Finite Complete Suites for CSP Refinement Testing

Ana Cavalcanti[1], Wen-ling Huang[2], Jan Peleska[2], and Adenilso Simao[3]

[1] University of York, United Kingdom
ana.cavalcanti@york.ac.uk
[2] University of Bremen, Germany
{peleska,huang}@uni-bremen.de
[3] University of São Paulo, Brazil
adenilso@icmc.usp.br

Abstract. In this paper, new contributions to testing Communicating Sequential Processes (CSP) are presented, with focus on the generation of complete, finite test suites. A test suite is complete if it can uncover every conformance violation of the system under test with respect to a reference model. Both reference models and implementation behaviours are represented as CSP processes. As conformance relation, we consider trace equivalence and trace refinement, as well as failures equivalence and failures refinement. Complete black-box test suites here rely on the fact that the SUT's true behaviour is represented by a member of a fault-domain, that is, a collection of CSP processes that may or may not conform to the reference model. We define fault domains by bounding the number of excessive states occurring in a fault domain member's representation as a normalised transition graph, when comparing it to the number of states present in the graph of the reference model. This notion of fault domains is quite close to the way they are defined for finite state machines, and these fault domains guarantee the existence of *finite* complete test suites.

Keywords: Model-based testing, CSP, Trace Refinement, Failures Refinement, Complete Test Suites

1 Introduction

Motivation Model-based testing (MBT) is an active research field that is currently evaluated and integrated into industrial verification processes by many companies. This holds particularly for the embedded and cyber-physical systems domain. While MBT is applied in different flavours, we consider the most effective variant to be the one where test cases and concrete test data, as well as checkers for the expected results (*test oracles*), are automatically generated from a reference model: it guarantees the maximal return of investment for the time and effort invested into creating the test model. The test suites generated in this way, however, usually have different test strength, depending on the generation algorithms applied.

For the safety-critical domain, test suites with guaranteed fault coverage are of particular interest. For black-box testing, guarantees can be given only if certain hypotheses are satisfied. These hypotheses are usually specified by a *fault domain*: a set of models that may or may not conform to the SUT. The so-called *complete* test strategies guarantee to uncover every conformance violation of the SUT with respect to a reference model, provided that the true SUT behaviour is captured by a member of the fault domain.

Generation methods for complete test suites have been developed for various modelling formalisms. In this paper, we use *Communicating Sequential Processes (CSP)* [6, 10]; this is a mature process-algebraic approach that has been shown to be well-suited for the description of reactive control systems in many publications over almost five decades. Industrial success has also been reported.

Contributions This paper presents complete black-box test suites for divergence-free[4] CSP processes interpreted both in the trace and the failures semantics. Our results complement work published by two of the authors in [2]. There, fault domains are specified as collections of processes refining a "most general" fault domain member. With this concept of fault domains, complete test suites may be finite or infinite. While this gives important insight into the theory of complete test suites, we are particularly interested in finite suites when it comes to their practical application.

Therefore, a complementary approach to the definition of CSP fault domains is presented in this paper. To this end, we observe that every finite-state CSP process can be semantically represented as a finite normalised transition graph, whose edges are labelled by the events the process engages in, and whose nodes are labelled by minimal acceptances or, alternatively, maximal refusals [9]. The maximal refusals express the degree of nondeterminism that is present in a given CSP process state that is in one-one-correspondence to a node of the normalised transition graph. Inspired by the way that fault-domains are specified for finite state machines, we define them here as the set of CSP processes whose normalised transition graphs do not exceed the size of the reference model's graph by more than a give constant.

Our main contributions in this paper consist in the proof that for fault domains of the described type, finite, complete test suite generation methods can be given for verifying (1) trace equivalence, (2) trace refinement, (3) failures equivalence, and (4) failures refinement.

The existence of – possibly infinite – complete test suites has been established for process algebras in general and for CSP in particular by several authors, e.g. [5, 11, 8, 7, 2]. To our best knowledge, however, this article is the first to present *finite*, complete test suites associated with this class of fault domains.

[4] The assumption of divergence freedom is usually applied in black-box testing, since it cannot distinguish between divergence and deadlock.

It should be noted that the presented results are of a theoretical nature: despite being finite, the resulting test suite size will still be too large to be applied in practice to real-world problems. We discuss a number of promising options how the test suite size can be reduced to become practically applicable.

Overview In Section 2, we present the background material relevant to our work.

@todo

2 Preliminaries

2.1 CSP and Refinement

Communicating Sequential Processes @todo

FiXme Warning: alcc: I can make this small contribution.

