# Dynamic Cut-Off Algorithm for Parameterised Refinement Checking*

Antti Siirtola[1][0000−0001−9118−5087] and Keijo Heljanko[2][0000−0002−4547−2701]

[1] University of Oulu, Faculty of Information Technology and Electrical Engineering, M3S Research Group, Finland
antti.siirtola@oulu.fi
[2] Aalto University, Department of Computer Science, Finland
University of Helsinki, Department of Computer Science, Finland
Helsinki Institute for Information Technology (HIIT)
keijo.heljanko@iki.fi

**Abstract.** The verification of contemporary software systems is challenging, because they are heavily parameterised containing components, the number and connections of which cannot be a priori fixed. We consider the multi-parameterised verification of safety properties by refinement checking in the context of labelled transition systems (LTSs). The LTSs are parameterised by using first-order constructs, sorts, variables, and predicates, while preserving compositionality. This allows us to parameterise not only the number of replicated components but also the system topology, the connections between the components. We aim to solve a verification task in the parameterised LTS formalism by determining cut-offs for the parameters. As the main contribution, we convert this problem into the unsatisfiability of a first-order formula and provide a SAT modulo theories (SMT)-based semi-algorithm for dynamically, i.e., iteratively, computing the cut-offs. The algorithm will always terminate for topologies expressible in the $\exists^*\forall^*$ fragment of first-order logic. It also enables us to consider systems with topologies beyond this fragment, but for these systems, the algorithm is not guaranteed to terminate. We have implemented the approach on top of the Z3 SMT solver and successfully applied it to several system models. As a running example, we consider the leader election phase of the Raft consensus algorithm and prove a cut-off of three servers which we conjecture to be the optimal one.

**Keywords:** labelled transition systems · refinement checking · safety properties · compositional verification · parameterized systems · cut-off · first-order logic · satisfiability modulo theories

## 1 Introduction

Contemporary software systems are not only highly concurrent and distributed but also heavily parameterised containing components, the number and connections of which cannot be a priori fixed. Since these systems are everywhere

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-02146-7_13.

around us, it is essential to verify that at least the most critical of them operate properly in all circumstances, i.e., for all possible parameter values. Moreover, since some subsystems (e.g., external software packages and subsystems concurrently under construction) can only be available in an interface specification form, we often need to be able to do their verification in a compositional way.

**Contribution** We consider the multi-parameterised verification of safety properties in the context of labelled transition systems (LTSs) with trace refinement preorder and parallel composition and hiding operators. The LTSs are parameterised by using the constructs of first-order logic (FOL), *sorts* (a.k.a. *types*), typed *variables*, and *predicates*, such that compositional verification is possible in the parameterised setting, too. Sorts are used to parameterise the number of replicated components whereas predicates enable us to parameterise the system topology, i.e., the connections between the components.

Our goal is to solve a verification task in the parameterised LTS (PLTS) formalism by determining cut-offs for the parameters such that in order to prove a parameterised system implementation correct with respect to its specification, it is sufficient to consider only finitely many instances up to the cut-offs. As the main result, we show how this problem can be converted into the unsatisfiability of a first-order formula (Thm 23). The result is accompanied by a SAT modulo theories (SMT)-based semi-algorithm for computing the cut-offs. The algorithm is called dynamic because it computes the cut-offs iteratively until the unsatisfiability condition is met. The algorithm is implemented and successfully applied to several parameterised system models, including the repeatable read property of taDOM2+ XML database protocol [10] with the tree topology and the leader election phase of the Raft consensus protocol with the quorum topology [17, 21].

Our approach is based on the precongruence reduction (PR) technique previously used to prove *static* cut-offs for PLTSs with predicates defined in the universal fragment ($\forall^*$) of FOL [24] and for PLTSs with special quorum functions [21]. The technique is also adapted to parameterised modal interface automata without predicates [23]. In general, the PR technique applies to implementation-specification pairs the topology of which is downward-closed in the sense that any (big) instance can be represented as a composition of smaller instances. For example star, bipartite, totally connected, linear, tree, and quorum topologies are such, but not all their combinations.

Static cut-off results [24, 21] are syntax-based and restricted to topologies specifiable in fragments of FOL. Consequently, a separate result is needed for each such fragment. The dynamic algorithm introduced here is not restricted to any syntactic fragment and we can basically use the full expressive power of FOL. The dynamic algorithm will always terminate for topologies expressible in the $\exists^*\forall^*$ fragment of first-order logic, because the fragment is decidable [8] and such topologies are downward-closed [24]. The algorithm also enables us to consider systems with downward-closed topologies beyond this fragment, but for these systems, termination depends on the capabilities of the used SMT solver. Nevertheless, the dynamic algorithm not only enables us to treat the

parameters of [24, 21] in a uniform way but it also allows for the use of parameters that are beyond those handled by the static methods. Moreover, based on our experiments, the dynamic algorithm produces at least as tight or even smaller cut-offs than the static cut-off methods. For example, the static cut-off for the number of servers in Raft is seven whereas the computed dynamic cut-off is only three, which we conjecture to be the optimal one. Earlier, the cut-offs of 5-7 are proved for other consensus algorithms [15].

**Related Work** The distinctive features of our solution to the parameterised verification problem are compositionality, the support for multiple and topology related parameters, and the dynamic computation of cut-offs.

As regards compositionality, the process algebraic approaches of Valmari & Tienari [25], Lazić [13], and Creese [2] are the closest works. Valmari & Tienari [25] present a generic induction method for parameterised verification. However, a crucial part of the technique is to come up with an invariant process which is a task that cannot be automated in general. Lazić considers data-independent systems which can handle infinite or arbitrarily large data types. His approach allows multiple parameters and provides static cut-offs for the size of data types. There is also a more general version of the results based on infinite automata [14] but in this context, compositionality is not considered. Moreover, neither approach allows the number of concurrent components nor the system topology to be parameterised. The limitation is overcome by Creese who combines the data-independence results with the induction method [2]. Simultaneously, however, full automation is lost.

Multi-parameterised verification is also considered by Emerson & Kahlon [4], Hanna et al. [9], and Yang & Li [26]. The approaches [4, 26] are based on static cut-offs, whereas [9] uses the iterative computation of cut-offs. The methods apply to systems with guarded broadcasts [4], shared actions [9], or rendezvous communication [26] and specifications are given in temporal logic [4, 9] or as property automata [26]. However, the formalisms do not lend support to compositionality. In addition to [25, 2, 9], dynamic cut-off computation is previously considered by Kaiser, Kroening & Wahl [12], too. Their approach can be used for the verification of reachability in boolean programs, but it does not lend support to multiple parameters nor compositional reasoning.

Completely different approaches to parameterised verification are based on abstract interpretation [27] and infinite-state verification algorithms [5]. While some of these techniques lend support to multiple parameters, they are typically not compositional. The additional benefit of our approach over these techniques is that we can exploit efficient finite-state model checkers for verification and since abstraction is not involved, false error traces are avoided.

