Object-Oriented Database Evolution

Jean-Bernard Lagorce, Arūnas Stočkus,
Emmanuel Waller
Paris-Orsay

ICDT'97

Context

Object-Oriented:

- Object identity
- Inheritance and attribute redefinition

Database:

- Schema:
  - classes (C)
  - inheritance links ($\leq$)
  - attribute definitions (Σ)
- Instance:
  1. class populations (ν)
  2. references (μ)
- Typing constraints

Motivation

Evolution = specification of a new state of the database from the current one

$$(C, \leq, \Sigma)(\nu, \mu) \mapsto (C', \leq', \Sigma')(\nu', \mu')$$

Evolution language = combination of schema and instance update languages

The database programmer should be provided with a tool allowing to describe the evolutions

Existing approaches do not offer:

- Programming language for evolution of the database (schema + instance)
- Consistency checking for this language

Instance Update, Syntax

Primitives:

- $move\ x\ c$
- $new\ x\ c$ $\qquad$ $delete\ x$
- $set\ x\ a\ x'$ $\qquad$ $cut\ x\ a\ x'$

Program over a schema S =
set of primitive calls

Instance Update, Semantics

- Program over an instance I = the variables are instanciated with objects in I

- First order formula associated to each instruction of program P

  - $new\ x_1\ c$ or $move\ x_1\ c$:
    $\forall z\ (x_1(z) \rightarrow$
    $(c(z) \wedge \bigwedge_{c' \in C - \{c\}} \neg c'(z)))$
  - $cut\ x_1\ a\ x_2$:
    $\forall z\ (x_1(z) \rightarrow$
    $\forall z'\ (x_2(z') \rightarrow \neg a(z, z')))$

$\psi_{S,P}$ = conjunction of the formulas associated to each instruction

Semantics of a Program

Given $\psi_{S,P}$ and an instance I of S

- What is not affected by the program is $invariant(I) =$ $Max\{I' \mid I' \subseteq I$ and $\exists I'' \supseteq I'$ s.t. $(I'', v) \models \psi\}$
- the semantics of P over I with the parameters is $Min\{I' \mid invariant(I) \subseteq I'$ and $(I', v) \models \psi\}$

Consistency

Inconsistent instance:

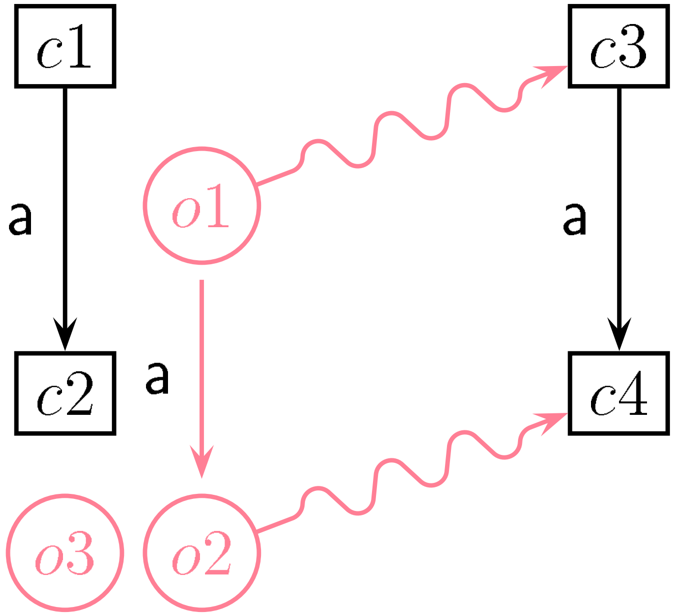
- objects belong to one and only one class
- no ill-typed references
- no undefined values

A program is inconsistent iff there exists an instance and an instantiation s.t. the semantics of the program is inconsistent

Consistency of a program is decidable

Problem

- Function migrating object in $c1$ to $c3$ and object referred through a to $c4$



$\{x_1 : c1;\ x_2 : c2\}$
$\{move\ x_1\ c3, move\ x_2\ c4\}$
is inconsistent

- State that $x_2 = x_1.a$

- Path defined from a typed variable x, $l.a$, $l.a^{-1}$, $l : c$, $l \cap l'$, $l \cup l'$, $l - l'$

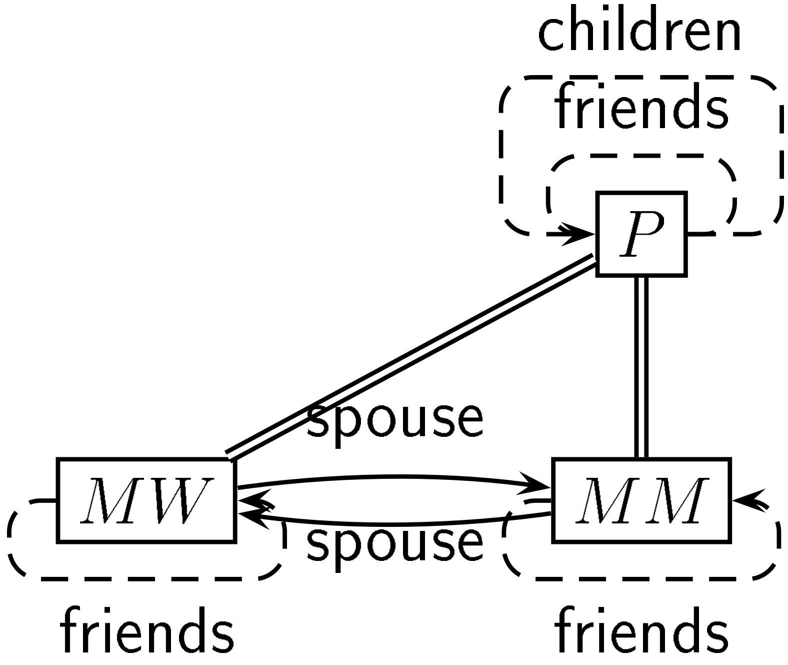
Instance Update using Paths

- FO2 = First Order
  - with two variables
  - no function symbol
- Satisfiability of path expressions is decidable
- Concrete program = variables replaced by paths
- expresses more functions
- consistency of a concrete program is decidable

