An approach for inferring geometric expressions in topology-based geometric modeling
Revisited as a program synthesis problem

Romain Pascual
Pascale Le Gall, Hakim Belhaouari, and Agnès Arnould

October 6, 2023
FM&AI working group at LMF
1. Introduction
1. Introduction
1. Introduction
1. Introduction
Inferring modeling operations

Inferring the generation of an object:

- Inverse procedural modeling: retrieving parameters.\(^1\)
- L-systems: retrieving formal rules.\(^2\)
- Constructive solid geometry: retrieving sequences of operations.\(^3\)
- Polyhedral decomposition: retrieving a graph grammar. Illustration from (Merrell 2023)

\(^1\)Wu et al. 2014; Emilien et al. 2015.
\(^3\)Sharma et al. 2018; Kania et al. 2020; Xu et al. 2021.
Inferring modeling operations

Inferring the generation of an object

Pure geometry

- Retrieve non-linear weights of a Loop-based subdivision scheme for mesh refinement. Illustration from (Liu et al. 2020).
Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

1. Introduction
Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

1. Build a theorem "for all input, there exists an output such that the specification holds."
2. Construct a proof of the theorem (proof assistant)
3. Derive a program from the proof
Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

Syntax-guided$^1$

- Search-based approaches that leverage a syntactic template

$^1$Alur et al. 2013.
Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

Syntax-guided\textsuperscript{1}

( Neural approaches, LLM, etc. )

\textsuperscript{1} Alur et al. 2013.

1. Introduction
Example from (Alur et al. 2018)

Consider the specification

$$\forall x \forall y \ (x \leq f(x, y)) \land (y \leq f(x, y)) \land (f(x, y) \in \{x, y\})$$
Example from (Alur et al. 2018)

Consider the specification

\[ \forall x \forall y \ (x \leq f(x, y)) \land (y \leq f(x, y)) \land (f(x, y) \in \{x, y\}) \]

Consider the context-free grammar generated by

\[ T := x | y | 0 | 1 | T + T | ITE(C, T, T) \]
\[ C := (T \leq T) | \neg C | (C \land C) \]
Example from (Alur et al. 2018)

Consider the specification

$$\forall x \forall y \ (x \leq f(x, y)) \land (y \leq f(x, y)) \land (f(x, y) \in \{x, y\})$$

Consider the context-free grammar generated by

$$T ::= x | y | 0 | 1 | T + T | ITE(C, T, T)$$
$$C ::= (T \leq T) | \neg C | (C \land C)$$

Possible expression

$$f(x, y) = ITE((x \leq y), y, x)$$
Syntax-guided program synthesis

Given

- a function $f$, specified by a formula $\varphi$ in a theory $T$
- a language $L$ of admissible expressions

Find an expression $e \in L$ such that

$$\varphi[f/e] \text{ is valid modulo } T$$
Syntax-guided program synthesis

Given

- a function \( f \), specified by a formula \( \varphi \) in a theory \( T \)
- a language \( L \) of admissible expressions

Find an expression \( e \in L \) such that

\[
\varphi[f/e] \text{ is valid modulo } T
\]

Programming by example\(^1\)

- \( \varphi \) derived from an input-output example
- \( L \) is a domain-specific language

---

\(^1\)Gulwani et al. 2012.
1. Introduction
1. Introduction
1. Introduction
1. Introduction
1. Introduction
Embedded generalized maps

► How to represent objects?
Generalized maps\(^1\) (topology)

Legend: 0, 1, 2

\(^1\)Damiand et al. 2014.

2. Embedded generalized maps
Generalized maps\textsuperscript{1} (topology)

Orbit: Sub-graph induced by a subset \langle o \rangle of dimensions

Legend: 0, 1, 2

Vertices: orbits \langle 1, 2 \rangle

\textsuperscript{1}Damiand et al. 2014.

2. Embedded generalized maps
Generalized maps\(^1\) (topology)

*Orbit:* Sub-graph induced by a subset \(\langle o \rangle\) of dimensions

Legend: 0, 1, 2

Vertices: orbits \(\langle 1, 2 \rangle\)

Faces: orbits \(\langle 0, 1 \rangle\)

\(^1\)Damiand et al. 2014.