**Outline** The next three sections are preliminaries; they cover the basics of LTSs, FOL, and PLTSs. In Sect. 5, we present our main contribution, the dynamic cut-off algorithm. The paper concludes with discussion on future research.

## 2 Labelled Transition Systems

In this section, we briefly recall a Communicating Sequential Processes (CSP)-like LTS-based process calculus with parallel composition and hiding operators and trace refinement preorder [18]. We use LTSs to express system components.

We assume a countably infinite set of *events*. One of the events is *invisible*, denoted $\tau$, and the other ones are *visible*. The visible events have an explicit channel and data part; we assume countably infinite sets $\mathbb{C}$ and $\mathbb{A}$ of, respectively, *channels* and *atoms* and that each visible event is of the form $c(a_1, \ldots, a_n)$, where $c$ is a channel and $a_1, \ldots, a_n \in \mathbb{A}$ are atoms.

A *labelled transition system (LTS)* is a four-tuple $L := (S, E, R, \dot{s})$, where (1) $S$ is a finite non-empty set of *states*, (2) $E$ is a finite set of visible events, also called an *alphabet*, (3) $R \subseteq S \times (E \cup \{\tau\}) \times S$ is a set of *transitions*, and (4) $\dot{s}$ is the *initial* state.

Let $L_i$ be an LTS $(S_i, E_i, R_i, \dot{s}_i)$ for both $i \in \{1, 2\}$. The *parallel composition (of $L_1$ and $L_2$)* is an LTS $(L_1 \parallel L_2) := (S_1 \times S_2, E_1 \cup E_2, R_\parallel, (\dot{s}_1, \dot{s}_2))$, where $R_\parallel$ is the set of all triples $((s_1, s_2), \alpha, (s'_1, s'_2))$ such that either (1) $\alpha \neq \tau$ and $(s_i, \alpha, s'_i) \in R_i$ for both $i \in \{1, 2\}$; (2) $(s_1, \alpha, s'_1) \in R_1$, $\alpha \notin E_2$, $s_2 \in S_2$, and $s'_2 = s_2$; or (3) $(s_2, \alpha, s'_2) \in R_2$, $\alpha \notin E_1$, $s_1 \in S_1$, and $s'_1 = s_1$.

Let $L$ be an LTS $(S, E, R, \dot{s})$ and $E'$ a set of visible events. The LTS $L$ *after hiding* $E'$ is an LTS $(L \setminus E') := (S, E \setminus E', R_\setminus, \dot{s})$, where $R_\setminus$ is the set of (1) all triples $(s, \alpha, s') \in R$ such that $\alpha \notin E'$; and (2) all triples $(s, \tau, s')$ such that $(s, \alpha, s') \in R$ for some $\alpha \in E'$.

A finite alternating sequence $(s_0, \alpha_1, s_1, \ldots, \alpha_n, s_n)$ of states and events of $L$ is an *execution of $L$* if $s_0$ is the initial state and $(s_{i-1}, \alpha_i, s_i)$ is a transition of $L$ for every $i \in \{1, \ldots, n\}$. A finite sequence of visible events is a *trace (of $L$)*, if there is an execution of $L$ such that the sequence can be obtained from the execution by erasing all the states and the invisible events. The set of all the traces of $L$ is denoted by $\mathrm{tr}(L)$. An LTS $L_1$ is a *trace refinement* of an LTS $L_2$, denoted $L_1 \preceq_{\mathrm{tr}} L_2$, if $L_1$ and $L_2$ have the same alphabet and $\mathrm{tr}(L_1) \subseteq \mathrm{tr}(L_2)$ [11]. The LTSs $L_1$ and $L_2$ are *trace equivalent*, denoted $L_1 \equiv_{\mathrm{tr}} L_2$, if and only if $L_1 \preceq_{\mathrm{tr}} L_2$ and $L_2 \preceq_{\mathrm{tr}} L_1$. Clearly, $\preceq_{\mathrm{tr}}$ is a preorder (i.e., a reflexive and transitive relation) and $\equiv_{\mathrm{tr}}$ an equivalence relation on the set of LTSs.

The operators and the trace relations have many useful properties [11, 18, 24], which are also exploited in the proofs. The parallel composition is commutative, associative, and idempotent with respect to $\equiv_{\mathrm{tr}}$ (i.e., $L \parallel L \equiv_{\mathrm{tr}} L$ for all LTSs $L$) and a single-state LTS $L_{id} := (\{\dot{s}\}, \emptyset, \emptyset, \dot{s})$ with the empty alphabet and no transition is the identity element of $\parallel$. This allows us to extend $\parallel$ to every finite set $I = \{i_1, \ldots, i_n\}$ and all LTSs $L_{i_1}, \ldots, L_{i_n}$ by defining $(\|_{i \in I} L_i) = (\|_{k=1}^{n} L_{i_k}) := (L_{i_1} \parallel (\|_{i \in I \setminus \{i_1\}} L_i))$, when $n > 0$, and $(\|_{i \in I} L_i) = (\|_{k=1}^{n} L_{i_k}) := L_{id}$, when $n = 0$. Moreover, distributing hiding over parallel composition results in an LTS greater in the preorder; $(L_1 \parallel L_2) \setminus E \preceq_{\mathrm{tr}} (L_1 \setminus E) \parallel (L_2 \setminus E)$ for all LTSs $L_1, L_2$ and every set $E$ of visible events [24]. Finally, $\equiv_{\mathrm{tr}}$ is compositional with respect to the parallel composition and hiding operators; if $L_1 \preceq_{\mathrm{tr}} L_2$, then $L_1 \parallel L_3 \preceq_{\mathrm{tr}} L_2 \parallel L_3$ and $L_1 \setminus E \preceq_{\mathrm{tr}} L_2 \setminus E$ for all LTSs $L_1, L_2, L_3$ and all sets $E$ of visible events. Hence, $\preceq_{\mathrm{tr}}$ is a precongruence and $\equiv_{\mathrm{tr}}$ a congruence on LTSs.

## 3 Many-Sorted First-Order Logic

In this section, we introduce first-order logic (FOL) [6] with *sorts* (a.k.a. *types*), *variables*, and *predicates*. We use FOL to express system topologies.

We assume sets of sorts, variables, and predicates, denoted by $\mathbb{T}$, $\mathbb{X}$, and $\mathbb{F}$, respectively, that are disjoint and countably infinite. We assume that for each atom $a \in \mathbb{A}$, there is a sort $T_a \in \mathbb{T}$ and for each sort $T \in \mathbb{T}$, the set $\mathbb{A}_T := \{a \in \mathbb{A} \mid T_a = T\}$ is countably infinite. Hence, $\mathbb{A}_T$ and $\mathbb{A}_S$ are disjoint whenever $T$ and $S$ are different sorts. Moreover, we assume that for each variable $x \in \mathbb{X}$ there is a sort $T_x \in \mathbb{T}$, and for each predicate $F$ there is an arity $n_F \in \mathbb{Z}_+$ and a tuple of sorts $\mathbf{T}_F = (T_F^1, \ldots, T_F^{n_F})$ specifying the domain of the predicate.

