

A Game Characterization for Contrasimilarity

Benjamin Bisping^{id} Luisa Montanari^{id}

Technische Universität Berlin
Berlin, Germany

benjamin.bisping@tu-berlin.de, luisa.montanari@campus.tu-berlin.de

We present the first game characterization of contrasimilarity, the weakest form of bisimilarity. The game is finite for finite-state processes and can thus be used for contrasimulation equivalence checking, of which no tool has been capable to date. A machine-checked Isabelle/HOL formalization backs our work and enables further use of contrasimilarity in verification contexts.

1 Introduction

Contrasimilarity is the weakest equivalence for systems with internal τ -transitions in Rob van Glabbeek’s linear-time–branching-time spectrum [10] that instantiates to bisimilarity if there is no internal behavior. It differs from the more commonly used weak bisimilarity in granting the additional equalities **CS**: $\tau.(\tau.X + Y) = \tau.X + Y$, which it shares with coupled similarity, and **C**: $a.(\tau.X + \tau.Y) = a.X + a.Y$. Contrasimilarity is about “as weak as one can get” with respect to τ -steps without losing the distinguishing powers of strong bisimilarity with respect to visible behavior.

Although this position “on the edge” makes contrasimilarity (and the contrasimulation preorder) particularly interesting, there has been only little research into it. It has been investigated as an equivalence notion justifying parallelizing transformations in compilers by Bell [2], as a strong way of relating context-free processes to their encodings as push-down automata by Baeten et al. [1, 13], and as the limit of arbitrarily-nested statements about impossible futures by Voorhoeve and Mauw [20].

In this paper, we give the first game characterization of the contrasimulation preorder and prove its correctness. All the stronger (branching-time) notions of the linear-time–branching-time spectrum with internal steps already have game characterizations [8, 4]. All the weaker (linear-time) notions can readily be characterized by slightly adapting the games [7, 5] of the linear-time–branching-time spectrum without internal steps [11]. So, our contribution inserts the last puzzle piece to have game characterizations for the whole linear-time–branching-time spectrum with internal steps of [10].

Contributions and Structure. This paper makes the following contributions:

- We assemble a concise *overview of the bisimulation-like properties of contrasimilarity* in Section 2.
- In Section 3, we present a *new game for the contrasimulation preorder* based on subset constructions, which is finite (albeit exponential) for finite-state processes.
- Section 4 proves the *correctness of the game characterization* by relating defender winning strategies and contrasimulations, which turns out to be slightly trickier than for stronger equivalences.
- Section 5 links our contributions to other research and hints at the relationship to modal-logical characterizations.
- All definitions and proofs of this paper have been *formalized in the interactive proof assistant Isabelle/HOL* [21]. Each lemma comes with a footnote pointing to its Isabelle proof at <https://concurrency-theory.org/contrasimulation-game/Contrasimulation/>.

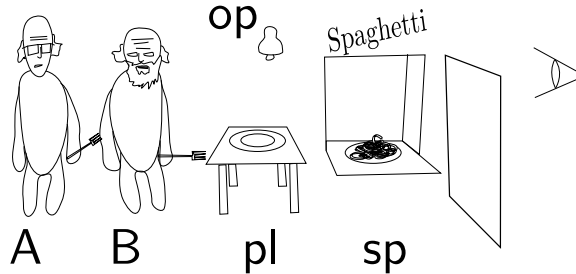
2 Contrasimilarity: The Weakest Bisimilarity

Notions of equivalence formalize when two process models can be considered indistinguishable given a certain model of communication or observation. The most famous such notion is bisimilarity, which holds if two processes can match each other's transitions repeatedly. Contrasimilarity is a reformulation of this, saying that every transition sequence by one process can be matched by the other process in such a way that they could continue with their roles swapping each time. This section explains systems with internal behavior (Subsection 2.1) and gives a formal definition of contrasimilarity (Subsection 2.2). We then show that contrasimilarity and bisimilarity actually are the same concept unless there is internal behavior (Subsection 2.3) and briefly look at systems that are not contrasimilar (Subsection 2.4). For this section, we omit the proofs; the Isabelle proofs can be found via the footnotes.

2.1 Systems with Internal Steps

In order to illustrate what kinds of systems contrasimilarity equates, we start with an example of two systems with internal communication behavior that are equivalent with respect to contrasimilarity, but not with respect to weak bisimilarity. (The example behaves similarly to the one in [2]. Its transition system will be given in Figure 1.)

Example 1 (Dining Hall Philosophers). Two philosophers A and B want to eat pasta. They can get their spaghetti (sp) once the dining hall counter opens (op). However, there is only one plate (pl), which must be taken before picking up the spaghetti (regardless of whether the counter has opened). We are waiting for them in the dining area outside; so we only observe whether the counter has opened and who eats, whereas the plate and spaghetti grabbing are invisible to us.



The following CCS structure models the situation of the philosophers waiting at the counter P_c in the notation of Milner [15]. The resources correspond to sending actions like \overline{sp} (which can be consumed only once) and obtaining the resources corresponds to receiving actions like sp . The subprocesses run in parallel ($\dots | \dots$) and internal communication is hidden from the outside ($\dots \setminus \{sp\}$).

$$P_c \stackrel{\text{def}}{=} (pl.sp.aEats | pl.sp.bEats | \overline{pl} | op.\overline{sp}) \setminus \{pl, sp\}$$

With sp being hidden: Does it make a difference to the observer if it is not the spaghetti counter waiting for the opening event before handing out pasta, but rather the philosophers waiting for the event before grabbing the pasta? This would amount to the following model of patient philosophers P_p .

$$P_p \stackrel{\text{def}}{=} (pl.op.sp.aEats | pl.op.sp.bEats | \overline{pl} | \overline{sp}) \setminus \{pl, sp\}$$

If one's application needs to abstract over "where exactly" internal waiting is happening, and thus equate P_c and P_p , one has to pick contrasimilarity (or a weaker equivalence) for the semantics.

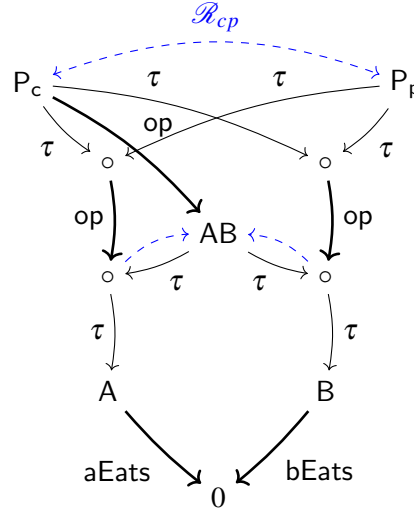


Figure 1: The reflexive closure of \mathcal{R}_{cp} is a contrasimulation on the philosopher system from Example 1.

Definition 1. A *labeled transition system*, or *LTS* for short, $(S, Act_\tau, \rightarrow)$ consists of a set of states S , of a set $Act_\tau = Act \cup \{\tau\}$ of visible actions Act and a special *internal action* $\tau \notin Act$, and of a transition relation $\rightarrow \subseteq S \times Act_\tau \times S$.

The transition system for Example 1 is depicted as the black part in Figure 1. All deadlocks are combined into the state 0. The transitions where communication internal to the system happens are labeled by τ , which denotes internal behavior. Where there are multiple internal transitions originating from the same process state, the system performs an *internal choice*. In process P_p , the internal choice happens at the start, whereas in P_c , the choice is interleaved with the observable occurrence of op .

