Validity of First-Order Knowledge Bases

John Debenham

Faculty of Information Technology
University of Technology, Sydney
PO Box 123 Broadway, NSW 2007, Australia
debenham@it.uts.edu.au

Abstract

A knowledge base is maintained by modifying its conceptual model and by using those modifications to specify changes to its implementation. The maintenance problem is to determine which parts of that model should be checked for correctness in response a change in the application. The maintenance problem is not computable for first-order knowledge bases. Two things in the conceptual model are joined by a maintenance link if a modification to one of them means that the other must be checked for correctness, and so possibly modified, if consistency of the model is to be preserved. In a unified conceptual model for first-order knowledge bases the data and knowledge are modelled formally in a uniform way. A characterisation is given of four different kinds of maintenance links in a unified conceptual model. Two of these four kinds of maintenance links can be removed by transforming the conceptual model. In this way the maintenance problem is simplified.

Introduction

The conceptual model of a knowledge base specifies what should be in an implementation of that knowledge base, but not what the implementation will be required to do. So the conceptual model may be used to drive the maintenance process (Debenham, 1997). The maintenance problem is to determine which parts of that model should be checked for correctness in response a change in the application. The maintenance problem is not computable for first-order knowledge bases. Maintenance links join two things in the conceptual model if a modification to one of them means that the other must be checked for correctness, and so possibly modified, if consistency of that model is to be preserved. If that other thing requires modification then the links from it to yet other things must be followed, and so on until things are reached that do not require modification. If node A is linked to node B which is linked to node C then nodes A and C are *indirectly* linked. In a *coherent* knowledge base everything is indirectly linked to everything else. A good conceptual model for maintenance will have a low density of maintenance links (Mayol and Teniente, 1999). The set of maintenance links should be *minimal* in than none may be removed.

Informally, one conceptual model is "better" than another if it leads to less checking for correctness. The aim of this work is to generate a good conceptual model. A classification into four classes is given here of the maintenance links for conceptual models expressed in the unified (Debenham, 1998) knowledge representation. Methods are given for removing two of these classes of link so reducing the density of maintenance links.

Approaches to the maintenance of knowledge bases are principally of two types (Katsuno and Mendelzon, 1991). First, approaches that take the knowledge base as presented and then try to *control* the maintenance process (Barr, 1999). Second, approaches that *engineer* a model of the knowledge base so that it is in a form that is inherently easy to maintain (Jantke and Herrmann, 1999) (Darwiche, 1999). The approach described here is of the second type because maintenance is driven by a maintenance link structure that is simplified by transforming the conceptual model.

The majority of conceptual models treat the "rule base" component separately from the "database" component. This enables well established design methodologies to be employed, but the use of two separate models means that the interrelationship between the things in these two models cannot be represented, integrated and manipulated naturally within the model (Debenham, 1998). Neither of these two separate models is able to address completely the validity of the whole knowledge base.

The terms data, information and knowledge are used here in the following sense. The *data* things in an application are the fundamental, indivisible things. Data things can be represented as simple constants or variables. If an association between things *cannot* be defined as a succinct, computable rule then it is an *implicit* association. Otherwise it is an *explicit* association. An *information* thing in an application is an implicit association between data things. Information things can be represented as tuples or relations. A *knowledge* thing in an application is an explicit association between information and/or data things. Knowledge can be represented either as programs in an imperative language or as rules in a declarative language.

Conceptual Model

Items are a formalism for describing all data, information and knowledge things in an application (Debenham, 1998). Items incorporate two powerful classes of constraints, and a single rule of decomposition is specified for items. The key to this unified representation is the way in which the "meaning" of an item, called its *semantics*, is specified. The semantics of an item is a function that *recognises* the members of the "value set" of that item. The value set of an item will change in time τ, but the item's semantics should remain constant. The value set of a data item at a certain time τ is the set of labels that are associated with a population that implements that item at that time. The value set of an information item at a certain time τ is the set of tuples that are associated with a relational implementation of that item at that time. Knowledge items have value sets too. Consider the rule "the sale price of parts is the cost price marked up by a universal mark-up factor"; this rule is represented by the item named *[part/sale-price, part/cost-price, mark-up]* with a value set of corresponding quintuples. The idea of defining the semantics of items as recognising functions for the members of their value set extends to complex, recursive knowledge items too.