The *atomic propositions* are of the form $\top$ (always true), $x = y$ (equivalence), and $F(x_1, \ldots, x_n)$ (predicate application), where $x$ and $y$ are variables, $F$ is a predicate with arity $n$, and $x_1, \ldots, x_n$ are variables of the sort $T_F^1, \ldots, T_F^n$, respectively. The *formulae* of FOL are defined by the grammar

$$\mathcal{V} ::= p \mid \neg\mathcal{V} \mid \mathcal{V} \wedge \mathcal{V} \mid \mathcal{V} \vee \mathcal{V} \mid \forall x.\mathcal{V} \mid \exists x.\mathcal{V} \,,$$

where $x$ denotes a variable and $p$ an atomic proposition. We also write $x \neq y$ (inequivalence) and $\mathcal{V}_1 \to \mathcal{V}_2$ (implication) short for $\neg(x = y)$ and $(\neg\mathcal{V}_1) \vee \mathcal{V}_2$, respectively. A *propositional formula* or a *guard* is a formula without quantified structures of the form $\forall x.\mathcal{V}$ and $\exists x.\mathcal{V}$. A formula is in the *prenex normal form* with *quantifier alternation* $Q_1, \ldots, Q_n$ if it is of the form $Q_1 x_1 . \cdots . Q_n x_n . \mathcal{V}$, where $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ are quantifiers and $\mathcal{V}$ is propositional.

A *signature function* maps a formula $\mathcal{V}$ to a finite set of sorts, variables, and predicates, which are the parameters of $\mathcal{V}$. The signature function, denoted par, is defined inductively:

1. $\mathrm{par}(\top) = \emptyset$,
2. $\mathrm{par}(x = y) = \{x, y, T_x, T_y\}$,
3. $\mathrm{par}(F(x_1, \ldots, x_n)) = \{F, x_1, \ldots, x_n, T_{x_1}, \ldots, T_{x_n}\}$,
4. $\mathrm{par}(\neg\mathcal{V}) = \mathrm{par}(\mathcal{V})$,
5. $\mathrm{par}(\mathcal{V}_1 \wedge \mathcal{V}_2) = \mathrm{par}(\mathcal{V}_1 \vee \mathcal{V}_2) = \mathrm{par}(\mathcal{V}_1) \cup \mathrm{par}(\mathcal{V}_2)$,
6. $\mathrm{par}(\forall x.\mathcal{V}) = \mathrm{par}(\exists x.\mathcal{V}) = (\mathrm{par}(\mathcal{V}) \setminus \{x\}) \cup \{T_x\}$ ($x$ is considered bound).

We write $\mathrm{par}_X(\mathcal{V})$ for the restriction $\mathrm{par}(\mathcal{V}) \cap X$ of the signature to a set $X$.

A formula is evaluated by using a *valuation* function which assigns values to sorts, variables, and predicates.

**Definition 1 (Valuation).** *A* valuation *is a function $\phi$ such that*

1. *the domain of $\phi$ is a finite set of sorts, variables, and predicates,*
2. *for each sort $T \in \mathrm{dom}(\phi)$, $\phi(T)$ is a finite non-empty subset of $\mathbb{A}_T$,*
3. *for each variable $x \in \mathrm{dom}(\phi)$, $T_x \in \mathrm{dom}(\phi)$ and $\phi(x) \in \phi(T_x)$, and*
4. *for each predicate $F \in \mathrm{dom}(\phi)$, $T_F^1, \ldots, T_F^{n_F} \in \mathrm{dom}(\phi)$ and $\phi(F)$ is a subset of $\phi(T_F^1) \times \ldots \times \phi(T_F^{n_F})$.*

We write $\mathrm{dom}_X(\phi)$ for the restriction $\mathrm{dom}(\phi) \cap X$ of the domain to a set $X$ and $\mathrm{Im}(\phi)$ for the set $\bigcup_{T \in \mathrm{dom}_\mathbb{T}(\phi)} \phi(T)$ of all atoms in the image of $\phi$. The complement $\overline{\phi(F)}$ of the value of a predicate $F$ is the set $\phi(T_F^1) \times \ldots \times \phi(T_F^{n_F}) \setminus \phi(F)$. A valuation $\phi$ is *compatible* with a formula $\mathcal{V}$ if $\mathrm{par}(\mathcal{V}) \subseteq \mathrm{dom}(\phi)$. The *instance* of $\mathcal{V}$ *generated by* a compatible valuation $\phi$, denoted $[\![\mathcal{V}]\!]_\phi$, is a truth value obtained in the usual way by substituting $\phi(T)$ for each sort $T$, $\phi(x)$ for each variable $x$ not bound in $\mathcal{V}$, and $\phi(F)$ for each predicate $F$ occurring in $\mathcal{V}$ and by evaluating the operators. We say that $\phi$ *satisfies* $\mathcal{V}$, if $[\![\mathcal{V}]\!]_\phi$ is true. The satisfiability problem in FOL asks whether for a given formula $\mathcal{V}$, there is a valuation satisfying $\mathcal{V}$. The problem is undecidable in general, but the fragment consisting of the formulae with the quantifier alternation $\exists^*\forall^*$ is decidable [8]. Many-sorted FOL considered here has several other known decidable fragments [1], too, but since many of them do not contain full $\exists^*\forall^*$, they are of limited use from the viewpoint of our algorithm introduced in Sect. 5.

## 4 Parameterised Labelled Transition Systems

In this section, we parameterise LTSs with first-order constructs, sorts, variables, and predicates, while preserving compositionality. With minor syntactic differences, this is done as in [24]. Parameterised LTSs can express systems with an unbounded number of replicated components and we use them to model both system implementations and specifications.

*Example 2.* As a running example, we consider the leader election phase of the Raft consensus algorithm [17]. In Raft, time is divided into *terms* of arbitrary length and a server can crash at any moment. When a server is running, it is in one of the three states, a follower, candidate, or leader. A server always (re)starts as a follower. A follower can vote for at most one server in a term. If a follower does not regularly receive messages from the leader, it increases its term and promotes itself to a candidate. A candidate sends vote requests to the other servers and if it receives a quorum of votes, it becomes a leader. Our goal is to formally prove that in each term, there is at most one leader independent of the number of terms and the size of the cluster.

For our Raft model, we pick a sort $T_S$ to represent the set $\{s_1, \ldots, s_n\} \subseteq \mathbb{A}_{T_S}$ of the identifiers of servers and a sort $T_T$ to represent the set $\{t_1, \ldots, t_m\} \subseteq \mathbb{A}_{T_T}$ of the identifiers of terms. We also use a predicate $Q_S$ with $\mathbf{T}_{Q_S} = (T_S, T_T, T_S)$ to assign each server and each term a set of servers from which the server needs a vote in order to become a leader in the term. Variables $x_i$ of the sort $T_S$ are used to refer to individual servers and a variable $y$ of the sort $T_T$ is used to refer to a specific term.

