform for a part, logistics information about a part, material, process, and unintegrated technical attributes (specification information is also used for the last three classes of information). In most cases the technical characteristics, the PID, and PID control files are used. Before describing in detail the

information extracted, we describe at a high level standardization and associated techniques used for extraction of these attributes.

Techniques for Standardization Consider the following row from the technical characteristics table

tc('010415788','GENERAL CHARACTERISTICS ITEM DESCRIPTION','COPPER 0.010 IN. THICK 1/4 HARD PER AST BM-152,FINISH TIN PLATE,3.101P 0.004 M 0.000 DIA,PART OF TANK ASSEMBLY PN65308-12023-042').

This single row contains information about the primary material of the NIIN, along with surface treatment, dimension, hardness, and part hierarchy. In order to extract this information, the *value* field of this record (the third field) must be standardized. The part attribute extractor uses a simplified form of standardization consisting of three phases. The first phase of *tokenization* is a relatively simple process whereby the field is broken up into a *list* of tokens representing the words and numbers, in which punctuation and whitespace have been removed. The XSB representation of this token list separates tokens by the comma symbol:

COPPER, decimal(0.010), IN, THICK, integer(1), integer(4), HARD, PER, AST, BM, integer(152), FINISH, TIN, PLATE, decimal(3.101), P, decimal(0.004), M, decimal(0.000), DIA, PART, OF, TANK, ASSY, PN, integer(65308), -, integer(12023), -, integer(042)

The second phase of part attribute standardization performs a simple `context free parse` which corrects common misspellings and transforms variant forms of words (such as abbreviation) into a common form. For instance, the context-free parse might transform the token `ASSEMBLY` into `ASSY`. This phase of technical attribute extraction allow later inference rules to be written more easily, since they will not have to check for misspellings and variant forms of words. At a technical level, the context-free parse uses *parse tables* to correct mispellings and to create *super tokens* such as `'BLACK OXIDE COATING'` out of one or more tokens. Parse tables are generated automatically from the various hierarchies, additional information can be entered by hand.

The third phase either uses Prolog's *Definite Clause Grammars* of DCGs or *keyword searches*. DCGs are sets of rules that specify how to parse, or partially parse textual information. Unfortunately, maintaining DCGs requires an XSB/Prolog programmer, and a detailed explanation of DCGs is beyond the scope of this paper. Keyword searches, however are maintainable without a knowledge of XSB and are explained below.We note that these phases of standardization also are used to extract the technical specifications of Section 3.

Structured and Unstructured Search Determining that copper is the primary material of an NIIN from the free-text `GENERAL CHARACTERISTICS ITEM DESCRIPTION` field above, or from free-text PID information is more difficult than, determining that copper is the surface treatment material designated by the row:

tc('000824869','SURFACE TREATMENT',' COPPER OVERALL FIRST LAYER').

Obviously, the characteristic name, `SURFACE TREATMENT` provides information on the context in which COPPER is used in this row. In general, it is easier for the part attribute extractor to use information when the characteristic name provides useful information (as with `SURFACE TREATMENT`, than when it doesn't as with `GENERAL CHARACTERISTICS ITEM DESCRIPTION` or PID information.

Of course the extractor can and does use both types of information, but the first class of information may sometimes be more reliable than the second class. Accordingly, the reason field in coherent view information for textual extraction from technical characteristics indicates whether the data was extracted via a *structured search* rule or an *unstructured search* rule. Structured searches used the *name field* (second field of the `tc` relation)) in addition to the *value field* (third field). Unstructured searches do not use this information.

Keyword Searches As mentioned above, explanation of the use of DCGs is beyond the scope of this paper. However, in certain cases, the full power of DCGs is not needed; rather, it suffices to search for the occurrence of a *keyword* in a tokenized field that has been corrected for misspellings. As an example, often the name field of a technical characteristic row indicates useful information about a part so that its value field does not need to be examined. For instance the technical characteristic name `KEYWAY NOMINAL WIDTH` indicates that the part (or a component of the part) was turned on a lathe, which is a type of machining operation. Thus, after tokenization and correcting for misspellings, if the word `KEYWAY` occurs in a characteristic name, the part can be inferred to be machined. Similarly, if the token `WELDING` is found in the PID information, one can determine that welding is needed to manufacture the part.