2. Embedded generalized maps
2. Embedded generalized maps
Embeddings (geometry)

Embedding: function $\pi : \langle o_\pi \rangle \rightarrow \tau_\pi$
with $\tau_\pi$ an abstract data type

Legend: 0, 1, 2  
position: $\langle 1, 2 \rangle \rightarrow \text{Point3}$  
color: $\langle 0, 1 \rangle \rightarrow \text{ColorRGB}$

2. Embedded generalized maps
Graph rewriting

- How to formalize object transformations?
Graph transformation rules\textsuperscript{1}

\textsuperscript{1}Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

3. Graph rewriting
Graph transformation rules\textsuperscript{1}

\textsuperscript{1}Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

3. Graph rewriting
Topological rewriting of Gmaps

3. Graph rewriting
Orbit rewriting

3. Graph rewriting
Orbit rewriting

Implicitly computed

3. Graph rewriting
Orbit rewriting

Implicitly computed

Local

Instantiated rule

3. Graph rewriting
Orbit rewriting

3. Graph rewriting
Orbit rewriting

Instantiated rule
Local

Implicitly computed

<0, 1>
<0, _>
<0, _>
<1, 2>
<_, 2>
1
0

1 0
<0, 1>
<0, _>

3. Graph rewriting
Orbit rewriting

Implicitly computed

Local

Instantiated rule

3. Graph rewriting
Orbit rewriting

3. Graph rewriting
Embedding expressions\(^1\) (towards \(L\))

Three families of expressions

- **Accessors**
  - a.color = \(\bullet\)
  - a.position = \(A\)

---

\(^1\)Bellet et al. 2017; Arnould et al. 2022.

3. Graph rewriting
Embedding expressions\(^1\) (towards \(L\))

Three families of expressions

- **Accessors**
- **Computations**

\[
middle(\{\bullet, \circ\}) = \bullet
\]

\(^1\)Bellet et al. 2017; Arnould et al. 2022.

3. Graph rewriting
Embedding expressions\(^1\) (towards \(L\))

Three families of expressions

- Accessors
- Computations
- Gmap traversals

\[a@0.\text{position} = D\]
\[\text{position}_{\langle 0,1 \rangle}(a) = \{A, B, C, D\}\]

---

\(^1\)Bellet et al. 2017; Arnould et al. 2022.

3. Graph rewriting
Extension to schemes

3. Graph rewriting
Extension to schemes

3. Graph rewriting
Extension to schemes

3. Graph rewriting
Extension to schemes

\[ \frac{1}{4}(A + B + C + D) \]
Extension to schemes

\[
\frac{1}{4}(A + B + C + D) = \text{middle}([A, B, C, D])
\]
Extension to schemes

\[ \frac{1}{4}(A + B + C + D) = \text{middle}(|A, B, C, D|) = \text{middle}(|\text{position}_{(0,1)}(a)|) \]
Extension to schemes

\[ \frac{1}{4}(A + B + C + D) = \text{middle}(\{A, B, C, D\}) \]
\[ = \text{middle}(\text{position}_{\langle0,1\rangle}(a)) \]
\[ = \text{middle}(\text{position}_{\langle0,1\rangle}(n0)) \]
Extension to schemes

n2.position : middle(position<0,1>(n0))
Inferring geometric expressions

- How to retrieve the embedding computation expressions?
Topological folding algorithm\textsuperscript{1}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{topological_folding_algorithm}
\caption{Topological folding algorithm.}
\end{figure}

\textsuperscript{1}Pascual et al. 2022.

4. Inferring geometric expressions
Topological folding algorithm\textsuperscript{1}

\begin{equation}
\begin{aligned}
<0, 1> & \quad \quad \quad \quad \quad \quad \quad \text{Implicitly computed} \\
<0, _> & \quad \quad \quad \quad \quad \quad \quad \text{Local}
\end{aligned}
\end{equation}

\textsuperscript{1}Pascual et al. 2022.

\section*{4. Inferring geometric expressions}
4. Inferring geometric expressions
Need for abstraction on schemes

Issues: Darts in the Gmap share the same expression.
Solution: Exploit the topology.

