Xcerpt et visXcerpt : Langages déductifs d'interrogation du Web

Sacha Berger, François Bry, Sebastian Schaffert

Institut d'Informatique, Université de Munich, Allemagne

RÉSUMÉ. *Les langages d'interrogation du Web comme XPath, XQuery et XSLT sont devenus des outils communs de réalisation d'applications Web. Cet article motive puis présente deux nouveaux langages d'interrogation expérimentaux pour le Web : Xcerpt et visXcerpt (cf. http ://xcerpt.org). Ces deux langages reposent sur les mêmes principes. Un des aspects essentiel de ces langages est qu'ils sont déductifs : il s'appuient sur une notion d'unification semblable à, bien que différente de l'unification des systèmes de preuve automatique et de la programmation logique ainsi que sur une notion d'inférence semblable à celle de la programmation logique et aux vues du langage SQL d'interrogation de bases de données relationnelles. visXcerpt est un langage visuel qui réalise et présente d'une manière originale les constructions textuelles de Xcerpt. Xcerpt et visXcerpt visent à faciliter le développement d'applications du Web Sémantique. L'objectif de cet article est d'introduire aux aspects essentiels de Xcerpt et visXcerpt.*

ABSTRACT. *Web Query languages like XPath, XQuery and XSLT are widely accepted tools for the development of Web applications today. This article presents and motivates two new and experimental query languages for the Web: Xcerpt and visXcerpt (see http://xcerpt.org). Both languages are based on the same principles. They are deductive languages, which is one of their central aspects: they use a sort of unification similar to the unification used in logic programming and deduction systems, as well as an inference mechanism similar to that of logic programming and to SQL views in databases. visXcerpt is a visual query language which provides a visual realization of Xcerpt's textual constructs. Xcerpt's and visXcerpt's main goal is to ease the development of applications for the Semantic Web applications. This article introduces the essential aspects of Xcerpt and visXcerpt.*

MOTS-CLÉS : *XML, langage d'interrogation, programmation logique, programmation visuelle*

KEYWORDS: *XML, query language, logic programming, visual programming*

1. Introduction

L'émergence de XML [W3 00] comme langage d'échange et de modélisation des données, en particulier des données semi-structurées [ABI 00], a grandement contribué à la popularité des langage d'interrogation du Web, de données XML et de données semi-structurées. La plupart de ces langages, en particulier les standards de fait XPath [xpa99], XQuery [xqu01] et XSLT [xsl00], peuvent être qualifiés de *navigationels* en ce sens qu'ils demandent que le programmeur exprime une requête d'interrogation comme une navigation dans le "document" interrogé ("document" est le terme consacré pour un article XML ou HTML à structures d'arbre ou de graphe avec racine). D'autres langages, en particulier UnQL [BUN 00] et Xcerpt [BRY 02b, BRY 03, BRY 02a, BER 03], permettent d'exprimer des requêtes d'interrogation en terme de "gabarits" semblables aux atomes de la logique et de la programmation logique et aux requêtes visuelles de QBE [ZLO 77]. Une interrogation en terme de "gabarits" peut être qualifiée de *positionelle* en ce qu'elle permet d'assigner des positions aux variables dans le gabarit décrivant les données recherchées. Il est raisonnable de soutenir qu'une interrogation par "gabarits" permet une expression plus simple d'interrogations complexes qu'une approche navigationelle. Un des objectifs de cet article est de donner des exemples démontrant la justesse de ce point de vue.

XQuery et XSLT s'appuient sur XPath pour la recherche de données : les interrogations élémentaires de XQuery et de XSLT sont des interrogations XPath. Alors que XPath ne permet que de sélectionner des sous-graphes d'articles XML – dans la terminologie XML des "éléments" du "document" considéré – XQuery et XSLT permettent de recomposer plusieurs parties d'articles sélectionnés à l'aide de XPath en nouveaux document dont la structure peut s'écarter autant que souhaité de la structure du document initial. Cela rappelle la génération de nouveaux atomes en programmation logique à l'aide de clauses, ou de nouvelles relations à l'aide de "vues" SQL. XQuery et XSLT n'ont toutefois pas de mécanisme aussi généraux que les clauses de la programmation logique ou les vues des langages d'interrogation des bases de données relationnelle comme SQL. Ces mécanismes voisins l'un de l'autre peuvent être vues comme des notions de procédures permettant, entre autres choses, de structurer des programmes en sorte que des calculs complexes puissent être exprimés en termes de calculs plus simples.

Xcerpt (cf. http ://xcerpt.org) est un langage expérimental d'interrogation destiné à simplifier le développement d'applications tant du Web actuel que du Web Sémantique en cours de construction. Xcerpt peut ainsi aussi bien être utilisé pour interroger le Web tel que'il existe aujourd'hui, c'est à dire consistant essentiellement de pages HTML (en de diverses versions de ce formalisme), que des applications avancées utilisant des ontologies – formulées par exemple en OWL [owl00] – ou des annotations RDF [rdf00]. Xcerpt combine de manière nouvelle et originale les principes suivants :

– interrogation positionelle comme en logique et en programmation logique au lieu de l'interrogation par nommage d'attributs comme en SQL et de l'interrogation navigationelle utilisée par XPath, XSLT et XQuery

– spécification incomplète des données recherchées comme les interrogations navigationelles de XPath le permettent sans toutefois utiliser l'interrogation navigationelle de XPath

– notion originale, en particulier non symétrique, d'unification, appelée "unification de simulation" [BRY 02b], inspirée de l'unification de la déduction automatique et de la programmation logique et adapté aux besoins de l'interrogation du Web et du Web Sémantique

– mécanisme simple d'inférence semblable à celui de la programmation logique

– évaluation des requêtes d'interrogation par raisonnement par contraintes

Suivant en cela le standard XML, Xcerpt ne suppose aucune notion d'objet ou d'héritage. Il est vrai que de telles notions ont été proposées dans des formalismes du Web dont le but est de compléter XML, en particulier dans le formalisme XML Schema [xsd01] visant à la description de la structure de documents XML. Cependant, ces propositions ne sont pas normatives et ne prétendent pas l'être : XML Schema n'impose pas que les documents XML soient valides par rapport à un schema. Des documents XML ne se pliant pas aux notions d'héritage de XML Schema ou de formalismes semblables sont donc présents et resteront présents sur le Web actuel et futur. Xcerpt ne suppose aucune notion d'objet ou d'héritage afin de pouvoir être utilisé pour interroger de tels documents. On notera que de telles notions peuvent aisément être programmées en Xcerpt bien qu'elles ne soient pas présentes dans le langage à l'état natif.

