Category Theory 101
Graph Transformations
Discrete Structures Day

F. Prost

PLUME team, LIP - ENS-Lyon

17th of December 2015
Introduction

- High level approach to programming: graph rewriting based on category theory.

- Much more difficult than term rewriting (which are just trees).
Introduction

- High level approach to programming: graph rewriting based on category theory.

- Much more difficult than term rewriting (which are just trees).

- Simulation of biological phenomena.

- Simulation of chemical reactions.

- Study of cloning:
  - Typically to produce a web site one starts to copy an existing one, then one modifies it accordingly to its will.
  - Social Data Anonymization techniques rely on finely tuned cloning operations.
Plan

1. Category Theory 101
2. Graph transformation and Categories
3. AGREE and Graph Generation
   - AGREE and Data Anonymization
   - Self-similar Graphs
4. Conclusion
Plan

1. Category Theory 101

2. Graph transformation and Categories

3. AGREE and Graph Generation
   - AGREE and Data Anonymization
   - Self-similar Graphs

4. Conclusion
Category Theory

- Early 40’s by MacLane and Eilenberg with a unifying aim: topology and algebra.

What are the fundamental structures of those two fields?
Category Theory

- Early 40’s by MacLane and Eilenberg with a unifying aim: topology and algebra.

⇒ What are the fundamental structures of those two fields?

- Results much more general than thought at first.

- Set theory is just a special case of category (Lawvere).

- In computer science E. Moggi was able to capture ideas previously thought to be outside of reach with the monads.

- In logic J.-Y. Girard and the linear logic.

- etc.
Definition

A category $\mathcal{C}$ is made of

- A collection of objects: $\text{Obj}(\mathcal{C})$
- $\forall x, y \in \text{Obj}(\mathcal{C})$ a set $\text{Hom}_{\mathcal{C}}(x, y)$
- $\forall x \in \text{Obj}(\mathcal{C})$ there is $\text{id}_x \in \text{Hom}_{\mathcal{C}}(x, x)$
- $\forall x, y, z \in \text{Obj}(\mathcal{C})$ a function
  $\circ : \text{Hom}_{\mathcal{C}}(x, y) \times \text{Hom}_{\mathcal{C}}(y, z) \to \text{Hom}_{\mathcal{C}}(y, z)$
Definition

A category $\mathcal{C}$ is made of

- A collection of objects: $\text{Obj}(\mathcal{C})$
- $\forall x, y \in \text{Obj}(\mathcal{C})$ a set $\text{Hom}_\mathcal{C}(x, y)$
- $\forall x \in \text{Obj}(\mathcal{C})$ there is $\text{id}_x \in \text{Hom}_\mathcal{C}(x, x)$
- $\forall x, y, z \in \text{Obj}(\mathcal{C})$ a function $\circ: \text{Hom}_\mathcal{C}(x, y) \times \text{Hom}_\mathcal{C}(y, z) \to \text{Hom}_\mathcal{C}(y, z)$

such that

1. Identity: $f \circ \text{id} = \text{id} \circ f = f$
2. Associativity: $(h \circ g) \circ f = h \circ (g \circ f)$
Example: Category of graphs

- Objects: \( G = (V, E, s, t) \) with \( s, t : E \rightarrow V \)
- Morphisms: \( f : G \rightarrow H \) must respect source and target functions, ie:

\[
\forall e \in E. f(s(e)) = s(f(e)) \\
\forall e \in E. f(t(e)) = t(f(e))
\]
Example: Category of graphs

- **Objects:** $G = (V, E, s, t)$ with $s, t : E \rightarrow V$
- **Morphisms:** $f : G \rightarrow H$ must respect source and target functions, ie:
  \[
  \forall e \in E. f(s(e)) = s(f(e)) \\
  \forall e \in E. f(t(e)) = t(f(e))
  \]

- **Examples:**

![Graph Examples](image-url)
Pullback

- Let's have: \( f : X \to Z \) and \( g : Y \to Z \)

- Fiber product: \( X \times_Z Y \) := \( \{ (x, w, y) \mid f(x) = w = g(y) \} \)
Co-construction of the pullback.

