Static analysis of incremental propagation graphs with process algebra

Théo Le Calvar, Fabien Chhel, Frédéric Jouault, Frédéric Saubion

Static analysis of incremental propagation graphs with process algebra

Théo Le Calvar[1,2], Fabien Chhel[2], Frédéric Jouault[2], Frédéric Saubion[1]

[1] LERIA, Université d'Angers, Angers, France [2] ERIS Team, ESEO Group, Angers, France

Abstract

Active Operations are a set of operations that can be composed to build incremental bidirectional OCL-like expressions on collections. The current implementation of active operations (AOF) relies on the Observer design pattern to propagate changes from each operation to its successors. These relations form an implicit directed acyclic propagation graph. In this work we present a new relational notation to describe propagation graphs. Along with this notation, we also present a new static analysis method of the propagation graph based on process algebra. This new method enables optimizations of the propagation graph not achievable with previous approaches, such as detection of parallelizable sections of the propagation graph or cache optimizations in specific situations.

Process Algebra

Process Algebra [2, 4] is a formalism used to formally describe process behaviors. It consists of a small sets of concepts and operators such as:

- Actions: a, b are simple actions, δ is a special action that corresponds to a failure.
- Sequential: $a.b$, a then b.
- Alternative: $a + b$, a or b.
- Parallel: $a \| b$, a and b in parallel.
- Communication: parallel processes a and b can communicate if $a|b$ is defined.
- Encapsulation: $\partial_H(X)$ replaces every action of X present in H by a δ. It can be used to force two actions to communicate.

Translation method

Translating an AOF expression into a corresponding process algebra formula is done in several steps:

1. Represent the expression as a propagation graph.
2. Extract sub-propagation graphs for each entry-point of the graph.
3. Add synchronization operations when there are several paths between an operation and the entry-point.
4. Break merges by splitting synchronization operations.
5. Generate a corresponding formula by applying the following function to each entry-point:

$$R2ACP(o) = \begin{cases} Next(o) = \emptyset, & o \\ Next(o) = o', & o \ . \ R2ACP(o') \\ Next(o) = \{o_1, ..., o_n\}, & o.\Big(R2ACP(o_1) \| ... \| R2ACP(o_n)\Big) \end{cases}$$

6. Merge all generated formulas:

$$\sum_{o \in Start} \partial_{\{s \in Syncs(o)\}}\big(R2ACP(o)\big)$$

Possible analysis

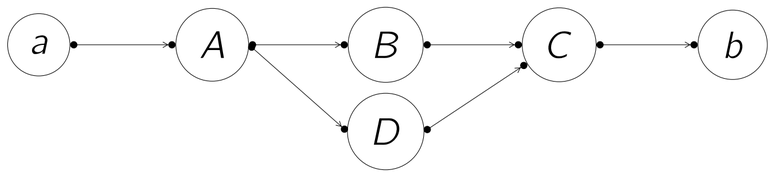
Using process algebra to describe AOF expressions offers several interesting analysis:

- The formula can be used to check if an operation evaluation ordering is correct.
- The formula can be used to generate correct orderings (operation evaluation).
- One specialized ordering per input of the expression.
- Parallelizable sections can be inferred from the formula.
- Possible detection of useless operation caches.
- Recursive formulas could be used to model expressions with cycles.
- Process algebra has tools that could be used to optimize propagation graphs.

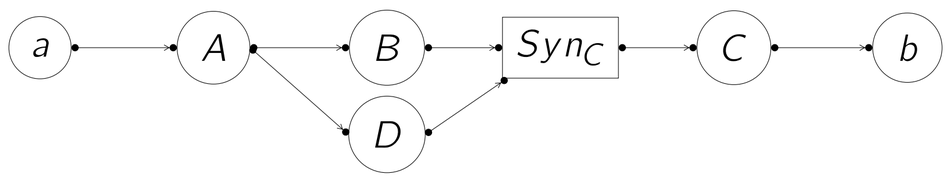
Example of transformation

```
def: F(a) =
  let v1 = a->A() in
  let v2 = v1->B() in
  let v3 = v1->D() in
  let b = v2->C(v3) in
  b
```

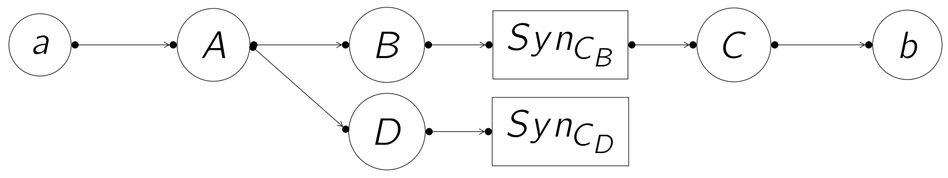
OCL-like expression with alignments issues



Steps 1 & 2 - Corresponding propagation graph for the only entry-point



Step 3 - Adding synchronization nodes



Step 4 - Splitting the synchronization nodes

$$F = a.\partial_H\Big(A.\Big((B.Syn_{C_B}.C.b) \| (D.Syn_{C_D})\Big)\Big)$$

$$H = \{Syn_{C_B}, Syn_{C_D}\}$$

$$Syn_{C_B}|Syn_{C_D} = S$$

Steps 5 & 6 - Corresponding ACP formula

References

[1] Olivier Beaudoux et al. "Active Operations on Collections". In: *MODELS*. Vol. 6394. Lecture Notes in Computer Science. Springer, 2010, pp. 91–105.

[2] J.A. Bergstra and J.W. Klop. "Process algebra for synchronous communication". In: *Information and Control* 60.1 (1984), pp. 109 –137.

[3] Frédéric Jouault et al. "Improving Incremental and Bidirectional Evaluation with an Explicit Propagation Graph". In: *Software Technologies: Applications and Foundations*. Ed. by Martina Seidl and Steffen Zschaler. Springer International Publishing, 2018, pp. 302–316.

[4] Robin Milner. *A Calculus of Communicating Systems*. Vol. 92. Lecture Notes in Computer Science. Springer, 1980.