Les aspects déclaratifs de Xcerpt mentionnés plus haut facilitent la réalisation d'un langage visuel reposant sur les mêmes notions. Un tel langage, visXcerpt [BER 03], a été conçu et implémenté.

L'objectif de cet article (dont une version anglaise a été publiée [BER 03]) est d'introduire aux aspects essentiels de Xcerpt et de présenter le langage visuel visXcerpt qui réalise de manière différente ces mêmes aspects. Cet article consiste en 4 parties, dont la première est cette introduction. La seconde partie introduit les principaux éléments du langage textuel Xcerpt et donne des exemples de programmes Xcerpt. La troisième partie décrit la visualisation en visXcerpt des principaux éléments de Xcerpt. La quatrième partie donne des conclusions et présente des perspectives pour de futurs développements de Xcerpt et de visXcerpt.

2. Principaux Éléments du Langage Textuel Xcerpt

2.1. *Représentation textuelle de Documents XML*

Xcerpt offre un formalisme plus lisible que celui de XML pour représenter des document XML, celui des "termes base de données". Les termes base de données de Xcerpt ont l'une des formes $l\{t_1, \ldots, t_n\}$ ou $l[t_1, \ldots, t_n]$ où :

– l est une balise d'élément (aussi appelée "étiquette" et, en anglais, "tag name")

– $\{t_1, \ldots, t_n\}$ est un ensemble non-ordonné de sous-termes base de données

– $[t_1, \ldots, t_n]$ est un ensemble ordonné de sous-termes base de données

Les termes base de données de Xcerpt peuvent contenir des référence, ce qui permet de définir des structures de graphes avec cycles : $a@t$ associe la référence (ou le "nom") a au sous-terme t, ^ a est un pointeur qui renvoie au sous-terme dont la référence est a. Le terme base de données suivant décrit une liste de publications. Des références sont utilisées afin de partager des données, des noms d'auteurs, comments à plusieurs publications.

```
bib {
   authors {
      a1 @ author {
         name { "Serge Abiteboul" }, publications { ^b1, ^b2 }
      },
      a2 @ author {
         name { "Peter Buneman" }, publications { ^b1 } },
      a3 @ author { name { "Dan Suciu" }, publications { ^b1 } },
      a4 @ author { name { "Richard Hull" }, publications { ^b2 }
      },
      a5 @ author { name { "Victor Vianu" }, publications { ^b2 }
      }
   },
   books {
     b1 @ book {
        title { "Data on the Web" },
        authors [ ^a1, ^a2, ^a3 ],
        price { "69.95" } },
     b2 @ book {
        title { "Foundations of Databases" },
        editors [ ^a1, ^a4, ^a5 ],
        price { "29.00" }
     }
   }
}
```

XML offre plusieurs mécanismes de références : les attributs ID-IDREF, les liens hypertexte, et les formalismes RDF et XLink. Le langage d'interrogation prototype Xcerpt ne connaît par contre actuellement qu'un seul mécanisme de références. Cette restriction doit être vue comme une abstraction utile durant le développement du langage qui peut être levée à tout moment : rien ne s'oppose à inclure dans Xcerpt plusieurs mécanismes de références.

La question de savoir si un langage d'interrogation du Web doit offrir une autre syntaxe pour les documents XML que la syntaxe originelle de XML a beaucoup été

débattue. Xcerpt propose une telle syntaxe pour les raisons suivantes : (1) Il y a un accord général que l'absence d'une telle syntaxe dans le langage XSLT est un obstacle non-négligeable à son utilisation ; (2) une syntaxe comme celle des termes bases de données de Xcerpt peut être aisément convertie en la syntaxe standard de XML.[1] La syntaxe XML du terme base de données Xcerpt donné ci-dessus est comme suit (les attributs `reference` et `refersto` de l'espace de noms préfixé `xcerpt` sont des attributs XML de types respectifs ID et IDREF) :

```
<bib xmlns:xcerpt="http://xcerpt.org" >
  <authors>
   <author  xcerpt:anchor="a1">
     <name>Serge Abiteboul</name>
     <publications>
      <xcerpt:reference  xcerpt:refersTo="b1"/>
      <xcerpt:reference  xcerpt:refersTo="b2"/>
     </publications>
   </author>
   ...
  <books>
   <book xcerpt:anchor="b1">
     <title >Data on the Web</title>
     <authors >
      <xcerpt:reference  xcerpt:refersTo="a1"/>
      <xcerpt:reference  xcerpt:refersTo="a2"/>
      <xcerpt:reference  xcerpt:refersTo="a3"/>
     </authors>
     <price >69.95</price>
   </book>
   ...
  </books>
</bib>
```

2.2. *Interrogations Élementaires*

Les termes d'interrogation de Xcerpt dont des gabarits spécifiant de manière éventuellement incomplète les données recherchées. Les parenthèses doubles {{ }} et [[ ]] indiquent des listes incomplètes de sous-termes, c'est à dire que des termes base de données ayant outre les sous-termes spécifiés dans la liste d'autres sous-termes sont des réponses possibles. Les parenthèses simples { } et [ ] indiquent des listes complètes de sous-termes, c'est à dire que seuls des termes base de données ayant exactement les

1. Un analyseur lexical ne suffit toutefois pas à réaliser cette conversion car les étiquettes des parenthèses fermantes doivent être mémorisée lors de la lecture de parenthèses ouvrantes ce qui est aisément réalisé en ajoutant une pile à un analyseur lexical élémentaire.