Normalised Transition Graphs for CSP Processes As shown in [9], any finite-state CSP process P can be represented by a *normalised transition graph*

$$G(P) = (N, \underline{n}, \Sigma, t : N \times \Sigma \nrightarrow N, r : N \to \mathbb{PP}(\Sigma)),$$

with nodes N, initial node $\underline{n} \in N$, and process alphabet Σ. The partial *transition function* t maps a node n and an event $e \in \Sigma$ to its successor node $t(n, e)$, if, and only if, (n, e) is in the domain of t, that is, there is a transition from n with label e. Normalisation of $G(P)$ is reflected by the fact that t is a function.

A finite sequence of events $s \in \Sigma^*$ is a *trace* of P, if there is a path through $G(P)$ starting at $\underline{n}$ whose edge labels coincide with s. The set of traces of P is denoted by $\mathrm{trc}(P)$. If $s \in \mathrm{trc}(P)$, then the process corresponding to P after having executed s is denoted by P/s. Since $G(P)$ is normalised, there is a unique node reached by applying the events from s one by one, starting in $\underline{n}$. Therefore, $G(P)/s$ is also well defined.

By $[n]^0$ we denote the *fan-out* of n: the set of events occurring as labels in any outgoing transitions.

$$[n]^0 = \{e \in \Sigma \mid (n, e) \in \mathrm{dom}\ t\}$$

We also use this notation for CSP processes: $[P]^0$ is the set of events P may engage into, in other words, the initials of P after the empty trace of events, that is, $\mathit{initials}(P/\langle\rangle)$ as defined in [10].

The total function r maps each node n to its *refulsals* $r(n) = \mathrm{Ref}(n)$. Each element of $r(n)$ is a set of events that the CSP process P might refuse to engage into, when in a process state corresponding to n. The number of refusal sets in $\mathrm{Ref}(P/s)$ specifies the degree of nondeterminism that is present in process state P/s: the more refusal sets contained in $\mathrm{Ref}(P/s)$, the more nondeterministic is the behaviour in state P/s. If P/s is deterministic, its refusals coincide with the set of subsets of $\Sigma - [P/s]^0$, including the empty set.

For finite CSP processes, since the refusals of each process state are subset-closed [6, 10], $\mathrm{Ref}(P/s)$ can be re-constructed by knowing the set of *maximal refusals* $\mathrm{maxRef}(P/s) \subseteq \mathrm{Ref}(P/s)$. More formally, the maximal refusals $\mathrm{maxRef}(P/s)$ are defined as

$$\mathrm{maxRef}(P/s) = \{R \in \mathrm{Ref}(P/s) \mid \forall\, R' \in \mathrm{Ref}(P/s) - \{R\} : R \not\subseteq R'\}$$

Conversely, with the maximal refusals at hand, we can reconstruct the refusals by subset-closure:

$$\mathrm{Ref}(P/s) = \{R' \in \mathbb{P}(\Sigma) \mid \exists\, R \in \mathrm{maxRef}(P/s) : R' \subseteq R\}.$$

To see that this approach works only for finite CSP processes, consider the example where Σ is infinite. In this case, $\mathrm{maxRef}(STOP/\langle\rangle)$ is empty, and so we cannot use this set to calculate the refusals of $STOP$, that is, $\mathrm{Ref}(STOP/\langle\rangle)$ as defined above. As with refusals, we also use the transition graph-oriented notation $\mathrm{maxRef}(n) \subseteq r(n)$ to denote the maximal refusals associated with graph state n: if n is the state reached in the transition graph by following the edge labels in trace s, then $\mathrm{maxRef}(n) = \mathrm{maxRef}(P/s)$.

Well-formed normalised transition graphs must not refuse an event of the fan-out of a state in *every* refusal applicable in this state; more formally,

$$\forall\, n \in N, e \in \Sigma : (n, e) \in \mathrm{dom}\ t \Rightarrow \exists\, R \in \mathrm{maxRef}(n) : e \notin R \qquad (1)$$

By construction, normalised transition graphs reflect the *failures semantics* of finite-state CSP processes: the traces s of a process are the sequences of transition associated with paths through the graph, starting at $\underline{n}$. The pairs (s, R) with $s \in \mathrm{trc}(P)$ and $R \in r(G(P)/s)$ represent the failures $\mathrm{failures}(P)$ of P.

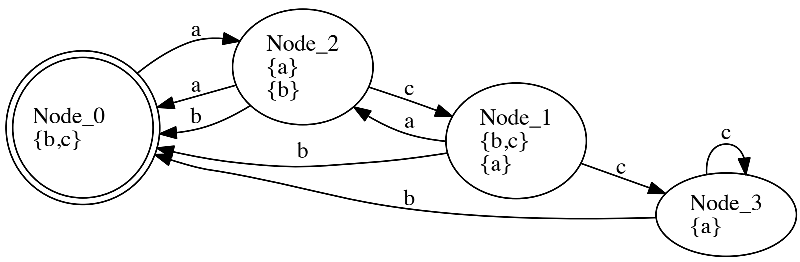

Fig. 1. Normalised transition graph of CSP process P from Example **??**.

