Les langages de requêtes : du modèle à valeurs complexes au modèle à objets

C. Delobel, V. Benzaken, E.Waller

Type et valeur complexe

Type	Valeur
D	a, b, 2
{D}	{1,2,a}
[const : D, mod : D]	[const:"peugeot",mod:406]
{{D}}	{{1,2},{2,3,1},{}}

C. Delobel, V. Benzaken, E.Waller

Système de types

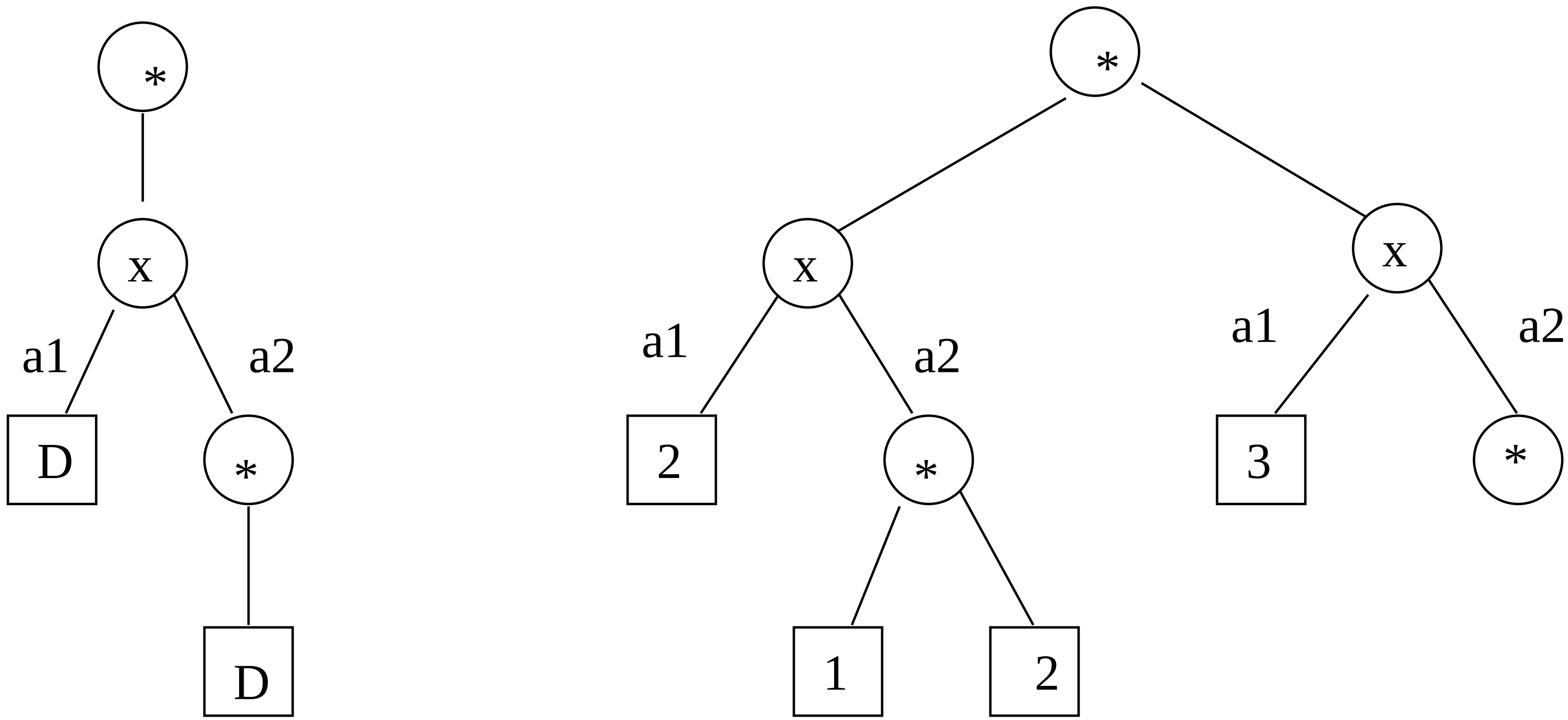
t ::= D |[a1 : t1, a2 : t2,......, an : tn] |{t}

Type est l'ensemble des expressions de type vérifiant la grammaire.

Interprétation de type

- [[D]] = D
- [[[a1:t1,....,an:tn]]] = {[a1:v1,...,an:vn] |vi app à ti, i=1,n}
- [[{t}]] = { {v1,v2,...,vj} | j   0, vi app à t}

Représentation



C. Delobel, V. Benzaken, E.Waller

Schéma et instance

- Définition : Un *schéma* S est une paire (R, T), où R est un ensemble de noms de relations et T une fonction de R dans Type.
- Une instance I d'un schéma S une fonction de R dans l'ensemble des valeurs, telle que pour tout r appartenant à R, I(r) est contenu dans [[T(r)]].
- I(r) par analogie avec le modèle relationnel sera appelé une relation de type T(r).

Exemple de schéma

R = { Monument, Restaurant}

T(Mom) =[nom: D, adresse: [ville:D, rue:D]]

T(Res) = [nom:D, adresse: [ville:D, rue:D], menus: {[type:D, prix:D]}]

Algèbre pour valeurs complexes

- Soit I, I1, I2, ... des relations de type t, t1, t2,...c'est-à-dire que I, I1, I2, ... sont des ensembles de valeurs sur t, t1,t2,...
- Opérations de base : si t1 = t2 alors I1 **union** I2, I1 **inter** I2, I1 - I2 sont des relations de type t1.