sous-termes spécifiés dans la liste peuvent être des réponses. Les parenthèses rondes (ou d'ensembles) {{ }} et { } indiquent que l'ordre des sous-termes est indifférent. Les parenthèses angulaire (ou de listes) [[ ]] [ ] signifient que l'ordre des sous-termes indiqués dans le terme d'interrogation doit être respecté par les réponses. XML ne connaît que des "termes ordonnés". Xcerpt considère également des "termes non-ordonnés" parce qu'ils sont utiles dans des applications de type bases de données. Le constructeur "descendant" *desc* sert à indiquer un sous-terme à une profondeur quelconque. Les variables sont indiqués par le préfixe *var*. Une variable peut être liée à un sous-terme par le constructeur $\rightsquigarrow$ (lu "as" ou "comme"). Les deux termes d'interrogation Xcerpt suivants peuvent être utilisés pour sélectionner des livres de la liste de publications donnée en exemple en partie 2.1.

```
books {{
  book {{ var TITLE ⇝ title {{}}, authors [[ var AUTHOR ]] }}
}}
```

Ce terme d'interrogation lie la variable `TITLE` aux éléments XML étiquettés `title` qui sont des fils d'éléments `book` eux-même fils d'éléments `books`. `title {{}}` signifie que le contenu des éléments étiquettés `title` à sélectionner est indifférent. En raison de la seconde spécification d'élément `authors [[ var AUTHOR ]]`, la variable `TITLE` n'est liée comme précédemment expliqué que dans les cas ou les éléments étiquettés `title` ont un "frère" étiqueté `authors`. La variable `AUTHOR` est liée à l'un des sous-éléments de tels frères. On notera que, comme en programmation logique, en SQL et en QBE, les réponses multiples, c'est-à-dire différentes liaisons pour les variables, sont possibles.

```
books {{
  book {{ desc var Section ⇝ section {{}}, title { /.*XML.*/ } }}
}}
```

Ce terme d'interrogation permet de retrouver les sections (variable `Section`) dont les titres contiennent la chaîne de caractère XML. On notera l'utilisation d'une expression régulière, `/.*XML.*/`, pour la sélection des titres. Comme de nombreux langages récents, Xcerpt utilise la syntaxe POSIX d'expressions régulières pour la spécification de chaînes de caractères.

Les termes d'interrogation de Xcerpt sont semblables aux atomes de la logique et de la programmation logique en ce qu'ils permettent d'exprimer les positions relatives des différents constituants d'une interrogation, qu'ils utilisent des variables logiques et qu'ils acceptent des réponses multiples. Ils diffèrent toutefois des atomes de la logique et de la programmation logique d'une part en ce qu'ils permettent des spécifications incomplètes tant en "largeur" à l'aide des parethèses doubles {{ }} et [[ ]] qu'en "profondeur" à l'aide du constructeur *desc*, d'autre part en ce qu'ils permettent de lier une variable à l'intérieur d'un terme d'interrogation à l'aide du constructeur $\rightsquigarrow$.

Ce dernier aspect peut-être à juste titre vu comme un “glaçage syntaxique”. Il contribue néanmoins de manière importante à réaliser la notion d’interrogation par gabarit mentionnée dans l’introduction.

Les termes d’interrogation de Xcerpt permettent également d’interroger les noms d’attributs, les valeurs d’attributs ainsi que les noms de balises d’éléments.

2.3. *Interrogations Complexes*

Les interrogations complexes sont obtenues par conjonctions et disjonctions d’interrogations élémentaires ou complexes ainsi que par utilisation des constructeurs de choix (`if then else` et `case of`). La conjonction et la disjonction de Xcerpt sont des opérateurs préfixes d’arités multiples notés $and\{t_1, \ldots, t_n\}$, $and[t_1, \ldots, t_n]$, $or\{t_1, \ldots, t_n\}$, et $or[t_1, \ldots, t_n]$. Les parenthèses rondes (ou d’ensemble) indiquent un ordre d’évaluation indifférent, les parenthèses angulaire (ou de listes) spécifient un ordre dans lequel les sous-requêtes d’interrogation doivent être évaluées. Une telle distinction est très utile en pratique, par exemple lorsque l’on sait que certains éléments sont beaucoup plus rares que d’autres. Dans de tels cas, imposer à une conjonction ou à une disjonction d’évaluer tout d’abord les termes d’interrogation livrant les plus petit nombres de réponses ou bien se rapportant à des éléments rares permet de très substantiels gains de temps pour l’évaluation de la requête globale d’interrogation. C’est par example le cas d’une requête d’interrogation cherchant une chambre d’hotel dans une ville à une certaine date et une place sur un vol vers cette ville à la même date. Si l’on sait que les chambres d’hotels encore libres à la date désirée sont beaucoup plus rares que les places sur des vols à cette même date, de meilleurs temps d’évaluation sont possibles en évaluant d’abord la requête partielle se référant à la chambre d’hotel, puis la requête partielle se référant au vol.

Les constructeurs de choix (`if then else` et `case of`) sont utiles pour une expression naturelle de conditions.

2.4. *Constructions*

Les termes de construction Xcerpt sont des gabarits de réponses. Il indiquent comment les variables liées lors de l’évaluation d’interrogations (élémentaires ou complexes) sont ré-assemblées en des document XML nouveaux.

Les termes de construction de Xcerpt peuvent contenir :

– des variables

– des occurrences du constructeur *all* servant à collecter les expressions correspondants aux différentes valeurs possibles des variables figurants dans leurs champs ;

– des occurrences du constructeur *some n* servant à collecter *n* expressions choisies de manière non-déterministe (don’t care non-determinism) correspondants à *n* différentes valeurs possibles des variables figurant dans leurs champs.

