"Fuzzy morphisms between graphs" (Perchant & Bloch, 2002) and "Fuzzy graphs" (Rosenfeld, 1975)

Robin Eklind

Disposition

1. Primer *Fuzzy Graph Theory*
2. What? *Fuzzy Graph Morphisms*
3. Why? *Applications of Fuzzy Graph Morphisms*
4. Conclusions
5. Demo!

A Primer to Fuzzy Graph Theory

1. Fuzzy Logic
2. Fuzzy Graph Theory

Fuzzy Logic

Definitions from "Fuzzy graphs" (Rosenfeld, 1975).

Fuzzy subsets

Definition: A *fuzzy subset* of a set S is a mapping $\sigma : S \to [0,1]$ which assigns each element $x \in S$ a degree of membership $0 \leq \sigma(x) \leq 1$.

Examples:

- $\mathit{Deina} \in \mathit{old}$ with $\sigma(\mathit{Deina}) = 0.3$
    - The element *Deina* (woman at age 34) is a member of the set *old* to a degree of 0.3
- $\mathit{Foufoutos} \in \mathit{old}$ with $\sigma(\mathit{Foufoutos}) = 0.5$
    - The element *Foufoutos* (man at age 45) is a member of the set *old* to a degree of 0.5
- $\mathit{Tade} \in \mathit{old}$ with $\sigma(\mathit{Tade}) = 0.1$
    - The element *Tade* (boy at age 6) is a member of the set *old* to a degree of 0.1

Fuzzy Logic

Fuzzy relations

Definition: A *fuzzy relation* on a set S is a mapping $\mu : S \times S \to [0,1]$ which assigns each ordered pair $(x, y) \in S \times S$ a degree of membership $0 \leq \mu(x, y) \leq 1$. In other words, a fuzzy relation on S is a fuzzy subset of $S \times S$.

Examples:

- $(Deina, Tade) \in mothers$ with $\mu(Deina, Tade) = 1.0$, where $mothers : person \times person$
  - The ordered pair $(Deina, Tade)$ is a $(mother, child)$-relation on the set *person* to a degree of 1.0
- $(Foufoutos, Tade) \in fathers$ with $\mu(Foufoutos, Tade) = 0.5$, where $fathers : person \times person$
  - The ordered pair $(Foufoutos, Tade)$ is a $(father, child)$-relation on the set *person* to a degree of 0.5

Fuzzy Logic

Definitions from "Fuzzy morphisms between graphs" (Perchant & Bloch, 2002).

Fuzzy relation on $\sigma_1 \times \sigma_2$

Let $\sigma_1 : S_1 \to [0,1]$ and $\sigma_2 : S_2 \to [0,1]$ be fuzzy subsets of S_1 and S_2, respectively.
Definition: The function $\mu : S_1 \times S_2 \to [0,1]$ is a *fuzzy relation* on $\sigma_1 \times \sigma_2$ iff:

$$\begin{aligned} &\forall (x,y) \in S_1 \times S_2 : \\ &\mu(x,y) \leq \sigma(x) \wedge \sigma(y) \end{aligned} \tag{1}$$

Note: $\wedge$ denotes *minimum* (and $\vee$ denotes *maximum*).

Intuition: A fuzzy relation cannot be "stronger" than either of its elements.

Fuzzy Graph Theory

Fuzzy graph

Definition: A *fuzzy graph* $F = (\sigma, \mu)$ is a pair of functions $\sigma : S \to [0, 1]$ and $\mu : S \times S \to [0, 1]$ which satisfy equation 1 (of previous slide).

Intuition: For a given graph $G(V, E)$, a fuzzy graph $F(\sigma, \mu)$ has a fuzzy subset σ of V and a fuzzy subset μ of $V \times V$. Since equation 1 (of previous slide) is satisfied, a "fuzzy edge" cannot be stronger than either of its "fuzzy endpoint vertices".

Max-min composition

Let μ_1 and μ_2 be two fuzzy relations on $\sigma_1 \times \sigma_2$ and $\sigma_2 \times \sigma_3$, respectively.
Definition A *max-min composition* of μ_1 and μ_2, denoted by $\mu_1 \circ \mu_2$, is defined as follows.

$$\begin{aligned} &\forall (u_1, u_3) \in S_1 \times S_3, (u_1 \circ u_2)(u_1, u_3) \\ &= \sup_{u_2 \in S_2} \{\mu_1(u_1, u_2) \wedge \mu_2(u_2, u_3)\} \end{aligned} \tag{2}$$

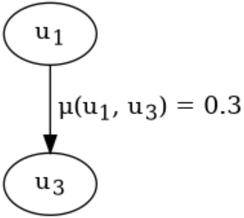
Note: sup (supremum) denotes *least upper bound* (and $\wedge$ denotes *minimum*).