Let's have: \( f : X \to Z \) and \( g : Y \to Z \)

disjoint sum with gluing: \( X +_Z Y := X + Y + Z / \sim \)

With \( \sim \) generated by \( f(z) \sim z \sim g(z) \)
Plan

1. Category Theory 101

2. Graph transformation and Categories

3. AGREE and Graph Generation
   - AGREE and Data Anonymization
   - Self-similar Graphs

4. Conclusion
Rule based transformations

- Rule-based term rewriting is easy: replace a tree by another one.
- Much more difficult with graphs (multiple incident edges).
- Categorical frameworks make it clean to express graph transformations systematically.

<table>
<thead>
<tr>
<th>PB</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone</td>
<td>merge</td>
</tr>
<tr>
<td>delete</td>
<td>add</td>
</tr>
<tr>
<td>comatch</td>
<td>match</td>
</tr>
<tr>
<td>global</td>
<td>local</td>
</tr>
</tbody>
</table>
AGREE extended rule

Extension of a framework proposed by A. Corradini, D. Duval, R. Echahed, F. Prost and L. Ribeiro [ICGT15].

Definition (AGREE rules and matches)

- A rule is

\[
L \xleftarrow{l} K \xrightarrow{r} R
\]

\[
T_L \xleftarrow{l'} T_K
\]

- A match of such a rule is composed of a mono \( L \xrightarrow{m} G \) and a typing morphism \( G \xrightarrow{\bar{m}} T_L \) and is such that \( l' \circ t = (\bar{m} \circ m) \circ l \).
Definition (AGREE rewriting)

Given \( \rho = (K \xrightarrow{l} L, K \xrightarrow{r} R, K \xrightarrow{t} T_K, T_K \xrightarrow{l'} T_L) \) and a match \( L \xrightarrow{m} G, G \xrightarrow{\overline{m}} T_L : G \Rightarrow_{\rho,m} H \) is computed as follows:

1. \( \text{Span } G \xleftarrow{g} D \xrightarrow{n'} T_K \) is the pullback of \( G \xrightarrow{\overline{m}} T(L) \xleftarrow{l'} T_K \). Since \( l' \circ t = \eta_L \circ l \) there is a unique \( K \xrightarrow{n} D \).

2. \( R \xrightarrow{p} H \xleftarrow{h} D \) is the pushout of \( D \xleftarrow{n} K \xrightarrow{r} R \).
AGREE rewrite step

Definition (AGREE rewriting)

Given \(\rho = (K \overset{l}{\to} L, \overset{r}{K} \to R, \overset{t}{K} \rightleftarrows T_K, T_K \overset{l'}{\to} T_L)\) and a match \(L \overset{m}{\Rightarrow} G, G \overset{m}{\Rightarrow} T_L : G \Rightarrow_{\rho,m} H\) is computed as follows:

1. \(\text{Span } G \overset{g}{\leftarrow} D \overset{n'}{\to} T_K\) is the pullback of \(G \overset{m}{\to} T(L) \overset{l'}{\leftarrow} T_K\). Since \(l' \circ t = \eta_L \circ l\) there is a unique \(K \overset{n}{\rightleftarrows} D\).

2. \(R \overset{p}{\to} H \overset{h}{\leftarrow} D\) is the pushout of \(D \overset{n}{\leftarrow} K \overset{r}{\to} R\).

\[\begin{array}{ccc}
L & \overset{l}{\leftarrow} & K \\
\downarrow m & & \downarrow n \\
G & \overset{g}{\leftarrow} & D \\
\downarrow m & & \downarrow h \\
T(L) & \overset{l'}{\leftarrow} & T_K \\
\end{array}\]
Example: copy of web pages

- The structure of a web site typically as two kind of links:
  - Internal links: file hierarchy (indirect link)
  - External links: references pointing outside of the site.
Example: copy of web pages

- The structure of a web site typically as two kinds of links:
  - Internal links: file hierarchy (indirect link)
  - External links: references pointing outside of the site.
- The cloning of a web site consists in duplicating all local files and keeping external links shared between the two copies.