An *item* is a named triple $A[S_A, V_A, C_A]$ with *item name* A, S_A is called the *item semantics* of A, V_A is called the *item value constraints* of A and C_A is called the *item set constraints* of A. The item semantics, S_A, is a λ-calculus expression that recognises the members of the value set of item A. The expression for an item's semantics may contain the semantics of other items $\{A_1,..., A_n\}$ called that item's *components*:

$$\lambda\, y^1_1...y^1_{m_1}...y^n_{m_n}\bullet[\, S_{A_1}(y^1_1,...,y^1_{m_1}) \wedge ..... \wedge S_{A_n}(y^n_1,...,y^n_{m_n}) \wedge J(y^1_1,..,y^1_{m_1},..,y^n_{m_n})\,]\bullet$$

The item value constraints, V_A, is a λ-calculus expression:

$$\lambda\, y^1_1...y^1_{m_1}...y^n_{m_n}\bullet[\, V_{A_1}(y^1_1,...,y^1_{m_1}) \wedge ..... \wedge V_{A_n}(y^n_1,...,y^n_{m_n}) \wedge K(y^1_1,..,y^1_{m_1},..,y^n_{m_n})\,]\bullet$$

that should be satisfied by the members of the value set of item A as they change in time; so if a tuple satisfies S_A then it should satisfy V_A (Johnson and Santos, 2000). The expression for an item's value constraints contains the value constraints of that item's *components*. The item set constraints, C_A, is an expression of the form:

$$C_{A_1} \wedge C_{A_2} \wedge ... \wedge C_{A_n} \wedge (L)_A$$

where L is a logical combination of:

- Card lies in some numerical range;
- Uni(A_i) for some i, $1 \le i \le n$, and
- Can(A_i, X) for some i, $1 \le i \le n$, where X is a non-empty subset of $\{A_1,..., A_n\} - \{A_i\}$;

subscripted with the name of the item A, "Uni(a)" means that "all members of the value set of item a must be in this association". "Can(b, A)" means that "the value set of the set of items A functionally determines the value set of item b". "Card" means "the number of things in the value set". The subscripts indicate the item's components to which that set constraint applies.

For example, each *part* may be associated with a *cost-price* subject to the "value constraint" that parts whose part-number is less that 1,999 should be associated with a cost price of no more than $300. A set constraint specifies that every part must be in this association, and that each part is associated with a unique cost-price. The information item named *part/cost-price* then is:

$$\begin{aligned} &\textit{part/cost-price}[\, \lambda xy\bullet[\, S_{part}(x) \wedge S_{cost\text{-}price}(y) \wedge \text{costs}(x, y)\,]\bullet, \\ &\quad \lambda xy\bullet[\, V_{part}(x) \wedge V_{cost\text{-}price}(y) \wedge ((x < 1999) \rightarrow (y \le 300))\,]\bullet, \\ &\quad C_{part} \wedge C_{cost\text{-}price} \wedge \\ &\quad (\text{Uni}(\textit{part}) \wedge \text{Can}(\textit{cost-price}, \{\textit{part}\})_{\textit{part/cost-price}}\,] \end{aligned}$$

Rules, or knowledge, can also be defined as items, although it is neater to define knowledge items using "objects". "Objects" are item building operators. The knowledge item *[part/sale-price, part/cost-price, mark-up]* which means "the sale price of parts is the cost price marked up by a uniform markup factor" is:

$$\begin{aligned} &\textit{[part/sale-price, part/cost-price, mark-up]}[\\ &\quad \lambda x_1x_2y_1y_2z\bullet[(\\ &\qquad S_{\textit{part/sale-price}}(x_1, x_2) \wedge S_{\textit{part/cost-price}}(y_1, y_2) \wedge \\ &\qquad S_{\textit{mark-up}}(z)\,) \wedge ((x_1 = y_1) \rightarrow (x_2 = z \times y_2))]\bullet, \\ &\quad \lambda x_1x_2y_1y_2z\bullet[\\ &\qquad V_{\textit{part/sale-price}}(x_1, x_2) \wedge V_{\textit{part/cost-price}}(y_1, y_2) \wedge \\ &\qquad V_{\textit{mark-up}}(z)\,) \wedge ((x_1 = y_1) \rightarrow (x_2 > y_2))]\bullet, \\ &\quad C_{\textit{[part/sale-price, part/cost-price, mark-up]}}\,] \end{aligned}$$

