Compositional Failure-based Semantic Equivalences for Reo Specifications

Mohammad Izadi
Department of Computer Engineering
sharif University of Technology
Tehran, Iran
izadi@ce.sharif.edu

Ali Movaghar
Department of Computer Engineering
sharif University of Technology
Tehran, Iran
movaghar@sharif.edu

ABSTRACT

Reo is a coordination language for modeling component connectors of component-based computing systems. We show that the failure-based equivalences NDFD and CFFD are congruences with respect to composition operators of Reo.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Languages*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Verification*; D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*Semantics*

General Terms

Languages, Semantics, Verification.

Keywords

Reo Specification Language, Constraint Automata, Failure-based Equivalences, Coordination, Component-based Systems, Semantics.

1. EXTENDED ABSTRACT

The concept of *component based systems*, especially component based software, is a philosophy or way of thinking to deal with the complexity in designing large scale computing systems. One of the main goals of this approach is to compose reusable components by some glue codes. The model or the way in which these components are composed is called *coordination model*. Sometimes there are some formal or programming languages which are used for specification of coordination models. Such languages are called as *coordination languages*. Reo, as one of the most recently proposed coordination languages, is a channel based exogenous coordination language in which complex coordinators are compositionally built out of simpler ones [1, 2, 3]. By using Reo specifications, complex component connectors can be organized in a network of channels and build in a compositional manner. Reo relies on a very liberal and simple notion of channels and can model any kind of peer-to-peer communication. The channels used in Reo networks can be considered as simple communicating processes and the only requirements for them are that channels should have two ends (or I/O interfaces), declared to be *sink* or *source* ends, and a user-defined semantics. At source ends data items enter the channel by performing corresponding write operations. Data items are received from a channel at sink ends by performing corresponding read operations. Reo allows for an open ended set of channel types with user defined semantics.

If we want to be able to reason about properties of specifications or verify their correctness, Reo, as well as any other process specification languages, should be given abstract semantics. The key question in giving a semantic model to a specification language is: "Whenever can we say that two specifications or two models are equivalent?" Numerous definitions of different equivalence-relations for transition system based models have been presented in the literature. *Trace equivalence* (automata-theoretic equivalence), *weak bisimilarity* presented by Milner [9] and *failure-based equivalences* (CSP-like equivalences) such as the equivalence presented by Hoare [7] are examples of these equivalences.

Constraint automaton, as an extension of finite or Büchi automaton, is a formalism proposed to capture the operational semantics of Reo [4]. In a constraint automaton, contrary to finite automata and labeled transition systems, the label of a transition is not a simple character or action name. A transition label contains a set of names and a (constraint) proposition. The set of names indicates the names of ports which are participant in doing the transition and the proposition expresses some constraint about the data of the ports.

In this presentation, we are interested to investigate failure-based equivalences for constraint automata as the abstract semantics of Reo and their congruency with respect to composition operators which are useful in composing Reo specifications. The ultimate goal is to prepare an environment for compositional model checking of Reo specifications using equivalence based reduction method. In this method, the models of components and connectors of a component-based system are reduced with respect to an equivalence relation before building the model of the complete system [5, 6]. An equivalence relation should have two properties in order to be useful in the equivalence based compositional reduction method: it should *preserve* the class of properties to be ver-

Sixth International Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2007), September 3-4, 2007, Cavtat near Dubrovnik, Croatia.

ified and also, it should be a *congruence* with respect to the syntactic operators which are used for composing of the components of the model. By congruence relation we mean that the replacement of a component of a model by an equivalent one should always yield a model which is equivalent with the original one. Fortunately, in the context of compositional failure based semantic models of process description languages such as CCS and LOTOS, there are two equivalence relations, called CFFD and NDFD, which have the preservation property: CFFD-equivalence preserves that fragment of linear time temporal logic which has no next-time operator and has an extra operator distinguishing deadlocks [10, 11] and NDFD-equivalence preserves linear time temporal logic without next-time operator [8]. It was also shown that CFFD and NDFD are the minimal equivalences preserving the above mentioned fragments of linear time temporal logic.

Now, we introduce an extended definition of constraint automaton by which not only the connectors but also the components can be modeled.

DEFINITION 1. *Let N be a set of port names and Data be a set of data. A* data constraint *g over names set N and data set Data is a proposition, which can be constructed by using the following grammar:*
$g \ ::= \ \text{true} \mid \mathrm{d}_A = d \mid g_1 \vee g_2 \mid \neg g \qquad d \in Data, \quad A \in N$
We use $DC(N, Data)$ as the set of all data constraints over names set N and data set Data.
A Constraint automaton *over data set Data is a quadruple $C = (Q, Nam, T, q_0)$ where, Q is a finite set of states, Nam is a finite set of names, such that, $\tau \notin Nam$, $T \subseteq Q \times ((2^{Nam} \times DC(Nam, Data)) \cup \{\tau\}) \times Q$, and $q_0 \in Q$ is the initial state.*
For each $(p, N, g, q) \in T$, it is required that, $N \neq \emptyset$ and $g \in DC(N, Data)$.