It is important to note that while DCGs require an XSB/Prolog programmer to maintain, keyword searches can be maintained by a knowledge engineer — as long as the knowledge engineer has a good understanding of the overall process of attribute extraction and of the underlying data. For instance, the following are tables that infer lathing and general machining operations from name fields of technical characteristics information.

```
lathing_keyword('KEYWAY').
lathing_keyword('PIN').
lathing_keyword('SHAFT').

machining_keyword('COUNTERBORE').
machining_keyword('BORE').
machining_keyword('SHAFT').
machining_keyword('SHANK').
machining_keyword('THREAD').
machining_keyword('WAVEGUIDE').
```

Thus, for ease of maintenance, a keyword search should be used whenever possibly but it should be noted that since DCGs are more general than keyword searches, there are many situations in which DCGs must be used.

Current Routines for Extracting Part Attributes We now characterize the textual extraction routines, by their source, by whether structured or unstructured search is used, and by whether DCGs are needed or a keyword search is used. We note that textual extraction routines are under development and are subject to change.

- *Dimension Information* is currently extracted from technical characteristics only.

  - *Structured Search*: The structured search for dimensions uses a simple DCG to parse information from the fields defined in the XSB relation *dimension_char*. Currently, this includes fields such as `OVERALL LENGTH`, `OVERALL THICKNESS` etc.
  - *Unstructured Search*: The unstructured search for dimensions is based on a DCG.

- *Part Hierarchy (Gozinta) Information* is currently extracted from the technical characteristics only using an unstructured search based on DCGs.

- *Logistics Information* is extracted from the PID using an unstructured search based on DCGs. Information extracted concerns whether government mylar is required, whether technical data is available or not, and whether first article testing is required.

- *Primary Material Information*

  - *PID*. A DCG is used that is parameterized using the material hierarchy in the knowledge base.
  - *Technical Characteristics*. A structured DCG search is used which excludes name fields that designate surface treatment. This search identifies alloys if such information is present, and infers the shape from the material and alloy if possible.

- *Platform information* is extracted from technical characteristics information using an unstructured search. This unstructured search is based on DCGs.

- *Process Information*

  - *PID* A keyword search is made using the process hierarchy, as well as the tables `machining_keyword` and `lathing_keyword`.
  - *Technical Characteristics*
    * *Structured Search* A keyword search is made of materials in surface treatment fields based in the material hierarchy.
    * *Unstructured Search* A keyword search is made using the process hierarchy.

- *Unintegrated and Other Information*

  - *Technical characteristics, structured search* A keyword search is made of name fields to determine whether the part is electrical. The keywords are drawn from the relation `electrical_keyword`.
  - *PID* DCG searches are made to determine the commercial part number, drawing number, and cage code of qualified manufacturers. This information is fed into a relation of NIINs, Cages, and Part numbers and used by further inferencing operations that extract part information from commercial part and drawing numbers. In addition, a search is made to determine whether an NSN is an assembly.

5 Extracting Information from Commercial Part Numbers

NSNs usually are associated with part numbers assigned by their commercial manufacturers. These commercial part numbers are available to the part extractor through two sources. The first is the *ncp* relation, which as mentioned above maintains information about the NIIN, a manufacturing Cage and a Commercial Part Number for that Cage. This relation may have been gathered from one of DLA's databases, such as the procurement database, or textually extracted from PID information. The *unintegrated* relation also contains commercial part numbers extracted from the PID in the cases where Cage extraction was not possible.