Two different items can share common knowledge and so can lead to a profusion of maintenance links. This problem can be avoided by using objects. An n-adic *object* is an operator that maps n given items into another item for some value of n. Further, the definition of each object will presume that the set of items to which that object may be applied are of a specific "type". The *type* of an m-adic item is determined both by whether it is a data item, an information item or a knowledge item and by the value of m. The type is denoted respectively by $\mathbf{D}^m$, $\mathbf{I}^m$ and $\mathbf{K}^m$. Items may also have unspecified, or free, type which is

denoted by $\mathbf{X}^m$. The formal definition of an object is similar to that of an item. An *object* named A is a typed triple A[E,F,G] where E is a typed expression called the *semantics* of A, F is a typed expression called the *value constraints* of A and G is a typed expression called the *set constraints* of A. For example, the *part/cost-price* item can be built from the items *part* and *cost-price* using the ***costs*** operator:

$$\begin{aligned}
&\textit{part/cost-price} = \textbf{\textit{costs}}(\textit{part}, \textit{cost-price})\\
&\textbf{\textit{costs}}[\lambda P{:}\mathbf{X}^1 Q{:}\mathbf{X}^1 \bullet \lambda xy \bullet [\, S_P(x) \wedge S_Q(y) \wedge \text{costs}(x,y)\,]\bullet\bullet,\\
&\quad \lambda P{:}\mathbf{X}^1 Q{:}\mathbf{X}^1 \bullet \lambda xy \bullet [V_P(x) \wedge V_Q(y) \wedge\\
&\qquad ((1000 < x < 1999) \rightarrow (y \le 300))\,]\bullet\bullet,\\
&\quad \lambda P{:}\mathbf{X}^1 Q{:}\mathbf{X}^1 \bullet [\, C_P \wedge C_Q \wedge\\
&\qquad (\text{Uni}(P) \wedge \text{Can}(Q, \{P\}))_{\nu(\textbf{\textit{costs}},P,Q)}\,]\bullet]
\end{aligned}$$

where ν(***costs***, P, Q) is the name of the item ***costs***(P, Q).

Data objects provide a representation of sub-typing. Rules are quite clumsy when represented as items; objects provide a far more compact representation. For example, consider the *[part/sale-price, part/cost-price, mark-up]* knowledge item which represents the rule "parts are marked-up by a universal mark-up factor". This item can be built by applying a knowledge object ***mark-up-rule*** of argument type ($\mathbf{I}^2$, $\mathbf{I}^2$, $\mathbf{D}^1$) to the items *part/sale-price*, *part/cost-price* and *mark-up*. That is:

[part/sale-price, part/cost-price, mark-up] =
mark-up-rule(*part/sale-price, part/cost-price, mark-up*)

Objects also represent value constraints and set constraints in a uniform way. A decomposition operation for objects is defined in (Debenham, 1999).

A *conceptual model* consists of a set of items and a set of maintenance links. The items are constructed by applying a set of object operators to a set of fundamental items called the *basis*. The *maintenance links* join two items if modification to one of them necessarily means that the other item has at least to be checked for correctness if consistency is to be preserved. Item join provides the basis for item decomposition (Debenham, 1997). Given items A and B, the item with name $A \otimes_E B$ is called the *join* of A and B on E, where E is a set of components common to both A and B. Using the rule of composition $\otimes$, knowledge items, information items and data items may be joined with one another regardless of type. For example, the knowledge item:

$$\begin{aligned}
&[\textit{cost-price, tax}]\,[\lambda xy \bullet [S_{\textit{cost-price}}(x) \wedge\\
&\qquad S_{\textit{tax}}(y) \wedge x = y \times 0.05]\bullet,\\
&\quad \lambda xy \bullet [V_{\textit{cost-price}}(x) \wedge V_{\textit{tax}}(y) \wedge x < y]\bullet,\\
&\quad C_{[\textit{cost-price, tax}]}\,]
\end{aligned}$$

can be joined with the information item *part/cost-price* on the set {*cost-price*} to give the information item *part/cost-price/tax*. In other words:

$$\begin{aligned}
&[\textit{cost-price, tax}] \otimes_{\{\textit{cost-price}\}} \textit{part/cost-price} =\\
&\textit{part/cost-price/tax}[\,\lambda xyz \bullet [\, S_{\textit{part}}(x) \wedge S_{\textit{cost-price}}(x) \wedge\\
&\qquad S_{\textit{tax}}(y) \wedge \text{costs}(x,y) \wedge z = y \times 0.05\,]\bullet,\\
&\quad \lambda xyz \bullet [\, V_{\textit{part}}(x) \wedge V_{\textit{cost-price}}(x) \wedge V_{\textit{tax}}(y) \wedge\\
&\qquad ((1000<x<1999) \rightarrow (0<y\le 300)) \wedge (z<y)\,]\bullet.\\
&\quad C_{\textit{part/cost-price/tax}}\,]
\end{aligned}$$

In this way items may be joined together to form more complex items. The $\otimes$ operator also forms the basis of a theory of decomposition in which each item is replaced by a set of simpler items. An item I is *decomposable* into the set of items $D = \{I_1, I_2, ..., I_n\}$ if: I_i has non-trivial semantics for all i, $I = I_1 \otimes I_2 \otimes ... \otimes I_n$, where each join is *monotonic*; that is, each term in this composition contributes at least one component to I. If item I is decomposable then it will not necessarily have a unique decomposition. The $\otimes$ operator is applied to objects in a similar way (Debenham (1999). The rule of decomposition is: "Given a conceptual model discard any items and objects which are decomposable". For example, this rule requires that the item *part/cost-price/tax* should be discarded in favour of the two items *[cost-price, tax]* and *part/cost-price*.

Maintenance Links

A *maintenance link* joins two items in the conceptual model if modification of one item means that the other item must be checked for correctness, and maybe modified, if the consistency of the conceptual model is to be preserved (Ramirez and de Antonio, 2000). The number of maintenance links can be very large. So maintenance links can only form the basis of a practical approach to knowledge base maintenance if there is some way of reducing their density on the conceptual model.

For example, given two items A and B, where both are n-adic items with semantics S_A and S_B respectively, if π is permutation such that:

$$(\forall x_1 x_2 ... x_n)[\, S_A(x_1, x_2, ..., x_n) \leftarrow S_B(\pi(x_1, x_2, ..., x_n))\,]$$

then item B is a *sub-item* of item A. These two items should be joined with a maintenance link. If A and B are both data items then B is a *sub-type* of A. Suppose that:

$$X = E\,D; \quad \text{where } D = C\,A\,B \qquad (1)$$

for items X, D, A and B and objects E and C. Item X is a sub-item of item D. Object E has the effect of extracting a

sub-set of the value set of item D to form the value set of item X. Item D is formed from items A and B using object C. Introduce two new objects F and J. Suppose that object F when applied to item A extracts the same subset of item A's value set as E extracted from the "left-side" (ie. the "A-side") of D. Likewise J extracts the same subset of B's value set as E extracted from D. Then:

$$X = C\,G\,K; \text{ where } G = F\,A \text{ and } K = J\,B \qquad (2)$$

so G is a sub-item of A, and K is a sub-item of B. The form (2) differs from (1) in that the sub-item maintenance links have been moved one layer closer to the data item layer, and object C has moved one layer away from the data item layer. This is illustrated in Fig. 1. Using this method repeatedly sub-item maintenance links between non-data items are reduced to sub-type links between data items.

It is shown now that there are four kinds of maintenance link in a conceptual model built using the unified knowledge representation. Consider two items A and B, and suppose that their semantics S_A and S_B have the form:

$$S_A = \lambda y_1^1 ... y_{m_1}^1 ... y_{m_p}^p \bullet [S_{A_1}(y_1^1,...,y_{m_1}^1) \wedge \ ..... \ \wedge$$
$$S_{A_p}(y_1^p,...,y_{m_p}^p) \wedge J(y_1^1,..,y_{m_1}^1,..,y_{m_p}^p)] \bullet$$
$$S_B = \lambda y_1^1 ... y_{n_1}^1 ... y_{n_q}^q \bullet [S_{B_1}(y_1^1,...,y_{n_1}^1) \wedge \ ..... \ \wedge$$
$$S_{B_q}(y_1^q,...,y_{n_q}^q) \wedge K(y_1^1,..,y_{n_1}^1,..,y_{n_q}^q)] \bullet$$