Next, we specify the values of $Q_S$. We require that for each server $x_0$ and each term $y$, the set $Q_{x_0,y} := \{x_1 \mid Q_S(x_0, y, x_1)\}$ of servers, from which $x_1$ needs a vote in order to became a leader in the term $y$, is a quorum set (the set covers more than half the servers including $x_0$ itself) or the empty set (the server cannot become a leader in the term $y$). Allowing the empty set is needed to

make the topology downward-closed so that our approach, introduced in the next section, can be successfully applied. Since we do not have an explicit construct for restricting the size of a set, we require that whenever $Q_{x_0,y}$ is non-empty then for every $x_1 \notin Q_{x_0,y}$, there is a unique $x_2 \in Q_{x_0,y}$. For this purpose, we introduce a partial function $f : (x_0, y, x_1) \mapsto x_2$ which is encoded as a predicate $F_S$ with $\mathbf{T}_{F_S} = (T_S, T_T, T_S, T_S)$ such that $f(x_0, y, x_1) = x_2$ if and only if $F_S(x_0, y, x_1, x_2)$. The values of $Q_S$ and $F_S$ are expressed as a formula

$$\begin{aligned} Qrm :=& \forall x_0.\forall y.((\forall x_1.\neg Q_S(x_0,y,x_1)) \vee \\ & (Q_S(x_0,y,x_0) \wedge \forall x_1.(\neg Q_S(x_0,y,x_1) \rightarrow \exists x_2.F_S(x_0,y,x_1,x_2)))) \wedge \\ & \forall x_0.\forall x_1.\forall x_2.\forall y.(F_S(x_0,y,x_1,x_2) \rightarrow (Q_S(x_0,y,x_2) \wedge \neg Q_S(x_0,y,x_1))) \wedge \\ & \forall x_0.\forall x_1.\forall x_2.\forall x_3.\forall y.((F_S(x_0,y,x_1,x_3) \wedge F_S(x_0,y,x_2,x_3)) \rightarrow x_1 = x_2) \wedge \\ & \forall x_0.\forall y.\exists x_1.\forall x_2.(\neg F_S(x_0,y,x_2,x_1) \wedge \neg F_S(x_0,y,x_1,x_2)) \ . \end{aligned}$$

The first three lines of the formula state the relationship between $Q_S$ and $F_S$, the next line says that the function represented by $F_S$ is injective, and the last line indirectly guarantees that if $Q_{x_0,y}$ is non-empty then it is larger than its complement, i.e., a quorum set covers more than half the servers. The formula is outside the decidable fragment $\exists^*\forall^*$, because it involves a quantifier alternation $\forall\exists\forall$, but our algorithm still terminates on this running example. □

Parameterised LTSs are constructed from LTSs with variables substituted for the atoms, propositional formulae used as guards, and (replicated) parallel composition and hiding constructs which can be thought as operators on parameterised LTSs. Replicated parallel composition allows for parameterising the number of components while guards are used to restrict the system topology.

**Definition 3 (PLTS).** Parameterised LTSs (PLTSs) *are defined inductively:*

1. *If $L$ is an LTS, $a_1, \ldots, a_n$ are the atoms occurring in its alphabet, and $x_1, \ldots, x_n$ are variables such that $T_{a_i} = T_{x_i}$ for all $i \in \{1, \ldots, n\}$, then a function $L(x_1, \ldots, x_n) := \lambda a_1, \ldots, a_n.L$ is an (elementary) PLTS.*
2. *If $\mathcal{P}$ is a PLTS and $G$ a guard, then $([G]\mathcal{P})$ is a (guarded) PLTS.*
3. *If $\mathcal{P}_1$ and $\mathcal{P}_2$ are PLTSs, then $(\mathcal{P}_1 \parallel \mathcal{P}_2)$ is a (parallel) PLTS.*
4. *If $\mathcal{P}$ is a PLTS and $x$ a variable, then $(\|_x \mathcal{P})$ is a (replicated parallel) PLTS.*
5. *If $\mathcal{P}$ is a PLTS and $C$ a finite set of channels, $(\mathcal{P} \setminus C)$ is a (hiding) PLTS.*

The signature function is extended to the set of PLTSs by setting:

1. $\mathrm{par}(L(x_1, \ldots, x_n)) = \{x_1, \ldots, x_n, T_{x_1}, \ldots, T_{x_n}\}$,
2. $\mathrm{par}([G]\mathcal{P}) = \mathrm{par}(G) \cup \mathrm{par}(\mathcal{P})$,
3. $\mathrm{par}(\mathcal{P}_1 \parallel \mathcal{P}_2) = \mathrm{par}(\mathcal{P}_1) \cup \mathrm{par}(\mathcal{P}_2)$,
4. $\mathrm{par}(\|_x \mathcal{P}) = (\mathrm{par}(\mathcal{P}) \setminus \{x\}) \cup \{T_x\}$ ($x$ is considered bound), and
5. $\mathrm{par}(\mathcal{P} \setminus C) = \mathrm{par}(\mathcal{P})$.

The signature determines the *parameters* of the PLTS and a valuation $\phi$ is said to be *compatible* with a PLTS $\mathcal{P}$ if $\mathrm{par}(\mathcal{P}) \subseteq \mathrm{dom}(\phi)$. We sometimes write $\mathcal{P}(x_1, \ldots, x_n)$ to emphasise that $\mathcal{P}$ has the variables $x_1, \ldots, x_n$ as parameters.

A PLTS is evaluated to an LTS by fixing the values of the parameters and by evaluating the operators. Handling binary operators is straightforward, but in order to evaluate a replicated construct $\|_x \mathcal{P}$ by using a compatible valuation $\phi$, we need to iterate over all the extensions of $\phi$ to $\{x\}$. Let $\phi$ be a valuation and $X$ a set of variables such that $T_x \in \mathrm{dom}_{\mathbb{T}}(\phi)$ for all $x \in X$. We write $\mathrm{ext}(\phi, X)$ for the set of all valuations $\phi'$ with the domain $\mathrm{dom}(\phi) \cup X$ such that $\phi'(x) \in \phi(T_x)$ for all $x \in X$ and $\phi'|_{\mathrm{dom}(\phi)\setminus X} = \phi|_{\mathrm{dom}(\phi)\setminus X}$, i.e., $\phi$ and $\phi'$ agree on the values of the parameters outside $X$. We can now extend the instantiation function $[\![\cdot]\!]_\phi$ to PLTSs $\mathcal{P}$ for which $\mathrm{par}(\mathcal{P}) \subseteq \mathrm{dom}(\phi)$.

