Closures of May and Must Convergence for Contextual Equivalence

Manfred Schmidt-Schauß and David Sabel

Fachbereich Informatik und Mathematik,
Institut für Informatik, Johann Wolfgang Goethe-Universität,
Postfach 11 19 32, D-60054 Frankfurt, Germany,
{schauss,sabel}@ki.informatik.uni-frankfurt.de

Technical Report Frank-35

December 10, 2008

Abstract. We show on an abstract level that contextual equivalence in non-deterministic program calculi defined by may- and must-convergence is maximal in the following sense. Using also all the test predicates generated by the Boolean, forall- and existential closure of may- and must-convergence does not change the contextual equivalence. The situation is different if may- and total must-convergence is used, where an expression totally must-converges if all reductions are finite and terminate with a value: There is an infinite sequence of test-predicates generated by the Boolean, forall- and existential closure of may- and total must-convergence, which also leads to an infinite sequence of different contextual equalities.

1 Introduction

We are interested in generalizations of may- and must-convergence predicates for contextual equivalence of non-deterministic and concurrent programming languages. Contextual equivalence in Morris' sense is based on termination, i.e. on may-convergence: $e{\downarrow} \iff \exists v : e \xrightarrow{*} v$ where v is a value. This notion is successfully used for deterministic calculi (for instance [Abr90,Pit97,MS99,Pit02]). If the investigation of contextual equivalence is applied to non-deterministic program calculi, then besides may-convergence – “there is some reduction to a value” – the branching structure of reduction sequences is also observed in the form of must-convergence, since contextual equivalence based on may-convergence only has insufficient discrimination power. E.g., bottom-avoiding choice can only be distinguished from erratic choice if contextual equivalence also tests for must-convergence [SSS08]. However, there are different versions of this test: One variant is the total must-convergence, denoted $e{\Downarrow}$, that is true iff all reductions originating in e are finite and terminate in a value. The other variant

is must-convergence, denoted $e\Downarrow$, that is true iff every successor of e is may-convergent. A conjunction of may- and total must-convergence is used in e.g. [KSS98,MSC99], and a conjunction of may- and must-convergence is used in e.g. [CHS05,SSS08,NSSSS07]. The latter combination is called *should testing* in the area of process algebras [RV07].
We will show in this paper that $\downarrow$ generates a finite class of test predicates using Boolean combinations and $\forall$ and $\exists$-generators, and that the corresponding contextual equivalence defined by the conjunction of $\downarrow$ and $\Downarrow$-testing already covers the equivalence w.r.t. the closure of $\downarrow$. We also show that the closure of $\Downarrow\!\!\Downarrow$ generates at least $\downarrow$ and $\Downarrow$ and in fact an infinite family of predicates leading to an infinite family of contextual congruences.
This shows that the combination of $\downarrow$ and $\Downarrow$ has the nice property of generating a contextual equivalence that it is invariant under closure of test predicates, which complements the advantage that fairness is built-in [CHS05,SSS08,RV07]. This is in contrast to the combinations with $\Downarrow\!\!\Downarrow$ whose closure leads to an infinite family of contextual equivalences, and, moreover is not useful for analyzing fairness.

2 May- and Must-Testing

The triple $(E, V, \to)$ is called a *reduction structure*, provided $V \subseteq E \neq \emptyset$, $\to \subseteq E \times E$, and $e \to e' \implies e \notin V$. 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 $eP \implies eQ$ 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:*

$$e(\exists P) := \exists e' : e \xrightarrow{*} e' \wedge e'P \qquad e(\neg P) := \neg eP$$
$$e(\forall P) := \forall e' : e \xrightarrow{*} e' \implies e'P \qquad e(P \wedge Q) := eP \wedge eQ$$
$$e(P \vee Q) := eP \vee eQ$$

Given a predicate (or a set of predicates) P, $B\forall\exists(P)$ denotes the closure under all predicate generators, $N\forall\exists(P)$ denotes the closure under $\forall, \exists$ and $\neg$, and $B(P)$ denotes the Boolean closure.

Note that the predicate closure corresponds to closing formulas in modal logic S4 (see [HC90]), where $\forall(P)$ corresponds to the modal operator $\Box P$, and $\exists(P)$ to the modal operator $\Diamond P$.

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:

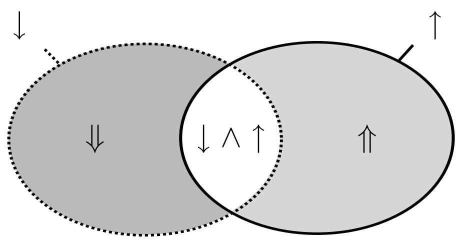
Lemma 2.2 (Simplification Rules). *For all predicates* P, Q*:*
1. $\neg\exists P = \forall\neg P$ *2.* $\neg\forall P = \exists\neg P$ *3.* $\forall\forall P = \forall P$
4. $\exists\exists P = \exists P$ *5.* $\exists(P \vee Q) = \exists P \vee \exists Q$ *6.* $\forall(P \wedge Q) = \forall P \wedge \forall Q$
7. $\forall\emptyset = \exists\emptyset = \emptyset$ *8.* $\forall\top = \exists\top = \top$ *9.* $\forall P \subseteq P \subseteq \exists P$

The predicates $\downarrow := \exists V$, $\Uparrow := \neg\downarrow$, $\uparrow := \exists\Uparrow$, and $\Downarrow := \neg\uparrow$ are called *may-convergence*, *must-divergence*, *may-divergence*, and *must-convergence*, respectively. Note that $\Uparrow = \neg\exists V = \forall\neg V$, $\uparrow = \exists\forall\neg V = \neg\forall\exists V$, and $\Downarrow = \forall\exists V$.
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* $\downarrow \vee \uparrow = \top$.