La sélection non-déterministe de données est très utile dans de nombreux cas pratiques d'interrogation du Web. C'est ainsi que la recherche d'une chambre d'hotel, d'une place sur un vol entre deux villes, ou d'une place de train conduit souvent à des options dont le choix est totalement indifférent. Le constructeur *some n* de Xcerpt permet d'exprimer un telnon-déterminisme (don't care non-determinism) de manière très simple et naturelle. Les languages ne disposant pas d'un tel constructeur de choix non-déterministe, comme la plupart des languages de programmation mais aussi comme XSLT et XQuery, imposent soit de programmer un choix arbitraire, soit de déterminer un ensemble (arbitrairement ordonné) de valeurs puis de sélectionner l'une, par exemple la première, de ces valeurs. Cela conduit tant à des programmes d'interrogation qu'à des évaluations d'interrogations plus complexes que nécessaire.

Le terme de construction Xcerpt suivant retourne, comme réponses différentes, les différentes paires possibles de titres et d'auteurs de publications lorsque les variables TITLE et AUTHORS sont évaluées par des requêtes d'interrogation comme celle données en partie 2.2.

```
result [ var TITLE, var AUTHOR ]
```

Le terme de construction Xcerpt suivant retourne la liste (dans l'ordre issu de l'évaluation) de toutes les réponses possibles à la requête d'interrogation précédente :

```
results { all result [ var TITLE, var AUTHOR ] }
```

Les constructeurs d'agrégation *all* et *some n* peuvent bien sûr être imbriqués. C'est ainsi que le terme de construction Xcerpt suivant retourne la liste de toutes les publications, donnant pour chaque auteur la liste des publications auxquelles it a contribué :

```
publist { all publication { var AUTHOR, all var TITLE } }
```

Le terme de construction Xcerpt suivant est semblable au précédent bien que différent en un point essentiel : il retourne la liste de toues les publications, donnant pour chaque publication la liste de tous ses auteurs :

```
publist { all publication { all var AUTHOR, var TITLE } }
```

On notera la symétrie des deux termes de construction précédent qui reflète parfaitement la symétrie de leur expression en langue naturelle. On notera que les langages XQuery et XSLT ne présentent pas de telle symétrie. Pour cette raison, il peuvent être considérés comme moins déclaratifs que Xcerpt.

Les constructeurs d'agrégation *all* et *some* correspondent à des méta-prédicats de Prolog comme *setof*. Une différence essentielle est que Xcerpt réalise l'agrégation sans faire appel à la méta-programmation. En terme de sémantique procédurale, la différence est minimale. Elle est plus importante du point de vue de la sémantique déclarative.

2.5. *Règles Construction-Interrogation*

Un programme Xcerpt consiste en une collection (finie) de “règles construction-interrogation” de la forme :

```
CONSTRUCT
   <terme-de-construction>
FROM
   in <ressource>
   <interrogation>
WHERE
   <condition>
END
```

Les variables figurant dans le terme de construction doivent toutes figurer dans l’interrogation. En d’autre termes, les règles construction-interrogation de Xcerpt sont à champs restreints. Cette contrainte permet que les règles construction-interrogation aiment une sémantique claire et puissent être évaluées tant par chaînage avant que arrière, permettant ainsi des optimisations qui peuvent être indispensables dans le contexte hautement distribué du Web.

L’interrogation est une interrogation élémentaire (cf. partie 2.2) ou complexe (cf. partie 2.3). Le constructeur *in* , qui est optionnel, sert à indiquer une ou plusieurs ressources Web, spécifiées par leur URI, à interroger. Si aucune ressource n’est spécifiée le programme Xcerpt lui-même est la ressource à considérer.

La partie “condition” est optionnelle. Elle sert à indiquer des relations entres valeurs qui ne peuvent pas être naturellement exprimées dans les gabarits dont l’interrogation est composée, par exemple : `PRICE < 40`.

L’exemple suivant de règle construction-interrogation montre comment des interrogations et des construction sont réunis en une règle construction-interrogation :

```
CONSTRUCT
   books {
      all book {
         var TITLE,
         price-a { var PRICEA }, price-b { var PRICEB } }
   }
FROM
and {
   in { resource { "http://bn.com" },
      bib {{
         book {{ var TITLE ⇝ title{{}}, price { var PRICEA } }}
      }} },
   in { resource { "http://amazon.com" },
```

```
      reviews {{ entry {{ var TITLE ⇝ title{{}},
                          price { var PRICEB } }}
      }} }
   }
WHERE
   or { var PRICEA < 40, var PRICEB < 40 }
END
```

La règle précédente interroge deux ressources donnant des listes de livres, `http://bn.com` et `amazon.com`, et sélectionne les prix des livres mentionnés aux deux ressources à condition que les deux prix soient strictement inférieurs à 40. Les titres et prix sélectionnés sont assemblés en un élément étiqueté `books` dont les fils sont tous les éléments `book` donnant pour chaque titre sélectionné les deux prix correspondants.

Les règles Xcerpt peuvent être utilisées afin de structurer un calcul complexe. L'exemple suivant montre comment une table HTML peut être construite en deux étapes. Une première étape construit le contenu des cellules de la table. Cette première étape est réalisée par la règle ci-dessus. Une seconde étape de formattage produit la table HTML. Cette seconde étape est réalisée par la règle ci-dessous :

```
CONSTRUCT
  table {
    tr { td{"Booktitle"}, td{"Price at A"}, td{"Price at B"} },
    all tr { td{var TITLE}, td{var PRICEA}, td{var PRICEB} }
  }
FROM
  books {{
    book {
      title { var TITLE },
      price-a { var PRICEA },
      price-b { var PRICEB }
    }
  }}
END
```

Les règles Xcerpt peuvent être encainées et permettent des spécifications récursives.

3. visXcerpt : Visualisation du Langage Textuel Xcerpt

visXcerpt est un langage qui réalise les constructions du langage textuel Xcerpt de manière visuelle. Les notions de bases de visXcerpt et Xcerpt sont indentiques. En cela visXcerpt diffère de la plupart des langages visuels d'interrogation du Web et

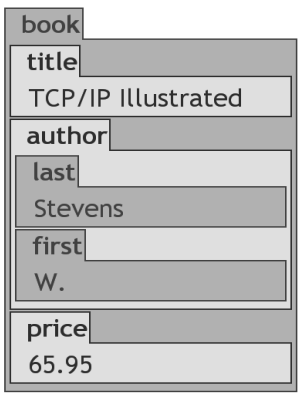
de données semi-structurés [CON 89, CER 99, PIE 01, MUN 00, PAP 02, ERW 00] qui traduisent de manière plus ou moins compliquées des constructions de langages textuels en des constructions visuelles reposant sur principes fondamentalement différents. Le langage visuel visXcerpt peut reposer les mêmes notions de base que le langage textuel Xcerpt grâce au concept déclaratif de Xcerpt de gabarit d'interrogation et de constructions : cette notion de gabarit est conceptuellement très proche de la programmation visuelle. En d'autres termes, c'est le caractère déclaratif de Xcerpt qui permet à sa réalisation visuelle visXcerpt de rester très proche de Xcerpt.

