Tiles for Reo ⋆

Farhad Arbab[1], Roberto Bruni[2], Dave Clarke[1], Ivan Lanese[3], and Ugo Montanari[2]

[1] CWI, Amsterdam, The Netherlands
{farhad,dave}@cwi.nl
[2] Dipartimento di Informatica, Università di Pisa, Italy
{bruni,ugo}@di.unipi.it
[3] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
lanese@cs.unibo.it

1 Extended abstract

Reo [1, 8] is an exogenous coordination model for software components. It is based on channel-like connectors that mediate the flow of data and signals among components. Notably, a small set of point-to-point primitive connectors is sufficient to express a large variety of interesting constraints over the behaviour of connected components, including various forms of mutual exclusion, synchronisation, alternation, and context-dependency. In fact, components and primitive connectors can be composed in a circuit fashion via suitable attach points, called Reo nodes. Typical primitive connectors are the synchronous / asynchronous / lossy channel and the asynchronous one-place buffer. The informal Reo's semantics has been matched by several proposals of formalisation, exploiting co-algebraic techniques [2], constraint-automata [3], and colouring tables [6].

Figure 1 shows a small but non-trivial example of Reo circuit for modelling an *exclusive router*, together with the explanation of how the different kinds of connectors are drawn as arrows. If some datum n is written on A then it flows to B on the synchronous channel s_A. Node B must push (copies of) n on the three outgoing channels ls_C, ls_D and sd. The datum can get lost on ls_C or on ls_D, because they are lossy, but not on both. In fact the synchronous drain sd can get n from B only if E can provide another datum. This is possible only if E receives the datum from C (via s_C) or from D (via s_D), and E is not allowed to take the datum from both synchronous channels. Therefore it must be the case that exactly one node between C and D receives n, which is then forwarded either to F or to G. The example suggests that the propagation of constraints can introduce some context-awareness in certain parts of the circuit (see [6]).

We aim to show that the Tile Model [9] offers a flexible and adequate semantic setting for Reo. The name 'tile' is due to the graphical representation of such rules (see Fig. 2). The tile α states that the *initial configuration* s can be triggered by the event a to reach the *final configuration* t, producing the *effect* b. Tiles can be composed in three different ways to generate larger steps: (i) horizontally (synchronisation), when the effect of one tile matches the trigger for another tile; (ii) vertically (composition

⋆ Research supported by the project FET-GC II IST-2005-16004 SENSORIA, by the Italian FIRB project TOCAI, by the Dutch NWO project n. 612.000.316 C-Quattro, and by the bilateral German-Dutch DFG-NWO project n. 600.643.000.05N12 SYANCO.

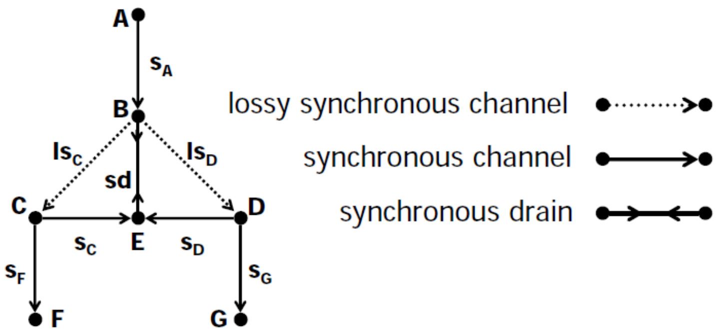

Fig. 1. Exclusive router (from A to either F or G) as a Reo circuit

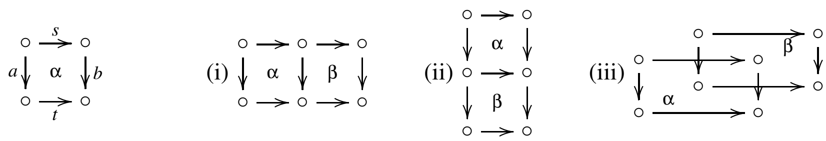

Fig. 2. Examples of tiles and their composition.

in time), when the final configuration of one tile matches the initial configuration of another tile; and (iii) in parallel (concurrency).

Tiles resemble Gordon Plotkin's SOS inference rules [14], but they can be composed horizontally, vertically and in parallel to build larger proof steps. They take inspiration from Andrea Corradini and Ugo Montanari's Structured Transition Systems [7] and generalise Kim Larsen and Liu Xinxin's context systems [12], by allowing for more general rule formats. The tile model also extends José Meseguer's rewriting logic [13] (in the non-conditional case) by taking into account rewrite with side effects and rewrite synchronisation. As rewriting logic, the tile model admits a purely logical formulation, where tiles are seen as sequents subject to certain inference rules.

The definition of a tile semantics for Reo has specific features:

- Concurrency aspects can be taken into account. In fact, tiles have been designed around concurrent systems, hence it is common to consider a monoidal structure of states that gives raise to a monoidal double-category of concurrent computations.
- Tile bisimilarity and tile trace equivalence offer standard abstract equivalences.
- Meta-theoretical results can be exploited to guarantee that tile bisimilarity is a congruence, thus reconciling the algebraic and co-algebraic views of connectors.

