Navigation in Object Graphs

Mitch Wand
and
Karl Lieberherr
Searching for Reachable Objects

• Task: Given an object o1 of class c1 in an object graph, find all objects of type c2 that are reachable from o1.

• Assumptions: we know the class structure that describes the object graph, but we know nothing else about the object graph except the class of the current object.
Search using meta information

• we could visit the entire object but that
  – would be wasteful or
  – might lead to wrong results
Classes and Objects: Basic Notations

Class c1 has a part e of type c2

Class c1 inherits from class c2

Object o1 is of class c1

Object o1 is of type c2 (i.e., its class is a subclass of c2)

Object o1 has a part e which is object o2
Finding the first step for the search

Which arrows might lead to an object of type C2?

Traversal Strategy: (// C1 C2)
Relations between Classes

- \( e(C_1, C_2) \)
- \( C(C_1, C_2) \) (that is, \( e(C_1, C_2) \) for some \( e \))
- \( \leq(C_1, C_2), \geq(C_2, C_1) \)
- Object \( o_1 \) is of type \( C_2 \): \( \text{Class}(o_1) \leq C_2 \)
Relations between Objects

\[ \text{o1: } \xrightarrow{e} \text{ o2: } \]

\( e(\text{o1, o2}) \)

\[ \text{o1: } \xrightarrow{} \text{ o2: } \]

\( O(\text{o1, o2}) \) (that is, \( e(\text{o1, o2}) \) for some \( e \))
Operations on Relations

- \( R.S = \{(x,z) \mid \text{exists } y \text{ s.t. } R(x,y) \text{ and } S(y,z)\}\)
- \( R^* = \text{reflexive, transitive closure of } R \)
Write graph in terms of relations

Set: \{S,T,X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3\}

Relations are sets of pairs, ordering is irrelevant.

Relations:
- x,X,Y,Z,T,<=,=>
- x(S,X1)
- y(X2,Y2)
- z(Y1,Z1)
- t(Z2,T)
- <=(Y2,Y1)
- <=(X2,X1)
- <=(X3,X2)
- =>(X2,X3)
- =>(X1,X2)
- =>(Z1,Z2)
- ...
Possible edges in the object graph

\[ e(o_1, o_2) \implies \text{class}(o_1) (\leq .e .\geq ) \text{class}(o_2) \]

in the class graph

“up, over, and down”

\[ O(o_1, o_2) \implies \text{class}(o_1) (\leq .C .\geq ) \text{class}(o_2) \]

in the class graph
Which edges to follow to C2?

From o1 of class C1, follow edge e iff there is some object graph O and some o2, o3 s.t.

1. e(o1,o2),
2. O*(o2,o3), and
3. class(o3) <= C2

The existential quantifier “there is some object graph” represents our lack of knowledge about the rest of the object graph.
from Basket to Orange = (from-to Basket Orange) = (// Basket Orange)

Example

class graph
premature termination
object graph

object graph slice
Example

(mapping:
  o1 b1
  o2 a1
  o3 a1
  e f

class graph)

(object graph)

(object graph slice)
Example B1

strategy

from A via T to D =
(join (// A T)
  (// T D))

class
graph

object
graph
Example B1

strategy from A via T to D = (join (// A T) (// T D))

class graph

object graph slice

Navigation in Object Graphs 15
Example B2

strategy from A via T to D =
(join (// A T)
 (// T D))
Example B2

strategy from A via T to D=
(join (/\ A T)
(\ T D))

object graph slice

class graph

Navigation in Object Graphs
Lack of Knowledge

• Objects of a given class may be very different.
• We want to go down edges without looking ahead!
• We don’t want to go down edges that are guaranteed to be unsuccessful (never reaching a target object).
Object graph conforms to class graph

• The object graph $O$ must follow the rules of the class graph: the object graph cannot contain more information than the class graph allows.

For all edges $e(o_1,o_2)$ in the object graph: $e(o_1, o_2)$ implies
\[ \text{class}(o_1) \quad (\leq .e .\geq ) \quad \text{class}(o_2) \]
in the class graph
From dynamic to static characterization

From o₁ of class c₁, follow edge e iff there is some object graph O and some o₂, o₃ s.t.
1. e(o₁,o₂),
2. O*(o₂,o₃), and
3. class(o₃) <= c₂

Let c’ be class(o₂), c’’ be class(o₃)

From o₁ of class c₁, follow edge e iff there are classes c’, c’’ s.t.
1. c₁ <=.e.=> c’
2. c’ (<=.C.=>)* c’’ and
3. c’’ <= c₂
from Basket to Orange