Example 1. ex:a Consider the CSP process P defined below, and that the set of events Σ is $\{a, b, c\}$.

$$\begin{aligned}P &= a \to (Q \sqcap R)\\ Q &= a \to P \mathrel{\Box} c \to P\\ R &= b \to P \mathrel{\Box} c \to R\end{aligned}$$

Its transition graph $G(P)$ is shown in Fig. 1. Process state $P/\langle\rangle$ is represented there as Node_0, with $\{b, c\}$ as the only maximal refusal, since a can never be refused, and no other events are accepted. Having engaged into a, the transition emanating from Node_0 leads to Node_2 representing the process state $P/a = Q \sqcap R$. The internal choice operator induces several refusal sets derived from Q and R. Since these processes accept their initial events in external choice, $Q \sqcap R$ induces two maximal refusal sets $\{b\}$ and $\{a\}$. Note that event c can never be refused, since it is not a member of any maximal refusal.

Having engaged into c, the next process state is represented by Node_1. Due to normalisation, there is only a single transition satisfying $t(\text{Node\_2}, c) = \text{Node\_1}$. This transition, however, can have been caused by either Q or R engaging into c, so Node_1 corresponds to process state $Q/c \sqcap R/c = P \sqcap R$. This is reflected by the two maximal refusals labelling Node_1.

Similar considerations explain the other nodes and transitions in Fig. 1.

Note that the node names are generated by the FDR tool (see next paragraph). The node numbering is generated by FDR during the normalisation procedure. Therefore, the node numbers do not reflect the distance from the initial node Node_0.

Refinement relations between finite-state CSP processes P, Q can be be expressed by means of their normalised transition graphs in the following way.

Lemma 1.

$$P \sqsubseteq_T Q \Leftrightarrow trc(G(Q)) \subseteq trc(G(P)) \tag{2}$$

$$P \sqsubseteq_F Q \Leftrightarrow trc(G(Q)) \subseteq trc(G(P)) \wedge \tag{3}$$

$$\forall\, s \in trc(G(Q)), R_Q \in maxRef(G(Q)/s) : \tag{4}$$

$$\exists\, R_P \in maxRef(G(P)/s) : R_Q \subseteq R_P \tag{5}$$

$$P \sqsubseteq_F Q \Leftrightarrow trc(G(Q)) \subseteq trc(G(P)) \wedge \tag{6}$$

$$\forall\, s \in trc(G(Q)), A_Q \in minAcc(G(Q)/s) : \tag{7}$$

$$\exists\, A_P \in minAcc(G(P)/s) : A_P \subseteq A_Q \tag{8}$$

Tool Considerations The FDR tool [4] supports model checking and semantic analyses of CSP processes. It provides an API [3] that can be used to construct normalised transition graphs for CSP processes.

The FDR graph nodes are labelled by *minimal acceptances* instead of maximal refusals as described above. Since such a minimal acceptance set is the complement of a maximal refusal, the function r mapping states to their refusals can be implemented by creating the complements of all minimal acceptances and then building all subsets of these complements. For practical applications, the subset closure is never constructed in an explicit way; instead, sets are checked with respect to containment in a maximal refusal.

2.2 Test Cases and Complete Test Suites

@todo

2.3 Minimal Hitting Sets

When testing for failures refinement, it will become apparent that the main idea of the underlying test strategy can be based on solving a *hitting set problem*. Given a collection of finite sets $C = \{A_1, \ldots, A_n\}$ that are each subsets of a universe Σ, a *hitting set* $H \subseteq \Sigma$ is a set satisfying

$$\forall A \in C : H \cap C \neq \varnothing. \tag{9}$$

A *minimal hitting set* is a hitting set which cannot be further reduced without losing the characteristinc property (9). The problem of determining minimal hitting sets is known to be NP-hard [1], but we will see below that it reduces the effort of testing for failures refinement from a factor of 2^{Σ} to a factor that equals the number of minimal hitting sets.

For testing in the failures model, the following lemmas about hitting sets are required.