The case of stateless connectors has been already considered in [4], in which case a normal form axiomatisation is available for tile bisimilarity, that coincides with tile trace equivalence and with the 2-colouring semantics of [6]. Roughly, Reo nodes and connectors are represented as hyper-edges (with typed incoming and outgoing tentacles) that can be composed sequentially (horizontally) and in parallel by connecting their

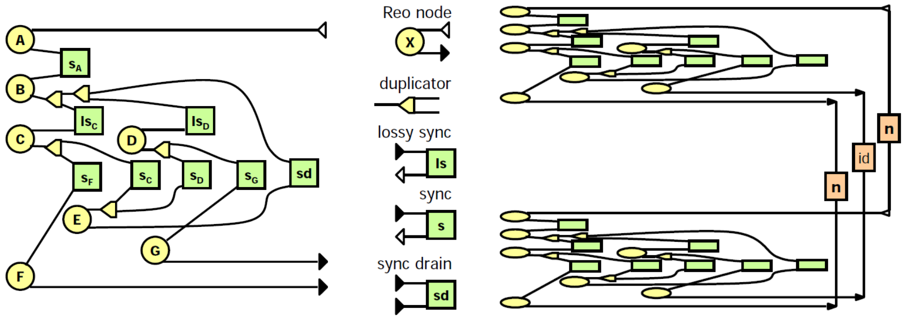

Fig. 3. Tile model for the exclusive router circuit

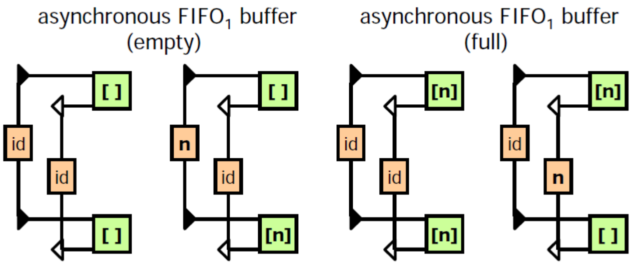

Fig. 4. Basic tiles for asynchronous FIFO$_1$ buffer

tentacles. The semantics of each connector c is defined by suitable basic tiles whose initial configuration is the hyper-edge c and whose triggers and effects define how the data can flow through c. Figure 3 shows the configuration diagram that corresponds to the exclusive router, together the explanation of how the different kinds of hyper-edges correspond to Reo elements and with an example of derived tile composition. To improve readability we use different shapes and colors for nodes, channels and vertical observations. A duplicator is a special kind of hyper-edge that allows to attach multiple connectors to the same node. White triangles are used to type incoming attach points and black triangles to type outgoing attach points: they are oriented according to the flow of data. The derived tile composition in Fig. 3 is obtained by horizontal and parallel pasting of basic tiles. The overall trigger is void, because the configuration has no attach point on the left. The overall effect models the routing of a datum n from the incoming interface of A to the outgoing interface of F, with G idle.

During the talk we will show that the semantics given in [4] can be extended to take into account stateful connectors, like one-place buffers (see Fig. 4) and, more importantly, it can deal with the finer 3-colouring semantics of [6], where the causes of inhibited interactions can be tracked. In the presence of stateful connectors, one advantage of tiles w.r.t. colouring tables is that the state of the connector after each step is made explicit in the final configurations of basic tiles (while it is not shown in colouring tables).

Finally, we observe that the Tile Model can offer a uniform setting for representing not only the ordinary execution of Reo systems but also dynamic reconfiguration strategies in the style of [5, 11, 10], thus reconciling relevant aspects that were dealt with separately in previous proposals.

References

1. F. Arbab. Reo: A channel-based coordination model for component composition. *Math. Struct. in Comput. Sci.*, 14(3):1–38, 2004.
2. F. Arbab and J. J. M. M. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Proceedings of WADT 2002*, volume 2755 of *Lect. Notes in Comput. Sci.*, pages 34–55. Springer, 2002.
3. C. Baier, M. Sirjani, F. Arbab, and J. J. M. M. Rutten. Modeling component connectors in Reo by constraint automata. *Sci. Comput. Program*, 61(2):75–113, 2006.
4. R. Bruni, I. Lanese, and U. Montanari. A basic algebra of stateless connectors. *Theoret. Comput. Sci.*, 366(1-2):98–120, 2006.
5. D. Clarke. Reasoning about connector reconfiguration II: Basic reconfiguration logic. In F. Arbab and M. Sirjani, editors, *Proceedings of FSEN'05*, volume 159 of *Elect. Notes in Th. Comput. Sci.*, pages 61–77. Elsevier Science, 2006.
6. D. Clarke, D. Costa, and F. Arbab. Connector colouring I: Synchronisation and context dependency. *Sci. Comput. Program*, 66(3):205–225, 2007.
7. A. Corradini and U. Montanari. An algebraic semantics for structured transition systems and its application to logic programs. *Theoret. Comput. Sci.*, 103:51–106, 1992.
8. CWI. *A repository of Reo connectors*. http://homepages.cwi.nl/˜proenca/webreo/.
9. F. Gadducci and U. Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 133–166. MIT Press, 2000.
10. C. Koehler, D. Costa, J. Proenca, and F. Arbab. Reconfiguration of Reo connectors triggered by dataflow. In C. Ermel, R. Heckel, and J. de Lara, editors, *Proceedings of GT-VMT'08*, Elect. Communic. of the EASST. EASST, 2008.
11. C. Koehler, A. Lazovik, and F. Arbab. Connector rewriting with high-level replacement systems. In C. Canal, P. Poizat, and M. Viroli, editors, *Proceedings of FOCLASA'07*, Elect. Notes in Th. Comput. Sci. Elsevier Science, 2007.
12. K. G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In M. Paterson, editor, *Proceedings of ICALP'90*, volume 443 of *Lect. Notes in Comput. Sci.*, pages 526–539. Springer, 1990.
13. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.
14. G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.