Example

static characterization

class graph

premature termination

object graph

object graph slice
Example

from Basket to Orange

static characterization

diagram:
c1 Basket
c’ Orange
c” Orange
e f
class graph

object graph slice

Basket --> Fruit --> Orange

Basket --> Vegetable

Apple

b1:Basket

a1:Orange

v1:Vegetable

object graph
from x1 to x2

Relational Formulation

From object o of class x1, to get to x2, follow edges in the set
POSS(x1,x2)={e | x1 <=.e.=>.(<=.C.=>)*.<= x2 }

Can easily compute these sets for every x1, x2 via transitive-closure algorithms.
POSS = abbreviation for: following these edges it is still possible to reach a x2-object for some x1-object rooted at o.
Relational Formulation

From object $o$ of class $x_1$, to get to $x_2$, follow edges in the set

$$\text{POSS}(x_1,x_2) = \{ e | x_1 \leq .e.=>.(\leq .C.=>)^* .\leq x_2 \}$$

Simplification for class graphs obtained from hw class graphs:

$$\text{POSS}(x_1,x_2) = \{ e | x_1 e.=>.(C.=>)^* .\leq x_2 \} \text{ (flat)}$$

up-over-and-down becomes over-and-down

$=>$ means only one edge

$\leq$ means only one edge
Negative formulation

Positive Formulation:
From object o of class x1, to get to x2, follow edges e in the set
POSS(x1,x2)={e | x1 <=.e.=>.(<=.C.=>)*.<= x2 }

Build paths anyway you like but don’t follow => (down)
immediately after <= (up) and the first has-a edge must
have label e.

Edge kinds:
Is-a (down, up)
Has-a

Forbidden

ok
up
down

11/1/2005 Navigation in Object Graphs 25
Followed-by relationships

<table>
<thead>
<tr>
<th>Followed by</th>
<th>Is-a: down</th>
<th>Is-a: up</th>
<th>Has-a e, C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is-a: down</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Is-a: up</td>
<td>NO</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Has-a</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Linking the terminologies

<table>
<thead>
<tr>
<th>program</th>
<th>theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>-&gt; A, b, B</td>
<td>Has-a</td>
</tr>
<tr>
<td>=&gt; A, B</td>
<td>Is-a in</td>
</tr>
<tr>
<td>=&gt; A, B</td>
<td>reverse</td>
</tr>
<tr>
<td>=&gt; A, B</td>
<td>Is-a</td>
</tr>
<tr>
<td>&lt;= A, B</td>
<td>up</td>
</tr>
</tbody>
</table>
Generalizations

• More complex strategies
• “from c1 through c2 to c3”
  – Use “waypoint navigation”; get to a c2 object, then search for a c3 object.
• More complex strategy graphs also doable in this framework
Examples
Example A

Strategy
\[ S \rightarrow T \]

Relations:
\[ x(S,X1) \]
\[ y(X1,X2) \]
\[ z(Y1,Z1) \]
\[ t(Z1,Z2) \]
\[ y(X2,Y2) \]
\[ z(X2,Y1) \]
\[ y(X3,Y3) \]
\[ t(Y3, Z3) \]

\[ x(S,X1) \rightarrow \] 
\[ y(X1,X2) \rightarrow \] 
\[ z(Y1,Z1) \rightarrow \] 
\[ t(Z1,Z2) \rightarrow \] 

go down x iff S and T are in relation: \[ \leq . x. \rightarrow . (\leq . c. \rightarrow )\ast . \leq \]
Example A

Strategy
\[ S \rightarrow T \]

Relations:
- \( x(S,X1) \)
- \( \Rightarrow(X1,X2) \)
- \( y(X2,Y2) \)
- \( \leq(Y2,Y1) \)
- \( z(Y1,Z1) \)
- \( \Rightarrow(Z1,Z2) \)
- \( t(Z2,T) \)
- \( \leq(X2,X1) \)
- \( \ldots \)

\( S \leq . x . \Rightarrow . (\leq . C . \Rightarrow )^* . \leq . T \)

go down \( x \) iff \( S \leq . x . \Rightarrow . (\leq . C . \Rightarrow )^* . \leq . T \)
**Example A**

**Strategy**

\[ S \rightarrow T \]

**Relations:**
- \( x(S,X1) \)
- \( =>(X1,X2) \)
- \( y(X2,Y2) \)
- \( <=(Y2,Y1) \)
- \( z(Y1,Z1) \)
- \( =>(Z1,Z2) \)
- \( t(Z2,T) \)
- \( <=(X2,X1) \)
...

**Navigation Rule:**

\[ go \ down \ x \ iff \ \text{S} \leq .x. \Rightarrow (\leq .C. \Rightarrow .\leq .C. \Rightarrow .\leq .C. \Rightarrow ).\leq T \]

---

11/1/2005 Navigation in Object Graphs
Just in terms of relations

Relations are sets of pairs, ordering is irrelevant.