Intuition: select the best "middle" vertex u_2 such that a path from u_1 to u_2 and u_2 to u_3 has the highest "quality", as measured by the lowest edge weight $\mu(x, y)$ along that path.

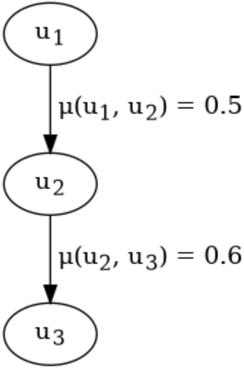
Fuzzy Graph Theory

Max-min composition

Going through u_2 improves "quality" of path from $\mu(u_1, u_3) = 0.3$ to $(u_1 \circ u_2)(u_1, u_3) = 0.5$.



(a) Path from u_1 to u_3.



(b) Path from u_1 through u_2 to u_3.

Figure: Find better "quality" path using max-min composition.

What?

(Sub)graph isomorphism requires a *bijective* mapping to match vertices and edges of two graphs. In other words, there may exist no missing or additional vertices or edges.

The real world is not ideal, as such we are often interested in *inexact* graph matches.

Fuzzy graph morphisms are used to formalize inexact graph matches.

Fuzzy Graph Morphisms

Fuzzy Graph Morphisms

Definition: A *fuzzy morphism* (ρ_σ, ρ_μ) between graphs G_1 and G_2 is a pair of mappings $\rho_\sigma : N_1 \times N_2 \to [0,1]$ and $\rho_\mu : N_1 \times N_2 \times N_1 \times N_2 \to [0,1]$ which satisfy:

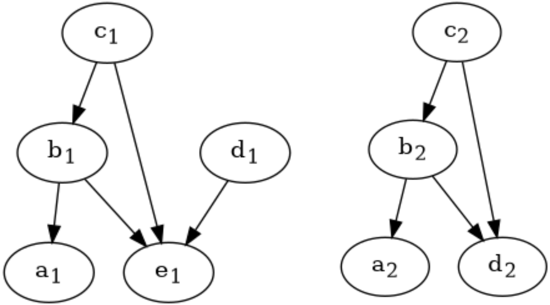
$$\begin{aligned} &\forall (u_1, v_1) \in N_1 \times N_1, \forall (u_2, v_2) \in N_2 \times N_2 : \\ &\rho_\mu(u_1, u_2, v_1, v_2) \leq \rho_\sigma(u_1, u_2) \wedge \rho_\sigma(v_1, v_2) \end{aligned} \tag{3}$$

Corollary: A fuzzy morphism (ρ_σ, ρ_μ) is a fuzzy graph, and equation 3 is analogous to equation 1.

The mapping ρ_σ is called a *vertex morphism* and ρ_μ a *edge morphism*.

Fuzzy Graph Morphisms

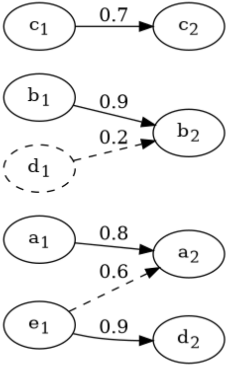
Example:



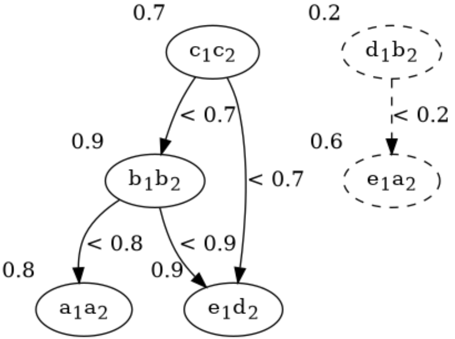
(a) G_1. (b) G_2.

Figure: Example graphs G_1 (left) and G_2 (right).

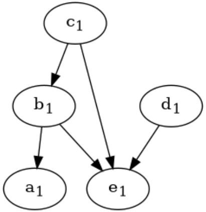
Internal Representation of Fuzzy Graph Morphisms



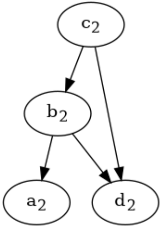
(a) Vertex morphism ρ_σ.