Definition 2. We employ the following notation for transitions in the system (with $p, p', q \in S$; $\alpha \in Act_\tau$):

- Strong steps: $p \xrightarrow{\alpha} p'$ iff $(p, \alpha, p') \in \rightarrow$.
- Internal steps: $p \Rightarrow p'$ iff $p \xrightarrow{\tau^*} p'$.
- Delay steps: $p \xRightarrow{\alpha} p'$ iff $\alpha = \tau$ with $p \Rightarrow p'$ or $\alpha \in Act$ with $p \Rightarrow \xrightarrow{\alpha} p'$.
- Weak steps: $p \xRightarrow{\hat{\alpha}} p'$ iff $p \xRightarrow{\alpha} p'$.
- Weak word steps: $p \xRightarrow{\vec{w}} p'$ iff $p \xRightarrow{\hat{w}_0} \xRightarrow{\hat{w}_1} \dots \xRightarrow{\hat{w}_n} p'$ with $\vec{w} = w_0 w_1 \dots w_n \in Act^*$ or $p \Rightarrow p'$ for the empty word $\vec{w} = \varepsilon$.
- Absence of steps: $p \not\xrightarrow{\alpha}$, $p \not\xrightarrow{\hat{\alpha}}$, and $p \not\xRightarrow{\vec{w}}$ to denote that certain transitions are not possible from p . p is called *stable* iff $p \not\xrightarrow{\tau}$.
- Lifting of steps to sets: $P \xrightarrow{\alpha} P'$ iff $P' = \{p' \mid \exists p \in P : p \xrightarrow{\alpha} p'\}$, and $P \xRightarrow{\alpha} p'$ iff there is a $p \in P$ with $p \xrightarrow{\alpha} p'$; analogously for \Rightarrow , $\xRightarrow{\alpha}$, $\xRightarrow{\hat{\alpha}}$, and $\xRightarrow{\vec{w}}$.

Note that delay steps \xRightarrow{a} differ from the more common weak steps $\xRightarrow{\hat{a}}$ ($a \in Act$) in that the step ends in the strong \xrightarrow{a} -transition with no trailing internal behavior. Also note that $P \xRightarrow{\alpha} \{p'\}$ is stronger than $P \xrightarrow{\alpha} p'$ in that it rules out possible other $\xrightarrow{\alpha}$ -transitions.

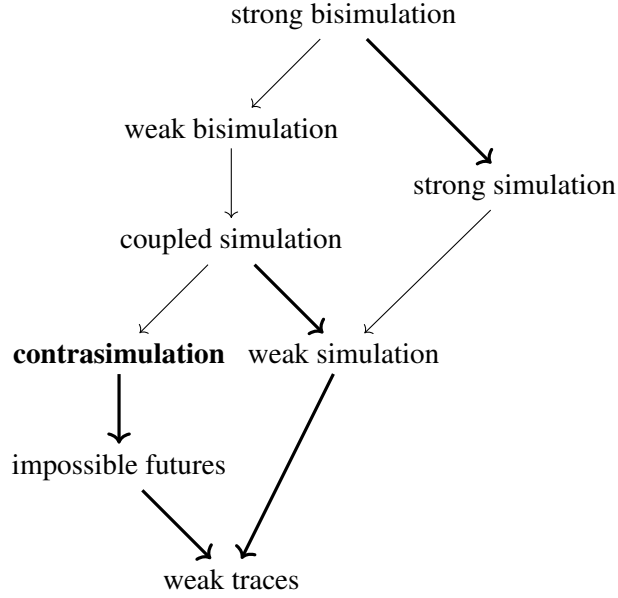


Figure 2: Hierarchy of equivalences. Arrows denote implications of preordering. Thinner arrows collapse into bi-implication for systems without internal steps.

2.2 Defining Contrasimilarity

It is common to define notions of behavioral equivalence and behavioral preorders in terms of relations between process states that fulfill certain properties. For instance, weak similarity is defined in terms of weak simulation relations, which also induce a weak simulation preorder. Contrasimilarity is obtained by making a subtle change to the weak simulation property.

Definition 3. A *weak simulation* is a relation \mathcal{R} where, for all $(p, q) \in \mathcal{R}$ with $p \xrightarrow{\alpha} p'$, there is a q' with $q \xrightarrow{\hat{\alpha}} q'$ and $(p', q') \in \mathcal{R}$. We say that p is *weakly simulated* by q , written $p \preceq_{WS} q$, iff there is a weak simulation \mathcal{R} with $(p, q) \in \mathcal{R}$. If $p \preceq_{WS} q \preceq_{WS} p$, the two are *weakly similar*, written $p \sim_{WS} q$. If \mathcal{R} is a symmetric weak simulation, the processes are called *weakly bisimilar* (\sim_{WB}), which implies all of the above.

On the processes of Example 1, $\{(p, p) \mid p \in \mathcal{S}\} \cup \{(P_p, P_c)\}$ is a weak simulation, because the steps from P_p are a subset of the steps from P_c . This implies $P_p \preceq_{WS} P_c$.

However, there is no weak simulation in the opposite direction, because P_c can op-step to a state where aEats and bEats are weakly enabled, while P_p cannot. Thus, weak similarity and hence also weak bisimilarity distinguish the two systems.

Due to its inherent recursiveness, Definition 3 can also be rephrased in terms of words instead of single actions:

Lemma 1. \mathcal{R} is a weak simulation precisely if, for all $(p, q) \in \mathcal{R}$ with $\vec{w} \in Act^*$ and $p \xrightarrow{\vec{w}} p'$, there is a q' with $q \xrightarrow{\vec{w}} q'$ and $(p', q') \in \mathcal{R}$.¹

This observation motivates why the following definition with p' and q' swapping sides at the end has been named *contra*-simulation:

¹ lemma Weak_Relations.weak_sim_word

Definition 4. A *contrasimulation* is a relation \mathcal{R} where, for all $(p, q) \in \mathcal{R}$ with $\vec{w} \in Act^*$ and $p \xrightarrow{\vec{w}} p'$, there is a q' with $q \xrightarrow{\vec{w}} q'$ and $(q', p') \in \mathcal{R}$. The contrasimulation preorder \preceq_C and contrasimilarity \sim_C are defined analogously to Definition 3.

The reflexive closure of the relation in Figure 1, $\mathcal{R}_{cp} \cup \{(p, p) \mid p \in S\}$, is a contrasimulation. Hence, P_c and P_p are contrasimilar, $P_c \sim_C P_p$.

Let us quickly stress that the contrasimulation preorder indeed *is* a sensible behavioral preorder (which is not true for its intransitive neighbors eventual simulation [2] and stability-coupled simulation [16]):

Lemma 2. *Properties of \preceq_C :*

- \preceq_C is transitive,² as the interleaved concatenation $\mathcal{R}_1\mathcal{R}_2 \cup \mathcal{R}_2\mathcal{R}_1$ of two contrasimulations \mathcal{R}_1 and \mathcal{R}_2 is itself a contrasimulation.³
- \preceq_C is reflexive.⁴ (More generally: $p \Rightarrow p'$ implies $p' \preceq_C p$.⁵)
- \preceq_C is itself a contrasimulation⁶ (and thus the greatest contrasimulation⁷).

2.3 Relationship of Contra- and Bi-similarity

Like the weak simulation preorder, the contrasimulation preorder is not symmetric for most systems. Like with weak simulation, the contrasimulation property can be used to characterize weak bisimulation:

Lemma 3 (Bisimulation characterization). *If a contrasimulation \mathcal{R} is symmetric, then \mathcal{R} moreover is a weak simulation. (Meaning that, in this case, $(p, q) \in \mathcal{R}$ implies $p \sim_{WB} q$.)⁸*

Unlike weak simulations, contrasimulations are “symmetric up to internal steps.” We call this property *coupling*, as it differentiates coupled simulations from weak simulations [6]:

Lemma 4 (Coupling). *If \mathcal{R} is a contrasimulation, then $(p, q) \in \mathcal{R}$ implies there is a q' such that $q \Rightarrow q'$ and $(q', p) \in \mathcal{R}$.⁹*

On stable states, coupling equates to local symmetry. This is nice for systems without internal behavior:

Lemma 5 (Contra/Bi-simulation). *If \rightarrow contains no τ -steps, and \mathcal{R} is a contrasimulation, then \mathcal{R} is symmetric and thus a bisimulation.¹⁰*

Accordingly, $\preceq_C = \sim_{WB} = \sim_{SB}$ for systems without internal steps. In this sense, contrasimilarity is closer to bisimilarity than weak similarity: A weak simulation on a τ -free system is just a strong simulation but does not need to be a bi-simulation. The hierarchy is depicted in Figure 2.