Proof. Using the representation above, the following is easy: $\neg\downarrow = \neg\exists V = \Uparrow$, $\neg\uparrow = \neg\exists\forall\neg V = \forall\exists V = \Downarrow$, $\neg\Uparrow = \neg\neg\exists V = \exists V = \downarrow$, and $\neg\Downarrow = \neg\neg\uparrow = \uparrow$. The subset relationships $\Downarrow \subseteq \downarrow$, $\Uparrow \subseteq \uparrow$ follow from Lemma 2.2. Hence the last equality holds. The relation $V \subseteq \Downarrow$ follows from irreducibility of elements e with eV and so the only reduction possibility is $e \xrightarrow{*} e$.

The following picture shows the complete set of expressions as a set diagram:



Theorem 2.4. $N\forall\exists(\downarrow) = \{\downarrow, \uparrow, \Uparrow, \Downarrow\}$.

Proof. We show by induction that constructing predicates cannot increase the set $\{\downarrow, \uparrow, \Uparrow, \Downarrow\}$. Lemma 2.3 shows that this holds for negation. It is sufficient to consider $\forall$-constructions. Obvious reasoning shows $\forall\downarrow = \Downarrow$, $\forall\Uparrow = \Uparrow$, and $\forall\Downarrow = \Downarrow$. The relation $\forall\uparrow = \Uparrow$ is proved as follows: Since $\Uparrow \subseteq \uparrow$, by monotonicity of $\forall$, we obtain $\Uparrow = \forall\Uparrow \subseteq \forall\uparrow$. To show the other direction, let $e\forall\uparrow$, and assume that $e\Uparrow$ is false. Then $e \xrightarrow{*} e'$ with $e'V$. However, since e' is irreducible, the predicate $e'\uparrow$ is wrong, hence we have a contradiction. This shows that $\forall\uparrow \subseteq \Uparrow$. □

Theorem 2.5. $B\forall\exists(\downarrow) = \{\emptyset, \downarrow, \uparrow, \Uparrow, \Downarrow, \downarrow \wedge \uparrow, \Downarrow \vee \Uparrow, \top\}$.

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 construction $\neg, \vee, \wedge$, and for $\forall$-constructions with the exception of $\forall(\downarrow \wedge \uparrow)$ and $\forall(\Downarrow \vee \Uparrow)$. It is sufficient to check the $\forall$-construction. Lemma 2.2 and the proof of Theorem 2.4

show $\forall\downarrow \wedge \forall\uparrow = \Downarrow \wedge \Uparrow = \emptyset$. For $\forall(\Downarrow \vee \Uparrow)$, we have $\forall(\Downarrow \vee \Uparrow) \subseteq \Downarrow \vee \Uparrow$ by Lemma 2.2. Since $e\Downarrow \implies e\forall(\Downarrow \vee \Uparrow)$ and $e\Uparrow \implies e\forall(\Downarrow \vee \Uparrow)$, we have proved $\forall(\Downarrow \vee \Uparrow) = \Downarrow \vee \Uparrow$. □

Definition 2.6. *Given a set $\mathcal{P}$ of predicates, we define the following preorders and equivalences on E:*

$$\begin{aligned} e_1 \leq_{\mathcal{P}} e_2 &:\iff \forall P \in \mathcal{P} : e_1 P \implies e_2 P \\ e_1 \sim_{\mathcal{P}} e_2 &:\iff \forall P \in \mathcal{P} : e_1 \leq_{\mathcal{P}} e_2 \wedge e_2 \leq_{\mathcal{P}} e_1 \end{aligned}$$

The following considerations for these orderings are transferrable also to contextually defined orderings and equivalences.

Lemma 2.7. *Let e_1, e_2 be expressions with $e_1\downarrow \iff e_2\downarrow$ and $e_1\Downarrow \iff e_2\Downarrow$. Then $e_1(\downarrow \wedge \uparrow) \iff e_2(\downarrow \wedge \uparrow)$ and $e_1(\Downarrow \vee \Uparrow) \iff e_2(\Downarrow \vee \Uparrow)$.*

The conclusion is that the equivalence corresponding to all test predicates is the same as the equivalence defined by the two test predicates $\downarrow$ and $\Downarrow$.

Main Theorem 2.8 $\sim_{\{\downarrow,\Downarrow\}} = \sim_{B\forall\exists(\downarrow)} = \sim_{N\forall\exists(\downarrow)}$.

This does not hold for respective preorders, since e.g. $\leq_{\{\downarrow,\Downarrow\}} \neq \leq_{\{\downarrow,\Uparrow\}}$.

3 Analyzing the Total-Must-Predicate

In this section we consider also the predicate that tests whether for an expression all (maximal) reduction sequences end in a value in V.

Definition 3.1. Total must-convergence *is defined as $e\Ddownarrow$ iff every $\to$-reduction sequence of e is finite and for every irreducible e' with $e \xrightarrow{*} e'$, it is $e'V$. The negation of $\Ddownarrow$ is defined as $e\upuparrows := \neg(e\Ddownarrow)$*

The following reduction structure $\mathcal{R} = (E_0, V_0, \to_0)$ is used to provide examples: The set E_0 is inductively defined as $\{p_0, \mathtt{T}, \bot\} \cup \{e_1 \oplus e_2 \mid e_1, e_2 \in E_0\}$, $V_0 := \{\mathtt{T}\}$, and $\to_0 = \{p_0 \to \mathtt{T}, p_0 \to p_0, \bot \to \bot, e_1 \oplus e_2 \to e_1, e_1 \oplus e_2 \to e_2\}$.

Lemma 3.2. *The following equivalences and relations hold:*
$\forall\Ddownarrow = \Ddownarrow, \forall\upuparrows = \Uparrow, \exists\Ddownarrow = \downarrow, \exists\upuparrows = \upuparrows$.
$\Ddownarrow \subset \Downarrow \subset \downarrow$, *and* $\Uparrow \subset \uparrow \subset \upuparrows$.

Proof. This can be proved by standard reasoning. The example p_0 of $\mathcal{R}$ satisfies $p_0\Downarrow$, but also $p_0\upuparrows$, and thus shows that $\Ddownarrow \neq \Downarrow$. □

Theorem 3.3. $N\forall\exists(\Ddownarrow) = \{\downarrow, \uparrow, \Uparrow, \Downarrow, \Ddownarrow, \upuparrows\}$.

