Exploiting the Layout Engine to Assess Diagram Completions

Steffen Mazanek, Sonja Maier, Mark Minas
Universität der Bundeswehr München, Germany

LED 2008, 15 September 2008
Editing modes

- **structured editing**
  - set of predefined editing operations
  - transform correct diagrams into other correct diagrams
  - advantage: guidance

- **free-hand editing**
  - arrange diagram components freely on the screen
  - syntax analysis for diagram recognition
  - advantage: more flexible, sketching etc.

- with our approach you do not need to decide anymore
  → get the best of both worlds!
Running Example: Nassi-Shneiderman Diagrams NSD

- representation of structured programs
- diagram components:

![Diagram components with code examples:]

n:=0

while x>1

y

x even

x:=x/2

until x>1

x:=3x+1

y

x even

n

- example diagram:
Problem 1: beginners need examples and error correction

<table>
<thead>
<tr>
<th>number of components</th>
<th>example diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>![Diagram 1]</td>
</tr>
<tr>
<td>2</td>
<td>![Diagram 2]</td>
</tr>
<tr>
<td>3</td>
<td>![Diagram 3]</td>
</tr>
</tbody>
</table>

What the hell is an NSD?

I wonder what‘s wrong here?!
Problem 2: correcting a diagram by hand is tedious

input diagram
two ways to (syntactically) correct it:

A

B

This is annoying

shortcut needed
Problem 3: geometric error correction might not be enough

- geometric error correction supported by several systems, e.g. DiaGen, Penguins/CIDER

- faster and easier to just roughly sketch arrangement of components, precision work done automatically

Still difficult to hit the mark ...

but what about this one?
Diagram completion

- changes to a diagram such that the resulting diagram is correct
  - existing diagram components can be moved or resized
  - new diagram components can be inserted

- diagram completions can be computed automatically from the syntax definition of the visual language

- Mazanek et al., Auto-completion for Diagram Editors based on Graph Grammars, VL/HCC 2008
Problem 1 revisited: many possible completions might exist

Which one should the editor propose as assistance?

Oh heck, where are my components?
Status quo: DiaGen tool demo

→ need for assessing and ranking of completions

Auto-completion
Ranking completions: Our approach in brief

- compute completions on the abstract syntax level
- translate them to concrete diagram changes
- compare possible target diagrams with respect to the changes they cause to existing diagram components
- apply best completion
Example: Hypergraphs as a model

```
while x > 1
    if x even
        n := n + 1
        x := x / 2
    else
        x := 3 * x + 1

Example: Hypergraphs as a model
stmt
cond
stmt
stmt
stmt
while
stmt
stmt
```

- diagram component → hyperedge
- type of diagram component → hyperedge label
- component's attachment area → node visited by edge
- corners touch → edges visit same node
Our approach by example

A simple, incomplete NSD, its abstract syntax graph, ...

... some hypergraph completions of size ...

0 1 2

a) b) c) d) e) f)

... and the resulting diagrams:

4 5 3 6 1 2

stmt₁ stmt₁ stmt₂ stmt₂ stmt₁ stmt₁

Assess Diagram Completions, Steffen Mazanek, 15 September 2008 (LED)
Metrics

- simplest: number of edges added
  - unfortunately not sufficient

- several metrics have been developed for mental map preservation in dynamic graph drawing
  - orthogonal ordering
  - taxicab
  - euclidean distance

- results of completions are even structurally different, so that all these metrics should work reasonably well for comparison
Integration in DiaGen: new editing process

Diagram

Scanner

SRG

Reducer

ASG

Parser

Layouter

Layout information

Attribute evaluation

Derivation structure

e.g. QOCA ensures minimal changes

Drawing tool

Update translator

Graph completion

Rank

selects operation

if desired: choose

asks for assistance

Editor user

Assess Diagram Completions, Steffen Mazanek, 15 September 2008 (LED)
New problem: \( n! \) possibilities

- roughly sketched example diagram:

  ... 
  ... 
  ... 
  ... 

  already \( 4! = 24 \) completions to check

- Solution 1: draw diagram in several steps!
- Solution 2: consider coordinates while parsing
Future Work

- further improve user interface
- usability studies
- extend scope by considering non-contextfree languages
Conclusion

- benefits of diagram completion in general:
  - syntactical assistance of the user
  - generation of example diagrams

- benefits of our approach:
  - choosing among completions is easier
  - quick fix shortcut, just roughly sketch your diagrams (increased productivity and accessibility)

- syntax assist becomes usable
DiaGen

- diagram editor generator developed by Mark Minas
- DiaGen editors have the following features:
  - freehand editing
  - syntax analysis
  - interactive layout
  - structured editing operations
- hypergraphs as a model
- syntax definition with hypergraph grammars
Syntax definition: Constraint Hypergraph Grammars

\[
a:\text{NSD} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
b:\text{Stmt} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
c:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

\[
b:\text{Stmt} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
c:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

\[
b:\text{Stmt} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
c:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

\[
b:\text{Stmt} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
c:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

\[
b:\text{while} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
b:\text{cond} ::\Rightarrow n_1 n_2 n_3 n_4
\]

\[
c:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

\[
d:\text{NSD} ::\Rightarrow n_3 n_4 n_1 n_2
\]

A derivation triggers layout constraints...

\[
a.x_1 = b.x_1, a.x_2 = b.x_2
\]

\[
a.y_1 = b.y_1, a.y_2 = b.y_2
\]

\[
a.x_1 = b.x_1 = c.x_1, a.x_2 = b.x_2 = c.x_2
\]

\[
a.y_1 = b.y_1, a.y_2 = c.y_2, b.y_2 = c.y_1
\]

\[
a.x_1 = b.x_1, a.x_2 = b.x_2 = c.x_2
\]

\[
a.y_1 = b.y_1, a.y_2 = b.y_2 = c.y_2
\]

\[
c.x_1 = b.xm, c.y_1 = b.ym
\]

\[
b.xm - b.x_1 \geq w_{\text{min}}, b.ym - b.y_1 \geq h_{\text{min}}
\]

\[
a.x_1 = b.x_1, a.x_2 = b.x_2 = c.x_2
\]

\[
a.y_1 = b.y_1, a.y_2 = b.y_2 = c.y_2
\]

\[
c.x_1 = b.xm, c.y_1 = b.ym
\]

\[
b.xm - b.x_1 \geq w_{\text{min}}, b.ym - b.y_1 \geq h_{\text{min}}
\]
Example Derivation