Set: \{S,T,X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3\}
Write graph in terms of relations

Set: \{S,T,X_1,X_2,X_3,Y_1,Y_2,Y_3,Z_1,Z_2,Z_3\}

Relations are sets of pairs, ordering is irrelevant.

Relations:
- x,y,z,t,\leq,\Rightarrow
- x(S,X_1)
- y(X_2,Y_2)
- z(Y_1,Z_1)
- t(Z_2,T)
- \leq(Y_2,Y_1)
- \leq(X_2,X_1)
- \leq(X_3,X_2)
- \Rightarrow(X_2,X_3)
- \Rightarrow(X_1,X_2)
- \Rightarrow(Z_1,Z_2)
- \ldots
Set: {S,T,X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3}

Compose relations

Relations are sets of pairs, ordering is irrelevant.

Relations: x,y,z,t,<=,=>

x(S,X1)
y(X2,Y2)
z(Y1,Z1)
t(Z2,T)
<= (Y2,Y1)
<= (X2,X1)
<= (X3,X2)
...
=> (X2,X3)
=> (X1,X2)
=> (Z1,Z2)
...

Set: \{S,T,X_1,X_2,X_3,Y_1,Y_2,Y_3,Z_1,Z_2,Z_3\}

**Just in terms of relations**

*Relations are sets of pairs, ordering is irrelevant.*

Relations: x, y, z, t, <=, =>

\[
\begin{align*}
x(S,X_1) \\
y(X_2,Y_2) \\
z(Y_1,Z_1) \\
t(Z_2,T) \\
\leq(Y_2,Y_1) \\
\leq(X_2,X_1) \\
\leq(X_3,X_2) \\
\ldots \\
\Rightarrow(X_2,X_3) \\
\Rightarrow(X_1,X_2) \\
\Rightarrow(Z_1,Z_2) \\
\ldots
\end{align*}
\]

The order in which we consume the pairs

\[
\begin{align*}
x(S,X_1) \\
\Rightarrow(X_1,X_2) \\
y(X_2,Y_2) \\
\leq(Y_2,Y_1) \\
z(Y_1,Z_1) \\
\Rightarrow(Z_1,Z_2) \\
t(Z_2,T)
\end{align*}
\]

**go down x iff S \leq x. \Rightarrow(\leq C.\Rightarrow\leq C.\Rightarrow\leq C\Rightarrow).\leq T**
How big is the relation?

How many pairs does the relation contain in this example?

\[
\leq . x. \Rightarrow . (\leq . C. \Rightarrow \leq . C. \Rightarrow . \leq . C\Rightarrow). \leq
\]
What is disallowed?

Where can you go from A?

go down x iff S and T are in relation: $\leq . x \Rightarrow (\leq . C . \Rightarrow )^* . \leq$
class dictionary
A = ["x" X] ["r" R].
B = ["b" B] D.
R = S.
S = ["t" T] C
C = D.
X = B.
T = R.
D = .

0..1

strategy
A -> T
T -> D

POSS(x1,x2)={e | x1 e.C* x2 }

Example B

class graph

object graph "r"
class dictionary
A = ["x" X] ["r" R].
B = ["b" B] D.
R = S.
S = ["t" T] C
C = D.
X = B.
T = R.
D = .

strategy
A -> T
T -> D
POSS(A,T) = 1 edge
POSS(R,T) = 1 edge
POSS(S,T) = 0 edges

Example B1

class graph

POSS(c1,c2)={e | c1 e.C* c2 }

object graph "r"
Example B1

strategy

A -> T
T -> D

POSS(A,T) = 1 edge
POSS(R,T) = 1 edge
POSS(S,T) = 0 edges

class dictionary

A = ["x" X] ["r" R].
B = ["b" B] D.
R = S.
S = ["t" T] C
C = D.
X = B.
T = R.
D = .