\[ \text{WWW} \]

should be cloned as follows
Web copy with AGREE rewriting
Web copy with AGREE rewriting
Web copy with AGREE rewriting
Web copy with AGREE rewriting
Graph transformation and Categories

Web copy with AGREE rewriting

(PB) → (PO)
Plan

1. Category Theory 101
2. Graph transformation and Categories
3. AGREE and Graph Generation
   - AGREE and Data Anonymization
   - Self-similar Graphs
4. Conclusion
Social Data Anonymization: concepts and challenges

- Big economical issue: more or less the backbone of the business models of internet giants (Google, Facebook, Yahoo etc.).
- Big political issue: Open Data Policy.
Social Data Anonymization: concepts and challenges

- Big economical issue: more or less the backbone of the business models of internet giants (Google, Facebook, Yahoo etc.).
- Big political issue: Open Data Policy.
- Raw problem: given a graph $G$ we would like to produce $G'$ such that
  - $\text{Stat}(G) \simeq \text{Stat}(G')$
  - It is not possible to reidentify nodes (or edges) of $G$ from knowing $G'$ (and some extra informations...).
Social Data Anonymization: concepts and challenges

- Big economical issue: more or less the backbone of the business models of internet giants (Google, Facebook, Yahoo etc.).
- Big political issue: Open Data Policy.
- Raw problem: given a graph $G$ we would like to produce $G'$ such that
  - $\text{Stat}(G) \simeq \text{Stat}(G')$
  - It is not possible to reidentify nodes (or edges) of $G$ from knowing $G'$ (and some extra informations...).
- Naïve approach doesn’t work: Netflix [NarayanShmatikov06].
- Anonymization is an active research field ... rather artistic at the time: approaches validated through experiments.
Social Data Anonymization: Dimensions and Principles

- Problem more down to the earth than non-interference:
  - Partial knowledge of the graph by the opponent.
  - Active attacker (embedding fake sub graphs to re-identify them).
  - Object of interests vary from one data set to another.
Problem more down to the earth than non-interference:
- Partial knowledge of the graph by the opponent.
- Active attacker (embedding fake sub graphs to re-identify them).
- Object of interests vary from one data set to another.

Hence three important points to consider:
1. Background Knowledge: What does the opponent know? Model of the opponent.
2. Privacy: what is attacked?
3. Usage: How the data is going to be analyzed?

⇒ Anonymizing techniques
Social Data Anonymization: Techniques

- Two families:
  - Clustering: group together edges and nodes.
  - k-anonymity (and l-diversity): there should be at least k-1 other candidates with similar features.
Social Data Anonymization: Techniques

- Two families:
  - Clustering: group together edges and nodes.
  - k-anonymity (and l-diversity): there should be at least k-1 other candidates with similar features.

- We focus on the k-anonymity approach: the problem amounts to create $G'$ such that $G' = G_1 \oplus G_2 \oplus \ldots \oplus G_k$ such that $G_i$s are isomorphic graphs.
Social Data Anonymization: Techniques

- Two families:
  - Clustering: group together edges and nodes.
  - k-anonymity (and l-diversity): there should be at least k-1 other candidates with similar features.

- We focus on the k-anonymity approach: the problem amounts to create $G'$ such that $G' = G_1 \oplus G_2 \oplus ... \oplus G_k$ such that $G_i$s are isomorphic graphs.

- It is NP-hard to find graph transformations minimizing the editing distance between a graph and a $k$-isomorphic graph.
Two families:
- Clustering: group together edges and nodes.
- k-anonymity (and l-diversity): there should be at least k-1 other candidates with similar features.

We focus on the k-anonymity approach: the problem amounts to create $G'$ such that $G' = G_1 \oplus G_2 \oplus \ldots \oplus G_k$ such that $G_i$s are isomorphic graphs.

It is NP-hard to find graph transformations minimizing the editing distance between a graph and a $k$-isomorphic graph.