Points of interest:

4. Inferring geometric expressions
Need for abstraction on schemes

Schemes induce a topological abstraction

4. Inferring geometric expressions
Need for abstraction on schemes

Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

4. Inferring geometric expressions
Need for abstraction on schemes

Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

Solution: Exploit the topology

4. Inferring geometric expressions
Need for abstraction on schemes

Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

Solution: Exploit the topology

Points of interest

4. Inferring geometric expressions
Points of interest

• $p_v$: vertex
• $p_e$: edge midpoint
• $p_f$: face barycenter
• $p_s$: volume barycenter
• $p_{cc}$: CC barycenter

4. Inferring geometric expressions
Points of interest

- \( p_v \): vertex
- \( p_e \): edge midpoint
- \( p_f \): face barycenter
- \( p_s \): volume barycenter
- \( p_{cc} \): CC barycenter

Looking for

d
Points of interest

with

• $p_v$: vertex

\[ p_v = \text{middle}(\text{position}_\langle \rangle(d)) \]
Points of interest

with

- \( p_v \) : vertex
- \( p_e \) : edge midpoint

\[ p_e = \text{middle}(\text{position}_{(0)}(d)) \]
Points of interest

with

- $p_v$: vertex
- $p_e$: edge midpoint
- $p_f$: face barycenter

$p_f = \text{middle}(\text{position}_{\langle 0,1 \rangle}(d))$
Points of interest

with

- $p_v$ : vertex
- $p_e$ : edge midpoint
- $p_f$ : face barycenter
- $p_s$ : volume barycenter

\[ p_s = \text{middle}(\text{position}_{\langle 0, 1, 2 \rangle}(d)) \]
Points of interest

with

- $p_v$: vertex
- $p_e$: edge midpoint
- $p_f$: face barycenter
- $p_s$: volume barycenter
- $p_{cc}$: CC barycenter

\[ p_{cc} = \text{middle(position}_{\langle 0,1,2,3 \rangle}(d)) \]

4. Inferring geometric expressions
Points of interest

with

- $p_v$: vertex
- $p_e$: edge midpoint
- $p_f$: face barycenter
- $p_s$: volume barycenter
- $p_{cc}$: CC barycenter

Looking for

$$f(p_v, p_e, p_f, p_s, p_{cc})$$
Points of interest

with

- $p_v$ : vertex
- $p_e$ : edge midpoint
- $p_f$ : face barycenter
- $p_s$ : volume barycenter
- $p_{cc}$ : CC barycenter

Looking for

\[ f(p_v, p_e, p_f, p_s, p_{cc}) = w_v p_v + w_e p_e + w_f p_f + w_s p_s + w_{cc} p_{cc} + t \]

$L$ is the set of affine expressions over the points of interest
Building the logical specification

The position expression of \( n_2 \) only depends on \( n_0 \)

4. Inferring geometric expressions
Building the logical specification

The position expression of $n2$ only depends on $n0$

Symbolic equation

$$n2.position = w_v n0.p_v + w_e n0.p_e + w_f n0.p_f + w_s n0.p_s + w_cc n0.p_{cc} + t$$

4. Inferring geometric expressions
Building the logical specification

The position expression of \textit{n2} only depends on \textit{n0}

- One equation per dart (8 darts).

Symbolic equation

\[
n2.\text{position} = w_v n0.p_v + w_e n0.p_e + w_f n0.p_f + w_s n0.p_s + w_{cc} n0.p_{cc} + t
\]
The position expression of \( n_2 \) only depends on \( n_0 \)

- One equation per dart (8 darts).
- Split per coordinate (on \( x, y, z \)).

Symbolic equation

\[
n_2.\text{position} = w_v n_0.\text{p}_v + w_e n_0.\text{p}_e + w_f n_0.\text{p}_f + w_s n_0.\text{p}_s + w_{cc} n_0.\text{p}_{cc} + t
\]
Building the logical specification

The position expression of \( n2 \) only depends on \( n0 \)

- One equation per dart (8 darts).
- Split per coordinate (on \( x, y, z \)).
- 24 equations and 8 variables.

Symbolic equation

\[
\begin{align*}
n2.\text{position} &= w_v n0.p_v + w_e n0.p_e + w_f n0.p_f + w_s n0.p_s + w_{cc} n0.p_{cc} + t 
\end{align*}
\]
Building the logical specification

The position expression of \( n_2 \) only depends on \( n_0 \)

- One equation per dart (8 darts).
- Split per coordinate (on \( x, y, z \)).
- 24 equations and 8 variables.