S_A contains $(p + 1)$ terms and S_B contains $(q + 1)$ terms. Let μ be a maximal sub-expression of $S_{A \otimes B}$ such that:

$$both \ S_A \rightarrow \mu \ and \ S_B \rightarrow \mu \qquad (a)$$

where μ has the form:

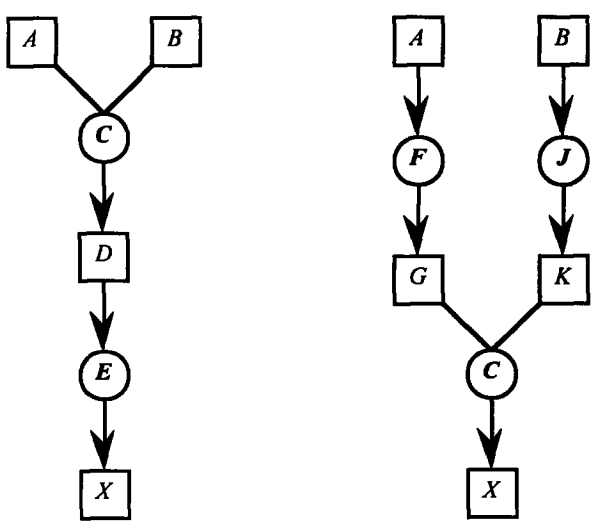

Figure 1. Reducing sub-item relationships

$$\lambda y_1^1 ... y_{d_1}^1 ... y_{d_r}^r \bullet [S_{C_1}(y_1^1,...,y_{d_1}^1) \wedge \ ..... \ \wedge S_{C_r}(y_1^r,...,y_{d_r}^r) \wedge$$
$$L(y_1^1,..,y_{d_1}^1,..,y_{d_r}^r)] \bullet$$

If μ is empty, ie. 'false', then the semantics of A and B are independent. If μ is non-empty then the semantics of A and B have something in common and A and B should be joined with a maintenance link.

Now examine μ to see *why* A and B should be joined. If μ is non-empty and if both A and B are items in the basis then:

A and B are a pair of basis items with logically dependent semantics (b)

If μ is non-empty and if A is *not* in the basis then there are three cases. First, if:

$$S_A \Leftrightarrow S_B \Leftrightarrow \mu \qquad (c)$$

then items A and B are equivalent and should be joined with an *equivalence link.* Second if (c) does not hold and:

$$either \ S_A \Leftrightarrow \mu \ or \ S_B \Leftrightarrow \mu \qquad (d)$$

then either A is a sub-item of B, or B is a sub-item of A and these two items should be joined with a *sub-item link.* Third, if (c) and (d) do not hold then if Δ is a minimal sub-expression of S_A such that $\Delta \rightarrow \mu$. Then:

$$either \ S_{A_j}(y_1^j,...,y_{m_j}^j) \in \Delta, \text{ for some } j \qquad (e)$$

$$or \ J(y_1^1,...,y_{m_j}^1,...,y_{m_p}^p) \in \Delta \qquad (f)$$

Both (e) and (f) may hold. If (e) holds then items A and B share one or more component items to which they should each be joined with a *component link.* If (f) holds then items A and B may be constructed with two object operators whose respective semantics are logically dependent. Suppose that item A was constructed by object operator C then the semantics of C will imply:

$$\Phi = \lambda Q_1{:}X_1^{i_1} Q_2{:}X_2^{i_2} ... Q_j{:}X_j^{i_j} \bullet \lambda y_1^1 ... y_{d_1}^1 ... y_{d_r}^r \bullet [$$
$$S_{P_1}(y_1^1,...,y_{d_1}^1) \wedge \ ..... \ S_{P_r}(y_1^r,...,y_{d_r}^r) \wedge$$
$$L(y_1^1,...,y_{d_1}^1,..,y_{d_r}^r)] \bullet$$

where the Q_i's take care of any possible duplication in the P_j's. Let E be the object $E[\Phi, T, \emptyset]$ then C is a sub-object of E; that is, there exists a non-tautological object F such that:

$$C \simeq_W E \otimes_M F \qquad (g)$$

for some set M and where the join is not necessarily monotonic. Items A and B are *weakly equivalent*, written $A \simeq_W B$, if there exists a permutation π such that:

$$(\forall x_1 x_2 ... x_n)[S_A(x_1,x_2,...,x_n) \leftrightarrow S_B(\pi(x_1,x_2,...,x_n))]$$

where the x_i are the n_i variables associated with the i'th component of A. If A is a sub-item of B and if B is a sub-item of A then items A and B are weakly equivalent.