Un prototype du langage visuel visXcerpt a été conçu comme une application Web en DHTML ("dynamic HTML") et en CSS [css01]. Une second prototype en cours de réalisation ne reposera plus que sur CSS, c'est à dire sera réalisé comme une pure visualisation de programme textuels Xcerpt. Pour cela, quelques extensions de CSS sont toutefois indispensables, en particulier afin de fournir des aspects dynamiques manquants actuellement à CSS. Ces extensions de CSS sont en cours de conception.

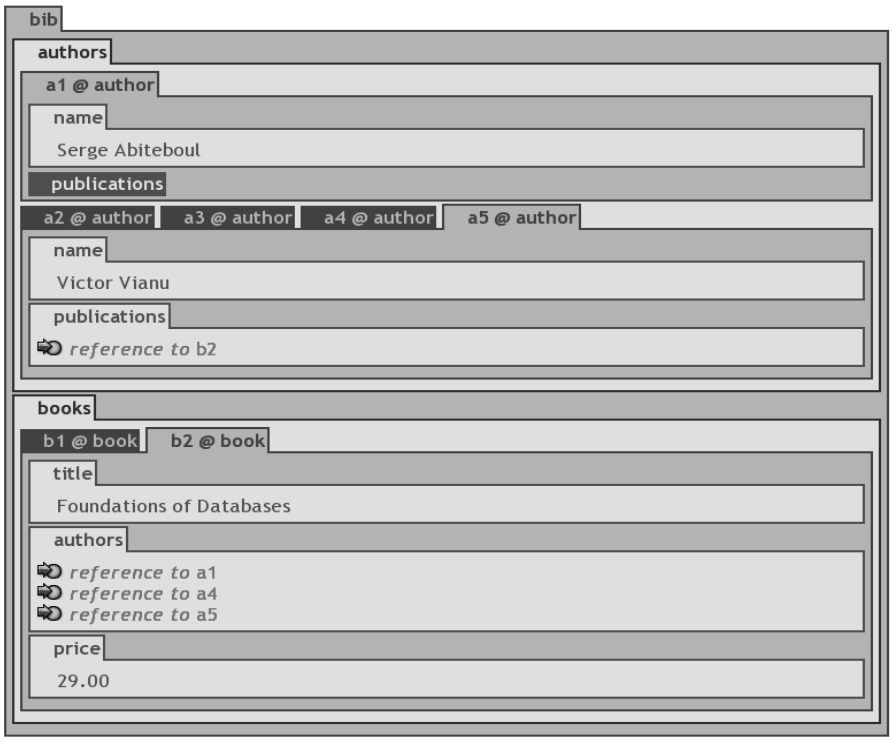
3.1. *Représentation Visuelle de Documents XML*

Un aspect essentiel d'un langage visuel comme visXcerpt fondé sur la notion de gabarits d'interrogation et de construction est la visualisation de documents XML. La visualisation de documents XML est au cœur de la visualisation de termes tant d'interrogation que de de construction. En visXcerpt, un document XML est visualisé par des rectangles imbriqués, chaque rectangle représentant un élément du document XML, l'imbrication des rectangles correspondant à l'imbrication des éléments : un représentant un élément père contiend donc autant de rectangles que cet élément père a d'éléments fils. L'étiquettes d'un élément figure dans un cavallier en partie supérieure gauche du rectangle représentant l'élément. Le contenu textuel (CDATA) d'un élément est visualisé sous forme bien sûr textuelle à l'intérieur du rectangle représentant cet élément. L'exemple suivant illustre cette visualisation en visXcerpt des éléments d'un document XML.

L'exemple ci-contre montre comment des couleurs sont utilisées pour distinguer les (rectangles représentant des) éléments : Une liste finie de couleurs est utilisée afin que deux éléments imbriqués aient des couleurs de fond différentes. En présence d'un nombre plus grand d'élément imbriqués que cette liste n'a de couleurs, les couleurs de cette liste sont tout simplement répétées dans l'ordre de la liste.



Il est souvent utile, voire indispensable en présence de gros documents ou d'interrogations ou de constructions complexes, de pouvoir cacher de manière temporaire une partie du document, de l'interrogation ou de la construction considérée. Avec visXcerpt, cela est



possible en “pliant” un élément dans son cavalier : de l’élément plié ou caché ne reste alors visible que son cavalier. On notera que les éléments cachés sont agencés dans le plan horizontalement alors que les éléments visibles sont agencés verticalement. Le pliage d’éléments est une des fonctionnalités de visXcerpt qui ne peut être réalisées en CSS, en son état actuel, car CSS n’offre aucune fonctionnalité dynamique de ce genre.

L’exemple suivant démontre non seulement comment des informations peuvent être cachées par pliage d’éléments mais aussi comment visXcerpt visualise les références de Xcerpt. Une référence est visualisée en visXcerpt par un lien hypertexte. Afin de faciliter la navigation le long des références, à toute référence est associé par visXcerpt un lien en direction contraire. L’exemple suivant est la visualisation en visXcerpt de la bibliographie considérée en partie 2.1.

Les termes ordonnés et non-ordonnés (cf. partie 2.1) sont distingués en visXcerpt à l’aide d’icônes placées en haut à gauche des (rectangles représentants les) éléments. Pour des raisons de place, cet aspect de visXcerpt n’est pas illustré ici. Par la suite, tous les exemples sont supposés correspondre à des documents ordonnés.