Proof. Follows from Lemma 3.2 and Theorem 2.4. □

The Boolean closure of $\{\downarrow, \uparrow, \Uparrow, \Downarrow, \Ddownarrow, \upuparrows\}$ are the 16 predicates generated from the mutually disjoint 4 predicates: $\Ddownarrow, (\upuparrows \wedge \Downarrow), (\downarrow \wedge \uparrow), \Uparrow$.

Corollary 3.4. $\sim_{\{\downarrow,\Downarrow,\Ddownarrow\}} = \sim_{B(\{\downarrow,\Downarrow,\Ddownarrow\})}$

Corollary 3.5. $\leq_{\{\downarrow,\Downarrow,\Ddownarrow\}} \neq \leq_{\{\downarrow,\Ddownarrow\}}$

3.1 Infinity of the Closure of Total Must-Convergence

We show below that the set $B\forall\exists(\Downarrow\!\Downarrow)$ is infinite. After having analyzed three levels by alternating Boolean- and $\forall$-closure, we could construct an infinite sequence of predicates, and an infinite sequence of elements of $\mathcal{R}$:

$$A_1 := \downarrow \wedge \uparrow \wedge \forall(\Downarrow\!\Downarrow \vee \uparrow) \qquad A_2 := \bar{A}_1 \wedge \forall(\Downarrow \vee \bar{A}_1 \vee \Uparrow)$$
$$\bar{A}_1 := \downarrow \wedge \uparrow \wedge \neg(\forall(\Downarrow\!\Downarrow \vee \uparrow)) \qquad \bar{A}_2 := \bar{A}_1 \wedge \neg(\forall(\Downarrow \vee \bar{A}_1 \vee \Uparrow))$$
$$A_i := \bar{A}_{i-1} \wedge \forall(\Downarrow \vee \bar{A}_{i-1} \vee A_{i-2} \vee \ldots \vee A_1 \vee \Uparrow)$$
$$\bar{A}_i := \bar{A}_{i-1} \wedge \neg(\forall(\Downarrow \vee \bar{A}_{i-1} \vee A_{i-2} \vee \ldots \vee A_1 \vee \Uparrow))$$

Let $a_1 := \mathtt{T} \oplus \bot$, $a_2 := \bot \oplus p_0$, $a_3 := a_1 \oplus p_0$, and for $i \geq 4$, let $a_i := a_{i-2} \oplus a_{i-3}$. Some obvious properties of $A_i, \bar{A}_i$ are

Lemma 3.6. *For all $i \geq 1$: $A_i \subseteq \downarrow \wedge \uparrow$ and $\bar{A}_i \subseteq \downarrow \wedge \uparrow$.*
For $i \geq 1$: $A_i \cap \bar{A}_i = \emptyset$ and for all $i \geq 2$: $A_i \cup \bar{A}_i = \bar{A}_{i-1}$.
For all $i \neq j$: $A_i \cap A_j = \emptyset$.

Lemma 3.7. *For all $i \geq 2$: $A_i = \bar{A}_{i-1} \wedge \neg(\exists A_{i-1})$ and $\bar{A}_i = \bar{A}_{i-1} \wedge \exists A_{i-1}$*

Proof. We compute an equivalent of $\neg(\forall(\Downarrow \vee \bar{A}_{i-1} \vee A_{i-2} \vee \ldots \vee A_1 \vee \Uparrow))$: The first step produces $\exists(\downarrow \wedge \uparrow \wedge \neg\bar{A}_{i-1} \wedge \neg A_{i-2} \wedge \ldots \wedge \neg A_1)$: We have that $\downarrow \wedge \uparrow \wedge \neg A_1 = \bar{A}_1$. By induction on j, we obtain that $\neg A_j \wedge \bar{A}_{j-1} = \bar{A}_j$. Finally, we obtain $\neg\bar{A}_{i-1} \wedge \bar{A}_{i-2} = A_{i-1}$. Hence, $\bar{A}_i = \bar{A}_{i-1} \wedge \exists A_{i-1}$. It is easy to see that this also implies $A_i = \bar{A}_{i-1} \wedge \neg(\exists A_{i-1})$. □

Corollary 3.8. $\bar{A}_i = \bar{A}_1 \wedge \exists A_1 \wedge \ldots \wedge \exists A_{i-1}$ *which is equivalent to* $(\downarrow \wedge \uparrow \wedge \exists(\Downarrow \wedge \Uparrow\!\uparrow)) \wedge \exists A_1 \wedge \ldots \wedge \exists A_{i-1}$.

Lemma 3.9. *For all i : $a_i A_i$ holds.*

Proof. Inspection of the definitions shows $a_1 A_1$. Since $a_2 \xrightarrow{*} p_0$, we have $a_2 \exists(\Downarrow \wedge \Uparrow\!\uparrow)$. Since $\bot\uparrow$, $p_0\downarrow$ and $p_0(\Downarrow \wedge \Uparrow\!\uparrow)$, we also have $a_2\bar{A}_1$. But then also $a_2 A_2$ holds. Similar arguments show $a_3\bar{A}_2$, and since $A_3 = \bar{A}_2 \wedge \forall(\Downarrow \vee \bar{A}_2 \vee A_1 \vee \Uparrow)$ and scanning the successors of a_3, we see that $a_3 \xrightarrow{*} a_1 A_1$, and that the second part holds, hence $a_3 A_3$.
By simultaneous induction on i we show the following 4 claims:

1. for all $i \geq 2 : a_i\bar{A}_1$.
2. For all $i \geq 3, j = 1, \ldots, i-2$: $a_i \exists A_j$.
3. For $i \geq 1 : a_i\bar{A}_{i-1}$.
4. For $i \geq 1 : a_i A_i$ holds.

Now we give the proofs for every item, where we can use the induction hypothesis for all claims and for smaller i.