Often, these part numbers have a meaning and signify the platform for which a part is designed, the subsystem of which the part forms a constituent, and so on. If it is known how a commercial part number encodes this information, a simple parsing routine can be written to extract the informatio [3]. As an example the Grummman part number 123CSH40013-611 indicates that the platform is a C-2A or E-2A, that the system is a control surface, and the manufacturing process uses honeycombing. Currently, there are four major encodings recognized by the CASP enabling technology: the drawing tree for Grumman E2-A, E6-A, and C2-A platforms, for the Grumman F-14, for Sikorsky platforms, and for Bell platforms. We present the format of each, and then consider how information from disparate manufacturers can be integrated.

5.1 Commercial Part Number Formats

Format of an E2-A, C2-A or E6-A part For the E2-A, C2-A and E6-A platforms, the commercial part number indicates not only the platform, but the aircraft system and the primary manufacturing process. The first 3 characters indicate the platform:

Characters	Platform
123	= E2-A or C2-A
128 or 1128	= E6-A

The next 1-2 characters indicate the system, and in certain cases may further specify the platform (e.g. CV is not used on E2-A).

[3] This encoding is sometimes called a *drawing-tree logic.*

Characters	System
W	WING SYSTEM
CS	CONTROL SURFACE
B	FUSELAGE
L	ALIGHTING GEAR
CV	CARRIER SUITABLE ALIGHTING GEAR
P	POWER PLANT
C	FLIGHT CONTROLS
AM	AUXILIARY MECHANISMS
AV	ELECTRONICS
AB	CREW AND EQUIPMENT INTEGRATION
EC	ENVIRONMENTAL CONTROLS
H	HYDRAULIC SYSTEM
EL	EQUIPMENT LINES

The next character indicates the process

Character	Process
M	MACHINING
F	FORGING
C	CASTING
P	PLASTIC
H	HONEYCOMB

Example 5.1

- Commercial Part: 123CSH40013-611 has
  - Platform = C-2A or E-2A
  - System = CONTROL SURFACE
  - Process = honeycomb
- Commercial Part: 128HM10076-511 has
  - Platform = E-6A
  - System = HYDRAULIC SYSTEM
  - Process = MACHINING

Format of an F-14 part For the F-14, the commercial part number indicates the platform, the version and the aircraft system for which the part is designed. The first 2 characters (A5) indicate that the platform is an F-14, while the next character (1,2,..) indicates the version. The next character indicates the aircraft system for which the part is used.

Character	System
A	AVIONICS
B	STRUCTURAL DESIGN
C	CONTROLS
D	SEATS AND SURVIVAL
E	ENVIRONMENTAL CONTROLS
F	FURNISHINGS
G	LANDING GEAR
H	FLUID POWER
J	PROTOTYPE ELECTRONICS
K	ARMAMENT SYSTEM
L	EQUIPMENT LINES
M	AUXILIARY MECHANISMS
N	INSTRUMENTATION
P	POWER PLANT

Example 5.2

- Commercial Part: A51B10506-15 has
  - Platform = F-14
  - Version = 1
  - System = STRUCTURAL DESIGN

Format of Sikorsky Drawing Numbers Many Sikorsky drawings have a common format that includes the model, the aircraft system, the aircraft location code, along with miscellaneous but useful information in the dash number. The first two characters indicate the model:

Characters	Model
14	UH55
15	UH56
58	UH58
60	UH60
61	UH61
62	UH62
64	UH64
65	UH65
67	UH67
69	UH69
70	UH70
72	UH72
75	UH75
76	UH76

The next 3 characters taken as a number, Num, indicate the system for which the part is used:

System	Range
ROTORS	if Num $\geq$ 100 and Num $<$ 150
BLADES	if Num $\geq$ 150 and Num $<$ 200
AIRFRAME	if Num $\geq$ 200 and Num $<$ 250
ALIGHTING GEAR	if Num $\geq$ 250 and Num $<$ 300
POWER PLANT	if Num $\geq$ 300 and Num $<$ 350
TRANSMISSION	if Num $\geq$ 350 and Num $<$ 400
MECHANICAL FLIGHT CONTROLS	if Num $\geq$ 400 and Num $<$ 450
CREW STATION CONTROL INSTRMNTS	if Num $\geq$ 450 and Num $<$ 500
FURNISHINGS	if Num $\geq$ 500 and Num $<$ 550
ELECTRICAL	if Num $\geq$ 550 and Num $<$ 600
AVIONICS	if Num $\geq$ 600 and Num $<$ 650
HYDRAULIC SYSTEM	if Num $\geq$ 650 and Num $<$ 700
SUPPORT/TEST/GROUND HANDLING EQPMT	if Num $\geq$ 700 and Num $<$ 725
TRAINERS	if Num $\geq$ 725 and Num $<$ 750
ARMAMENT SYSTEM	if Num $\geq$ 750 and Num $<$ 800
WING/NACELLE/SPONSON	if Num $\geq$ 800 and Num $<$ 850
CARGO HANDLING SYSTEM	if Num $\geq$ 850 and Num $<$ 900
ELECTRONIC FLIGHT CONTROLS	if Num $\geq$ 900 and Num $<$ 950

The next 2 characters indicate the aircraft location.

Characters	Aircraft Location Code
01	NOSE SECTION
02	MID SECTION
03	AFT SECTION
04	MAIN ROTOR PYLON
05	TAILCONE
06	TAIL ROTOR PYLON
07	HORIZONTAL STABILIZER
08	MAIN TRANS/ROTOR HOOD
09	MAIN ROTOR BLADES
10	POWER PACKAGE
11	TAIL ROTOR
12	MAIN LANDING GEAR
13	TAIL LANDING GEAR

The numbers, Num, after a dash, also can indicate interesting information about a part

Dash Information	Range
'CASTING' or 'EXTRUSION' or 'FORGING'	if Num $\geq$ 1 and Num $<$ 10
'INSTALLATION'	if Num $\geq$ 10 and Num $<$ 40
'ASSY'	if Num $\geq$ 40 and Num $<$ 100
'DETAIL PART'	if Num $\geq$ 100 and Num $<$ 400
'ASSY'	if Num $\geq$ 400 and Num $<$ 450
'MINOR DEVIATION'	if Num $\geq$ 450 and Num $<$ 490

Example 5.3 • The Sikorsky commercial part: '70219-04209-001' has

  - Model = UH-70
  - System = AIRFRAME
  - Aircraft Location Code = MAIN ROTOR PYLON
  - Dash = CASTING or EXTRUSION or FORGING

- The Sikorsky commercial part: '65300-12161-101' has
  - Platform = UH-65
  - System = POWER PLANT
  - Aircraft Location Code = MAIN LANDING GEAR
  - Dash = DETAIL

Format of Bell Part Numbers Bell part numbers provide the model and aircraft system along with information on whether the part is a detail part. The first three characters indicate the model:

Characters	Model
204	'BELL 204'
205	'BELL 205'
206	'BELL 206'
209	'BELL 209'
212	'BELL 212'
406	'BELL 406'
407	'BELL 407'
412	'BELL 412'

The next three numeric characters indicate the system to which the part belongs:

Characters	System
'MAIN ROTOR'	if Num $\geq$ 10 and Num $\leq$ 12
'POWER RETRACTION'	if Num = 13
'BLADES'	if Num $\geq$ 15 and Num $\leq$ 16
'WING'	if Num $\geq$ 20 and Num $\leq$ 23
'FUSILGE'	if Num $\geq$ 30 and Num $\leq$ 34
'TRANSMISSION'	if Num = 40
'ALIGHTING GEAR'	if Num $\geq$ 50 and Num $\leq$ 53
'POWER PLANT'	if Num $\geq$ 60 and Num $\leq$ 64
'FIXED EQUIPMENT'	if Num $\geq$ 70 and Num $\leq$ 71
'ARMAMENT SYSTEM'	if Num = 72
'ENVIRONMENTAL CONTROLS'	if Num = 73
'ELECTRONICS'	if Num = 74
'ELECTRICAL'	if Num $\geq$ 75 and Num $\leq$ 76
'HYDRAULIC'	if Num = 76
'AVIONICS'	if Num = 77

Finally the last three numeric characters indicate whether the part is a detail part, according to the following rule:

The part is a detail part if ($Num \geq 020$ and $Num \leq 023$) or ($Num \geq 030$ and $Num \leq 034$) or ($Num = 040$) or ($Num \geq 060$ and $Num \leq 064$) or ($Num = 070$) or ($Num = 072$) or ($Num = 076$).