² lemma Contrasimulation.contrasim_trans

³ lemma Contrasimulation.contrasim_trans_constructive

⁴ lemma Contrasimulation.contrasim_refl

⁵ lemma Contrasimulation.contrasim_tau_step

⁶ lemma Contrasimulation.contrasim_preorder_is_contrasim

⁷ lemma Contrasimulation.contrasim_preorder_is_greatest

⁸ lemma Contrasimulation.symm_contrasim_is_weak_bisim

⁹ lemma Contrasimulation.contrasim_coupled

¹⁰ lemma Contrasimulation.contrasim_weakest_bisim

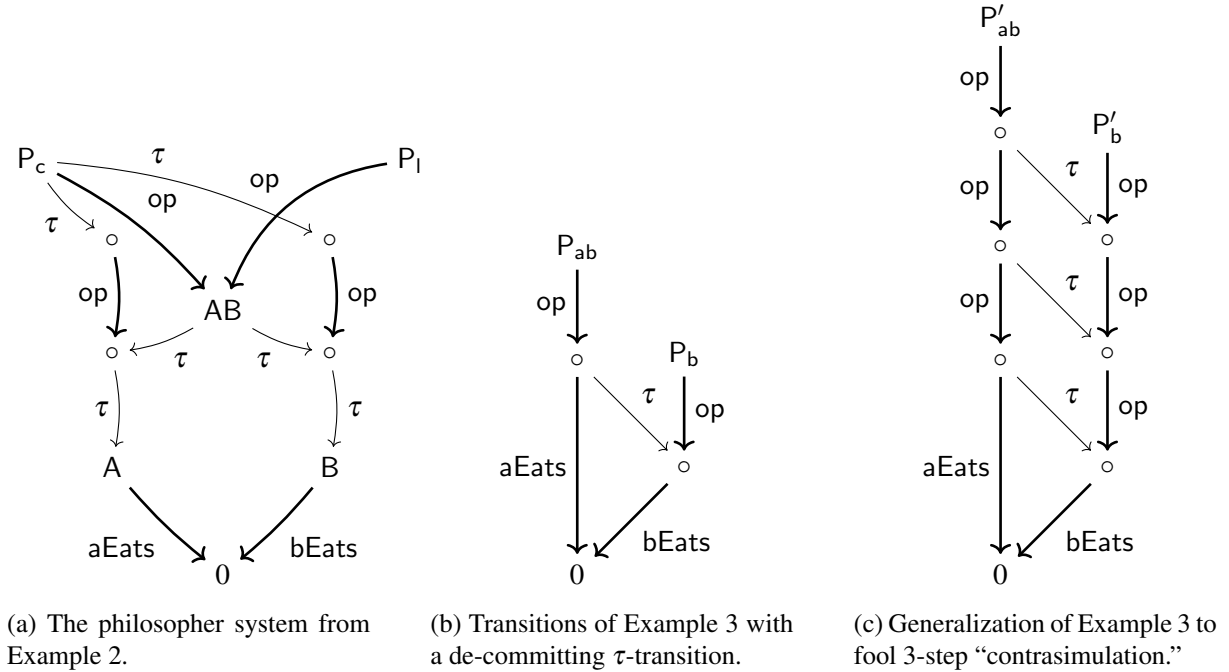


Figure 3: Processes that are not contrasimilar.

2.4 No Shortcuts

We already mentioned that contrasimilarity grants the equality $a.(\tau.X + \tau.Y) = a.X + a.Y$. However, this does not mean that internal choice may commute over actions: The equality $a.(\tau.X + \tau.Y) = \tau.a.X + \tau.a.Y$ is *not sound* for contrasimilarity, as the following example illustrates:

Example 2 (Locked-out Philosophers). Consider this slight modification of P_c , where already the scarce plate pl is guarded by the opening op :

$$P_l \stackrel{\text{def}}{=} (pl.sp.aEats \mid pl.sp.bEats \mid op.\bar{pl} \mid \bar{sp}) \setminus \{pl, sp\}$$

As depicted in Figure 3a, the result of this change is that the decision between philosophers A and B can happen *only* after op .

The change is noticed by contrasimilarity, i.e. $P_c \not\leq_C P_l$. The reason is that $P_c \leq_C P_l$ with the left process resolving its choice would imply $P_l \leq_C op.\tau.aEats$. But this does not hold because $P_l \xrightarrow{op, bEats} 0$ but not $op.\tau.aEats \xrightarrow{op, bEats} 0$.

It is also worthwhile to observe that, in general, the definition of contrasimulation (Definition 4) cannot straight-forwardly be simplified to use single steps $\hat{\alpha} \Rightarrow$ instead of words $\vec{\alpha} \Rightarrow$, as is the case with its finer siblings like weak bisimulation. Such a simplification is used by de Hoop [13]. However, this is not sound for general systems due to the alternating sides in the definition.

Example 3 (Instable choice). Consider $P_{ab} \stackrel{\text{def}}{=} op.(aEats + \tau.bEats)$ and $P_b \stackrel{\text{def}}{=} op.bEats$ whose transitions can be found in Figure 3b. The processes are clearly not even weakly trace equivalent. But single-step "contrasimulation" cannot tell them apart, as neither $P_{ab} \xrightarrow{op} aEats + \tau.bEats$ nor $P_{ab} \xrightarrow{op} bEats$ (matched by $P_b \xrightarrow{op} bEats$) lead to a situation that would not (contra-)simulate $bEats$.

The counter-example can be generalized as hinted at in Figure 3c, which would need at least a word length of four to distinguish the named processes.

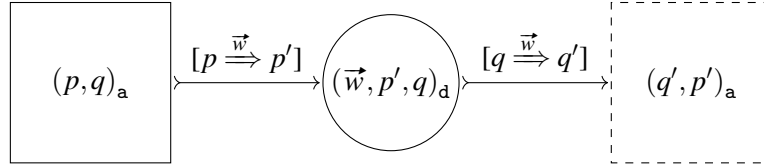


Figure 4: Schematic basic contrasimulation game. Boxes denote attacker positions, circles denote defender positions and arrows denote game moves. Each game move is only possible if the condition in square brackets is satisfied. Dashed boxes are attacker positions with a new variable assignment and admit analogous moves to the solid boxes.

3 The Contrasimulation Game

The contrasimulation preorder can be characterized by a game between two opposing players, the *attacker* and the *defender*. For a given $p, q \in S$, the attacker seeks to disprove $p \preceq_C q$, while the defender seeks to maintain $p \preceq_C q$. We first introduce some general thoughts about such games in Subsection 3.1, and then present the contrasimulation game at the core of our contribution in Subsection 3.2.

3.1 Preliminaries

For this paper, we use Gale-Stewart-style games in the tradition of Stirling [19] where the attacker wins by getting the defender stuck, and the defender wins by not getting stuck.

Definition 5 (Games). A *simple reachability game* $\mathcal{G}[g_0] = (G, G_d, \succrightarrow, g_0)$ consists of

- a set of *game positions* G , partitioned into
 - a set of *defender positions* $G_d \subseteq G$
 - and *attacker positions* $G_a := G \setminus G_d$,
- a graph of *game moves* $\succrightarrow \subseteq G \times G$, and
- an *initial position* $g_0 \in G$.

Definition 6 (Plays and wins). We call the infinite and finite paths $g_0 g_1 \dots \in G^\infty$ with $g_i \succrightarrow g_{i+1}$ *plays* of $\mathcal{G}[g_0]$. The defender *wins* infinite plays. If a finite play $g_0 \dots g_n \not\succrightarrow$ is stuck, the stuck player loses: The defender wins if $g_n \in G_a$, and the attacker wins if $g_n \in G_d$. Equivalently, the defender wins precisely those plays in which the defender is not stuck.