3.2. *Représentation Visuelle d'Interrogations Élémentaires*

Les requêtes d'interrogation élémentaires sont obtenues en étendant la représentation des documents XML au listes de sous-termes incomplètes, à des variables et à la construction "descendant" (`desc` dans le langage textuel Xcerpt). Toutes les constructions spécifiques au langage visXcerpt, c'est à dire qui correspondent à des mots réservés du langage textuel Xcerpt, sont caractérisés par deux propriétés suivantes :

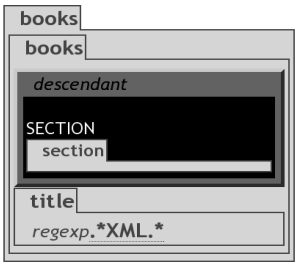
1) Les constructions spécifiques au langage visXcerpt sont représentées dans les couleurs noire, blanche ou grise (dans quelques cas également en un blanc partiellement transparent),

2) lorsque ces constructions spécifiques au langage visXcerpt sont accompagnées d'indications textuelles, ces textes figurent en italique.

Les exemples suivants de représentations en visXcerpt sont des visualisations d'exemples données précédemment dans les parties de l'article consacrées à Xcerpt. Pour cette raison, leur signification n'est pas rappelée.

L'exemple ci-contre contient deux variables, Title et AUTHOR. La variable Title est liée à l'un des éléments dont l'étiquette figure immédiatement sous cette variable. Il s'agit de la visualisation de la construction $\rightsquigarrow$ (ou "as", ou "comme") du langage textuel Xcerpt qui sert à restreindre les valeurs possibles d'une variable. On notera que les variables sont représentées en visXcerpt comme des rectangles à fond noirs, le nom de la variable figurant en blanc en haut à gauche de ce fond et une restriction des valeurs de cette variable comme un élément représenté à l'intérieur du rectangle associé à cette variable.

L'exemple ci-contre illustre également la visualisation en visXcerpt de la construction *descendant* et d'expressions régulières (servant à la sélection de portions de textes). La construction *descendant* est représentée en visXcerpt par un rectangle gris avec un effet de profondeur, une métaphore visuelle naturelle pour exprimer une profondeur d'imbrication d'éléments. Le mot *descendant* figure dans la représentation visXcerpt afin de faciliter la reconnaissance de la construction. Les expressions régulières sont représentées comme des textes (CDATA) toutefois soulignés d'une ligne pointillées et accompagnés du mot *regexp* afin de bien les distinguer visuellement de textes ordinaires.



3.3. *Représentation Visuelle d'Interrogations Complexes*

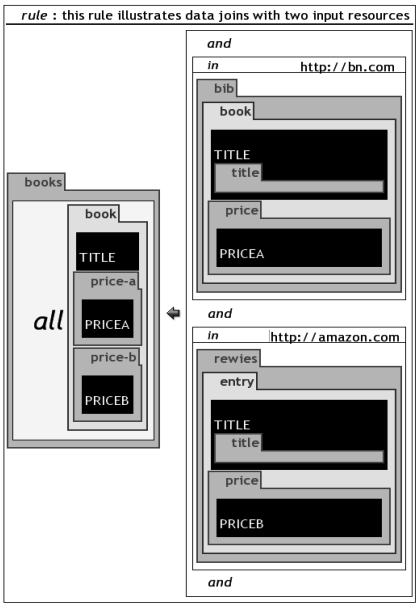
En visXcerpt, les conjonctions et disjonctions sont représentées comme des rectangles blancs bordés de noir. Les composants des conjonctions sont agencés verticalement, ceux des disjonctions horizontalement. Cela reflète des usages fréquents en logique et présente l'avantage de clairement distinguer les deux types de groupements d'interrogations. Les mots *and* (*or*) figurent entre les composants des conjonctions (des disjonctions). Un exemple est donné plus bas dans le cadre d'un exemple de représentation en visXcerpt d'une règle construction-interrogation (cf. partie 3.5).

3.4. *Représentation Visuelle de Constructions*

Les termes de construction sont représentés en visXcerpt de manière semblable aux termes d'interrogation.

L'exemple précédent gauche est la représentation en visXcerpt d'un des termes de construction donnés en exemple dans la partie consacrée au langage textuel Xcerpt.

Le terme de construction en visXcerpt précédent droit utilise la construction *all*. Cette construction est représentée en visXcerpt par un rectangle transparent blanc dans lequel figure pour une meilleure reconnaissance le mot réservé *all*.



3.5. *Représentation Visuelle de Règles Construction-Interrogation*

La visualisation des règles construction-interrogation est inspirée de la représentation traditionnelles des règles de la

programmation logique : une flèche orientée de droite à gauche réunit le terme de construction à gauche à l'interrogation à droite. La représentation d'une règle est bordé d'un mince trait noir et en partie supérieure figure le mot réservé *rule*. Une règle peut être "repliée" sous sa ligne de titre. Plusieurs règles forment un programme : elles sont agencées dans le plan verticalement.

3.6. *Langage visuel et Éditeur de Texte Avancé*

Un langage visuel tel que visXcerpt ressemble beaucoup à un éditeur de texte offrant des fonctionalités avancées telles que coloration de portions de textes reflétant la syntaxe du langage édité, pliage de portions de texte (définies d'après la syntaxe) et, bien sûr indentation suivant la syntax (souvent appelée "pretty print" en anglais). La question se pose donc, dans quel mesure un langage visuel tel que visXcerpt pourrait être conçu sur la base d'un éditeur de texte avancé.

Un langage visuel comme visXcerpt offre non seulement des fonctionalités standards d'u n éditeur de texte avancé et adapté à un language, mais aussi des fonctionalités spécifiques à l'interrogation en Xcerpt. En particulier, les caractéristiques suivantes de visXcerpt distinguent ce langage visuel d'un éditeur de textes avancé :

– visXcerpt impose que les termes (de données, d'interrogation, ou de construction) ainsi que les autres structures du langage (come, par exemple, des règles) soient nécessairement syntaxiquement bien formées. Un éditeur de texte avancé ne peut pas garantir dans tous les cas que seules des structures syntaxiquement bien formées du langage considéré puissent être écrites par l'utilisateur de l'éditeur.