1. For a_3, this can be seen by the same arguments. For $i \geq 4$: $a_{i-2}\bar{A}_1$, since $i - 2 \geq 2$ and by induction hypothesis, and hence also $a_i\bar{A}_1$.
2. The base cases are $i = 3, 4$. For a_3, claim (2), which is only $a_3 A_1$, follows from the definition. For a_4, we have $a_4 \xrightarrow{*} a_2$ and $a_4 \xrightarrow{*} a_1$. By induction hypothesis, the claims $a_j A_j$ hold for $j < i$. Now the general case is $a_i \xrightarrow{*} a_{i-2}$ and $a_i \xrightarrow{*} a_{i-3}$, and by induction and transitivity of $\xrightarrow{*}$, the claim is proved.
3. For $i \geq 1 : a_i\bar{A}_{i-1}$. Item (1) shows $a_i\bar{A}_1$. Item (2) shows that $a_i \exists A_j$ holds for all $j = 1, \ldots, i-2$. By Corollary 3.8, this shows $a_i\bar{A}_{i-1}$.

4. $a_i A_i$ holds: The base cases $i = 1, 2, 3$ are already proved. Let $i \geq 4$: we already have shown that $a_i \bar{A}_{i-1}$. Now it suffices to scan all successors. Either the successors are in $\Downarrow \vee \Uparrow$, or $a_i \bar{A}_{i-1}$ or for $j \leq i-2$: it is $a_j A_j$. This satisfies the definition $A_i = \bar{A}_{i-1} \wedge \forall(\Downarrow \vee \bar{A}_{i-1} \vee A_{i-2} \vee \ldots \vee A_1 \vee \Uparrow)$.

Theorem 3.10. *The set $B\forall\exists(\Downarrow\!\!\Downarrow)$ is not finite.*

Corollary 3.11. *There is no finite set of predicates $M' \subseteq B\forall\exists(\Downarrow\!\!\Downarrow)$ such that* $\sim_{M'} = \sim_{B\forall\exists(\Downarrow\!\Downarrow)}$.

Acknowledgements We thank Jan Schwinghammer for his valuable comments.

References

Abr90. Samson Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1990.

CHS05. Arnaud Carayol, Daniel Hirschkoff, and Davide Sangiorgi. On the representation of McCarthy's amb in the pi-calculus. *Theoret. Comput. Sci.*, 330(3):439–473, 2005.

HC90. G. E. Hughes and M. J. Cresswell. *Introduction to Modal Logic*. Routledge,, London, 1990.

KSS98. Arne Kutzner and Manfred Schmidt-Schauß. A nondeterministic call-by-need lambda calculus. In *International Conference on Functional Programming 1998*, pages 324–335. ACM Press, 1998.

MS99. A. K. D. Moran and D. Sands. Improvement in a lazy context: An operational theory for call-by-need. In *POPL 1999*, pages 43–56. ACM Press, 1999.

MSC99. Andrew K. D. Moran, David Sands, and Magnus Carlsson. Erratic fudgets: A semantic theory for an embedded coordination language. In *Coordination '99*, volume 1594 of *Lecture Notes in Comput. Sci.*, pages 85–102. Springer-Verlag, 1999.

NSSSS07. Joachim Niehren, David Sabel, Manfred Schmidt-Schauß, and Jan Schwinghammer. Observational semantics for a concurrent lambda calculus with reference cells and futures. *Electron. Notes Theor. Comput. Sci.*, 173:313–337, 2007.

Pit97. Andrew M. Pitts. Operationally-based theories of program equivalence. In *Semantics and Logics of Computation*. Cambridge University Press, 1997.

Pit02. Andrew M. Pitts. Operational semantics and program equivalence. In J. T. O'Donnell, editor, *Applied Semantics*, volume 2395 of *Lecture Notes in Computer Science*, pages 378–412. Springer-Verlag, 2002.

RV07. Arend Rensink and Walter Vogler. Fair testing. *Inform. and Comput.*, 205(2):125–198, 2007.

SSS08. David Sabel and Manfred Schmidt-Schauß. A call-by-need lambda-calculus with locally bottom-avoiding choice: Context lemma and correctness of transformations. *Math. Structures Comput. Sci.*, 18(03):501–553, 2008.

APPENDIX

A Analyzing the Closure for Total Must Testing

A.1 The First Level

The table 1 shows the predicates that correspond to $\forall P$ for all Boolean combinations P of the four basic sets $\Ddownarrow, (\upuparrows \wedge \Downarrow), (\downarrow \wedge \uparrow), \Uuparrow$. It is sufficient to look for the $\forall$-construction only. The only predicate that cannot be represented is $\forall(\Ddownarrow \vee \uparrow)$: It is obvious that $\Downarrow \vee \Uuparrow \subseteq \forall(\Ddownarrow \vee \uparrow) \subseteq \Ddownarrow \vee \uparrow$. We only have to show that the inclusions are proper. The element $\bot \oplus \mathtt{T}$ does not satisfy $\Downarrow \vee \Uuparrow$, but $\forall(\Ddownarrow \vee \uparrow)$. The element $\bot \oplus p_0$ satisfies $\Ddownarrow \vee \uparrow$, but not $\forall(\Ddownarrow \vee \uparrow)$, since p_0 does mot satisfy $\Ddownarrow \vee \uparrow$.