**Definition 4 (Instance of a PLTS).** *Let $\mathcal{P}$ be a PLTS and $\phi$ a compatible valuation. The* $\phi$-instance of $\mathcal{P}$ *or the* instance of $\mathcal{P}$ (generated by $\phi$) *is denoted by $[\![\mathcal{P}]\!]_\phi$ and determined inductively as follows:*

1. $[\![L(x_1, \ldots, x_n)]\!]_\phi = L(\phi(x_1), \ldots, \phi(x_n))$,
2. $[\![[G]\mathcal{P}']\!]_\phi = \begin{cases} [\![\mathcal{P}']\!]_\phi, \text{ if } [\![G]\!]_\phi \text{ is true}, \\ L_{id}, \quad \text{if } [\![G]\!]_\phi \text{ is false (the instance has no behaviour)}, \end{cases}$
3. $[\![\mathcal{P}_1 \parallel \mathcal{P}_2]\!]_\phi = [\![\mathcal{P}_1]\!]_\phi \parallel [\![\mathcal{P}_2]\!]_\phi$,
4. $[\![\|_x \mathcal{P}']\!]_\phi = \|_{\phi' \in \mathrm{ext}(\phi, \{x\})} [\![\mathcal{P}']\!]_{\phi'}$,
5. $[\![\mathcal{P}' \setminus C]\!]_\phi = [\![\mathcal{P}']\!]_\phi \setminus \{c(a_1, \ldots, a_n) \mid c \in C, a_1, \ldots, a_n \in \mathbb{A}\}$.

*Example 5.* For the Raft specification, we use an event $\mathit{leader}(x_0, y)$ to denote that the server $x_0$ is chosen as a leader in the term $y$. First, we consider the specification from the viewpoint of two servers, $x_0$ and $x_1$, and a term $y$. PLTS $\mathit{Spec2}(x_0, x_1, y)$ on the left of Fig. 1 formally says that no two servers can become a leader during the same term but repeating a leader notification is fine.

Recall the definition of the predicate $Q_S$ from Ex. 2. As we let the variable $y$ to range over all term identifiers and $x_0$, $x_1$, and $x_2$ over all server identifiers, we obtain the model of the full specification as a PLTS

$$\mathit{Spec} := \|_{x_0} \|_{x_1} \|_{x_2} \|_{y} [Q_S(x_0, y, x_2) \wedge Q_S(x_1, y, x_2)] \mathit{Spec2}(x_0, x_1, y) \ ,$$

which says that for each term, there is at most one leader. The guard guarantees that for each term, we only consider servers with (non-empty) overlapping quorum sets. This is not a restriction since any two quorum sets are overlapping. Since $\mathit{Spec2}(x_0, x_1, y)$ has no bound variable, $\mathrm{par}(\mathit{Spec2}(x_0, x_1, y)) = \{x_0, x_1, y, T_S, T_T\}$, but $\mathrm{par}(\mathit{Spec}) = \{Q_S, T_S, T_T\}$ as *Spec* has only bound variables.

In order to visualize *Spec*, let us consider a valuation $\phi$ such that $\phi(T_T) = \{t_1\}$, $\phi(T_S) = \{s_1, \ldots, s_n\}$, and $\{x \mid (s_i, t_1, x) \in \phi(Q_S)\}$ is a quorum set for all $i \in \{1, \ldots, n\}$. Obviously, the valuation is compatible with *Spec* and the $\phi$-instance of *Spec* is a star-shaped LTS on the right of Fig. 1, which indeed says that there is at most one leader for the term $t_1$. □

We complete the PLTS formalism by extending the trace refinement relation to the set of PLTSs while preserving compositionality. However, instead of a single relation, there will be infinitely many, since we use formulae to define the allowed values of parameters.

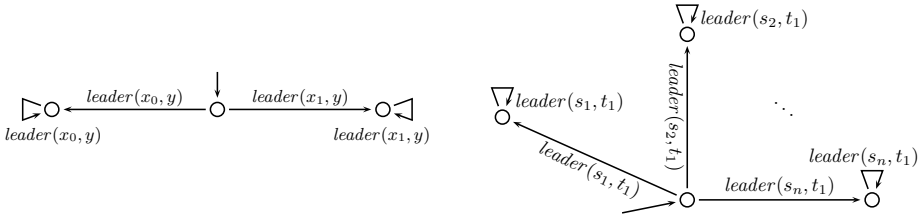


**Fig. 1.** On the left: PLTS $\mathit{Spec2}(x_0, x_1, y)$ representing the Raft specification from the viewpoint of two servers, $x_0$ and $x_1$, and a term $y$. On the right: the instance of *Spec* representing the Raft specification from the viewpoint of $n$ servers $s_1, \ldots, s_n$ and a term $t_1$.

**Definition 6 (Trace refinement on PLTSs).** *Let $\mathcal{P}$ and $\mathcal{Q}$ be PLTSs and $\mathcal{V}$ a formula. We write $\mathcal{P} \preceq_{\mathrm{tr}}^{\mathcal{V}} \mathcal{Q}$, if $[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\phi$ for all valuations $\phi$ that are compatible with $\mathcal{P}$ and $\mathcal{Q}$ and satisfy $\mathcal{V}$.*

Obviously, the definition can also be restricted to valuations with the minimal domain $\mathrm{par}(\mathcal{P} \parallel \mathcal{Q}) \cup \mathrm{par}(\mathcal{V})$. Therefore, we write $\mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ for the set of all valuations $\phi$ which have the domain $\mathrm{par}(\mathcal{P} \parallel \mathcal{Q}) \cup \mathrm{par}(\mathcal{V})$ and satisfy $\mathcal{V}$.

Parameterised verification tasks can now be expressed as follows: Given an implementation PLTS $\mathcal{P}$, a specification PLTS $\mathcal{Q}$, and a topology formula $\mathcal{V}$, we consider $\mathcal{P}$ to be *correct (with respect to $\mathcal{Q}$) (when $\mathcal{V}$)*, if $\mathcal{P} \preceq_{\mathrm{tr}}^{\mathcal{V}} \mathcal{Q}$. This allows for the verification of safety properties. Parameterised trace refinement relations also enable compositional verification since they are precongruences. This follows from Def. 4 and the precongruence of $\preceq_{\mathrm{tr}}$.

**Proposition 7.** *For all formulae $\mathcal{V}$, the relation $\preceq_{\mathrm{tr}}^{\mathcal{V}}$ is a precongruence on the set of PLTSs [24].*

*Example 8.* For the Raft implementation, we use an event $\mathit{vote}(x_0, y, x_1)$ to denote that the server $x_0$ votes for the server $x_1$ in the term $y$ and an event $\mathit{candidate}(x_0, y)$ to denote that the server $x_0$ promotes itself to a candidate in the term $y$. The behaviour of the Raft implementation is modelled in the same fashion as the specification. First, we capture it in the follower/candidate mode from the viewpoint of three servers $x_0, x_1, x_2$ and a term $y$ in a PLTS $\mathit{Flw3}(x_0, x_1, x_2, y)$ on the left of Fig. 2. The PLTS says that in the term $y$, the server $x_0$ can vote for either $x_1$, $x_2$, or itself or become a candidate and vote for itself. When we let the variables $x_1$, $x_2$, and $y$ to range over all values in their domain (with the restriction that the values of $x_1$ and $x_2$ are different), we obtain the model of a single server $x_0$ running in the follower/candidate mode as a PLTS

$$\mathit{Flw}(x_0) := \mathop{\|}_{x_1} \mathop{\|}_{x_2} \| [x_1 \neq x_2] \mathop{\|}_{y} \mathit{Flw3}(x_0, x_1, x_2, y) \ ,$$

which states that a server can vote for at most one server in the term or become a candidate.

