Undoing Operational Steps of Domain-Specific Modeling Languages

Tim Hartmann, Daniel A. Sadilek
Humboldt-Universität zu Berlin
Outline

- Introduction
  - Development of executable DSMLs
  - Animated execution
  - Operational semantics
- Undoing operational steps
- Open issues
- Conclusion
Use Case: DSML Development

Draft

Iterative development cycle

Prototype

- Metamodel
- Graphical editor
- Operational semantics

Evaluation

Error

Carl Adam Petri

Language engineer
2nd Iteration of DSML Development Example

Draft

Carl Adam Petri

Language engineer

Prototype

- Metamodel
- Graphical editor
- Corrected operational semantics

Iterative development cycle

Evaluation

Runtime state
Operational Semantics

- Interpretable operational semantics

- Transition system: $\langle \Gamma, \rightarrow \rangle$

- Configurations: $\Gamma$

- Transition relation: $\rightarrow \subseteq \Gamma \times \Gamma$
Operational Semantics

• Configurations are represented as models:
  \[ \{ 1 \rightarrow \square \rightarrow 0, \quad 0 \rightarrow \square \rightarrow 1, \quad \ldots \} \in \Gamma \]

• Configurations are defined by a metamodel

• Transition relation \( \rightarrow \) can be defined with a model-to-model transformation
Configuration Metamodel

Petri Net example

```plaintext
context Transition::activated: bool
derive: src->forall(runtimeToken > 0)
```

```
Net
+ running: int
```

```
Transition
+ name: String
+ activated: bool
```

```
Place
+ name: String
+ runtimeToken: int
+ initToken: int
```

Graphical representation of a Petri Net model with places, transitions, and their relationships.

9/20
Transition Transformation

Part of Petri Net Java semantics (erroneous version)

```java
protected void run(Net net) {
    Transition t = getActivated(net);
    if (t != null) {
        consume(t.getSrc.get(0));
        produce(t.getSnk.get(0));
    }
}
```
protected void run(Net net) {
    Transition t = getActivated(net);
    if (t != null) {
        for (Place p : t.getSrc()) {
            consume(p);
        }
        for (Place p : t.getSnk()) {
            produce(p);
        }
    }
}
Executing model test.petri...
Initialising.
Performed step 1.
Performed step 2.
Executing model test.petri...

Initialising.

Performed step 1.

Performed step 2.

Stepped back to step 1.

Performed step 2.
Undoing Operational Steps

- **Undo**: reverse changes
- **Observer for model changes**

- **Execution step**: single unit of work
- **Composition of elementary changes**

- **Change history**:
- **Shared command stack for editor and execution**
Synchronization

Editor

Configuration

user input

transition

visualize

Execution

Configuration

1

read values

perform step

2
Open Issues

- Breakpoints between execution steps
- Declarative breakpoint description?
- Users can produce invalid configurations
- How to describe and implement constraints?
- Changing operational semantics can affect previous configurations
- How to step back to last state that is consistent with changed semantics?
Conclusion

- New debug feature for DSML prototyping
- Adapting undo of editors for stepwise model execution

- Implementation experience: many building blocks available in EMF
Undoing Operational Steps of Domain-Specific Modeling Languages

Discussion

Tim Hartmann, Daniel A. Sadilek
Humboldt-Universität zu Berlin