$\forall \Ddownarrow$	$= \Ddownarrow$
$\forall \Uuparrow$	$= \Uuparrow$
$\forall(\upuparrows \wedge \Downarrow)$	$= \emptyset$
$\forall(\downarrow \wedge \uparrow)$	$= \emptyset$
$\forall \Downarrow$	$= \Downarrow$
$\forall(\Ddownarrow \vee (\downarrow \wedge \uparrow))$	$= \Ddownarrow$
$\forall(\Ddownarrow \vee \Uuparrow)$	$= \Ddownarrow \vee \Uuparrow$
$\forall((\Downarrow \wedge \upuparrows) \vee (\downarrow \wedge \uparrow))$	$= \emptyset$
$\forall((\Downarrow \wedge \upuparrows) \vee \Uuparrow)$	$= \Uuparrow$
$\forall \uparrow$	$= \Uuparrow$
$\forall(\Downarrow \vee (\downarrow \wedge \uparrow))$	$= \Downarrow$
$\forall(\Downarrow \vee \Uuparrow)$	$= \Downarrow \vee \Uuparrow$
$\forall(\Ddownarrow \vee \uparrow)$	= a new test predicate
$\forall \upuparrows$	$= \Uuparrow$

Fig. 1. Predicates using $\forall$ on the first level

For convenience, we abbreviate two new components as follows:

$$A := \downarrow \wedge \uparrow \wedge \forall(\Ddownarrow \vee \uparrow)$$
$$\bar{A} := \downarrow \wedge \uparrow \wedge \neg(\forall(\Ddownarrow \vee \uparrow))$$

Now the sets on this level can be illustrated in the following diagram. There are now 5 basic sets:

<table>
<tr><td rowspan="2">$\wedge$</td><td rowspan="2">$\Downarrow$</td><td colspan="2">$\downarrow \wedge \uparrow$</td><td rowspan="2">$\Uuparrow$</td></tr>
<tr><td>$\bar{A}$</td><td>A</td></tr>
<tr><td>$\Ddownarrow$</td><td></td><td>$\emptyset$</td><td>$\emptyset$</td><td>$\emptyset$</td></tr>
<tr><td>$\upuparrows$</td><td></td><td></td><td></td><td></td></tr>
</table>

Using the refined sets we have to check 32 combinations on the next level, among them 16 new combinations, which are presented in the table 2.

$\forall A$	$= \emptyset$	
$\forall \bar{A}$	$= \emptyset$	
$\forall(\Downarrow\!\!\Downarrow \vee A)$	$= \Downarrow\!\!\Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee \bar{A})$	$= \Downarrow\!\!\Downarrow$	
$\forall((\upuparrows \wedge \Downarrow) \vee A)$	$= \emptyset$	
$\forall((\upuparrows \wedge \Downarrow) \vee \bar{A})$	$= \emptyset$	
$\forall(A \vee \Uparrow)$	$= \Uparrow$	
$\forall(\bar{A} \vee \Uparrow)$	$= \Uparrow$	
$\forall(\Downarrow \vee A)$	$= \Downarrow$	
$\forall(\Downarrow \vee \bar{A})$	$= \Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee A \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee A \vee \Uparrow$	see Lemma A.1
$\forall(\Downarrow\!\!\Downarrow \vee \bar{A} \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee \Uparrow$	see Lemma A.2
$\forall((\Downarrow \wedge \upuparrows) \vee A \vee \Uparrow$	$= \Uparrow$	
$\forall((\Downarrow \wedge \upuparrows) \vee \bar{A} \vee \Uparrow$	$= \Uparrow$	
$\forall(\Downarrow \vee A \vee \Uparrow)$	$= \Downarrow \vee A \vee \Uparrow$	see Lemma A.3
$\forall(\Downarrow \vee \bar{A} \vee \Uparrow)$	= a new test predicate	see Lemma A.4

Fig. 2. New cases using $\forall$ on the second level

Lemma A.1. $\forall(\Downarrow\!\!\Downarrow \vee A \vee \Uparrow) \;=\; \Downarrow\!\!\Downarrow \vee A \vee \Uparrow$

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee A \vee \Uparrow)$. So assume that sA. We have to show that for every s' with $s \xrightarrow{*} s'$: $s'(\Downarrow\!\!\Downarrow \vee A \vee \Uparrow)$. Note that sA means $s(\downarrow \wedge \uparrow \wedge \forall(\Downarrow\!\!\Downarrow \vee \uparrow))$. The condition $s\forall(\Downarrow\!\!\Downarrow \vee \uparrow)$ shows that $s'\neg(\upuparrows \wedge \Downarrow)$. So, it remains to show that $s'(\downarrow \wedge \uparrow)$ implies that $s'A$. Suppose that this is false. Then $s'(\downarrow \wedge \uparrow \wedge \neg(\forall(\Downarrow\!\!\Downarrow \vee \uparrow)))$, which is equivalent to $s'(\downarrow \wedge \uparrow \wedge \exists(\upuparrows \wedge \Downarrow)))$. Then there is some s'' with $s' \xrightarrow{*} s''$ and $s''(\upuparrows \wedge \Downarrow)$. But this contradicts the facts $s \xrightarrow{*} s' \xrightarrow{*} s''$ and $s(\forall(\Downarrow\!\!\Downarrow \vee \uparrow))$.

Lemma A.2. $\forall(\Downarrow\!\!\Downarrow \vee \bar{A} \vee \Uparrow) \;=\; \Downarrow\!\!\Downarrow \vee \Uparrow$.

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee \bar{A} \vee \Uparrow)$. So assume that $s\bar{A}$. Note that $s\bar{A}$ means $s(\downarrow \wedge \uparrow \wedge \neg(\forall(\Downarrow\!\!\Downarrow \vee \uparrow)))$, which in turn is equivalent to $s(\downarrow \wedge \uparrow \wedge \exists(\upuparrows \wedge \Downarrow))$. The condition $s\exists(\upuparrows \wedge \Downarrow))$ contradicts $s(\forall(\Downarrow\!\!\Downarrow \vee \bar{A} \vee \Uparrow))$.

Lemma A.3. $\forall(\Downarrow \vee A \vee \Uparrow) \;= \Downarrow \vee A \vee \Uparrow$

Proof. It is easy to see that $\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow \vee A \vee \Uparrow)$. So assume that sA. We have to show that for every s' with $s \xrightarrow{*} s'$: $s'(\Downarrow \vee A \vee \Uparrow)$. Note that sA means $s(\downarrow \wedge \uparrow \wedge \forall(\Downarrow\!\!\Downarrow \vee \uparrow))$. The condition $s\forall(\Downarrow\!\!\Downarrow \vee \uparrow)$ shows that $s'\neg(\upuparrows \wedge \Downarrow)$. So, it remains to show that $s'(\downarrow \wedge \uparrow)$ implies that $s'A$.
Suppose that this is false. Then $s'(\downarrow \wedge \uparrow \wedge \neg(\forall(\Downarrow\!\!\Downarrow \vee \uparrow)))$, which is equivalent to $s'(\downarrow \wedge \uparrow \wedge \exists(\upuparrows \wedge \Downarrow)))$. Then there is some s'' with $s' \xrightarrow{*} s''$ and $s''(\upuparrows \wedge \Downarrow)$. But this contradicts the facts $s \xrightarrow{*} s' \xrightarrow{*} s''$ and $s(\forall(\Downarrow\!\!\Downarrow \vee \uparrow))$.