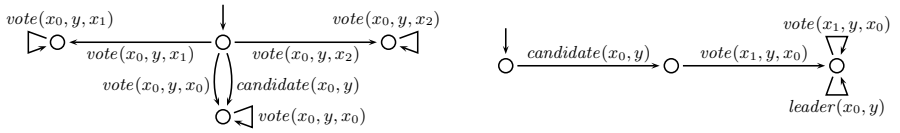


**Fig. 2.** On the left: PLTS $\mathit{Flw3}(x_0, x_1, x_2, y)$ representing the Raft implementation in the follower/candidate mode from the viewpoint of three servers $x_0$, $x_1$, $x_2$ and a term $y$. On the right: PLTS $\mathit{Ldr2}(x_0, x_1, y)$ representing the Raft implementation in the candidate/leader mode from the viewpoint of two servers $x_0, x_1$ and a term $y$.

Second, we model the Raft implementation in the candidate/leader mode from the viewpoint of two servers $x_0, x_1$ and a term $y$ as a PLTS $\mathit{Ldr2}(x_0, x_1, y)$ on the right of Fig. 2. This model says that once the server $x_0$ becomes a candidate and receives a vote from the server $x_1$, it can promote itself to a leader in the term $y$. As we let $y$ to range over all term ids and $x_1$ to range over all server ids in the quorum set of the server $x_0$ for the term $y$, the model of a single server $x_0$ running in the candidate/leader mode is obtained as a PLTS

$$\mathit{Ldr}(x_0) := \mathop{\|}_{y} \mathop{\|}_{x_1} [Q_S(x_0, y, x_0) \wedge Q_S(x_0, y, x_1)] \mathit{Ldr2}(x_0, x_1, y) \ ,$$

which says that in order for a server to become a leader, it needs to become a candidate and then receive a vote from a quorum of servers, including itself.

When we compose the partial models in parallel and let $x_0$ to range over all server ids, we obtain the model of the Raft implementation with an arbitrary many servers and terms as a PLTS $\mathit{Raft} := \|_{x_0}(\mathit{Ldr}(x_0) \parallel \mathit{Flw}(x_0))$. Finally, we hide the events irrelevant to the specification yielding to a PLTS $\mathit{Raft}' := \mathit{Raft} \setminus \{\mathit{vote}, \mathit{candidate}\}$. Now, the problem on the correctness of Raft can be formalised as the question whether $\mathit{Raft}' \preceq_{\mathrm{tr}}^{\mathit{Qrm}} \mathit{Spec}$ holds. Code for the Raft example is found in the online appendix [22]. □

## 5 Dynamic Cut-Off Algorithm

In this section, we show how a parameterised trace refinement checking task can be reduced to finitely many refinement checks among LTSs by determining a cut-off set. As the main result, we convert the problem of determining a cut-off set into the unsatisfiability in FOL and introduce an SMT-based semi-algorithm for computing such a set.

**Definition 9 (Cut-off set).** *Let $\mathcal{P}$ be an implementation PLTS, $\mathcal{Q}$ a specification PLTS, $\mathcal{V}$ a topology formula, and $\Phi \subseteq \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ a finite set of valuations. The set $\Phi$ is a* cut-off set (for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$)*, if $\mathcal{P} \preceq_{\mathrm{tr}}^{\mathcal{V}} \mathcal{Q}$ if and only if $[\![\mathcal{P}]\!]_\phi \preceq [\![\mathcal{Q}]\!]_\phi$ for all $\phi \in \Phi$.*

Our approach can find cut-off sets for implementation-specification pairs where the system topology is downward-closed and the specification does not

involve hiding. Both the restrictions are necessary for decidability [24], but the latter one is less severe since hiding is typically only applied on the implementation side. Hence, from now on, *an implementation PLTS* refers to any PLTS, whereas *a specification PLTS* means a PLTS which does not involve hiding. Results similar to Prop. 15 are proved in [24, 23, 21] but the main result, Thm 23, the supporting lemmata, Lemmata 20 and 22, and the related dynamic cut-off algorithm are completely new.

Intuitively, our verification technique consists of the following steps. First, we show that if a big instance of the implementation PLTS $\mathcal{P}$ (resp., a specification PLTS $\mathcal{Q}$) is composed of the same components as a set of small instances and each small instance of $\mathcal{P}$ is a trace refinement of the corresponding instance of $\mathcal{Q}$, then the big instance of $\mathcal{P}$ is a trace refinement of the big instance of $\mathcal{Q}$, too (Prop. 15). Second, if the system topology is downward-closed, i.e, all the instances are covered by a finite set of small instances, then we can reduce a refinement checking on PLTSs to finitely many refinement checks on LTSs (Prop. 18). Third, we convert the sufficient condition for a cut-off set into the unsatisfiability of a first-order formula (Thm 23). Finally, we give an SMT-based semi-algorithm for computing a cut-off set and successfully apply it to several system models (Algorithm 1).

In order to present the technique in detail, we first formalise the notion of a small instance. After that, in Lemma 14, we show that small instances are generated by small valuations, called subvaluations.

**Definition 10.** *Let $\mathcal{P}$ be a PLTS and $\phi$ a compatible valuation.* The set of the components (of the $\phi$-instance of $\mathcal{P}$)*, denoted by* $\mathrm{comp}(\mathcal{P}, \phi)$*, is defined inductively:*

1. $\mathrm{comp}(L(x_1, \ldots, x_n), \phi) = \{L(\phi(x_1), \ldots, \phi(x_n))\}$,
2. $\mathrm{comp}([G]\mathcal{P}', \phi) = \begin{cases} \mathrm{comp}(\mathcal{P}', \phi), & \text{if } [\![G]\!]_\phi \text{ is true,} \\ \emptyset, & \text{otherwise,} \end{cases}$
3. $\mathrm{comp}(\mathcal{P}_1 \parallel \mathcal{P}_2, \phi) = \bigcup_{i \in \{1,2\}} \big(\{i\} \times \mathrm{comp}(\mathcal{P}_i, \phi)\big)$,
4. $\mathrm{comp}(\|_x \mathcal{P}', \phi) = \bigcup_{\phi' \in \mathrm{ext}(\phi, \{x\})} \big(\{\phi'(x)\} \times \mathrm{comp}(\mathcal{P}', \phi')\big)$, *and*
5. $\mathrm{comp}(\mathcal{P}' \setminus C, \phi) = \mathrm{comp}(\mathcal{P}', \phi)$.

*We say that the $\phi$-instance of $\mathcal{P}$ is* smaller than (or equal to) *the $\psi$-instance of $\mathcal{P}$ if* $\mathrm{comp}(\mathcal{P}, \phi)$ *is a subset of* $\mathrm{comp}(\mathcal{P}, \psi)$.