The main difference of our definition of constraint automaton and its original definition (defined in [4]) is that in our definition, τ-transitions are permitted, while in its original definition it is not. We use τ-transitions because τ can be used as a symbol for each kind of internal action which is occurred in an actual system but its real type is not important in the modeling process. Thus, by using this kind of constraint automaton, not only the observational behavior of connectors, but also all internal and observable behavior of components can be modeled. Also note that, in principle, a *hiding* operator can hide all port-names of a transition. In such cases, we replace the transition label by τ. Our definition of Constraint automaton is departed from the original one by dropping the requirement that all runs have to be infinite. We also deal with finite runs, which are necessary to argue about deadlock configurations.

Now, we introduce two new composition operators for constraint automata: join (production) of two automata with respect to their common port names and hiding of a port name in all transition labels of an automaton.

DEFINITION 2. *Let $C_1 = (Q_1, Nam_1, T_1, q_{01})$ and $C_2 = (Q_2, Nam_2, T_2, q_{02})$ be two Constraint automata. The* product (join) Constraint automaton *of C_1 and C_2 is:*
$C_1 \bowtie C_2 = (Q_1 \times Q_2, Nam_1 \cup Nam_2, T, q_{01} \times q_{02})$ in which,
1) If $(q_1, N_1, g_1, p_1) \in T_1$ and $(q_2, N_2, g_2, p_2) \in T_2$ and $N_1 \cap Nam_2 = N_2 \cap Nam_1$, then,
$(< q_1, q_2 >, N_1 \cup N_2, g_1 \wedge g_2, < p_1, p_2 >) \in T$,
2) If $(q, N, g, p) \in T_1$ and $N \cap Nam_2 = \emptyset$, then,
$(< q, q' >, N, g, < p, q' >) \in T$,
3) If $(q, N, g, p) \in T_2$ and $N \cap Nam_1 = \emptyset$, then,
$(< q', q >, N, g, < q', p >) \in T$,
4) If $(q, \tau, p) \in T_1$ then, $(< q, q' >, \tau, < p, q' >) \in T$,
5) If $(q, \tau, p) \in T_2$ then, $(< q', q >, \tau, < q', p >) \in T$.
Let $C = (Q, Nam, T, q_0)$ be a Constraint automaton and B be a name, $B \in Nam$. The Constraint automaton resulted by hiding *of B in A is $\exists B[C] = (Q, Nam \backslash \{B\}, T_{\exists B}, q_0)$ where,*
(1) If $(q, \{B\}, g, p) \in T$ then, $(q, \tau, p) \in T_{\exists B}$.
(2) If $(q, N, g, p) \in T$ and $N \backslash \{B\} \neq \emptyset$ then $(q, N \backslash \{B\}, \exists B[g], p) \in T_{\exists B}$, where $\exists B[g] = \vee_{d \in Data} g[d_B/d]$.
(3) If $(q, \tau, p) \in T$ then, $(q, \tau, p) \in T_{\exists B}$.

Now, we can show that failure-based equivalences CFFD and NDFD are congruence with respect to join and hiding operators of constraint automata.

THEOREM 1. *NDFD and CFFD-equivalences are congruences with respect to the product (join) and hiding operators defined for finite constraint automata.*

Based on these congruency results and because of the linear time temporal logic preservation properties of CFFD and NDFD equivalences and their minimality properties (proved in [8]), they will be useful candidates for compositional reduction of models in the process of verifying the properties of component based systems, which their connectors are specified by Reo.

2. REFERENCES

[1] Arbab F., *Reo: A Channel-based Coordination Model for Component Composition*, Math. Struc. in Computer Science, **14(3)**, (2004), 329-366.

[2] Arbab F., *Abstract Behaiviour Types: A foundation model for components and their composition*, science of Computer Programming, **55**, (2005), 3-52.

[3] Arbab F., Mavadat F., *Coordination Through Channel Composition*, Proceedings of Coordination Languages and Models 2002, LNCS, **2315**, Springer-Verlag, (2002).

[4] Baier C., Sirjani M., Arbab F., Rutten J., *Modelling Component connectors in Reo by Constraint Automata*, Science of Computer Programming, **61**, (2006), 75-113.

[5] Clarke E., Long D., McMillan K., *Compositional Model Checking*, Proc. of 4th IEEE Symp. on Logic in Computer Science, (1989), 353-362.

[6] Graf S., Steffen B., *Compositional Minimization of Finite-State Systems*, Proc. of CAV'90, Springer, (1991), 186-196.

[7] Hoare C.A.R., "Communicating Sequential Processes", Prentice-hall, (1985).

[8] Kaivola R., Valmari, A., *The Weakest Semantic Equivalence Preserving Nexttime-less Linear Temporal Logic*, LNCS **630**, Springer-Verlag, (1992), 207-221.

[9] Milner R., "Communication and Concurrency", Prentice-Hall, (1989).

[10] Valmari A., Tienari M., *An Improved Failure Equivalence for Finite State Systems with a Reduction Algorithm*, "Protocol Specification, Testing and Verification", **XI**, (1991), 3-18.

[11] Valmari A., Tienari M., *Compositional Failure Based Semantic Models for Basic LOTOS*, Formal Aspects of Computing **7**, (1995), 440-468.