Definition 7 (Strategies and winning strategies). A *defender strategy* is a partial mapping from initial play fragments to next moves $f \subseteq \{(g_0 \dots g_n, g_{n+1}) \mid g_n \in G_d \wedge g_n \succrightarrow g_{n+1}\}$. A play g is consistent with a defender strategy f iff, for each move $g_i \succrightarrow g_{i+1}$ with $g_i \in G_d$ where $f(g_0 \dots g_i)$ is defined, we have $g_{i+1} = f(g_0 \dots g_i)$. We denote the set of plays g consistent with f by $G_f^\infty[g_0]$ for the initial game position g_0 . If every stuck or infinite play $g \in G_f^\infty[g_0]$ is won by the defender, then f is a winning strategy for the defender. The player with a winning strategy for $\mathcal{G}[g_0]$ is said to *win* $\mathcal{G}[g_0]$.

All simple games are *determined*, that is, either the defender or the attacker wins. The winning regions of finite simple games can be computed in linear time in the number of game moves (cf. [12]). Therefore, it is desirable to find finite game characterizations of equivalences. For many weak equivalences, such characterizations can be obtained directly from their standard coinductive characterization.

For contrasimilarity, this direct route is less helpful due to its word-based definition. That is why we only briefly discuss it here, and then move on to a different approach in the next subsection.

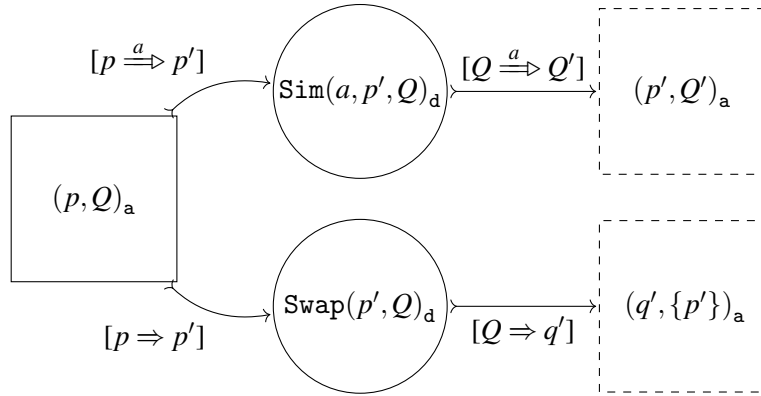


Figure 5: Schematic contrasimulation set game (Definition 8).

If we directly transfer Definition 4 into a game, we arrive at the following game alternating between attacker and defender: The attacker may challenge $p \preceq_C q$ by selecting a word $\vec{w} \in Act^*$ and a $p' \in S$ with $p \xrightarrow{\vec{w}} p'$, to which the defender has to name a $q' \in S$ with $q \xrightarrow{\vec{w}} q'$. The sides of the game then swap, and the attacker can go on to question $q' \preceq_C p'$. The attacker wins if the defender is unable to answer with an appropriate q' , and the defender wins if the play goes on forever.

A schematic model of the basic contrasimulation game is given in Figure 4. We proved its correctness in the Isabelle formalization.¹¹ Unfortunately, in finite-state processes with cycles, the attacker can challenge with infinitely many words. So, such games will have an infinite number of game positions, even for finite systems. Therefore, we decided not to devote more space to this game in this paper and rather move on to a different game in the following subsection.

3.2 The Game

Let us now turn to the contrasimulation game that is the core contribution of this paper. The idea is to restrict the players' moves to single actions $\alpha \in Act_\tau$. This means breaking the attacker's word challenge for $p \preceq_C q$ into a simulation phase and a swap request. During the simulation phase, the defender plays a *set of states*:

Definition 8 (\preceq_C game). For a transition system $(S, Act_\tau, \rightarrow)$, the *contrasimulation set game* $\mathcal{G}_C[g_0] = (G, G_d, \succ, g_0)$ consists of

- attacker positions $(p, Q)_a \in G_a$ with $p \in S, Q \subseteq S$,
- defender simulation positions $\text{Sim}(a, p, Q)_d \in G_d$ with $a \in Act, p \in S, Q \subseteq S$, and
- defender swapping positions $\text{Swap}(p, Q)_d \in G_d$ with $p \in S, Q \subseteq S$

and the following game moves

- simulation challenges $(p, Q)_a \succ \text{Sim}(a, p', Q)_d$ if $p \xrightarrow{a} p'$,
- swap challenges $(p, Q)_a \succ \text{Swap}(p', Q)_d$ if $p \Rightarrow p'$,
- simulation answers $\text{Sim}(a, p', Q)_d \succ (p', Q')_a$ if $Q \xrightarrow{a} Q'$, and
- swap answers $\text{Swap}(p', Q)_d \succ (q', \{p'\})_a$ if $Q \Rightarrow q'$.¹²

¹¹ theorem Basic_Contrasim_Game.winning_strategy_in_basic_game_iff_contrasim

¹² locale Contrasim_Set_Game.c_set_game

To check whether $p \preceq_C q$ holds we play the contrasimulation set game from the initial attacker position $(p, \{q\})_a$. A schematic model of the game is given in Figure 5.

In each simulation phase, the attacker challenges the defender to successively simulate the actions w_i of a word $\vec{w} = w_0 \dots w_n$. Here, the defender does not play a single state, but rather the *set of all* states reachable by the known prefix $w_0 \dots w_k$ of \vec{w} . When the attacker is done choosing \vec{w} in $p \xrightarrow{\vec{w}} p'$, they request a swap. The defender must then select a *specific* state q' with $q \xrightarrow{\vec{w}} q'$ from their state set. The players then change sides and the attacker may now challenge $q' \preceq_C p'$. Hence, the defender postpones the decision of how exactly to simulate the challenged word until a swap is requested.

Example 4 (Contrasimulation game on P_c, P_p). A possible play of \mathcal{G}_C for the philosopher transition system from Example 1 would be: $(P_c, \{P_p\})_a \rightsquigarrow \text{Sim}(\text{op}, \text{AB}, \{P_p\})_d \rightsquigarrow (\text{AB}, \{\tau.\text{aEats}, \tau.\text{bEats}\})_a \rightsquigarrow \text{Swap}(\tau.\text{aEats}, \{\tau.\text{aEats}, \tau.\text{bEats}\})_d \rightsquigarrow (\tau.\text{aEats}, \{\tau.\text{aEats}\})_a \rightsquigarrow \text{Sim}(\text{aEats}, 0, \{\tau.\text{aEats}\})_d \rightsquigarrow (0, \{0\})_a \rightsquigarrow \text{Swap}(0, \{0\})_d \rightsquigarrow \dots$. The play ends in an infinite loop of swaps and is thus won by the defender. The crucial point of this game is the defender move highlighted in blue, where the defender answers a swap challenge by matching the processes on both sides. After this, it becomes impossible for the attacker to win. If the defender picked $\tau.\text{bEats}$ instead, they would lose. However, the defender has a winning strategy no matter which moves the attacker chooses for $\mathcal{G}_C[(P_c, \{P_p\})_a]$.

There are no easy ways of switching to single-action moves without using the subset construction: The defender does not know the full word \vec{w} that the attacker will choose, but only the word prefix $w_0 \dots w_k$ challenged thus far up to a point $k \leq n$. Deciding for *single* states early would thus put the defender at a disadvantage: There *might* be several states q' with $q \xrightarrow{w_0 \dots w_k} q'$ for every such k , of which only some also satisfy $q \xrightarrow{w_0 \dots w_k} q' \xrightarrow{w_{k+1} \dots w_n} q''$ for any $q'' \in S$. Dually, forcing early swapping would be disadvantageous to the attacker when the attack has to pass through instable states as seen in Example 3.

Crucially, this construction yields a finite game for any finite-state process. As with the well-known subset construction when transforming nondeterministic into deterministic finite automata, the game size is exponential in the size of the state space S .