– visXcerpt interprête les liens hypertextes comme tels et permets de les suivre de la manière usuelle. Cette fonctionalité n'est en général pas offertes par des éditeurs de textes, précisemment parcequ'ils se réfèrent au textes sources (par exemple des textes source HTML ou XML) et non pas au rendu de ces textes sources (que réalise, par exemple, un browser).

– visXcerpt utilise le plan pour le rendu graphique de programme Xcerpt. C'est ainsi que des requêtes complexes d'interrogation sont rendues tant en largeur qu'en hauteur et que le pliage est possible, selon les structures, dans les deux dimensions. Un éditeur de textes, parcequ'il se réfère au texte, ne connait en général que la ligne et des portions linéaires de textes.

Il n'en reste pas moins que la frontière entre langage visuel et éditeur de textes avancé ne peut pas par essence être clairement tracée : ces deux types de systèmes ont de nombreux points en commun et l'évolution des travaux dans leur domaines peut très bien conduire à les rapprocher considérablement.

4. Comparaisons, Conclusion et Perspectives

Xcerpt et visXcerpt sont deux langage déclaratifs et déductif d'interrogation du Web. Ils reposent sur une notion de "gabarit" tant pour les requêtes d'interrogation que pour les constructions permettant de ré-assembler les données sélectionnées. Comme

les langages traditionnels d'interrogation du Web, XPath, XQuery et XSLT, ils permettent une spécification incomplète des données à sélectionner. Comme les langages de la programmation logique, ils ont une notion de règle de déduction apparentée aux vues des langages d'interrogation des bases de données relationnelles.

L'interrogation par "gabarits" et la séparation stricte entre "gabarits" d'interrogation et "gabarits" de construction de Xcerpt et visXcerpt permettent dans de très nombreux cas des requêtes d'interrogation plus simples, plus lisibles et bien plus naturelles que les requêtes navigationelles de langages tels que XSLT et XQuery. Aucun cas n'est connu des auteurs où des requêtes d'interrogation en Xcerpt ou visXcerpt seraient plus complexes que les requêtes correspondantes en XSLT ou XQuery. Le pouvoir d'expression de Xcerpt et visXcerpt n'est toutefois pas plus grand que celui de ces langages : comme XSLT, XQuery et d'autres langages d'interrogation du Web, Xcerpt et visXcerpt ont le pouvoir d'expression de machines de Turing.

L'interrogation par "gabarits" de Xcerpt et visXcerpt démarque ces langages non seulement de XSLT et XQuery, mais aussi de SQL. Pour interroger une relation "carnet" dont chaque tuple contient un "nom", un "prénom" et une "adresse", SQL n'utilise pas d'atome ou de "gabarit" (par exemple "carnet(N, P, A)") comme la logique et la programmation logique mais au contraire des "chemins" assez semblables à ceux de XPath s'appuyant sur les noms des attributes (come par exemple "carnet.nom"). Dans le cas de tuples plats comme ceux des bases de données relationelles pour lesquelles SQL a été conçu, la différence entre de tels "chemins" et les atomes de la logique et de la programmation logique n'est pas frappante – comme le suggère le fait que cette différence est souvent négligée. Lorsque des structures imbriquées comme des documents XML sont considérées, les "chemins" à la SQL et XPath deviennent souvent difficiles à comprendre pour le programmeur alors que des "gabarits" comme ceux de Xcerpt et visXcerpt ou les atomes de la logique et de la programmation logique sont très intuitifs. "Gabarits" d'interrogation ou atomes peuvent en effet être vus comme des formulaires plus ou moins remplis décrivant les données recherchées.

Un aspect supplémentaire qui distingue Xcerpt et visXcerpt d'une part de XSLT, XQuery et SQL d'autre part est la déclarativité de Xcerpt et visXcerpt. Ces deux derniers langages peuvent incontestablement être considérés comme moins procéduraux que XSLT, XQuery et SQL en raison tant de leurs "gabarits" d'interrogation, de leurs constructeurs de agrégation "some" et "all" que de la forme particulière d'unification ("unification de simulation" [BRY 02b]) sur laquelle ils s'appuient.

Xcerpt et visXcerpt permettent une interrogation du Web et du Web Sémantique plus simple et plus naturelle qu'une combination de systèmes d'exploration du Web et de Prolog (ou d'un autre langage de programmation logique). En effet, Xcerpt et visXcerpt permettent sans programmation particulière d'exprimer des requêtes d'interrogation ne spécifiant les données recherchées (sur le Web ou Web Sémantique) que de manière incomplête. Pour simple que cette fonctionalité puisse apparaître à un programmeur, elles nécessite la forme particulière d'unification, "l'unification de simulation" [BRY 02b], qui est un des aspects essentiels des langages Xcerpt et visXcerpt. Il est important de noter qu'exprimer en Prolog une telle forme d'unification

ou des requêtes d'interrogation ne spécifiant les données recherchées que de manière incomplête est loin d'être une tâche de programmation triviale.

visXcerpt est une réalisation visuelle de Xcerpt qui repose sur les mêmes principes que Xcerpt. L'identité de principes de Xcerpt et visXcerpt a permis de réaliser visXcerpt pour une grande part comme une visualisation (du document XML qu'est un programme Xcerpt) à l'aide du langage de formattage CSS. Des aspects dynamique de visXcerpt, comme le "pliage" d'éléments, n'ont toutefois pas pu être implémentés à l'aide de CSS parce que ce langage de formattage n'offre aucune primitive dynamique.

Xcerpt et visXcerpt sont des supports d'activité de recherche. L'objectif de ces langage n'est donc pas de devenir des produits ou des standards, mais d'étudier des questions importantes dans la recherche actuelle. L'attention des auteurs est présentement consacrée en priorité à l'utilisation de techniques déductives pour la réalisation d'applications Web et Web Sémantique. La notion de règles de Xcerpt et visXcerpt inspirée des clauses de la programmation logique et des vues du langage d'interrogation des bases de données relationnelles SQL apparaît très utile pour réaliser de telles applications.