Example 5.4

- The Bell commercial part: '205-072-420' has
  - Model = 205
  - System = FIXED EQUIPMENT
  - Part Type = DETAIL
- The Bell commercial part: '212-010-710' has
  - Model = 212
  - System = MAIN ROTOR

5.2 Integrating Commercial Part Information from Disparate Manufacturers

As discussed in Section 5.1, commercial part numbers contain different information: for instance Sikorsky parts contain information on the aircraft location code, while Grumman E2-A pars contain information on the manufacturing process. Even for common information, such as the aircraft system, the information extracted may vary depending on whether the platform is a helicoptor or fixed-wing aircraft, or even on the manufacturer.

Within the coherent database, information extracted from commercial part numbers is added to process and platform tables, and kept in a new table:

dtl(Nsn,Commercial_Part_Number,System,Aircraft_location_code,Dash)

In the above table, the dash field maintains information about whether the part is an assembly or a detail part. Platform information extracted from the commercial part number is added to the platform table, while process information from Grumman E2-A, C2-A, and E6-A and from Sikorsky models is added to the process table. Note, however, that various fields in the *dtl* relation may always be null for certain platforms — for instance non-Sikorsky platforms will always have a null $Aircraft\_location\_code$.

There remains an issue of how to integrate information about the aircraft system of different manufacturers. In order to address this, a subsystem hierarchy is defined for aircraft, for fixed-wing aircraft, and for helicopters. In addition, a synonym table is maintained to convert subsystem names from those defined in the commercial part number formats into system names defined within the hierachy. It is these canonical names that are added to the coherent database.

6 Inferencing and Conflict Removal among Extracted Data

Once information has been extracted from the underlying databases, missing data can be inferred in certain cases, and conflicting data can often be resolved.

6.1 Inferring Missing Data

In certain cases information about, say a process of a part may have been extracted, while information about the primary material of a part may be missing. However, in certain cases the possibilities for the primary material are *constrained* by the process. For instance, if it is known that a process for a part is `PASSIVATING`, then the part itself must be steel — although no information is known about whether the steel is corrosion-resistant, low-alloy, or any other kind. Currently several classes of such constraints are used for inferencing about parts.

- Constraining the primary material via a process. An example of this is provided in the previous paragraph. Such constraints are specified via the relation `process_material_constraint`.
- Constraining the primary material shape via a specification. An instance of this can be seen from the Boeing process specification:

  BAC-5652: Inspection of Castings.

  This process specification says nothing about the material used in the casting — only about the shape. Such constraints are specified via the relation `spec_shape_constraint`.
- Constraining the primary material via a specification. An instance of this can be seen from the Boeing process specification:

  BAC-5925: Spot Welding, Roller Spot Welding and Seam Welding of Titanium Alloys

  This process specification primarily concerns welding, but the fact that titanium alloys are used is also of importance. Such constraints are specified via the relation `spec_material_constraint`.

6.2 Identifying Conflicts in Data

To summarize, the following relations are produced by the previous phases of standardization.

- *material(Niin,Material,Shape,Source,Reason)* produced by reasoning about specifications, textual information, and discovered data.
- *process(Niin,Process,Process_material,Source,Reason)* produced by reasoning about specifications, textual information, and commercial part numbers.
- *unintegrated(Niin,Value,Source,Reason)* produced by reasoning about specifications and textual information.
- *unintegrated(Niin,Value,Source,Reason)* produced by reasoning about specifications and textual information.
- *logistics(Niin,Value,Source,Reason)* produced by reasoning about textual information.
- *dtl(Niin,Commercial_part_number,System,Aircraft_location_code,Dash)* produced by reasoning about textual information.