If (g) holds then the maintenance links are of three different kinds. If the join in (g) *is* monotonic then (g) states that C may be decomposed into E and F. If the join in (g) is *not* monotonic then (g) states that either $C \simeq_W E$ or $C \simeq_W F$. So, if the join in (g) is not monotonic then *either* E will be weakly equivalent to C, *or* C will be a sub-object of E.

It has been shown above that sub-item links between non-data items may be reduced to sub-type links between data items. So if:

- the semantics of the items in the basis are all logically independent;
- all equivalent items and objects have been removed by re-naming, and
- sub-item links between non-data items have been reduced to sub-type links between data items

then the maintenance links will be between nodes marked with:

- a data item that is a sub-type of the data item marked on another node, these are called the *sub-type links*;
- an item and the nodes marked with that item's components, these are called the *component links*, and
- an item constructed by a decomposable object and nodes constructed with that object's decomposition, these are called the *duplicate links*.

If the objects employed to construct the conceptual model have been decomposed then the only maintenance links remaining will be the sub-type links and the component links. The sub-type links and the component links cannot be removed from the conceptual model.

Unfortunately, decomposable objects, and so too duplicate links, are hard to detect. Suppose that objects A and B are decomposable as follows:

$$A \simeq_W E \otimes_M F$$
$$B \simeq_W E \otimes_M G$$

Then objects A and B should both be linked to object E. If the decompositions of A and B have not been identified then object E may not have been identified and the implicit link between objects A and B may not be identified.

Conclusion

Maintenance links are used to maintain the validity of first-order knowledge bases. Maintenance links join two items in the conceptual model if modification of one of these items could require that the other item should be checked for correctness if the validity of the conceptual model is to be preserved. The efficiency of maintenance procedures depends on a method for reducing the density of the maintenance links in the conceptual model. One kind of maintenance link is removed by applying the rule of knowledge decomposition (Debenham, 1999). Another is removed by reducing sub-item relationships to sub-type relationships (Debenham, 1998). And another is removed by re-naming.

References

Barr, V. (1999). "Applying Reliability Engineering to Expert Systems" in *proceedings 12th International FLAIRS Conference*, Florida, May 1999, pp494-498.

Darwiche, A. (1999). "Compiling Knowledge into Decomposable Negation Normal Form" in *proceedings International Joint Conference on Artificial Intelligence, IJCAI'99*, Stockholm, Sweden, August 1999, pp 284-289.

Debenham, J.K. (1997). "From Conceptual Model to Internal Model", in *proceedings Tenth International Symposium on Methodologies for Intelligent Systems ISMIS'97*, Charlotte, October 1997, pp227-236.

Debenham, J.K. (1998). "*Knowledge Engineering*", Springer-Verlag, 1998.

Debenham, J.K. (1999). "Knowledge Object Decomposition" in *proceedings 12th International FLAIRS Conference*, Florida, May 1999, pp203-207.

Jantke, K.P. and Herrmann, J. (1999). "Lattices of Knowledge in Intelligent Systems Validation" in *proceedings 12th International FLAIRS Conference*, Florida, May 1999, pp499-505.

Johnson, G. and Santos, E. (2000). "Generalizing Knowledge Representation Rules for Acquiring and Validating Uncertain Knowledge" in *proceedings 13th International FLAIRS Conference*, Florida, May 2000, pp186-2191.

Katsuno, H. and Mendelzon, A.O. (1991). "On the Difference between Updating a Knowledge Base and Revising It", in *proceedings Second International Conference on Principles of Knowledge Representation and Reasoning, KR'91*, Morgan Kaufmann, 1991.

Mayol, E. and Teniente, E. (1999). "Addressing Efficiency Issues During the Process of Integrity Maintenance" *in proceedings Tenth International Conference DEXA99*, Florence, September 1999, pp270-281.

Ramirez, J. and de Antonio, A. (2000). "Semantic Verification of Rule-Based Systems with Arithmetic Constraints" in *proceedings 11th International Conference DEXA2000*, London, September 2000, pp437-446.