Lemma A.4. $\Downarrow \vee \Uparrow \subset \forall(\Downarrow \vee \bar{A} \vee \Uparrow) \;\subset \Downarrow \vee \bar{A} \vee \Uparrow$.

Proof. Lemma 2.2 shows that $\forall(\Downarrow \vee \bar{A} \vee \Uparrow) \subseteq \Downarrow \vee \bar{A} \vee \Uparrow$. It is easy to see that $\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow \vee \bar{A} \vee \Uparrow)$. Note that for a process s: $s\bar{A}$ means $s(\downarrow \wedge \uparrow \wedge \neg(\forall(\Ddownarrow \vee \uparrow)))$, which is equivalent to $s(\downarrow \wedge \uparrow \wedge \exists(\upuparrows \wedge \Downarrow))$.
Now we construct the examples. The following process $p_3 := (\mathtt{T} \oplus \bot) \oplus p_0$ satisfies $p_3\bar{A}$, but $p_3 \xrightarrow{*} (\mathtt{T} \oplus \bot)$ with $(\mathtt{T} \oplus \bot)A$. Hence $\forall(\Downarrow \vee \bar{A} \vee \Uparrow) \neq \Downarrow \vee \bar{A} \vee \Uparrow$.
For the element $p = (\bot \oplus p_0)$ it is obvious that $p\neg(\Downarrow \vee \Uparrow)$, but for every reduct s' of p the test $s'(\Downarrow \vee \bar{A} \vee \Uparrow)$ is true. Suppose that $(\Downarrow \vee \Uparrow)$ fails for s'. Then $s' = p$, which satisfies $p(\downarrow \wedge \uparrow \wedge \exists(\upuparrows \wedge \Downarrow))$, and hence $p\bar{A}$. Hence $\Downarrow \vee \Uparrow \neq \forall(\Downarrow \vee \bar{A} \vee \Uparrow)$.

If we use the abbreviation: $B := \bar{A} \wedge \forall(\Downarrow \vee \bar{A} \vee \Uparrow)$ and $\bar{B} := \bar{A} \wedge \neg(\forall(\Downarrow \vee \bar{A} \vee \Uparrow))$, then the following table illustrates the 6 basic sets on the next level:

$\wedge$	$\Downarrow$	$\downarrow \wedge \uparrow$			$\Uparrow$
		A		A	
		B	B		
$\Ddownarrow$		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\upuparrows$					

Some properties of A, B are:

Lemma A.5.

1. $\forall(\Downarrow \vee \bar{A} \vee \Uparrow) = \neg(\exists A)$. *Thus* $B = \bar{A} \wedge \neg(\exists A)$ *and* $\bar{B} = \bar{A} \wedge \exists A$.
2. $B \subseteq \forall(\neg(\bar{B}))$.

Proof. 1. We compute $\neg(\forall(\Downarrow \vee \bar{A} \vee \Uparrow))$: Then $\exists(\uparrow \wedge (\neg(\bar{A})) \wedge \downarrow)) = \exists((\uparrow \wedge \downarrow) \wedge (\Uparrow \vee \Downarrow \vee (\forall(\Ddownarrow \vee \uparrow)))) = \exists((\uparrow \wedge \downarrow \wedge \Uparrow) \vee (\uparrow \wedge \downarrow \wedge \Downarrow) \vee (\uparrow \wedge \downarrow \wedge \forall(\Ddownarrow \vee \uparrow))))$ $= \exists(\uparrow \wedge \downarrow \wedge \forall(\Ddownarrow \vee \uparrow)) = \exists(A)$.

2. Suppose there is some bB such that $b \xrightarrow{*} b'$ with $b'\bar{B}$. The latter is equivalent to $b'\bar{A} \wedge b'\neg(\forall(\Downarrow \vee \bar{A} \vee \Uparrow))$. In particular, there is some $b' \xrightarrow{*} b''$ with $b''\neg(\Downarrow \vee \bar{A} \vee \Uparrow)$. Transitivity of $\xrightarrow{*}$ shows that $b \xrightarrow{*} b''$. However, bB implies that $b\forall(\Downarrow \vee \bar{A} \vee \Uparrow))$. Hence there is no such b'.

Some witnesses for the elements of $A, \bar{A}, B, \bar{B}$ are in the following lemma:

Lemma A.6.

1. A *contains* $\mathtt{T} \oplus \bot$
2. $\bar{A}$ *contains* $p := \bot \oplus p_0$.
3. $B \subset \bar{A}$ *contains* $\bot \oplus p_0$
4. $\bar{B} \subset \bar{A}$ *contains* $(\mathtt{T} \oplus \bot) \oplus p_0$

A.2 The Third Level

The abbreviations and an alternative formulation are:

$$
\begin{aligned}
A &:= \downarrow \wedge \uparrow \wedge \forall(\Ddownarrow \vee \uparrow) \\
\bar{A} &:= \downarrow \wedge \uparrow \wedge \neg(\forall(\Ddownarrow \vee \uparrow)) \\
B &:= \bar{A} \wedge \forall(\Downarrow \vee \bar{A} \vee \Uparrow) &&= \bar{A} \wedge \neg(\exists A) \\
\bar{B} &:= \bar{A} \wedge \neg(\forall(\Downarrow \vee \bar{A} \vee \Uparrow)) &&= \bar{A} \wedge \exists A
\end{aligned}
$$

Using the refined sets we have to check 64 combinations on the next level, among them 32 new combinations, the combinations without A are presented in table 3.