(b) Edge morphism ρ_μ.



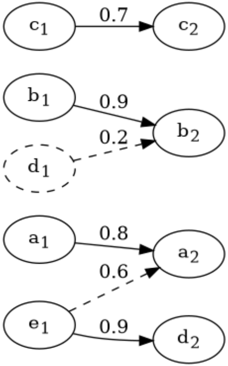
(c) G_1.



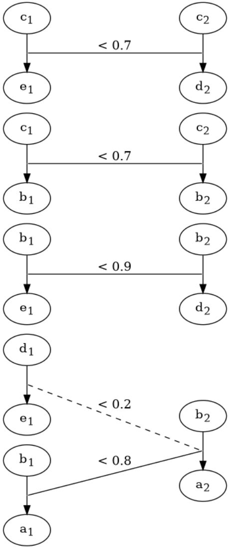
(d) G_2.

Figure: Internal representation of fuzzy graph morphism $\rho(\sigma, \mu)$.

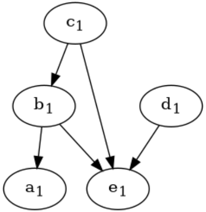
External Representation of Fuzzy Graph Morphisms



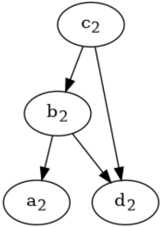
(a) Vertex morphism ρ_σ.



(b) Edge morphism ρ_μ.



(c) G_1.



(d) G_2.

Figure: External representation of fuzzy graph morphism $\rho(\sigma, \mu)$.

Why?

Applications of Fuzzy Graph Morphisms

- Binary analysis
  - Function identification by matching fuzzy graph morphisms of callgraphs, control flow and data flow graphs.
- Pattern recognition
- Image analysis
  - Fuzzy graph morphisms of regions (vertices) and relation between regions (edges).
- Recognition of brain structures
  - Fuzzy graph morphisms of neurons (vertices) and synapses (edges)

Conclusions

1. Significance of papers
2. Relation to other research

Significance of papers

Rosenfeld was among the first to formalized fuzzy graph theory in the seminal paper "Fuzzy graphs" (1975), which has since received 910 citations (and it also cited in "Fuzzy morphisms between graphs").

Perchant and Bloch established a formalism for inexact graph matchings in "Fuzzy morphisms between graphs". The paper remains less influential with 61 citations.

Relation to other research

Prior-art

Rosenfeld was a leading researcher of image analysis and essentially established the field; among others writing the first textbook on the topic ("Picture Processing by Computer" in 1969, later followed by "Digital picture processing" in 1976, a book with 8547 citations).

Similarly, Perchant and Bloch have a background in image analysis.

Follow-up research

Bloch went on to applied research related to cognitive science and have among others co-authored the paper "Diffeomorphic demons: Efficient non-parametric image registration", which has received 1106 citations to date.

In general, follow-up research seem to be more oriented towards application rather than theory.

Demo

Time for a demo!

Function identification through fuzzy callgraph matching

Example: source program compiled for three different machine architectures.

```
.text:00447558 ; =============== S U B R O U T I N E =====================
.text:00447558
.text:00447558
.text:00447558 ; void __fastcall OperateStoryBook(int pnum, int i)
.text:00447558 OperateStoryBook proc near              ; CODE XREF: Operat
.text:00447558                                         ; SyncOpObject:loc_
.text:00447558                 push    esi
.text:00447559                 push    edi
.text:0044755A                 mov     edi, edx
.text:0044755C                 xor     eax, eax
.text:0044755E                 mov     esi, edi
.text:00447560                 imul    esi, 78h ; 'x'
.text:00447563                 cmp     byte ptr object._oSelFlag[esi], al
.text:00447569                 jz      short loc_4475B8
.text:0044756B                 cmp     deltaload, eax
.text:00447571                 jnz     short loc_4475B8
.text:00447573                 cmp     qtextflag, al
.text:00447579                 jnz     short loc_4475B8
.text:0044757B                 cmp     ecx, myplr
.text:00447581                 jnz     short loc_4475B8
.text:00447583                 push    object._oy[esi] ; y
.text:00447589                 mov     eax, object._oVar4[esi]
.text:0044758F                 mov     edx, object._ox[esi] ; x
.text:00447595                 mov     object._oAnimFrame[esi], eax
```