POSS(c1,c2) = \{ e | c1 e C* c2 \}

object graph slice

object graph "r"
Example B2

POSS(A,T) = 1 edge
POSS(R,T) = 1 edge
POSS(S,T) = 1 edge
POSS(T,D) = 1 edge
POSS(R,D) = 1 edge

\[
\text{POSS}(c_1,c_2) = \{ e \mid c_1 e . C^* c_2 \}
\]

\[a_1:A\]
\[r_1:R\]
\[s_1:S\]
\[t_1:T\]
\[c_1:C\]
\[d_2:D\]
\[c_2:C\]

strategy

A → T
T → D

object graph slice

class graph

Navigation in Object Graphs
Example B2

POSS(A,T) = 1 edge
POSS(R,T) = 1 edge
POSS(S,T) = 1 edge
POSS(T,D) = 1 edge
POSS(R,D) = 1 edge

strategy
A -> T
T -> D

object graph slice
POSS(c1,c2) = \{ e | c1 e C* c2 \}

class graph

0..1
Only node paths shown for space reasons

Example C

Object graph:

```
A
  /\ 
X  R  X
  |  |
X2:X
  |  |
C
```

Class graph:

```
A
  /\ 
R  S  B
  |  |
X
```

**Example C**

**Object graph**

```
strategy SG:
{A -> B
 B -> C}
```

**Class graph**

```
strategy SG:
{A -> B
 B -> C}
```

**Object graph**

```
OG : A X  R X C
OG' : A X  B X C
SG : A      B  C
```

**Class graph**

```
BOpt
```

**Example C**
Write Java code that does the traversal

Example C

strategy:  {A -> B
            B -> C}

Object graph

class graph

Write Java code that does the traversal
Not the complete story

Example C

strategy: 
{A -> B 
  B -> C}
Not the complete story: traversal must look for further B

Example C

strategy: 
{A -> B
 B -> C}

Object graph

A

x1:X

R

c1:C

c2:C

c3:C

From A to B

From B to C

class graph

A

B

C

S

Empty

BOpt

X

C

11/1/2005

Navigation in Object Graphs
Not the complete story: traversal must look for further B

Example C

strategy:  
{A -> B  
   B -> C}  

Object graph

From A to B

First B found

e1:Empty

Premature Termination: No more B

class graph

Strategy

s          t

A -- B -- C

A -- B

B: Empty

X: BOpt

C: Empty
Example C1

strategy SG: 
{A -> s} 
s -> C

Object graph

Only node paths shown for space reasons

early termination

11/1/2005 Navigation in Object Graphs 49
$S = \text{from } \text{BusRoute} \text{ through } \text{Bus} \text{ to } \text{Person}$

Example D

Diagram:

- **BusRoute**
  - **buses** to **BusList**
  - **busStops** to **BusStopList**
  - **NGasPowered**
  - **waiting** to **PersonList**

- **BusList**
  - **0..*** to **Bus**
  - **0..** to **passengers**

- **Bus**
  - **buses** from **BusRoute**
  - **0..** to **passengers**

- **Person**
  - **0..*** to **waiting**
  - **0..*** to **PersonList**
  - **DieselPowered**

Navigation in Object Graphs
Example D1

Only node paths shown for space reasons

Route1:BusRoute \rightarrow \text{buses} \rightarrow \text{BusList}

\text{busStops} \rightarrow :\text{BusStopList}

CentralSquare:BusStop \rightarrow \text{waiting} \rightarrow :\text{PersonList}

Paul:Person \rightarrow \text{passengers} \rightarrow \text{Joan:Person, Seema:Person}

S = \text{from BusRoute through Bus to Person}

11/1/2005 Navigation in Object Graphs
Example D2

Only node paths shown for space reasons

$S = \text{from BusRoute via NGasPowered to Person}$
from c1 bypassing \{x_1, x_2, \ldots , x_n\} to c2

Relational Formulation

From object o of class c1, to get to c2, follow edges in the set

\[ \text{POSS}(c_1, c_2) = \{ e \mid c_1 \leq e \Rightarrow (\leq . C. \Rightarrow )^* \leq c_2 \} \]

POSS = abbreviation for: following these edges it is still possible to reach a c2-object for some c1-object rooted at o.

Delete x_1, x_2, \ldots , x_n and all edges incident with these nodes from the class graph (assumed to be different from c1, c2).
Example D

\[ S = \text{from BusRoute bybassing Bus to Person} \]
\[ S = \text{from BusRoute bybassing Bus to Person} \]

**Example D**

![Diagram showing object graph with relationships between BusRoute, BusStopList, PersonList, and Person.]

- **BusRoute** → **BusStopList** (buses)
- **BusStopList** → **PersonList** (waiting)
- **PersonList** → **Person** (0..*)
- **BusRoute** → **NGasPowered** (0..*)
- **BusRoute** → **DieselPowered**
Conclusions

• Programming language elements are mathematical objects having precise mathematical definitions.
• Exercise in applying an abstract algorithm to concrete inputs. Mapping abstract situations to concrete situations.
• Separation of concerns is also useful for defining programming language elements
  – separate subgraph selected from
  – how the subgraph is traversed (depth-first etc.)
• In earlier works: meaning of a traversal strategy for an object graph
  – was a traversal history
  – now it is a subgraph of the object graph. A traversal history can be defined ...