Lemma 2. *Let P, Q be two finite-state CSP processes satisfying $P \sqsubseteq_T Q$. For each $s \in trc(P)$, let $MINHIT(s)$ denote the collection of all minimal hitting sets of $minAcc(P/s)$. Then the following statements are equivalent.*

1. *$P \sqsubseteq_F Q$*
2. *For all $s \in trc(P) \cap trc(Q)$ and $H \in MINHIT(s)$, H is a (not necessariliy minimal) hitting set of $minAcc(Q/s)$.*

Proof. For showing "1 $\Rightarrow$ 2", assume that $P \sqsubseteq_F Q$ and suppose that $s \in \mathrm{trc}(P) \cap \mathrm{trc}(Q)$. Then Lemma 1, (6), implies that

$$\forall A_Q \in \mathrm{minAcc}(G(Q)/s) : \exists A_P \in \mathrm{minAcc}(G(P)/s) : A_P \subseteq A_Q$$

Therefore, every minimal acceptance set $A_Q \in \mathrm{minAcc}(G(Q)/s)$ contains a minimal acceptance set from minAcc(G(P)/s). As a consequence, any hitting set $H \in \mathrm{MINHIT}(s)$ also has a non-empty intersection with every $A_Q \in \mathrm{minAcc}(G(Q)/s)$.

To prove "2 $\Rightarrow$ 1", assume $P \sqsubseteq_T Q$, but $P \not\sqsubseteq_F Q$. According to Lemma 1, (6), there exists $s \in \mathrm{trc}(P) \cap \mathrm{trc}(Q)$ such that

$$\exists A_Q \in \mathrm{minAcc}(G(Q)/s) : \forall A_P \in \mathrm{minAcc}(G(P)/s) : A_P \not\subseteq A_Q(*)$$

Define

$$\overline{H} = \bigcup_{A_P \in \text{minAcc}(G(P)/s)} \big(A_P \setminus A_Q\big).$$

Since $A_P \setminus A_Q \neq \varnothing$ because of (*), $\overline{H}$ is a hitting set of $\text{minAcc}(G(P)/s)$ which has a non-empty intersection with A_Q. Minimising $\overline{H}$ induces the existence of a minimal hitting set $H \in \text{MINHIT}(s)$ which is *not* a hitting set of of $\text{minAcc}(G(Q)/s)$. This completes the proof of the lemma. □

3 Finite Complete Test Suites for CSP Trace Refinement

4 Finite Complete Test Suites for CSP Failures Refinement

5 Applications

5.1 Testing for Trace Equivalence

5.2 Testing for Trace Refinement

5.3 Testing for Failures Equivalence

5.4 Testing for Failures Refinement

6 Related Work

7 Conclusion

References

1. Book, R.V.: Karp richard m.. reducibility among combinatorial problems. complexity of computer computations, proceedings of a symposium on the complexity of computer computations, held march 20-22, 1972, at the ibm thomas j. watson center, yorktown heights, new york, edited by miller raymond e. and thatcher james w., plenum press, new york and london 1972, pp. 85–103. Journal of Symbolic Logic 40(4), 618–619 (1975)
2. Cavalcanti, A., da Silva Simão, A.: Fault-based testing for refinement in CSP. In: Yevtushenko, N., Cavalli, A.R., Yenigün, H. (eds.) Testing Software and Systems - 29th IFIP WG 6.1 International Conference, ICTSS 2017, St. Petersburg, Russia, October 9-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10533, pp. 21–37. Springer (2017), https://doi.org/10.1007/978-3-319-67549-7_2
3. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.: Failures Divergences Refinement (FDR) Version 3 (2013), https://www.cs.ox.ac.uk/projects/fdr/
4. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.: FDR3 — A Modern Refinement Checker for CSP. In: brahm, E., Havelund, K. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 8413, pp. 187–201 (2014)

5. Hennessy, M.: Algebraic Theory of Processes. MIT Press, Cambridge, MA, USA (1988)
6. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1985)
7. Peleska, J., Siegel, M.: Test automation of safety-critical reactive systems. South African Computer Jounal 19, 53–77 (1997)
8. Peleska, J., Siegel, M.: From testing theory to test driver implementation. In: Gaudel, M., Woodcock, J. (eds.) FME '96: Industrial Benefit and Advances in Formal Methods, Third International Symposium of Formal Methods Europe, Co-Sponsored by IFIP WG 14.3, Oxford, UK, March 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1051, pp. 538–556. Springer (1996), https://doi.org/10.1007/3-540-60973-3_106
9. Roscoe, A.W. (ed.): A Classical Mind: Essays in Honour of C. A. R. Hoare. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK (1994)
10. Roscoe, A.W.: Understanding Concurrent Systems. Springer, London, Dordrecht Heidelberg, New York (2010)
11. Schneider, S.: An operational semantics for timed csp. Inf. Comput. 116(2), 193–213 (Feb 1995), http://dx.doi.org/10.1006/inco.1995.1014