We chose to present the game as an alternating game where attacker and defender take turns. Several modifications could be made to simplify some aspects of the game: Note, for example, that the $\text{Sim}(\dots)_d$ -positions are not strictly necessary, as the defender has exactly one move originating from each such position. Additionally, parts of the game moves could be broken up into smaller steps on \rightarrow instead of \Rightarrow . However, both changes would make the game non-alternating. For the purpose of this paper, especially for intuitive proofs like in the following section, we consider the alternating formulation superior.

4 Correctness of the Contrasimulation Game

Let us now demonstrate that the \preceq_C game does indeed correspond to the contrasimulation preorder in the sense that the defender wins the game $\mathcal{G}_C[(p, \{q\})_a]$ precisely if $p \preceq_C q$. In Subsection 4.1, we first establish soundness of the characterization, that is, defender winning strategies in the game imply contrasimulations on the LTS. Subsection 4.2 then shows completeness of the characterization by constructing a defender winning strategy from the greatest contrasimulation.

Our proofs must bridge the gap between the single-action game and the word-transition definition of the contrasimulation property. While transition relations on single actions usually are non-deterministic, the transitions lifted to sets of states *are* deterministic. In order to exploit this in proofs, we first define the word successor function from delay steps:

Definition 9. We define the word successor function $\text{succs} : \text{Act}^* \times 2^S \rightarrow 2^S$ recursively as follows:¹³

$$\begin{aligned} \text{succs}(\varepsilon, Q) &= Q \\ \text{succs}(\vec{w}a, Q) &= \{q' \mid \text{succs}(\vec{w}, Q) \xrightarrow{a} q'\} \end{aligned}$$

Intuitively, succs computes the set of states reachable with a given word \vec{w} from a starting set Q . Thus, we can use succs to compute the state set of defender simulation positions in the contrasimulation set game. Note that $\text{succs}(\vec{w}, Q)$ will return the empty set if no state in Q admits a \vec{w} -transition.

Lemma 6. Let $\vec{w} \in \text{Act}^*$ be a word and let q, q' be states in S . Then $q \xrightarrow{\vec{w}} q'$ implies $\text{succs}(\vec{w}, \{q\}) \Rightarrow q'$.¹⁴ Furthermore, $q' \in \text{succs}(\vec{w}, \{q\})$ implies $q \xrightarrow{\vec{w}} q'$.¹⁵

4.1 Soundness of the Contrasimulation Game

Let us first prove that the \preceq_C game is sound with respect to the contrasimulation preorder, that is, $p \preceq_C q$ holds if the defender wins $\mathcal{G}_C[(p, \{q\})_a]$. To this end, we first prove an intermediate result stating that the defender is always able to answer simulation challenges over the prefix \vec{v} of a nonempty word $\vec{w} = \vec{v}w_n$ in $p \xrightarrow{\vec{w}} p'$.

Lemma 7 (Word challenge building). *Let f be a defender strategy on $\mathcal{G}_C[g_0]$ for some $g_0 \in G$ and let $(p, \{q\})_a$ be an attacker position in a play consistent with f . Let $\vec{w} = \vec{v}w_n \in \text{Act}^*$ be a nonempty word and assume $p \xrightarrow{\vec{w}} p'$ for some $p' \in S$. Then there exist $p_0, p_1 \in S$ with $p \xrightarrow{\vec{v}} p_0 \xrightarrow{w_n} p_1 \Rightarrow p'$ such that the defender position $\text{Sim}(w_n, p_1, \text{succs}(\vec{v}, \{q\}))_a$ can be reached in some play consistent with f .¹⁶*

Proof. We will prove this by nonempty induction on $\vec{w} = \vec{v}w_n \in \text{Act}^*$:

- *Base Case:* $\vec{w} = w_n = \varepsilon w_n$.

From $p \xrightarrow{w_n} p'$ we know there exists a p_1 such that $p \xrightarrow{w_n} p_1 \Rightarrow p'$. Hence, the attacker can move from $(p, \{q\})_a$ to $\text{Sim}(w_n, p_1, \{q\})_a = \text{Sim}(w_n, p_1, \text{succs}(\varepsilon, \{q\}))_a$. This play is still consistent with f as f is only defined for defender moves.


- *Induction step:* $\vec{w} = \vec{v}w_n$ for some nonempty word $\vec{v} \in \text{Act}^*$.


Then there exists a $\vec{u} \in \text{Act}^*$ such that $\vec{v} = \vec{u}w_{n-1}$ and $p \xrightarrow{\vec{u}} p_0 \xrightarrow{w_{n-1}} p_1 \xrightarrow{w_n} p'$ for some $p_0, p_1 \in S$. We assume the lemma holds for \vec{v} , i.e. there exists a $p_{01} \in S$ with $p_0 \xrightarrow{w_{n-1}} p_{01} \Rightarrow p_1$ such that the position $\text{Sim}(w_{n-1}, p_{01}, \text{succs}(\vec{u}, \{q\}))_a$ can be reached in some play consistent with f .


Because there always exists a (potentially empty) set Q' with $\text{succs}(\vec{u}, \{q\}) \xrightarrow{w_{n-1}} Q'$, the defender is not stuck at $\text{Sim}(w_{n-1}, p_{01}, \text{succs}(\vec{u}, \{q\}))_a$. There must therefore exist a position $(p_{01}, Q')_a = f(g_0 \dots \text{Sim}(w_{n-1}, p_{01}, \text{succs}(\vec{u}, \{q\}))_a)$ that the defender can move to. For Q' we have


$$Q' = \{q' \mid \text{succs}(\vec{u}, \{q\}) \xrightarrow{w_{n-1}} q'\} = \text{succs}(\vec{u}w_{n-1}, \{q\}) = \text{succs}(\vec{v}, \{q\}).$$

From our assumptions $p_{01} \Rightarrow p_1$ and $p_1 \xrightarrow{w_n} p'$ we can infer the existence of a p'_1 with $p_{01} \Rightarrow p_1 \xrightarrow{w_n} p'_1 \Rightarrow p'$, which we can shorten to $p_{01} \xrightarrow{w_n} p'_1 \Rightarrow p'$. Hence, the attacker can move from $(p_{01}, Q')_a$ to $\text{Sim}(w_n, p'_1, Q')_a = \text{Sim}(w_n, p'_1, \text{succs}(\vec{v}, \{q\}))_a$, and therefore the defender position

¹³  primrec Weak_Transition_Systems.dsuccs_seq_rec

¹⁴  lemma Weak_Transition_Systems.word_reachable_implies_in_dsuccs

¹⁵  lemma Weak_Transition_Systems.in_dsuccs_implies_word_reachable

¹⁶  lemma Contrasim_Set_Game.def_sim_pos_with_prefix_in_play

$\text{Sim}(w_n, p'_1, \text{succs}(\vec{v}, \{q\}))_d$ is reachable in a play consistent with f . Furthermore, the second implication $p \xrightarrow{\vec{v}} p_{01} \xrightarrow{w_n} p'_1 \Rightarrow p'$ follows immediately from $p \xrightarrow{\vec{u}} p_0 \xrightarrow{w_{n-1}} p_{01} \xrightarrow{w_n} p'_1 \Rightarrow p'$ and $\vec{v} = \vec{u}w_{n-1}$. \square

With this result, we are now able to prove the soundness of the \preceq_C game.

Lemma 8 (Soundness). *Let f be a winning strategy for the defender on $\mathcal{G}_C[(p_0, \{q_0\})_a]$ for some $p_0, q_0 \in S$. Then we have $p_0 \preceq_C q_0$.¹⁷*

Proof. We construct a relation $\mathcal{R} \subseteq S \times S$ where

$$\mathcal{R} = \{(p_0, q_0)\} \cup \{(q, p) \mid \exists g \in G_f^\infty[(p_0, \{q_0\})_a] : \exists k \in \mathbb{N} : \exists Q \subseteq S : g_k = \text{Swap}(p, Q)_d \wedge g_{k+1} = (q, \{p\})_a\}.$$

Informally, \mathcal{R} contains the states p_0, q_0 of the initial position $(p_0, \{q_0\})_a$ and the states of all attacker positions following a defender swap position in any play consistent with f . We aim to prove that \mathcal{R} is a contrasimulation:

Let $p, p', q \in S$ be states and assume $(p, q) \in \mathcal{R}$ and $p \xrightarrow{\vec{w}} p'$ for some $\vec{w} \in \text{Act}^*$. We shall prove that a state $q' \in S$ exists such that $q \xrightarrow{\vec{w}} q'$ and $(q', p') \in \mathcal{R}$.