Le langage Xcerpt a été présenté dans les articles suivants [BRY 03, BRY 02a]. Le langage visXcerpt a été présenté dans l'article [BER 03]. Des études sur la sémantique de Xcerpt et de visXcerpt ont été présentées dans les publications suivantes [BRY 02c, BRY 04]. Certains aspects de Xcerpt et de visXcerpt (comme les primitives d'édition de visXcerpt sûre, c'est à dire respectant la validité syntaxique et structurelle des programmes visuels) n'ont pas été décrites dans le présent article pour des raisons de place.

5. Remerciements

Cette recherche a été financée par la Commision Européenne et par le bureau fédéral suisse pour l'éducation et la science dans le cadre du project du 6ème programme cadre REWERSE N° 506779 (cf. `http://rewerse.net/`)

6. Bibliographie

[ABI 00] ABITEBOUL S., BUNEMAN P., SUCIU D., *Data on the Web. From Relations to Semistructured Data and XML*, Morgan Kaufmann, 2000.

[BER 03] BERGER S., BRY F., SCHAFFERT S., WIESER C., « Xcerpt and visXcerpt : From Pattern-Based to Visual Querying of XML and Semistructured Data », *Proc. Intl. Conference on Very Large Databases (VLDB03)*, Berlin, Germany, 2003.

[BRY 02a] BRY F., SCHAFFERT S., « A Gentle Introduction into Xcerpt, a Rule-based Query and Transformation Language for XML », *Proc. Int. Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, June 2002, (invited article).

[BRY 02b] BRY F., SCHAFFERT S., « Towards a Declarative Query and Transformation Language for XML and Semistructured Data : Simulation Unification », *Proc. Int. Conf. on Logic Programming (ICLP)*, LNCS 2401, Springer-Verlag, 2002.

[BRY 02c] BRY F., SCHAFFERT S., « Towards a Declarative Query and Transformation Language for XML and Semistructured Data : Simulation Unification », *Proceedings of the Int. Conf. on Logic Programming (ICLP)*, Copenhagen, Denmark, July 2002, Springer-Verlag, LNCS 2401.

[BRY 03] BRY F., SCHAFFERT S., « An Entailment for Reasoning on the Web », rapport n° PMS-FB-2003-5, 2003, Institute for Computer Science, University of Munich.

[BRY 04] BRY F., SCHAFFERT S., SCHROEDER A., « A contribution to the Semantics of Xcerpt, a Web Query and Transformation Language. », *18. Workshop Logische Programmierung (WLP)*, Potsdam, Germany, 2004.

[BUN 00] BUNEMAN P., FERNANDEZ M., SUCIU D., « UnQL : A Query Language and Algebra for Semistructured Data Based on Structural Recursion », *VLDB Journal*, vol. 9, n° 1, 2000, p. 76-110.

[CER 99] CERI S., DAMIANI E., FRATERNALI P., PARABOSCHI S., TANCA L., « XML-GL : A Graphical Language for Querying and Restructuring XML Documents », *Sistemi Evoluti per Basi di Dati*, 1999.

[CON 89] CONSENS M., MENDELZON A., « Expressing Structural Hypertext Queries in GraphLog », *Second ACM Hypertext Conf.*, 1989, p. 269–292.

[css01] W3C, http ://www.w3.org/TR/css3-roadmap/, « Introduction to CSS3 », 2001.

[ERW 00] ERWIG M., « A Visual Language for XML », *IEEE Symp. on Visual Languages*, 2000, p. 47–54.

[MUN 00] MUNROE K. D., PAPAKONSTANTINOU Y., « BBQ : A Visual Interface for Integrated Browsing and Querying of XML », *VDB*, 2000.

[owl00] W3C, http ://www.w3.org/TR/owl-features/, « OWL Web Ontology Language Overview », 2000.

[PAP 02] PAPAKONSTANTINOU Y., PETROPOULOS M., VASSALOS V., « QURSED : Querying and Reporting Semistructured Data », *ACM SIGMOD*, 2002.

[PIE 01] PIETRIGA E., QUINT V., VION-DURY J.-Y., « VXT : A Visual Approach to XML Transformations », *ACM Symp. on Document Engineering*, 2001.

[rdf00] W3C, http ://www.w3.org/RDF/, « Resource Description Framework (RDF) », 2000.

[W3 00] W3 Consortium, http ://www.w3.org/TR/REC-xml, « Extensible Markup Language (XML) 1.0, Second Edition », October 2000.

[xpa99] W3 Consortium, http ://www.w3.org/TR/xpath, « XML Path Language (XPath) », 1999.

[xqu01] W3C, http ://www.w3.org/TR/xquery/, « XQuery : A Query Language for XML », Feb 2001.

[xsd01] W3C, http ://www.w3.org/TR/xmlschema-0/, « XML Schema Part 0 : Primer », 2001.

[xsl00] W3C, http ://www.w3.org/Style/XSL/, « Extensible Stylesheet Language (XSL) », 2000.

[ZLO 77] ZLOOF M. M., « Query-by-Example : A Data Base Language », *IBM Systems Journal*, vol. 16, n° 4, 1977, p. 324-343.

Annexe pour le service de fabrication

Article pour les actes :
JFPLC'04

Auteurs :
Sacha Berger, François Bry, Sebastian Schaffert

Titre de l'article :
Xcerpt et visXcerpt : Langages déductifs d'interrogation du Web

Titre abrégé :
Xcerpt et visXcerpt

Traduction du titre :
Xcerpt and visXcerpt : deductive web querying languages

Date de cette version :
16 avril 2004

Coordonnées des auteurs :

– téléphone : +49-89-2180-9316
– télécopie : +49-89-2180-9311
– Email : sacha.berger@ifi.lmu.de

Logiciel utilisé pour la préparation de cet article :
LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.10 du 2000/03/21.

Formulaire de copyright :
Joindre le formulaire de copyright signé, récupéré sur le web à l'adresse
`http://www.hermes-science.com`