- *platform(Niin,Dimension_type,Dimension_value,Tolerance,Source,Reason)* produced by reasoning about textual information.
- *platform(Niin,Platform,Version,Source,Reason)* produced by reasoning about textual information and commercial part numbers.
- *platform(Niin,Platform,Version,Source,Reason)* produced by reasoning about textual information and commercial part numbers.
- *gozinta(Subassembly,Assembly,Source,Reason)* produced by reasoning about textual information.

While this data has been standardized and inferred it is not necessarily free from conflicts. For instance, it may be the case that a textual-based reasoning produces the record

material('000088975','STEEL',null,tc,ss('GENERAL CHARACTERISTICS ITEM DESCRIPTION')).

while specification-based reasoning produces

material('000088975','CRES'(['CHROMIUM'(12.5000)]),['BAR','FORGING','TUBE','WIRE'],tc,'AMS 5612').

This information is not necessarily conflicting: it is much more likely that the specifications provide much more specific information than a textual search. Thus, while the first record may be important for supporting information, it is not necessary for a high-level description of the view.

Alternatively, consider a case in which the technical characteristics considers a part to be made of magnesium while the PID considers a part to be aluminum. This may be evidence of conflicting data due to re-engineering, (as has been seen for the Boeing KC-135) or may be due to an error in part attribute discovery. In either case the information should be flagged as conflicting and the conflict resolved, either manually or through some automated means.

Version 1.5.1 of the attribute extractor uses a form of Annotation Logic [3] both to determine the most specific information obtained from within a particular source, and to identify conflicting information between sources. This form of Annotation logic is called Amalgamation Logic [6], and is implemented using the tabling mechanisms of XSB (see [7] for details). The result of applying amalgamation logic rules to inferred data is called the coherent database. Data in the coherent database is maintained in the same form as inferred data, but the source field is tagged as `coherent`. Because the coherent database is created by applying amalgamation rules to inferred data, there could be alternate definitions of a coherent database. For instance, rather than tagging contradictory information, information from one database could be preferred over another (e.g. PID information preferred over technical characteristics, when there is a conflict), or specialized conflict handling rules could be constructed.

7 Discussion

Although the discovery processes outlined in this paper have been developed for on-demand manufacturing, the techniques are generally applicable whenever reasoning must be done about DLA

data. For instance, a query was recently performed for the casting consortium to determine all Richmond parts that were likely to be made from cast metal. Indeed, the attributes extracted: such as materials and manufacturing processes are applicable to most parts. The attribute extractor has been designed so that if it needed to be extended to work on a domain radically different from airplane parts, much of the work to do so would consist of designing a new knowledge base. This task can currently be done by an analyst who has a minimal knowledge of Prolog. In addition, tools can be constructed to allow analysts to maintain the knowledge base through a graphical front-end so that even this minimal knowledge would not be required. How to maintain more complicated rules in the system, such as definite clause grammars and amalgamation logic rules without using an XSB programmer remains an open question.

References

[1] B. Cui, T. Swift, and D. S. Warren. Using tabled logic programs and preference logic for data standardization. Available at http://www.cs.sunysb.edu/~tswift, 1998.

[2] J. Freire, P. Rao, K. Sagonas, T. Swift, and D. S. Warren. XSB: A system for efficiently computing the well-founded semantics. In *LPNMR*, pages 430–440. MIT Press, 1997.

[3] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Logic Programming*, 12(4):335–368, 1992.

[4] I. V. Ramakrishnan, A. Roychoudhury, and T. Swift. A standardization tool for data warehousing. In *Practical Applications of Prolog*, 1997.

[5] K. Sagonas, T. Swift, and D. S. Warren. XSB as an efficient deductive database engine. In *ACM SIGMOD Conference*, 1994.

[6] V. S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database Systems*, 19(2):291–331, 1994.

[7] T. Swift. Tabling for non-monotonic programming. *Annals of Mathematics and Artificial Intelligence*. To appear.

[8] T. Swift, C. Henderson, R. Holberger, J. Murphey, and E. Neham. CCTIS: an expert transaction processing system. In *Sixth Conference on Industrial Applications of Artificial Intelligence*, pages 131–140, 1994.