*Example 11.* Recall our Raft model and the definition of $Q_{x_0,y}$ from Ex. 2. Let $\theta \in \mathrm{va}(\mathit{Qrm} \mid \mathit{Raft}', \mathit{Spec})$ such that $\theta(T_S) = \{s_1, \ldots, s_n\}$, $\theta(T_T) = \{t_1, \ldots, t_m\}$, and $\{x \mid (s_i, t_l, x) \in \theta(Q_S)\}$ is a quorum set with $(s_i, t_l, s_i) \in \theta(Q_S)$ for all $i \in \{1, \ldots, n\}$ and $l \in \{1, \ldots, m\}$. Then

$$\mathrm{comp}(\mathit{Spec}, \theta) = \bigcup_{i=1}^{n} \bigcup_{j=1}^{n} \bigcup_{l=1}^{m} \bigcup_{\substack{k=1 \\ s_k \in Q_{s_i,t_l} \cap Q_{s_j,t_l}}}^{n} \big\{\big(s_i, \big(s_j, \big(s_k, \big(t_l, \mathit{Spec2}(s_i, s_j, t_l)\big)\big)\big)\big)\big\} \,. \quad \square$$

**Definition 12 (Subvaluation).** *Let $\Pi$ and $\Xi$ be sets of predicates. A valuation $\phi$ is a* $(\Pi, \Xi)$-subvaluation *of a valuation $\psi$ if and only if*

1. *the valuations have the same domain,*
2. $\phi(T) \subseteq \psi(T)$ *for all sorts* $T \in \mathrm{dom}_{\mathbb{T}}(\phi)$*,*
3. $\phi(x) = \psi(x)$ *for all variables* $x \in \mathrm{dom}_{\mathbb{X}}(\phi)$*,*
4. $\underline{\phi(F)} \subseteq \underline{\psi(F)}$ *for all predicates* $F \in \mathrm{dom}_{\mathbb{F}}(\phi) \cap \Pi$*, and*
5. $\overline{\phi(F)} \subseteq \overline{\psi(F)}$ *for all predicates* $F \in \mathrm{dom}_{\mathbb{F}}(\phi) \cap \Xi$*.*

The fact that $\phi$ is (not) a $(\Pi, \Xi)$-*subvaluation* of $\psi$ is denoted $\phi \subseteq^{\Pi}_{\Xi} \psi$ (respectively, $\phi \not\subseteq^{\Pi}_{\Xi} \psi$). Given a propositional formula $G$, we write $\mathrm{pr}^{+}(G)$ and $\mathrm{pr}^{-}(G)$ for the set of predicates occurring within, respectively, even and odd number of negations in $G$. The notation is extended to PLTSs $\mathcal{P}$ by defining $\mathrm{pr}^{\oplus}(\mathcal{P})$, where $\oplus \in \{+, -\}$, as the union of all $\mathrm{pr}^{\oplus}(G)$ as $G$ ranges over all guards in $\mathcal{P}$.

*Example 13.* Let $\theta$ be a valuation as in Ex. 11 and $\Theta$ the set of all valuations $\theta' \in \mathrm{va}(\mathit{Qrm} \mid \mathit{Raft}', \mathit{Spec})$ such that $\theta'(T_T) = \{t_l\}$, $\theta'(T_S) = \{s_i, s_j, s_k\}$, and $\theta'(Q_S) \subseteq \theta(Q_S)$ for some $l \in \{1, \ldots, m\}$ and $i, j, k \in \{1, \ldots, n\}$. Since *Spec* involves a single predicate, $Q_S$, without negation, $\mathrm{pr}^{+}(\mathit{Spec}) = \{Q_S\}$ and $\mathrm{pr}^{-}(\mathit{Spec}) = \emptyset$. This implies that $\Theta$ is a finite set of $(\mathrm{pr}^{+}(\mathit{Spec}), \mathrm{pr}^{-}(\mathit{Spec}))$-subvaluations of $\theta$. It is also easy to see that for all $\theta' \in \Theta$, $\mathrm{comp}(\mathit{Spec}, \theta')$ is a subset of $\mathrm{comp}(\mathit{Spec}, \theta)$, i.e., $[\![\mathit{Spec}]\!]_{\theta'}$ is smaller than $[\![\mathit{Spec}]\!]_{\theta}$ . □

**Lemma 14.** *Let* $\mathcal{P}$ *be a PLTS,* $G$ *a propositional formula, and* $\psi, \phi$ *valuations compatible with* $\mathcal{P}$ *and* $G$ *such that* $\phi$ *is a* $(\emptyset, \emptyset)$*-subvaluation of* $\psi$*.*

1. *If* $\mathcal{P}$ *is an elementary PLTS, then* $[\![\mathcal{P}]\!]_{\psi} = [\![\mathcal{P}]\!]_{\phi}$*.*
2. *If* $\phi$ *is a* $(\mathrm{pr}^{+}(G), \mathrm{pr}^{-}(G))$*-subvaluation of* $\psi$ *and* $[\![G]\!]_{\phi}$*, then* $[\![G]\!]_{\psi}$*.*
3. *If* $\phi$ *is a* $(\mathrm{pr}^{+}(\mathcal{P}), \mathrm{pr}^{-}(\mathcal{P}))$*-subvaluation of* $\psi$*, then* $[\![\mathcal{P}]\!]_{\phi}$ *is smaller than* $[\![\mathcal{P}]\!]_{\psi}$*.*

*Proof.* (1) Since $\phi|_{\mathbb{X}} = \psi|_{\mathbb{X}}$ and an instance of $\mathcal{P}$ is completely defined by the values of variables, the claim is evident. (2) Put $G$ into negation normal form $\overline{G}$ and argue by induction on the structure of $\overline{G}$ by using the claim as an induction hypothesis. (3) By induction on the structure of $\mathcal{P}$ by using (1) and (2) and the lemma as an induction hypothesis. □

With the aid of the lemma above, we can show that the correctness of a (big) implementation instance can be derived from the correctness of smaller instances if the big implementation and specification instances are composed of the same components as the smaller, respectively, implementation and specification instances. This is formalised as Prop. 15.