$\forall B$	$= \emptyset$	
$\forall \bar{B}$	$= \emptyset$	
$\forall(\Downarrow\!\!\Downarrow \vee B)$	$= \Downarrow\!\!\Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee \bar{B})$	$= \Downarrow\!\!\Downarrow$	
$\forall((\Uparrow\!\!\Uparrow \wedge \Downarrow) \vee B)$	$= \emptyset$	
$\forall((\Uparrow\!\!\Uparrow \wedge \Downarrow) \vee \bar{B})$	$= \emptyset$	
$\forall(B \vee \Uparrow)$	$= \Uparrow$	
$\forall(\bar{B} \vee \Uparrow)$	$= \Uparrow$	
$\forall(\Downarrow \vee B)$	$= \Downarrow$	
$\forall(\Downarrow \vee \bar{B})$	$= \Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee B \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee \Uparrow$	see Lemma A.7
$\forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee \Uparrow$	see Lemma A.8
$\forall((\Downarrow \wedge \Uparrow\!\!\Uparrow) \vee B \vee \Uparrow$	$= \Uparrow$	
$\forall((\Downarrow \wedge \Uparrow\!\!\Uparrow) \vee \bar{B} \vee \Uparrow$	$= \Uparrow$	
$\forall(\Downarrow \vee B \vee \Uparrow)$	$= \Downarrow \vee B \vee \Uparrow$	see Lemma A.9
$\forall(\Downarrow \vee \bar{B} \vee \Uparrow)$	$= \Downarrow \vee \Uparrow$	see Lemma A.10

Fig. 3. New cases without A using $\forall$ on the third level

Lemma A.7. $\forall(\Downarrow\!\!\Downarrow \vee B \vee \Uparrow) \;=\; \Downarrow\!\!\Downarrow \vee \Uparrow$.

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee B \vee \Uparrow) \subseteq \Downarrow\!\!\Downarrow \vee B \vee \Uparrow$. We only have to consider sB. Since $B \subseteq \bar{A}$, the claim follows from Lemma A.2.

Lemma A.8. $\forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee \Uparrow) = \Downarrow\!\!\Downarrow \vee \Uparrow$.

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee \Uparrow)$. So assume that $s\bar{B}$. Since $\bar{B} \subseteq \bar{A}$, the claim follows from Lemma A.2.

Lemma A.9. $\forall(\Downarrow \vee B \vee \Uparrow) = \Downarrow \vee B \vee \Uparrow$

Proof. It is easy to see that $\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow \vee B \vee \Uparrow) \subseteq \Downarrow \vee B \vee \Uparrow$. So assume that sB. We have to show that for every s' with $s \xrightarrow{*} s'$: $s'(\Downarrow \vee B \vee \Uparrow)$. Note that sB means $s(\bar{A} \wedge \forall(\Downarrow \vee \bar{A} \vee \Uparrow))$. The condition $s(\forall(\Downarrow \vee \bar{A} \vee \Uparrow))$ shows that $s'(\Downarrow \vee \bar{A} \vee \Uparrow)$. The case $s'\bar{B}$ is not possible due to Lemma A.5. Hence $s'(\Downarrow \vee B \vee \Uparrow)$ holds, and the lemma is proved.

Lemma A.10. $\forall(\Downarrow \vee \bar{B} \vee \Uparrow) \;= \Downarrow \vee \Uparrow$.

Proof. The relations $\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow \vee \bar{B} \vee \Uparrow) \subseteq \Downarrow \vee \bar{B} \vee \Uparrow$ follow easily. Note that $s\bar{B}$ means $s\bar{A} \wedge \exists A$. Hence there is some $s'A$ with $s \xrightarrow{*} s'$. Hence $s\neg\forall(\Downarrow \vee \bar{B} \vee \Uparrow)$. and the Lemma is proved.

$\forall B \vee A$	$= \emptyset$	
$\forall \bar{B} \vee A$	$= \emptyset$	
$\forall(\Downarrow\!\!\Downarrow \vee A \vee B)$	$= \Downarrow\!\!\Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee A)$	$= \Downarrow\!\!\Downarrow$	
$\forall((\Uparrow \wedge \Downarrow) \vee B \vee A)$	$= \emptyset$	
$\forall((\Uparrow \wedge \Downarrow) \vee \bar{B} \vee A)$	$= \emptyset$	
$\forall(B \vee A \vee \Uparrow)$	$= \Uparrow$	
$\forall(\bar{B} \vee A \vee \Uparrow)$	$= \Uparrow$	
$\forall(\Downarrow \vee B \vee A)$	$= \Downarrow$	
$\forall(\Downarrow \vee \bar{B} \vee A)$	$= \Downarrow$	
$\forall(\Downarrow\!\!\Downarrow \vee B \vee A \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee \Uparrow$	see Lemma A.11
$\forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee A \vee \Uparrow)$	$= \Downarrow\!\!\Downarrow \vee A \vee \Uparrow$	see Lemma A.12
$\forall((\Downarrow \wedge \Uparrow) \vee B \vee A \vee \Uparrow$	$= \Uparrow$	
$\forall((\Downarrow \wedge \Uparrow) \vee \bar{B} \vee A \vee \Uparrow$	$= \Uparrow$	
$\forall(\Downarrow \vee B \vee A \vee \Uparrow)$	$= \Downarrow \vee B \vee A \vee \Uparrow$	see Lemma A.13
$\forall(\Downarrow \vee \bar{B} \vee A \vee \Uparrow)$	$\supset \Downarrow \vee A \vee \Uparrow$	see Lemma A.14

Fig. 4. New cases with A using $\forall$ on the third level

Now we present the new combinations with A in table 4.

Lemma A.11. $\forall(\Downarrow\!\!\Downarrow \vee B \vee A \vee \Uparrow) \;=\; \Downarrow\!\!\Downarrow \vee A \vee \Uparrow$.