\( \varphi \) is the concrete system induced by the input-output example
Building the logical specification

The position expression of $n_2$ only depends on $n_0$

- One equation per dart (8 darts).
- Split per coordinate (on $x$, $y$, $z$).
- 24 equations and 8 variables.

$\varphi$ is the concrete system induced by the input-output example

Solved via an SMT solver (Z3, OR-Tools)
Solving the barycentric triangulation

Symbolic equation

\[ n2.position = w_v n0.p_v + w_e n0.p_e + w_f n0.p_f + w_s n0.p_s + w_{cc} n0.p_{cc} + t \]
Solving the barycentric triangulation

Symbolic equation

\[ n2.\text{position} = w_v n0.\text{p}_v + w_e n0.\text{p}_e + w_f n0.\text{p}_f + w_s n0.\text{p}_s + w_{cc} n0.\text{p}_{cc} + t \]

Generated system

\[
\begin{align*}
(0.5; 0.5) &= w_v \times (0; 0) + w_e \times (0.5; 0) + w_f \times (0.5; 0.5) + w_s \times (0.5; 0.5) + w_{cc} \times (0.5; 0.5) + (tx; ty) \\
(0.5; 0.5) &= w_v \times (1; 0) + w_e \times (0.5; 0) + w_f \times (0.5; 0.5) + w_s \times (0.5; 0.5) + w_{cc} \times (0.5; 0.5) + (tx; ty) \\
(0.5; 0.5) &= w_v \times (1; 1) + w_e \times (1; 0.5) + w_f \times (0.5; 0.5) + w_s \times (0.5; 0.5) + w_{cc} \times (0.5; 0.5) + (tx; ty) \\
\vdots &= \vdots \\
\end{align*}
\]

4. Inferring geometric expressions
Solving the barycentric triangulation

Symbolic equation

\[ n2.\text{position} = w_\text{v} n0.\text{p}_\text{v} + w_\text{e} n0.\text{p}_\text{e} + w_\text{f} n0.\text{p}_\text{f} + w_\text{s} n0.\text{p}_\text{s} + w_\text{cc} n0.\text{p}_\text{cc} + t \]

Generated system

\[
\begin{align*}
(0.5; 0.5) &= w_\text{v} \cdot (0; 0) + w_\text{e} \cdot (0.5; 0) + w_\text{f} \cdot (0.5; 0.5) + w_\text{s} \cdot (0.5; 0.5) + w_\text{cc} \cdot (0.5; 0.5) + (tx; ty) \\
(0.5; 0.5) &= w_\text{v} \cdot (1; 0) + w_\text{e} \cdot (0.5; 0) + w_\text{f} \cdot (0.5; 0.5) + w_\text{s} \cdot (0.5; 0.5) + w_\text{cc} \cdot (0.5; 0.5) + (tx; ty) \\
(0.5; 0.5) &= w_\text{v} \cdot (1; 1) + w_\text{e} \cdot (1; 0.5) + w_\text{f} \cdot (0.5; 0.5) + w_\text{s} \cdot (0.5; 0.5) + w_\text{cc} \cdot (0.5; 0.5) + (tx; ty) \\
\vdots & \vdots \vdots \vdots \vdots \vdots \vdots \vdots \vdots \vdots \vdots \\
\end{align*}
\]

Solution found

- \( w_\text{v} = 0.0 \)
- \( w_\text{e} = 0.0 \)
- \( w_\text{f} = 1.0 \)
- \( w_\text{v} = 0.0 \)
- \( w_\text{cc} = 0.0 \)
- \( t = (0.0, 0.0) \)

4. Inferring geometric expressions
JerboaStudio and applications

- Implementation in Jerboa
5. JerboaStudio and applications
// no translation
Point3 res = new Point3(0.0, 0.0, 0.0);

// face
Point3 p2 = Point3::middle(<0,1>_position(n0));

// weight
p2.scale(1.0);

// added to the result
res.addVect(p2);

// return the value
return res;
Menger Sponge

5. JerboaStudio and applications
Menger Sponge

Node $n_1$

```java
Point3 res = new Point3(0.0, 0.0, 0.0);
Point3 p0 = Point3::middle(<>, _position(n0));
p0.scale(0.3333333134651184);
res.addVect(p0);
Point3 p1 = Point3::middle(<0>, _position(n0));
p1.scale(0.6666666865348816);
res.addVect(p1);
return res;
```
Menger Sponge