Since $(p, q) \in \mathcal{R}$, there exists a $g \in G_f^\infty[(p_0, \{q_0\})_a]$ and a $k \in \mathbb{N} \cup \{-1\}$ such that $g_{k+1} = (p, \{q\})_a$ is an attacker position in g . We will distinguish between empty and nonempty words \vec{w} :

- Case 1: $\vec{w} = \varepsilon$.

By assumption, we have $p \xrightarrow{\varepsilon} p'$ and thus $p \Rightarrow p'$. Hence, the attacker can move from $(p, \{q\})_a$ to a defender position $\text{Swap}(p', \{q\})_d$. Because f is a winning strategy, the defender is not stuck at $\text{Swap}(p', \{q\})_d$; there must therefore exist a state $q' \in S$ such that the defender can move to the position $f(g_0 \dots \text{Swap}(p', \{q\})_d) = (q', \{p'\})_a$. It follows that $q \Rightarrow q'$ and that there exists a play $g \in G_f^\infty[(p_0, \{q_0\})_a]$ in which $(q', \{p'\})_a$ can be reached. Since the last position played is a $\text{Swap}(\dots)_d$ node, we therefore have $(q', p') \in \mathcal{R}$ by our construction of \mathcal{R} .


- Case 2: $\vec{w} = \vec{v}w_n$ for some $\vec{v} \in \text{Act}^*$ and $w_n \in \text{Act}$.

By application of Lemma 7, we know there exist p_0, p_1 with $p \xrightarrow{\vec{v}} p_0 \xrightarrow{w_n} p_1 \Rightarrow p'$ such that the position $\text{Sim}(w_n, p_1, \text{succs}(\vec{v}, \{q\}))_d$ can be reached in some play $g \in G_f^\infty[(p_0, \{q_0\})_a]$. Because f is a winning strategy, the defender is not stuck at $\text{Sim}(w_n, p_1, \text{succs}(\vec{v}, \{q\}))_d$; there must therefore exist a set Q_1 allowing the defender to move to a position $f(g_0 \dots \text{Sim}(w_n, p_1, \text{succs}(\vec{v}, \{q\}))_d) = (p_1, Q_1)_a$. For Q_1 we have

$$Q_1 = \{q_1 \mid \text{succs}(\vec{v}, \{q\}) \xrightarrow{w_n} q_1\} = \text{succs}(\vec{v}w_n, \{q\}) = \text{succs}(\vec{w}, \{q\}).$$

By our assumption $p_1 \Rightarrow p'$, the attacker can keep moving to a defender position $\text{Swap}(p', Q_1)_d$. Again, the defender is not stuck, and there exists a $q' \in S$ allowing the defender to move to the position $f(g_0 \dots \text{Swap}(p', Q_1)_d) = (q', \{p'\})_a$. It follows that $Q_1 \Rightarrow q'$. For q' we have

$$\begin{aligned} q' &\in \{q' \mid \exists q_1 \in Q_1 : q_1 \Rightarrow q'\} \\ &= \{q' \mid \exists q_1 \in \text{succs}(\vec{w}, \{q\}) : q_1 \Rightarrow q'\} \\ &\subseteq \{q' \mid \exists q_1 \in \text{succs}(\vec{w}, \{q\}) : q \xrightarrow{\vec{w}} q_1 \wedge q_1 \Rightarrow q'\} && \text{(Lemma 6)} \\ &\subseteq \{q' \mid q \xrightarrow{\vec{w}} q'\}. \end{aligned}$$

¹⁷  lemma Contrasm_Set_Game.set_contrasim_game_sound

Hence, there exists a q' satisfying $q \xrightarrow{\vec{w}} q'$. Because the defender moves according to f from a position $(p, \{q\})_a$ in a play $g \in G_f^\infty[(p_0, \{q_0\})_a]$, the position $(q', \{p'\})_a$ must also be reachable in a play consistent with f . Then we also have $(q', p') \in \mathcal{R}$, since the last played position was a $\text{Swap}(\dots)_d$ node.

Thus, \mathcal{R} is a contrasimulation by Definition 4. From $(p_0, q_0) \in \mathcal{R}$ then follows $p_0 \preceq_C q_0$. \square

4.2 Completeness of the Contrasimulation Game

Let us now prove that the \preceq_C game is complete with respect to the contrasimulation preorder, that is, the defender wins $\mathcal{G}_C[(p, \{q\})_a]$ if $p \preceq_C q$. To this end, we define an auxiliary function F with which we are able to construct a defender strategy f_C from the contrasimulation preorder \preceq_C . We will prove that if $p \preceq_C q$, then f_C is a winning strategy for the defender on $\mathcal{G}_C[(p, \{q\})_a]$.

We define the function $F : 2^{S \times 2^S} \rightarrow 2^{S \times 2^S}$ as follows:¹⁸

$$F(R) = \{(p', \text{succs}(\vec{w}, Q)) \mid \exists p \in S : (p, Q) \in R \wedge p \xrightarrow{\vec{w}} p'\}$$

Furthermore, we define a type-congruent relation $C \subseteq S \times 2^S$ from the contrasimulation preorder with $C = \{(p, \{q\}) \mid p \preceq_C q\}$.

This yields a relation $F(C) \subseteq S \times 2^S$ which we will use extensively in the following proofs. The motivation behind C and $F(C)$ becomes clear when we consider a play in the game $\mathcal{G}_C[(p_0, \{q_0\})_a]$ with $p_0 \preceq_C q_0$: Here, C contains tuples $(p, \{q\})$ of attacker positions after the word challenge has been completed, i.e. positions $(p, \{q\})_a$ following a $\text{Swap}(\dots)_d$ node. The relation $F(C)$ then expands C to also include attacker positions in the *simulation phase* of the word challenge, i.e. positions $(p, Q)_a$ following a $\text{Sim}(\dots)_d$ node. Thus, using $F(C)$ we can construct a defender strategy that is well-defined for both $\text{Sim}(\dots)_d$ and $\text{Swap}(\dots)_d$ positions.


Lemma 9. *We have $R \subseteq F(R)$ for all $R \subseteq S \times 2^S$.*¹⁹


Lemma 10. *Let $a \in \text{Act}$ and let $p, p' \in S$ be states and $Q \subseteq S$ be a set of states such that $(p, Q) \in F(C)$.*


- *If $p \xrightarrow{a} p'$, then there exists a set $Q' \subseteq S$ such that $Q \xrightarrow{a} Q'$ and $(p', Q') \in F(C)$,*²⁰ *and*
- *if $p \Rightarrow p'$, then there exists a state $q' \in S$ such that $Q \Rightarrow q'$ and $(q', \{p'\}) \in F(C)$.*²¹


Proof. By construction of F and our assumption $(p, Q) \in F(C)$, we know there exist states $p_0, q_0 \in S$ such that $(p_0, \{q_0\}) \in C$, $p_0 \xrightarrow{\vec{w}} p$ and $Q = \text{succs}(\vec{w}, \{q_0\})$ for some $\vec{w} \in \text{Act}^*$. Hence, we also have $p_0 \preceq_C q_0$. We will distinguish between τ -steps and delay steps:

- For $p \xrightarrow{a} p'$, we have $p_0 \xrightarrow{\vec{w}} p \xrightarrow{a} p'$ and thus $p_0 \xrightarrow{\vec{w}a} p'$. With $p_0 \preceq_C q_0$, there must then exist a $q' \in S$ such that $q_0 \xrightarrow{\vec{w}a} q'$. It follows from Lemma 6 that $\text{succs}(\vec{w}a, \{q_0\}) \Rightarrow q'$. For $Q' := \text{succs}(\vec{w}a, \{q_0\})$, we then have $(p', Q') \in F(C)$ by our construction of F and $Q \xrightarrow{a} Q'$ by Definition 9.
- For $p \Rightarrow p'$, we have $p_0 \xrightarrow{\vec{w}} p \Rightarrow p'$ and thus $p_0 \xrightarrow{\vec{w}} p'$. With $p_0 \preceq_C q_0$, there must then exist a $q' \in S$ such that $q_0 \xrightarrow{\vec{w}} q'$ and $q' \preceq_C p'$. Thus, we have $(q', \{p'\}) \in C \subseteq F(C)$ and $q' \in \text{succs}(\vec{w}, \{q_0\}) = Q$ by application of Lemma 6. It follows immediately that $Q \Rightarrow q'$. \square

¹⁸  definition Contrasimulation.F

¹⁹  lemma Contrasimulation.R_is_in_F_of_R

²⁰  lemma Contrasimulation.F_of_C_guarantees_action_succ

²¹  lemma Contrasimulation.F_of_C_guarantees_tau_succ

We can now construct a positional defender strategy f_C using $F(C)$.

Definition 10 (Defender strategy f_C on \mathcal{G}_C). Wherever possible, a defender strategy f_C derived from $F(C)$ maps the current play fragment $g_0 \dots g_k$ to next positions as follows:

- If the last played position g_k is a swap node $\text{Swap}(p', Q)_d$, move to some attacker position $(q', \{p'\})_a$ with $(q', \{p'\}) \in F(C)$ and $Q \Rightarrow q'$, and
- if the last played position g_k is a simulation node $\text{Sim}(a, p', Q)_d$, move to the attacker position $(p', Q')_a$ where $Q \xrightarrow{a} Q'$.²²

Where there are no applicable moves, we leave f_C undefined. Note that there may exist several strategies satisfying these conditions. In the following, f_C is one of these defender strategies—it makes no difference which one.²³

Lemma 11 (F invariant). *Let $p_0, q_0 \in S$ be states and let g be a play of $\mathcal{G}_C[(p_0, \{q_0\})_a]$ consistent with f_C . If $p_0 \preceq_C q_0$, then we have $(p, Q) \in F(C)$ for all attacker positions $(p, Q)_a$ in g .*²⁴

Proof. This follows immediately from $C \subseteq F(C)$ for the initial position $(p_0, \{q_0\})_a$. All other attacker positions can only be reached if the defender moves to them, and, following f_C , the defender always moves in accordance with $F(C)$. \square

Lemma 12 (Completeness). *Let $p_0, q_0 \in S$ be states with $p_0 \preceq_C q_0$. Then f_C is a winning strategy on $\mathcal{G}_C[(p_0, \{q_0\})_a]$.*²⁵

Proof. We show by induction on $g \in G_{f_C}^\infty[(p_0, \{q_0\})_a]$ that the defender is never stuck. Let $g = g_0 g_1 \dots g_k$ be the current play fragment on $\mathcal{G}_C[(p_0, \{q_0\})_a]$.

At the initial position $g_0 = (p_0, \{q_0\})_a$, the defender cannot be stuck, since g_0 is an attacker position. We will assume the lemma holds for the current play fragment $g_0 \dots g_k$. For g_{k+1} we have the cases:

- g_{k+1} is an attacker position. Then the defender cannot be stuck by the same reasoning as for the base case.
- g_{k+1} is a defender position. Then there exists a position $g_k = (p, Q)_a$ from which the attacker has moved to g_{k+1} , therefore $(p, Q) \in F(C)$ must hold by Lemma 11. We will distinguish between types of defender positions for g_{k+1} :
 - $g_{k+1} = \text{Sim}(a, p', Q)_d$ is a simulation position. Then we have $p \xrightarrow{a} p'$. It follows from $(p, Q) \in F(C)$ and Lemma 10 that there is a set $Q' \subseteq S$ such that $Q \xrightarrow{a} Q'$ and $(p', Q') \in F(C)$. Thus, the defender can move to $(p', Q')_a$ with f_C and is therefore not stuck.
 - $g_{k+1} = \text{Swap}(p', Q)_d$ is a swapping position. Then we have $p \Rightarrow p'$. From $(p, Q) \in F(C)$ and Lemma 10, it follows that there exists a state $q' \in S$ such that $Q \Rightarrow q'$ and $(q', \{p'\}) \in F(C)$. Thus, the defender can move to $(q', \{p'\})_a$ with f_C and is therefore not stuck.

Hence, the defender is never stuck in a play consistent with f_C , and thus f_C is a winning strategy for the defender. \square

Combining Lemma 8 and Lemma 12, we get:

Theorem 13. *The defender wins $\mathcal{G}_C[(p, \{q\})_a]$ precisely if $p \preceq_C q$.*²⁶

²² `fun Contrasm_Set_Game.strategy_from_F_of_C`

²³ In our Isabelle/HOL formalization, we used its Hilbert's choice operator SOME.

²⁴ `lemma Contrasm_Set_Game.set_game_strategy_retains_F`

²⁵ `lemma Contrasm_Set_Game.set_contrasm_game_complete`

²⁶ `theorem Contrasm_Set_Game.winning_strategy_in_set_game_iff_contrasm`

5 Discussion and Related Work

The presented game closes the last interesting gap in the *landscape of game characterizations for equivalences in the linear-time–branching-time spectrum with internal steps* [10]. De Frutos Escrig et al.’s games for branching, delay, η and weak bisimulation [8] and ours for coupled simulation [6] are polynomial in the size of the system state space. Due to the subset construction, the presented contrasimulation game needs exponentially many game positions. Compared to defining the equivalence games on words as Chen and Deng [7], the subset construction is still preferable, as it yields finite games for finite-state systems. Our group has used the same subset approach in [5]. There, the virtue of the single-action construction lay in making attacker strategies correspond closely with all relevant distinguishing Hennessy–Milner logic formulas.

A side-effect of our work is that we have formalized contrasimilarity (and its characterizing games) in *Isabelle/HOL*, based on our prior work in [3]. This is not the first such formalization. Peters and van Glabbeek [17, 18] provided an Isabelle theory of contrasimilarity tailored to reduction semantics and to the analysis of encodings between formalisms. Moreover, Bell [2] developed a Coq formalization to support the verification of compilers using contrasimilarity, still available at <http://people.csail.mit.edu/cj/par/>.

Voorhoeve and Mauw [20] examined a *modal-logical characterization* of contrasimilarity consisting of \top , \perp , conjunction $\varphi \wedge \psi$, disjunction $\varphi \vee \psi$ and a special necessity operator on paths with immediate negation $\Box_W \neg \varphi$, which is true at p iff φ is true at all p' with $p \xrightarrow{\vec{w}} p'$ and $\vec{w} \in W$. Although this paper has not been about modal logic, there is a strong link between games and modal logic. One can read our game in Figure 5 as a way of enumerating possible distinguishing Hennessy–Milner logic (HML) formulas (with $\langle \varepsilon \rangle$ denoting points of possible internal behavior). Then, the Sim-branch corresponds to a delayed observation $\langle \varepsilon \rangle \langle a \rangle \varphi$ and the Swap-branch to a delayed logical nor $\langle \varepsilon \rangle \neg (\varphi_1 \vee \varphi_2 \vee \dots)$.²⁷ For instance, a distinguishing formula for the non-contrasimilar systems of Example 2 would be $\langle \varepsilon \rangle \neg \langle \varepsilon \rangle \langle \text{op} \rangle \langle \varepsilon \rangle \langle \text{aEats} \rangle$.