(a) x86 assembly.

```
ROM:8005D7B8  # =============== S U B R O U T I N E ==============
ROM:8005D7B8
ROM:8005D7B8
ROM:8005D7B8  # void __cdecl OperateStoryBook(int pnum, int i)
ROM:8005D7B8 OperateStoryBook__Fii:                    # CODE XREF:
ROM:8005D7B8                                           # SyncOpObje
ROM:8005D7B8
ROM:8005D7B8 var_10          = -0x10
ROM:8005D7B8 var_C           = -0xC
ROM:8005D7B8 var_8           = -8
ROM:8005D7B8
ROM:8005D7B8                 addiu   $sp, -0x20
ROM:8005D7BC                 sw      $s1, 0x20+var_C($sp)
ROM:8005D7C0                 move    $s1, $a1
ROM:8005D7C4                 sll     $v0, $s1, 1
ROM:8005D7C8                 addu    $v0, $s1
ROM:8005D7CC                 sll     $v0, 2
ROM:8005D7D0                 subu    $v0, $s1
ROM:8005D7D4                 sw      $s0, 0x20+var_10($sp)
ROM:8005D7D8                 sll     $s0, $v0, 2
ROM:8005D7DC                 sw      $ra, 0x20+var_8($sp)
ROM:8005D7E0                 lb      $v0, object._oSelFlag($s0)
ROM:8005D7EC                 nop
ROM:8005D7F0                 beqz    $v0, loc_8005D894
```

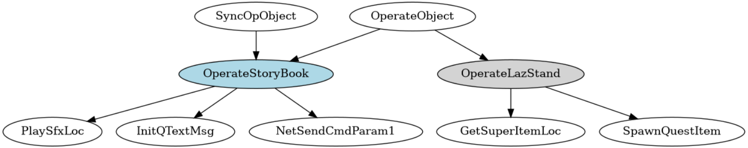
(b) MIPS assembly.

```
seg000:0008B9C4 # =============== S U B R O U T I N E =================
seg000:0008B9C4
seg000:0008B9C4
seg000:0008B9C4 sub_8B9C4:                                 # CODE XREF: su
seg000:0008B9C4                                            # sub_8C0A0+104
seg000:0008B9C4
seg000:0008B9C4 .set sender_sp, -0x40
seg000:0008B9C4 .set var_8, -8
seg000:0008B9C4 .set var_4, -4
seg000:0008B9C4 .set sender_lr,  8
seg000:0008B9C4
seg000:0008B9C4                 stw     r31, var_4(r1)
seg000:0008B9C8                 mflr    r0
seg000:0008B9CC                 stw     r30, var_8(r1)
seg000:0008B9D0                 mr      r30, r4
seg000:0008B9D4                 mulli   r4, r30, 0x78 # 'x'
seg000:0008B9D8                 stw     r0, sender_lr(r1)
seg000:0008B9DC                 lwz     r0, off_F1948 # dword_17E3D4
seg000:0008B9E0                 stwu    r1, sender_sp(r1)
seg000:0008B9E4                 add     r31, r0, r4
seg000:0008B9E8                 lbz     r0, 0x40(r31)
seg000:0008B9EC                 extsb.  r0, r0
seg000:0008B9F0                 beq     loc_8BA54
seg000:0008B9F4                 lwz     r4, off_F1F60 # dword_1795D0
```

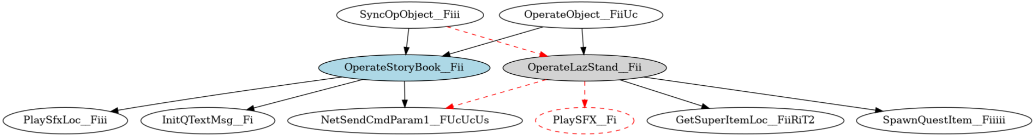
(c) PowerPC assembly.

Figure: Machine code of the same source function in x86 (left), MIPS (middle) and PowerPC (right) assembly.

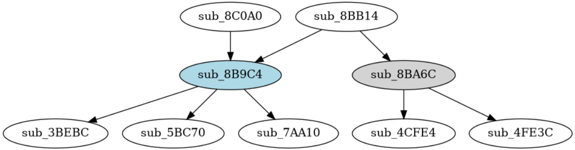
Function identification through fuzzy callgraph matching



(a) x86 callgraph.