Node \( n_7 \)

```cpp
Point3 res = new Point3(0.0, 0.0, 0.0);
Point3 p0 = Point3::middle(<> _position(n0));
p0.scale(0.3333333134651184);
res.addVect(p0);
Point3 p2 = Point3::middle(<0,1> _position(n0));
p2.scale(0.66666666865348816);
res.addVect(p2);
return res;
```
Node \textit{n16}

```cpp
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<>_position(n0));
p0.scale(0.3333333134651184);
res.addVect(p0);
Point3 p3 = Point3::middle(<0,1,2>_position(n0));
p3.scale(0.66666666865348816);
res.addVect(p3);
return res;
```
(2, 2, 2)-Menger Polycube$^1$

$^1$Richaume et al. 2019.
(2, 2, 2)-Menger Polycube\textsuperscript{1}

\textsuperscript{1}Richaume et al. 2019.

5. JerboaStudio and applications
Geology inspired

Before

After

Positions and colors

5. JerboaStudio and applications
Geology inspired

Before
Geology inspired

After
Limits

Von Koch’s snowflake generated by L-systems

Inferred
Limits

Von Koch’s snowflake generated by L-systems

Inferred
6. Conclusion
Geometric inference and program synthesis

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule level</td>
<td>Rule scheme</td>
</tr>
<tr>
<td></td>
<td>Instantiated rule</td>
</tr>
</tbody>
</table>

Corresponds to

Affine combinations of points of interest

Concrete system derived from the example

Property

Finite Encodes redundancies

Extend with

- other points of interest
- other computations
- multi-examples
- counter-examples

6. Conclusion
Geometric inference and program synthesis

<table>
<thead>
<tr>
<th>Rule level</th>
<th>Rule scheme</th>
<th>Instantiated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Affine combinations of points of interest</td>
<td>Concrete system derived from the example</td>
</tr>
<tr>
<td>φ</td>
<td>Corresponds to</td>
<td></td>
</tr>
</tbody>
</table>

6. Conclusion
## Geometric inference and program synthesis

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule level</td>
<td>Rule scheme</td>
</tr>
<tr>
<td>Corresponds to</td>
<td>Affine combinations of points of interest</td>
</tr>
<tr>
<td>Property</td>
<td>Finite</td>
</tr>
</tbody>
</table>

### 6. Conclusion
## Geometric inference and program synthesis

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule level</td>
<td>Rule scheme</td>
</tr>
<tr>
<td>Corresponds to</td>
<td>Affine combinations of points of interest</td>
</tr>
<tr>
<td>Property</td>
<td>Finite</td>
</tr>
<tr>
<td>Extend with</td>
<td>• other points of interest</td>
</tr>
<tr>
<td></td>
<td>• other computations</td>
</tr>
</tbody>
</table>

6. Conclusion
Geometric inference vs program synthesis?

Similarities
Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
Geometric inference vs program synthesis?

Similarities

• Formal specification of the expected result
• DSL
• Resolution delegated to a solver

Differences

• Pretreatment induced by the topology
• Exploit symmetries
• Not so easy to play with examples

Is there any added value in re-thinking in terms of program synthesis?

6. Conclusion
Geometric inference vs program synthesis?

Similarities

• Formal specification of the expected result
• DSL
• Resolution delegated to a solver

Differences
Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology
Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology
- Exploit symmetries

6. Conclusion
Geometric inference vs program synthesis?

Similarities
- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences
- Pretreatment induced by the topology
- Exploit symmetries
- Not so easy to play with examples
Geometric inference vs program synthesis?

Similarities
- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences
- Pretreatment induced by the topology
- Exploit symmetries
- Not so easy to play with examples

Is there any added value in rethinking in terms of program synthesis?

Alur, Rajeev et al. (Nov. 20, 2018). “Search-based program synthesis”. In: Communications of the ACM 61.12, pp. 84–93. ISSN: 0001-0782. DOI: 10.1145/3208071.

Arnould, Agnès et al. (2022). “Preserving consistency in geometric modeling with graph transformations”. In: Mathematical Structures in Computer Science. DOI: 10.1017/S0960129522000226.


References IV