Instance Update Example

- Static aspect Real World:
  Persons = not-married + married men + married women
  Attributes = friends, children, spouse
  Constraints = "my friends are like me"

- Static aspect Database



Instance Update Program

```
$\{$
$move\ xx\ MW,$
$move\ xy\ MM,$
$set\ xx\ spouse\ xy,$
$set\ xy\ spouse\ xx,$
$cut\ xx\ friends$
      $(xx.friends - (xx.friends : MW)),$
$cut\ xy\ friends$
      $(xy.friends - (xy.friends : MM))$
$\}$
```

This program is consistent

It will be used many times. There are people who marry every day using this procedure.

Schema update

Syntax

- primitives:
  - classes: $+c$, $-c$
  - inheritance links: $+(c, c')$, $-(c, c')$
  - attribute definitions: $+d$, $-d$
- program $=$ set of primitive calls

Semantics

- Trivial semantics associated to each instruction
- Semantics of program $=$ combination of semantics of each primitive of the program

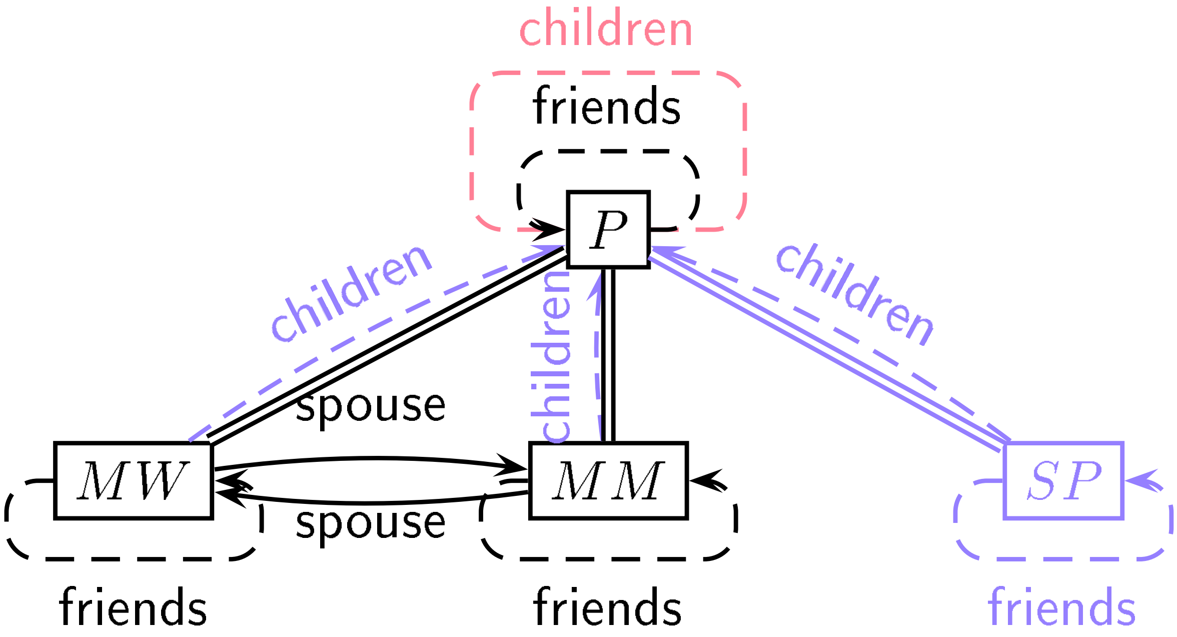
Database Evolution

Schema S, Instance I

- Evolution program $=$ instance update program P_I+ schema update program P_S
- Consistency checking of P_I in $S \cup P_S(S)$
- P_I performed over I in schema $S \cup P_S(S)$

consistency is decidable

Evolution Example



$\{\ +_c\ SP,\ \ +_a\ children : SP \to P,$
$+_a\ children : MW \to P,\ \ -_a\ children\ P \to P,$
$+_a\ children : MM \to P,$
$+_a\ friends : SP \to SP,\ \ +_h\ SP\ P\}$

$z =$ all the objects in the database
$singleparents =$
$z.children^{-1} - (z.children^{-1} :$
$MW \cup z.children^{-1} : MM)$

P $= \{move\ singleparents\ SP,$
$cut\ singleparents\ friends\ (z - singleparents)\}$

Conclusion

- Statically typed instance update language featuring object migration
- Evolution mechanism

Future Works

- Evolution usable with different instance update languages by isolating some specific parts of the language that allow consistency checking
- O2 prototype, extension in progress

References

[BKKK87] J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and implementation of schema evolution in object-oriented databases. In *SIGMOD*, 1987.

[FMZ$^+$95] F. Ferrandina, T. Meyer, R. Zicari, G. Ferran, and J. Madec. Schema and database evolution in the o2 object database system. In *VLDB*, pages 170–181, 1995.

[MMW94] A. Mendelzon, T. Milo, and E. Waller. Object migration. In *PODS*, 1994.