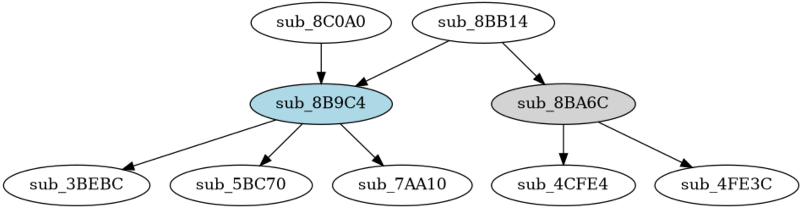
(b) MIPS callgraph.



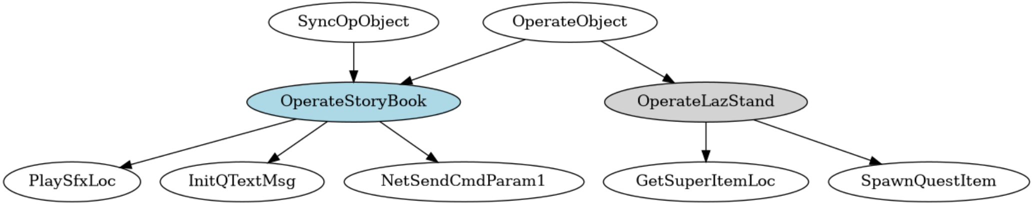
(c) PowerPC callgraph.

Figure: Corresponding callgraphs for x86 (top), MIPS (middle) and PowerPC (bottom) assembly.

Function identification through fuzzy callgraph matching



(a) MIPS callgraph *before* fuzzy callgraph matching.



(b) MIPS callgraph *after* fuzzy callgraph matching.

Figure: Results of fuzzy callgraph matching used to assign names to unknown functions.

Function identification through fuzzy callgraph matching

The example source program has a callgraph with 2551 vertices and 6400 edges, illustrating the need for performant algorithms and the benefit of automating function identification.

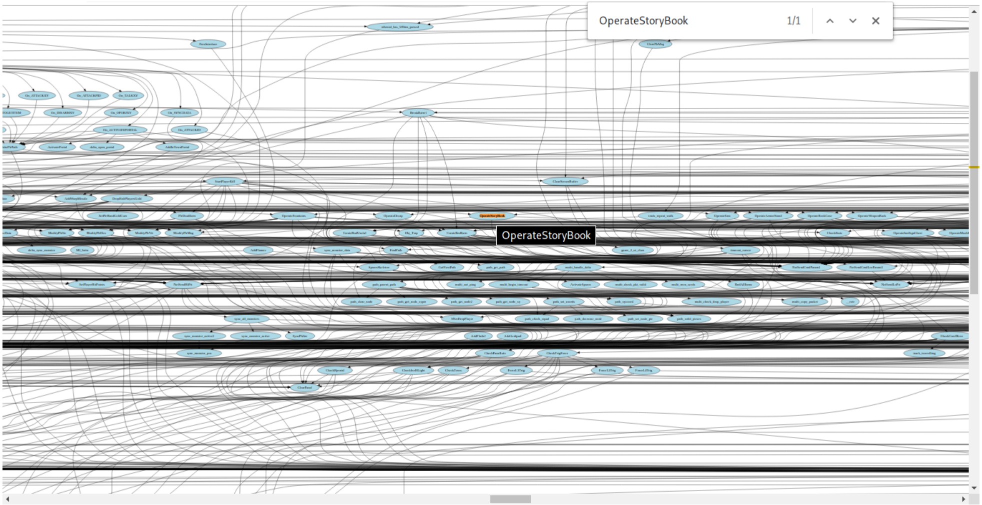


Figure: An extract of the callgraph of the source program.

Function identification through fuzzy callgraph matching

For reference, this is the original source code of the C source function `OperateStoryBook`.

```c
void OperateStoryBook(int pnum, int i)
{
	if (object[i]._oSelFlag != 0 && !deltaload && !qtextflag && pnum == myplr) {
		object[i]._oAnimFrame = object[i]._oVar4;
		PlaySfxLoc(IS_ISCROL, object[i]._ox, object[i]._oy);
		InitQTextMsg(object[i]._oVar2);
		NetSendCmdParam1(FALSE, CMD_OPERATEOBJ, i);
	}
}
```

Figure: Original source code of the C source function `OperateStoryBook`.