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee B \vee A \vee \Uparrow) \subseteq \Downarrow\!\!\Downarrow \vee B \vee A \vee \Uparrow$. Lemma A.2 shows that $\Downarrow\!\!\Downarrow \vee A \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee B \vee A \vee \Uparrow)$. We only have to consider sB. Since $B \subseteq \bar{A}$, the claim follows similar as in the proof of Lemma A.2.

Lemma A.12. $\forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee A \vee \Uparrow) \;=\; \Downarrow\!\!\Downarrow \vee A \vee \Uparrow$.

Proof. It is easy to see that $\Downarrow\!\!\Downarrow \vee A \vee \Uparrow \subseteq \forall(\Downarrow\!\!\Downarrow \vee \bar{B} \vee A \vee \Uparrow)$. So assume that $s\bar{B}$. Since $\bar{B} \subseteq \bar{A}$, the claim follows similar as in the proof of Lemma A.2.

Lemma A.13. $\forall(\Downarrow \vee B \vee A \vee \Uparrow) \;= \Downarrow \vee B \vee A \vee \Uparrow$

Proof. It is easy to see that $\Downarrow \vee A \vee \Uparrow \subseteq \forall(\Downarrow \vee B \vee \Uparrow) \subseteq \Downarrow \vee B \vee A \vee \Uparrow$. The claim now follows from Lemmas A.3 and A.9.

Lemma A.14. $= \Downarrow \vee A \vee \Uparrow \subset \forall(\Downarrow \vee \bar{B} \vee A \vee \Uparrow) \subset \Downarrow \vee \bar{B} \vee A \vee \Uparrow)$

Proof. The relations $\Downarrow \vee A \vee \Uparrow \subseteq \forall(\Downarrow \vee \bar{B} \vee A \vee \Uparrow) \subseteq \Downarrow \vee \bar{B} \vee A \vee \Uparrow$ follow easily and from Lemma A.3.
The element $b_3 := ((\texttt{choice T} \perp) \oplus p_0$ is in $\bar{B} \subset \bar{A}$, and it is $b_3 \xrightarrow{*} (\texttt{T} \oplus \perp)A$. Hence $b\forall(\Downarrow \vee \bar{B} \vee A \vee \Uparrow)$. Let $b_4 := (\texttt{T} \oplus \perp) \oplus (\perp \oplus p_0)$. Then $b_4\bar{A}$, since p_0 is a successor. Moreover, $b_4(\exists A)$, since $(\texttt{T} \oplus \perp)$ is a successor, and it has $(\texttt{T} \oplus \perp)$ as a successor in B. Thus $b_4\neg\forall(\Downarrow \vee \bar{B} \vee A \vee \Uparrow)$

B Abstract Properties

Let us assume that the sets E have some structure like a programming language as follows:

1. Given expressions e_1, e_2, the expression $\texttt{amb}\ e_1\ e_2$ is also an expression in E with $\dfrac{e_1 \to e_1'}{\texttt{amb}\ e_1\ e_2 \to \texttt{amb}\ e_1'\ e_2}$, $\dfrac{e_2 \to e_2'}{\texttt{amb}\ e_1\ e_2 \to \texttt{amb}\ e_1\ e_2'}$, $\dfrac{e_1 W}{\texttt{amb}\ e_1\ e_2 \to e_1}$, and $\dfrac{e_2 W}{\texttt{amb}\ e_1\ e_2 \to e_2}$.
2. Given expressions e_1, e_2, e_3, the expression $\texttt{if}\ e_1 == w\ \texttt{then}\ e_2\ \texttt{else}\ e_3$ is in E such that: $\dfrac{wW}{\texttt{if}\ w == w\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \to e_2}$, and $\dfrac{wW, w'W, w \neq w'}{\texttt{if}\ w' == w\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \to e_3}$,
3. There are at least two elements $w_1, w_2, \ldots$ in W.
4. There is an element $\bot$ with $\bot \Uparrow$.

We say the relation $\sim$ is a *congruence*, iff it is an equivalence relation and for all contexts C constructed from amb or if-then-else, and for all elements e_1, e_2, the relation $e_1 \sim e_2$ implies $C[e_1] \sim C[e_2]$.

Lemma B.1. *Assume that $\sim_{\Downarrow}$ and $\sim_{\downarrow}$ are congruences. Then for all expressions s, t: If $s \leq_{\Downarrow} t$, then $t \leq_{\downarrow} s$.*

Proof. Let $s \leq_{\Downarrow} t$, $t \downarrow$, and assume for contradiction that $s \Uparrow$. Let $w \in W$ be an element, such that for some $w' \in W : w \neq w'$ and $t \xrightarrow{*} w'$.
Let C be the context $C[\ ] := \texttt{if (amb [ ]}\ w) == w\ \texttt{then}\ w\ \texttt{else}\ \bot$. Then $C[s] \sim_{\Downarrow} C[t]$ by the congruence assumption. We also have $C[s] \Downarrow$, which implies $C[t] \Downarrow$. This, however, contradicts the fact that t may reduce to a value $w' \neq w$. Hence, $s \Uparrow$ is false, which means $s \downarrow$ holds.

Corollary B.2. *Assume that $\sim_{\Downarrow}$ and $\sim_{\downarrow}$ are congruences. Then for all expressions s, t: If $s \sim_{\Downarrow} t$, then $s \sim_{\downarrow} t$.*

Proof. Lemma B.1 applied twice shows that $s \sim_{\downarrow} t$.

Corollary B.3. *Assume that $\sim_{\Downarrow}$ and $\sim_{\downarrow}$ are congruences. Then for all expressions s, t: If $s \leq_{\Downarrow,\downarrow} t$, then $s \sim_{\downarrow} t$.*

Proof. Lemma B.1 applied once shows that $t \leq_{\downarrow} s$. Since the assumptions includes $s \leq_{\downarrow} t$, this also shows $s \sim_{\downarrow} t$.

Note that the our method is too weak to show the corresponding theorems for the non-deterministic higher-order language with amb (see [SSS08]) , since lambda-abstractions cannot be compared in such a simple way.