Connectors: Refinement and test case generation using Z3

Sihan Wu[1] and Xueyi Tan[2]

[1] Yuanpei College, Peking University, No.5 Yiheyuan Road, Haidian District, Beijing 100871, China,
2300017743@stu.pku.edu.cn,
[2] School of Mathematical Sciences, Peking University, No.5 Yiheyuan Road, Haidian District, Beijing 100871, China,
2300010816@stu.pku.edu.cn

Abstract. test. test.

Keywords: test, test, test

1 Introduction

2 Preliminaries

2.1 The coordination language Reo

Reo is a channel-based exogenous coordination language where complex coordinators, called connectors, are compositionally built out of simpler ones [2]. Exogenous coordination imposes a purely local interpretation on each inter-component communication, engaged in as a pure I/O operation on each side, that allows components to communicate anonymously, through the exchange of untargeted passive data.

Complex connectors in Reo are organized in a network of primitive connectors with well-defined behavior, called *channels*, such as synchronous channels, FIFO channels, etc. A connector provides the protocol that controls and organized the communication, synchronization and cooperation among the components/services that they interconnect. Each channel has two *channel ends*: *source* ends and *sink* ends. A source channel end accepts data into the channel, and a sink channel end dispenses data out of the channel. It is possible for the ends of a channel to be both sinks or both sources. Reo places no restriction on the behavior of a channel and thus allows an open-ended set of different channel types to be used simultaneously together. Each channel end can be connected to at most one component instance at any given time. Some simple channel types in Reo are shown in sectionModeling Connectors.

Complex connectors are constructed by composing simpler ones via the *join* and *hiding* operations. Channels are joined together at nodes. A node consists of a set of channel ends. The set of channel ends coincident on a node A is disjointly partitioned into the sets Src(A) and Snk(A), denoting the sets of source and sink channel ends that coincide on A, respectively. Nodes are categorized into *source*, *sink* and *mixed nodes*, depending on whether all channel ends that coincide on a node are source ends, sink ends or a combination of the two. The hiding operation is used to hide the internal topology of a component connector. The hidden nodes can no longer be accessed or observed from outside. A complex connector has a graphical representation, called a *Reo circuit*, which is a finite graph where the nodes are labeled with pair-wise disjoint, non-empty sets of channel ends, and the edges represent their connecting channels. The behavior of a Reo circuit is formalized by means of the data-flow at its sink and source nodes. Intuitively, the source nodes of a circuit are analogous to the input ports, and the sink nodes to the output ports of a component, while mixed nodes capture its hidden internal details.

A component can write data items to a source node that it is connected to. The write operation succeeds only if all (source) channel ends coincident on the node accept the data item, in which case the data item is transparently written to every source end coincident on the node. A source node, thus, acts as a replicator. A component can obtain data items, by an input operation, from a sink node that it is connected to. A take operation succeeds only if at least one of the (sink) channel ends coincident on the node offers a suitable data item; if more than one coincident channel end offers suitable data items, one is selected non-deterministically. A sink node, thus, acts as a non-deterministic merger. A mixed node nondeterministically selects and takes a suitable data item offered by one of its coincident sink channel ends and replicates it into all of its coincident source channel ends. Note that a component cannot connect to, take from, or write to mixed nodes.

2.2 Z3

Z3 [5] if an efficient SMT (Satisfiability Modulo Theories) solver freely available from Microsoft. It has been used in various software verification and analysis applications. Z3 expands to deciding the satisfiability (or dully the validity) of first order formulas with respect to combinations of theories such as: arithmetic, bit-vectors, arrays, and uninterpreted functions. Given the data and time constraints of connectors, it allows us to verify the satisfiability of properties or refinement relations. Z3 provides bindings for several programming languages. In this paper, we use *Z3 python-bindings* to construct the models and carry out refinement checking.

3 Modeling Connectors

3.1 Basic connectors

We first develop the design model for a set of basic Reo connectors, i.e., channels.

3.2 Timer connectors

3.3 Probabilistic connectors

The specification of channels with probabilistic behavior can be captured by the disjunction or conjunction of different predicates about time and data distributions. We consider four types of probabilistic channels: *message-corrupting synchronous channel*, *randomized synchronous channel*, *probabilistic lossy synchronous channel*, and *faulty FIFO1 channel*. Specifications of other primitive channels are ignored here and can be found at [1].

CptSync: The message-corrupting synchronous channel $-\!\!\!p\!\!\!\rightarrow$ is a synchronous channel which has an extra parameter p compared with the primitive synchronous channel. The delivered message can be corrupted with probability p. Hence, if a data item flows into the channel through the source end, then the correct data value will be obtained at the sink end with probability $1 - p$ and a corrupted data value $\perp$ will be obtained with probability p.

RandomSync: The randomized synchronous channel $\xrightarrow{rand(0,1)}$ can generate a random number $b \in \{0, 1\}$ with equal probability when it is activated through an arbitrary write operation on its source end, and this random number will be taken on the sink end simultaneously.

ProbSync: The message transmitted by the probabilistic synchronous channel $\overset{q}{--}\rightarrow$ can get lost with a certain probability q. It can also act like a *Sync* channel and the message will be delivered successfully with probability $1 - q$.

FaultyFIFO1: The messages flowing into a faulty FIFO1 channel $\overset{r}{\cdots}\square\rightarrow$ can get lost with probability r when it is inserted into the buffer. In this case, the buffer remains empty. It can also behave as a normal *FIFO1* channel when the insertion of data into the buffer is successful with probability $1 - r$.

3.4 Composing connectors

3.5 Refinement of connectors

4 Refinement checking and test case generation in Z3

4.1 Test case for connectors

4.2 Refinement checking

References

1. The source code., https://github.com/Zhang-Xiyue/Prob-Reo
2. Arbab, F.: Reo: A channel-based coordination model for component composition. Mathematical Structures in Computer Science 14, 329–366 (06 2004)
3. Meng, S.: Connectors as designs: The time dimension. In: Margaria, T., Qiu, Z., Yang, H. (eds.) Sixth International Symposium on Theoretical Aspects of Software Engineering, TASE 2012, 4-6 July 2012, Beijing, China. pp. 201–208. IEEE (2012), http://doi.ieeecomputersociety.org/10.1109/TASE.2012.36

4. Meng, S., Arbab, F., Aichernig, B.K., Aştefănoaei, L., de Boer, F.S., Rutten, J.: Connectors as designs: Modeling, refinement and test case generation. Science of Computer Programming 77(7), 799–822 (2012), https://www.sciencedirect.com/science/article/pii/S0167642311001006, (1) FOCLASA'09 (2) FSEN'09
5. de Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
6. Sun, M.: Towards formal modeling and verification of probabilistic connectors in coq xiyue zhang (2018), https://api.semanticscholar.org/CorpusID:85541303
7. Zhang, X., Hong, W., Li, Y., Sun, M.: Reasoning about connectors using coq and z3. Science of Computer Programming 170, 27–44 (2019), https://www.sciencedirect.com/science/article/pii/S0167642318304076