Observation $\langle a \rangle \varphi$ and logical nor $\neg (\varphi_1 \vee \varphi_2 \vee \dots)$ form a functionally complete set of operators for HML and thus characterize (strong) bisimilarity. So it is aesthetically pleasing that “weakening” this observation language by alternating its constructs and $\langle \varepsilon \rangle$, thereby allowing internal behavior “in between the connectives” leads precisely to contrasimilarity. This view provides further evidence why contrasimilarity is a quite sensible way of generalizing bisimilarity to systems with internal steps as discussed in Section 2.

Contrasimilarity is an only slightly coarser sibling of coupled similarity [16]. For coupled similarity, our group has been able to prove that it can be decided in cubic time, and cannot be cheaper than deciding weak similarity [4]. The exponential game of the present paper induces an exponential algorithm for deciding contrasimilarity. We have integrated the contrasimilarity algorithm into our prototypical coupled similarity checker on <https://coupledsim.bbisping.de>. So, a side effect of the work presented here is to provide the first tool support for checking contrasimilarity.

It seems strange that deciding contrasimilarity, which coincides with coupled similarity in many cases, should be so much more expensive. We have not yet thought of a decisive argument for the complexity of contrasimilarity checking. Due to the results for the spectrum without internal steps [14], one may presume the coarser siblings of contrasimilarity from weak impossible futures down to weak trace equivalence to all be PSPACE-hard. But while deciding weak possible futures and nestings of possible

²⁷For more on why focusing the defender on one position is conjunction and swapping sides is negation (which together amounts to NOR), see our group’s recent paper on linear-time–branching-time spectroscopy [5].

futures equivalence is PSPACE, deciding its arbitrarily nested limit, weak bisimilarity, is PTIME. So how come that the arbitrarily nested limit of impossible futures equivalence, contrasimilarity, seems to be beyond PTIME?

The possibility of fooling depth-restricted approximations of contrasimilarity using instable external choice points as in Example 3 suggests that one indeed must cater for the complexity of handling words. We would very much like to be proven wrong with regard to this. Also, not every formalism is expressive enough to raise this issue. In order to avoid the problem, it might suffice to ensure that all observations are committed, which would rule out Example 3. Fournet and Gonthier [9] showed that parts of the hierarchy of equivalences around contrasimilarity collapse for reduction semantics if transient barbs are excluded. Then, however, cheaper algorithms for easier predicates such as coupled simulation should also be applicable right away. More research would be needed in order to pinpoint where exactly one cannot do without arbitrary-length word contrasimulation.

For this, and all the other nice things one could do using contrasimilarity, everyone is welcome to use our Isabelle/HOL formalization.²⁸

Acknowledgments. We are thankful to the members of the TU Berlin research group Modelle und Theorie Verteilter Systeme, especially Uwe Nestmann, for supporting us in the preparation of this paper. Many thanks also to the EXPRESS/SOS reviewers!

References

- [1] J. C. M. Baeten, P. J. L. Cuijpers & P. J. A. van Tilburg (2008): *A Context-Free Process as a Pushdown Automaton*. In Franck van Breugel & Marsha Chechik, editors: *CONCUR 2008 - Concurrency Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 98–113, doi:10.1007/978-3-540-85361-9_11.
- [2] Christian J. Bell (2013): *Certifiably sound parallelizing transformations*. In Georges Gonthier & Michael Norrish, editors: *Certified Programs and Proofs*, 8307, Springer International Publishing, pp. 227–242, doi:10.1007/978-3-319-03545-1_15.
- [3] Benjamin Bisping (2019): *Isabelle/HOL proof and Apache Flink program for TACAS 2019 paper: Computing Coupled Similarity*, doi:10.6084/m9.figshare.7831382.v1.
- [4] Benjamin Bisping & Uwe Nestmann (2019): *Computing coupled similarity*. In: *Proceedings of TACAS, LNCS*, Springer, pp. 244–261, doi:10.1007/978-3-030-17462-0_14.
- [5] Benjamin Bisping & Uwe Nestmann (2021): *A Game for Linear-time–Branching-time Spectroscopy*. In Jan Friso Groote & Kim Guldstrand Larsen, editors: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer International Publishing, Cham, pp. 3–19, doi:10.1007/978-3-030-72016-2_1.
- [6] Benjamin Bisping, Uwe Nestmann & Kirstin Peters (2020): *Coupled similarity: the first 32 years*. *Acta Informatica* 57(3), pp. 439–463, doi:10.1007/s00236-019-00356-4.
- [7] Xin Chen & Yuxin Deng (2008): *Game characterizations of process equivalences*. In G. Ramalingam, editor: *Programming Languages and Systems*, 5356, Springer Berlin Heidelberg, pp. 107–121, doi:10.1007/978-3-540-89330-1_8.
- [8] David De Frutos Escrig, Jeroen J. A. Keiren & Tim A. C. Willemse (2017): *Games for bisimulations and abstraction*. *Logical Methods in Computer Science* 13(4), pp. 1–40, doi:10.23638/LMCS-13(4:15)2017.
- [9] Cédric Fournet & Georges Gonthier (2005): *A hierarchy of equivalences for asynchronous calculi*. *The Journal of Logic and Algebraic Programming* 63(1), pp. 131–173, doi:10.1016/j.jlap.2004.01.006.
- [10] Rob J. van Glabbeek (1993): *The linear time–branching time spectrum II*. In: *International Conference on Concurrency Theory*, Springer, pp. 66–81, doi:10.1007/3-540-57208-2_6.

²⁸Available from <https://github.com/luisamontanari/ContrasimGame>.

- [11] Rob J. van Glabbeek (2001): *The linear time–branching time spectrum I. The semantics of concrete, sequential processes*. In: *Handbook of Process Algebra*, Elsevier, pp. 3–99, doi:10.1016/B978-044482830-9/50019-9.
- [12] Erich Grädel (2007): *Finite model theory and descriptive complexity*. In: *Finite Model Theory and its Applications*, Springer, pp. 125–230, doi:10.1007/3-540-68804-8_3.
- [13] Zeno de Hoop (2017): *Context-Free Processes and Push-Down Processes*. Master’s thesis, Universiteit van Amsterdam. Available at <https://eprints.iillc.uva.nl/id/eprint/1561>.
- [14] Hans Hüttel & Sandeep Shukla (1996): *On the Complexity of Deciding Behavioural Equivalences and Preorders. A Survey*. *BRICS Report Series* 3(39), doi:10.7146/brics.v3i39.20021.
- [15] Robin Milner (1989): *Communication and Concurrency*. PHI Series in computer science, Prentice Hall Englewood Cliffs.
- [16] Joachim Parrow & Peter Sjödin (1994): *The complete axiomatization of Cs-congruence*. In: *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, pp. 555–568, doi:10.1007/3-540-57785-8_171.
- [17] Kirstin Peters & Rob J. van Glabbeek (2015): *Analysing and Comparing Encodability Criteria*. In: *Proceedings of the Combined 22th International Workshop on Expressiveness in Concurrency and 12th Workshop on Structural Operational Semantics, and 12th Workshop on Structural Operational Semantics, EXPRESS/SOS*, pp. 46–60, doi:10.4204/EPTCS.190.4.
- [18] Kirstin Peters & Rob J. van Glabbeek (2015): *Analysing and Comparing Encodability Criteria for Process Calculi*. *Archive of Formal Proofs*. https://isa-afp.org/entries/Encodability_Process_Calculi.html, Formal proof development.
- [19] Colin Stirling (1999): *Bisimulation, modal logic and model checking games*. *Logic Journal of IGPL* 7(1), pp. 103–124, doi:10.1093/jigpal/7.1.103.
- [20] Marc Voorhoeve & Sjouke Mauw (2001): *Impossible futures and determinism*. *Information Processing Letters* 80(1), pp. 51–58, doi:10.1016/S0020-0190(01)00217-4.
- [21] Makarius Wenzel et al. (2021): *The Isabelle/Isar Reference Manual*. Available at <https://isabelle.in.tum.de/doc/isar-ref.pdf>.