C. Delobel, V. Benzaken, E.Waller

Sélection

- I est une relation de type t = [a1:t1,..., ak:tk]
- **select** (C,I) est une relation de type t où C est une condition définie par
  - ai = d
  - ai = aj
  - ai in aj
  - ai = aj.a’ (si tj contient l’attribut a’)
  - aj.a’ = d
- **select** (C,I) = {v | v **in** I, v |= C }
- La conséquence logique v|=C est définie de façon classique.

C. Delobel, V. Benzaken, E.Waller

Projection

- **project** ((a1,...,al), I), lŠk, est une relation de type [a1:t1,..., al:tl]
- **project** ((a1,...,al), I) = { [a1:v1,...,al:vl] | **existe** vl+1,...,vk tel que [a1:v1,..., ak:vk] in I}

C. Delobel, V. Benzaken, E.Waller

Opérations constructives et destructives

- Powerset : **powerset** (I) = {v | v contenu dans I }
- **tup_create** ((a1,...,an), (I1,...,In) = { [a1:v1,...,an:vn] | vi in Ii }
- **set_create** (I) = {I}
- I de type t = {t'}, **set_destroy** (I) = {w |**exist** v **in** I, w **in** v }
- I de type t = [a:t'], **tuple_destroy** (I) = {v |[a:v] **in** I }

C. Delobel, V. Benzaken, E.Waller

Requêtes

- Soit S un schéma où les noms de relations sont R1,R2,...,Rn. Une requête q de type t retourne un ensemble de valeurs de type t.
- q est définie à partir des requêtes de base, comme suit :
  - Ri est une requête dont le résultat est I(Ri)
  - {d} est une requête dont le résultat est {d}
- si q, q1, q2,... sont des requêtes alors : q1 **inter** q2, q1 **union** q2, q1 - q2, **select** (C, q), **project** (A, q), **tuple_create** (A, q1,..,qk), **powerset** (q),... sont des requêtes

Exemple

nom des monuments de la ville de Paris ?

project (nom, **select** (adresse.ville = ’Paris’, Mom))

Extensions algébriques

- Renommage des attributs d'une relation
- Produit cartésien de deux relations
- Jointure
- nest et unnest

C. Delobel, V. Benzaken, E.Waller

Calcul

- A partir d'un schéma S et d'une instance I, il est possible de voir l'ensemble des faits comme une interprétation où les noms de relation sont des noms de prédicat.
- faits (I) = { R(v) | v in [[T(R)]], R est un nom de S }
- Le langage du calcul est construit sur un alphabet comprenant les noms de relation et des variables x, y, ... associées à un type.

Définition du calcul

- Une variable est atomique si son type est D.
- Un terme est soit une constante, une variable, ou une expression de la forme x.a (x est de type tuple contenant l'attribut a).
- Un litéral positif est une expression de la forme R(t), t = t', t **in** t', t **contdans** t'.
- Les formules sont formées à partir du processus habituel et des connecteurs logiques, **et**, **ou**, **non**, **exist**, **pourtout**,...
- Une requête est une expression de la forme { x | F(x)}

C. Delobel, V. Benzaken, E.Waller

Exemple

nom des monuments de la ville de Paris ?

{ u | **exist** y, x (Mom (x) **et** x.nom = u **et** y = x.adresse **et** y.ville = ’Paris’) }

restaurants situés dans la même rue que le musée d'Orsay et ayant un menu
à moins de 100 francs

{ x | **exist** y, m (Mom(y) **et** Res(x) **et** x.adresse = y.adresse **et**
y.nom = 'Musée Orsay' **et** m **in** x.menus **et** m.prix Š 1000 }

C. Delobel, V. Benzaken, E.Waller

Théorème d'équivalence

- Si de l'algèbre on retire l'opérateur **powerset** et ajoute **nest** on obtient une algèbre moins puissante notée ALG $^{-}$
- Théorème : ALG $^{-}$ et CAL sont équivalents

Calcul à objets

- Les racines de persistance peuvent être vues comme une valeur complexe à condition de considérer les identificateurs d'objet comme aassociés à un type particulier.
- Attention il faut prendre en compte le cas des racines qui ne font référence qu'à un seul objet.
- Les requêtes du calcul à objets sont construites sur les mêmes règles que le calcul à valeurs complxes

Déréférençage

- Dans le modèle à objets Val(O) définit l'esmble des valeurs possibles construites à partir de O.
- Etant donné un objet o, on doit pouvoir parler de sa valeur (l'état interne de o). Le passage de o à sa valeur s'appelle le déréférençage de o. L'opération de déréférençage est notée ^.

C. Delobel, V. Benzaken, E.Waller

Exemple de calcul à objets

exist x,y (x **in** MesActeursPreferes **et** y **in** ActeursPreferesduPublic **et**
x = y)

exist y (y **in** ActeursPreferesduPublic **et** y = RealisateurPrefere)

{ x | x = Realisateur }

{ y | **exist** x (x **in** MesActeursPreferes **et** x **in** ActeursPreferesduPublic **et**
x^. nom = y)

{ t | **exists** f, x (x= RealisateurPrefere **et** f **in** x^.film et f^.titre = t }

Introduction des méthodes

nom de mes acteurs préférés ?

{ y | **exist** x (x **in** MesActeursPreferes **et** y = x.lire_nom() }

C. Delobel, V. Benzaken, E.Waller

Conclusion

- Les langages de requêtes pour les modèles à objets sont des extensions de l'algèbre et du calcul pour les modèles à valeurs complexes.
- Ces langages sont trop théoriques pour déboucher sur des langages opérationnels. C'est pour cela que OQL a été développé à partir de principes simples.

C. Delobel, V. Benzaken, E.Waller