One solution: select $1/k$ nodes randomly, create $k$ clones, link the clones together easy to program with AGREE approach.
Using \textit{AGREE} for $k$-anonymity

- Progaming with types!
- $L$ is just a cloud of nodes, and $K$ is made of $k$ clones of $L$.
- Standard $T_L$ is:

\begin{center}
\begin{tikzpicture}
    \node (L) at (0,0) {$\circ$};
    \node (R) at (1,0) {$\star$};
    \node (M) at (1,1) {$\circ$};
    \node (ML) at (1,2) {$\circ$};
    \draw (L) edge [loop left] (L);
    \draw (L) edge [loop right] (L);
    \draw (L) edge (M);
    \draw (M) edge (R);
    \draw (R) edge (M);
    \draw (M) edge (L);
\end{tikzpicture}
\end{center}

- Simplest $T_K$ is:

\begin{center}
\begin{tikzpicture}
    \node (L) at (0,0) {$\circ^1$};
    \node (R) at (1,0) {$\star$};
    \node (M) at (1,1) {$\circ$};
    \node (ML) at (1,2) {$\circ$};
    \node (ML1) at (1,3) {$\circ$};
    \node (ML2) at (1,4) {$\circ$};
    \draw (L) edge [loop left] (L);
    \draw (L) edge [loop right] (L);
    \draw (L) edge (M);
    \draw (M) edge (R);
    \draw (R) edge (M);
    \draw (M) edge (L);
    \draw (M) edge (ML);
    \draw (ML) edge (ML1);
    \draw (ML1) edge (ML2);
\end{tikzpicture}
\end{center}
Types and structural graph properties

- The simplest $k$-clones are not connected to each others.
Types and structural graph properties

- The simplest $k$-clones are not connected to each other.
- AGREE allows the use of the graph structure to reconnect them:
Types and structural graph properties

- The simplest $k$-clones are not connected to each others.
- AGREE allows the use of the graph structure to reconnect them:

```
1

2 --> 3
```

- Degree problems (nodes of degree 1).
Types and structural graph properties

- The simplest $k$-clones are not connected to each others.
- AGREE allows the use of the graph structure to reconnect them:

\[ \begin{array}{c}
\circlearrowleft \\
\circlearrowright \\
\end{array} \quad \quad \begin{array}{c}
\circlearrowright \\
\circlearrowleft \\
\end{array} \quad \quad \begin{array}{c}
\circlearrowleft \\
\circlearrowright \\
\end{array} \]

- Degree problems (nodes of degree 1). One possibility is to type differently the edges, eg:

\[ \begin{array}{c}
\circlearrowleft \\
\circlearrowright \\
\end{array} \quad \quad \begin{array}{c}
\circlearrowright \\
\circlearrowleft \\
\end{array} \quad \quad \begin{array}{c}
\circlearrowleft \\
\circlearrowright \\
\end{array} \]
Self-similar graphs

- Every vertex is replaced by a copy of the graph.
- Interconnections between copies of the original “mimic” the ones in the target graph.
Implementation in the AGREE Framework

\[ L \leftarrow K \]

\[ \text{clique}(\star \oplus K) \]

\[ t_{kl1} \]

\[ l_K \]
Implementation in the AGREE Framework

\[ G \xrightarrow{m} L \leftarrow K \]

\[ G \xrightarrow{\bar{m}} \text{clique}(\star \oplus K) \]

\[ t_{k/1} \]

\[ l_K \]
Implementation in the AGREE Framework
Implementation in the AGREE Framework

\[
\begin{align*}
L & \xleftarrow{m} G & K & \xrightarrow{t_k} TL_1 & \text{clique}(\star \oplus K) \\
G & \xleftarrow{\overline{m}} & & & \\
G & \xleftarrow{l_1} & TL_1 & \xrightarrow{k_{l_1}} & \\
G & \xleftarrow{l_K} & & & \\
\end{align*}
\]
Implementation in the AGREE Framework
Plan

1. Category Theory 101
2. Graph transformation and Categories
3. AGREE and Graph Generation
   - AGREE and Data Anonymization
   - Self-similar Graphs
4. Conclusion
Categorical frameworks allow simple and mathematically workable definition of complex transformations.

Only basic constructs are needed: pushouts, pullbacks.

An very generic implementation is scheduled.

Open questions:
  - matching ? (random match does not lead to scale-free networks)
  - What statistics can be interesting (Ramsey-like theory) ?
  - What kind of certificate can be produced ?