Improved Conflict Detection for Graph Transformation with Attributes

GaM 2015

Géza Kulcsár, Frederik Deckwerth, Malte Lochau, Gergely Varró, Andy Schürr
geza.kulcsar@es.tu-darmstadt.de
Motivation: Conflict of Attribute Operations

- **Conflict**: having two alternative operations on the same object, applying the second after the first leads to a different result than applying the first after the second (or at least one sequence is not possible)

```
Node n
int x = 42
```

```
x := x + 1
```

```
x := x + 2
```

```
Node n
int x = 43
```

```
x := x + 2
```

```
Node n
int x = 44
```

```
x := x + 1
```

```
Node n
int x = 45
```

**non-conflicting**
Representing Attributes: Symbolic Graphs

**Symbolic graphs (Orejas):** graphs enriched with: (i) attribute nodes holding variables, (ii) attribute edges and (iii) a first-order logic formula to assign values to attribute variables

- note that a symbolic graph can specify multiple models (attribute value assignments)
Graph Transformation with Attributes

- Graph transformation: a rule-based approach to modify graph-based models
- **Example rule:** attribute value is increased by 1

![Graph Transformation Diagram]

match $x = 42$
Parallel independence of two direct derivations: no direct derivation deletes any element which is matched by the other direct derivation.

...but still, $42 + 1 + 2 = 42 + 2 + 1$
Parallel independence is intuitively too strict for analysing attribute operations (resulting in false positives, i.e., situations where the derivation sequences are equivalent may be recognized as conflict)

- A new, more precise conflict condition is needed to take the semantics of attribute operations into account
Contribution: an Improved Conflict Notion for Graphs with Attributes

- Not independent of the classical approach based on parallel dependence

- Performed as a 3-step process (for a given pair of alternative direct derivations)
  1. Checking the two rules if there is any chance of a conflict
  2. Checking the parallel dependence of the direct derivations
  3. Refining the conflict detection by checking **direct confluence** in case of parallel dependence (**contribution**)
A Conflict Detection Process

1. \((L_1 \cap L_2 = \emptyset) \lor D \not\models \Phi_1 \land \Phi_2\)
   - true: OUT: „no conflict“
   - false: If the two left-hand sides are not overlapping or there does not exist a model where both rules are applicable, there is no conflict

2. \(SH_1 \xleftrightarrow{r_1} SG \xrightarrow{r_2} SH_2\) is parallel independent
   - true: OUT: „no conflict“
   - false: OUT: „no conflict“

3. \(SH_1 \xleftrightarrow{r_1} SG \xrightarrow{r_2} SH_2\) is directly confluent
   - true: Our proposed step
   - false: OUT: \(SH_1 \xleftrightarrow{r_1} SG \xrightarrow{r_2} SH_2\)
Direct Confluence

- **Direct confluence** is proposed as a less conservative conflict condition.
Direct Confluence
Result: Completeness

- Our approach has the completeness property:

Although direct confluence is less restrictive than parallel independence, our approach is still able to detect each possible conflict situation (according to the direct confluence condition).
Conclusion

- Conflict detection is important for real-life rule-based applications, e.g., program refactorings, visual languages, ...

- Our approach takes the semantics of attribute operations into account, which can potentially reduce the number of false positives while retaining completeness

- It can be used as a static analysis technique (by lifting the detection process to rule level using minimal contexts)

- Moreover, by using symbolic graphs, an implementation of the approach can use any off-the-shelf SMT solver for the formula part (ongoing work)