Information Processing Letters 110 (2010) 232–235

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Closures of may-, should- and must-convergences for contextual equivalence

Manfred Schmidt-Schauß *, David Sabel

FB Informatik und Mathematik, Institut für Informatik, Goethe-Universität, Postfach 11 19 32, D-60054 Frankfurt, Germany

ARTICLE INFO

Article history:
Received 19 January 2009
Received in revised form 13 August 2009
Accepted 11 January 2010
Available online 18 January 2010
Communicated by M. Yamashita

Keywords:
Formal semantics
Programming calculi
Program correctness

1. Introduction

For analyzing the semantics of non-deterministic and/or concurrent programming languages, there is an increasing use of combinations of may-, should-, and must-convergence predicates in connection with contextual equivalence. For deterministic calculi (for instance [1,9,5,10]), Morris' contextual equivalence based on termination, i.e. on may-convergence appears to be sufficient, where may-convergence means: $e{\downarrow} \Leftrightarrow \exists v : e \xrightarrow{*} v$ where v is a value, and where two program expressions s, t are contextually equivalent, if $P[s]{\downarrow} \Leftrightarrow P[t]{\downarrow}$ for every program context P.

For non-deterministic program calculi, may-convergence alone is arguably insufficient, since, e.g., the programs 0 and $(\texttt{choice}\ 0\bot)$ cannot be distinguished, and since under may-convergence alone, simple `choice` and bottom-avoiding `amb` cannot be distinguished. There are investigations using the combination of may-convergence and must-convergence (must-testing equivalence) (see [4,6]) and others based on the combination of may- and should-convergences (should-testing equivalence); see e.g. [2,12,8, 11]. Must-convergence of an expression e holds, denoted $e{\Downarrow}$, if there are no infinite reduction sequences starting from e, and should-convergence of e holds, denoted $e{\Downarrow}$, if every reduct of e is may-convergent (denoted $\Box{\downarrow}$ in this paper). Simplicity of the definition is the advantage of must-testing equivalence whereas the invariance of the contextual equivalence under the restriction to fair reduction sequences is an advantage of should-testing equivalence (see e.g. [2,12]). Should- and must-convergences have a different view on weakly divergent expressions where e is weakly divergent (see [7]) if it reduces infinitely but never looses the possibility to terminate successfully. A weakly divergent expression e is should-convergent, but not must-convergent.

As a consequence, should-testing equivalence is insensitive w.r.t. weak divergences, and on the other hand must-testing equivalence identifies weakly divergent expressions with expressions that may reduce to an error (which strongly diverges). Thus, it may also be reasonable to argue that all three convergences should be combined.

The modal–logical construction of should-convergence gives rise to the question, whether there are further such constructions resulting in new test predicates and corresponding contextual equivalences. We answer this question by the following results: Starting with $\downarrow$, and generating further convergence predicates using $\Box, \Diamond, \vee, \wedge$, and

* Corresponding author.
E-mail addresses: schauss@cs.uni-frankfurt.de (M. Schmidt-Schauß), sabel@cs.uni-frankfurt.de (D. Sabel).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.01.001

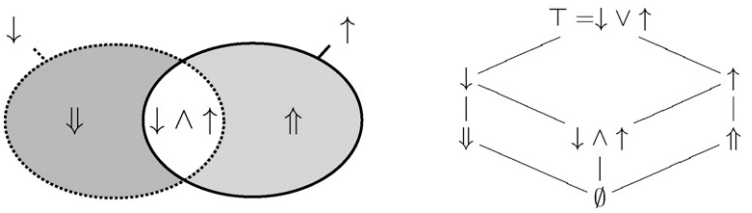

Fig. 1.

¬ leads to 8 convergence predicates, but w.r.t. contextual equivalence there is no change: only should-testing equivalence can be defined (Main Theorem 2.8). The second result (Theorem 3.9) is that starting with ⇓, using the same generators, an infinite family of convergence predicates will be generated, including ↓ and ⇓, and in general an infinite family of contextual equivalences.

2. Should- and must-testings

The triple $(E, V, \to)$ is called a *reduction structure*, provided $V \subseteq E \neq \emptyset$, $\to \subseteq E \times E$, and $e \to e' \Rightarrow e \notin V$. The reflexive transitive closure of $\to$ is denoted as $\xrightarrow{*}$. The idea is that E is the set of expressions of a programming calculus, $\to$ the small-step reduction relation, and V the (irreducible) values, i.e. successful outcomes of reductions. Note that there may be irreducible elements $e \in E$ with $e \notin V$, where $e \in E$ is called *irreducible*, iff there is no $e' \in E$ with $e \to e'$. We will analyze unary predicates over E, which are always written in postfix. The first predicate is eV, which holds iff $e \in V$. Note that $(eV \wedge e \xrightarrow{*} e')$ implies that $e = e'$. This predicate, however, will not be used for observations. We will also use the predicates $\top$ and $\emptyset$, where $e\top$ is always true, and $e\emptyset$ is always false. For predicates P, Q we write $P \subseteq Q$ if $eP \Rightarrow eQ$ for all reduction structures $(E, V, \to)$ and for all $e \in E$, and $P = Q$ iff $P \subseteq Q$ and $Q \subseteq P$. We write $P \neq Q$, iff for some reduction structure $(E, V, \to)$ and some $e \in E$, $eP \neq eQ$. The notation $P \subset Q$ means that $P \subseteq Q$ but $P \neq Q$.

Definition 2.1. We define the following predicate-generators: Given predicates P, Q, the following new predicates can be defined:

$$e(\Diamond P) := \exists e' : e \xrightarrow{*} e' \wedge e'P,$$

$$e(\Box P) := \forall e' : e \xrightarrow{*} e' \Rightarrow e'P,$$

$$e(\neg P) := \neg eP,$$

$$e(P \wedge Q) := eP \wedge eQ,$$

$$e(P \vee Q) := eP \vee eQ.$$

Given a predicate (or a set of predicates) P, $B^{\Box}_{\Diamond}(P)$ denotes the closure under all predicate generators, $N^{\Box}_{\Diamond}(P)$ denotes the closure under $\Box, \Diamond$ and $\neg$, and $B(P)$ denotes the Boolean closure.

Since the predicate closure corresponds to closing formulas in modal logic S4 (see [3]), we have chosen the corresponding modal operators $\Box$ and $\Diamond$ for universal and existential quantifications.

It is obvious that the usual propositional laws hold for the Boolean combinations. The proof of the following simple laws is left to the reader:

Lemma 2.2 *(Simplification rules). For all predicates P, Q:*

1. $\neg\Diamond P = \Box\neg P$,
2. $\neg\Box P = \Diamond\neg P$,
3. $\Box\Box P = \Box P$,
4. $\Diamond\Diamond P = \Diamond P$,
5. $\Diamond(P \vee Q) = \Diamond P \vee \Diamond Q$,
6. $\Box(P \wedge Q) = \Box P \wedge \Box Q$,
7. $\Box\emptyset = \Diamond\emptyset = \emptyset$,
8. $\Box\top = \Diamond\top = \top$,
9. $\Box P \subseteq P \subseteq \Diamond P$.

The predicates $\downarrow := \Diamond V$, $\Uparrow := \neg\downarrow$, $\uparrow := \Diamond\Uparrow$, and $\Downarrow := \neg\uparrow$ are called *may-convergence*, *must-divergence*, *strong may-divergence*, and *should-convergence*, respectively. Note that $\Uparrow = \neg\Diamond V = \Box\neg V$, $\uparrow = \Diamond\Box\neg V = \neg\Box\Diamond V$, and $\Downarrow = \Box\Diamond V$.

Since $\xrightarrow{*}$ is transitive and sV implies that s is irreducible, we obtain:

Lemma 2.3. *The set of predicates $\{\downarrow, \uparrow, \Uparrow, \Downarrow\}$ is closed w.r.t. negation. Also $\Downarrow \subseteq \downarrow$, $\Uparrow \subseteq \uparrow$, $V \subseteq \Downarrow$, and $\downarrow \vee \uparrow = \top$.*

The picture in Fig. 1 shows the complete set of expressions as a set diagram and a Hasse diagram (w.r.t. the $\subseteq$-ordering)

Theorem 2.4. $N^{\Box}_{\Diamond}(\downarrow) = \{\downarrow, \uparrow, \Uparrow, \Downarrow\}$.

Proof. This follows easily by induction and using Lemmas 2.3 and 2.2, and the remarks above. □

Theorem 2.5. $B^{\Box}_{\Diamond}(\downarrow) = \{\emptyset, \downarrow, \uparrow, \Uparrow, \Downarrow, \downarrow \wedge \uparrow, \Downarrow \vee \Uparrow, \top\}$.

Proof. This is shown by induction on the construction of predicates. Lemmas 2.2, 2.3 and Theorem 2.4 show that the claim holds for the constructions $\neg, \vee, \wedge$, and for $\Box$-construction with the exception of $\Box(\downarrow \wedge \uparrow)$ and $\Box(\Downarrow \vee \Uparrow)$. It is sufficient to check the $\Box$-construction. Lemma 2.2 and the proof of Theorem 2.4 show $\Box\downarrow \wedge \Box\uparrow = \Downarrow \wedge \Uparrow = \emptyset$. For $\Box(\Downarrow \vee \Uparrow)$, we have $\Box(\Downarrow \vee \Uparrow) \subseteq \Downarrow \vee \Uparrow$ by Lemma 2.2. Since $e\Downarrow \Rightarrow e\Box(\Downarrow \vee \Uparrow)$ and $e\Uparrow \Rightarrow e\Box(\Downarrow \vee \Uparrow)$, we have proved $\Box(\Downarrow \vee \Uparrow) = \Downarrow \vee \Uparrow$. □

Definition 2.6 *(Contextual preorder and equivalence).* Given a set $\mathcal{P}$ of predicates and set $\mathcal{F}$ of functions $f : E \to E$ (the