**Proposition 15.** *Let* $\mathcal{P}$ *be an implementation PLTS,* $\mathcal{Q}$ *a specification PLTS,* $\psi$ *a valuation compatible with* $\mathcal{P}$ *and* $\mathcal{Q}$*, and* $\Phi$ *a finite set of* $(\mathrm{pr}^{+}(\mathcal{P} \,\|\, \mathcal{Q}), \mathrm{pr}^{-}(\mathcal{P} \,\|\, \mathcal{Q}))$*-subvaluations of* $\psi$*. If* $\mathrm{comp}(\mathcal{P} \,\|\, \mathcal{Q}, \psi) = \bigcup_{\phi \in \Phi} \mathrm{comp}(\mathcal{P} \,\|\, \mathcal{Q}, \phi)$ *and* $[\![\mathcal{P}]\!]_{\phi} \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_{\phi}$ *for all* $\phi \in \Phi$*, then* $[\![\mathcal{P}]\!]_{\psi} \preceq_{\mathrm{tr}} \|_{\phi \in \Phi} [\![\mathcal{P}]\!]_{\phi} \preceq_{\mathrm{tr}} \|_{\phi \in \Phi} [\![\mathcal{Q}]\!]_{\phi} \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_{\psi}$*.*

*Proof.* First, we argue that if $\mathrm{comp}(\mathcal{P}, \psi) = \bigcup_{\phi \in \Phi} \mathrm{comp}(\mathcal{P}, \phi)$, then $[\![\mathcal{P}]\!]_{\psi} \preceq_{\mathrm{tr}} \|_{\phi \in \Phi} [\![\mathcal{P}]\!]_{\phi}$. The proof proceeds by induction on the structure of $\mathcal{P}$ by using the claim as an induction hypothesis.

The cases when $\mathcal{P}$ is an elementary PLTS, a (replicated) parallel PLTS, or a hiding PLTS are similar to the proofs of Lemmata 23 and 25 in [24]: the base

case (an elementary PLTS) follows from the idempotence of $\|$, the case when $\mathcal{P}$ is a (replicated) parallel PLTS utilises the associativity and commutativity of $\|$ (and, respectively, the identity property), and the case when $\mathcal{P}$ is a hiding PLTS follows from the distributivity of hiding over parallel composition. Hence, we only need to consider the case when $\mathcal{P}$ is $[G]\mathcal{P}'$. If $[\![G]\!]_\psi$ is false, then by the second item of Lemma 14, $[\![G]\!]_\phi$ is false for all $\phi \in \Phi$. This implies that $[\![\mathcal{P}]\!]_\psi \equiv_{\mathrm{tr}} L_{id} \equiv_{\mathrm{tr}} \|_{\phi\in\Phi}[\![\mathcal{P}]\!]_\phi$. Let us then assume that $[\![G]\!]_\psi$ is true and let $\Phi_t$ be the set of all $\phi \in \Phi$ such that $[\![G]\!]_\phi$ is true. Since $\mathrm{comp}(\mathcal{P}, \phi)$ is empty for all $\phi \in \Phi \setminus \Phi_t$, it means that $\mathrm{comp}(\mathcal{P}, \psi) = \bigcup_{\phi\in\Phi_t} \mathrm{comp}(\mathcal{P}, \phi)$. By the induction hypothesis and the identity of $L_{id}$, it implies that $[\![\mathcal{P}]\!]_\psi \equiv_{\mathrm{tr}} [\![\mathcal{P}']\!]_\psi \preceq_{\mathrm{tr}} \|_{\phi\in\Phi_t}[\![\mathcal{P}']\!]_\phi \equiv_{\mathrm{tr}} \|_{\phi\in\Phi}[\![\mathcal{P}]\!]_\phi$.

The proof that $\mathrm{comp}(\mathcal{Q}, \psi) = \bigcup_{\phi\in\Phi} \mathrm{comp}(\mathcal{Q}, \phi)$ implies $\|_{\phi\in\Phi}[\![\mathcal{Q}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\psi$ is similar but simpler because there is no need to consider the case with hiding.

Finally, we argue like in the proof of Proposition 27 in [24]. If $\mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi) = \bigcup_{\phi\in\Phi} \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \phi)$ and $[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\phi$ for all $\phi \in \Phi$, then by above and the precongruence of $\preceq_{\mathrm{tr}}$, $[\![\mathcal{P}]\!]_\psi \preceq_{\mathrm{tr}} \|_{\phi\in\Phi}[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} \|_{\phi\in\Phi}[\![\mathcal{Q}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\psi$. □

*Example 16.* Let $\theta$ and $\Theta$ be as in Ex. 11 and 13. Since every element of $\mathrm{comp}(\mathit{Spec}, \theta)$ depends on the identifiers of at most three servers and one term, it is easy to see that $\mathrm{comp}(\mathit{Spec}, \theta) = \bigcup_{\theta'\in\Theta} \mathrm{comp}(\mathit{Spec}, \theta')$, i.e., the $\theta$-instance of *Spec* is composed of the same components as the set of $\theta'$-instances, where $\theta' \in \Theta$. Similarly, we can see that every element of $\mathrm{comp}(\mathit{Raft}', \theta)$ depends on the identifiers of at most three servers and one term, which implies that $\mathrm{comp}(\mathit{Raft}', \theta) = \bigcup_{\theta'\in\Theta} \mathrm{comp}(\mathit{Raft}', \theta')$. Since $\Theta$ is finite, by Prop. 15, it means that if $[\![\mathit{Raft}']\!]_{\theta'} \preceq_{\mathrm{tr}} [\![\mathit{Spec}]\!]_{\theta'}$ for all $\theta' \in \Theta$, then $[\![\mathit{Raft}']\!]_\theta \preceq_{\mathrm{tr}} [\![\mathit{Spec}]\!]_\theta$, too. □

Valuations that can be obtained from each other by injective renaming result in equivalent verification tasks. A function (injection) $g : A \to B$, where $A, B \subseteq \mathbb{A}$, is a *sortwise function* (respectively, *injection*) if $g(a) \in \mathbb{A}_{T_a}$ for each $a \in A$. Let $\phi$ be a valuation and $g$ a sortwise function. We write $g(\phi)$ for a valuation $\phi'$ which is obtained from $\phi$ by mapping the atoms in the image using $g$. Valuations $\phi_1$ and $\phi_2$ are *(non-)isomorphic*, if there is (respectively, not) a sortwise injection $g : \mathrm{Im}(\phi_1) \to \mathrm{Im}(\phi_2)$ such that $\phi_2 = g(\phi_1)$. It is easy to see that for isomorphic valuations $\phi_1$ and $\phi_2$, $[\![\mathcal{P}]\!]_{\phi_1} \preceq [\![\mathcal{Q}]\!]_{\phi_1}$ if and only if $[\![\mathcal{P}]\!]_{\phi_2} \preceq [\![\mathcal{Q}]\!]_{\phi_2}$ [24].

Prop. 15 and the notion above imply that if the system topology is downward-closed in the sense that all the instances are covered by a finite set of (injectively renamed) small instances, then we can reduce a refinement checking on PLTSs to finitely many refinement checks on LTSs. This is stated formally as Prop. 18.

**Definition 17 (Downward-closed topology).** *A topology $\mathcal{V}$ of an implementation-specification pair $(\mathcal{P}, \mathcal{Q})$ is* downward-closed*, if there is a finite set $\Phi \subseteq \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ of valuations for which the following holds: For all $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ and all $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$, there is a valuation $\phi \in \Phi$ and a sortwise injection $g$ from $\mathrm{Im}(\phi)$ such that $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, g(\phi))$ and $g(\phi)$ is a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))$-subvaluation of $\psi$. The set $\Phi$ is called a* witness *for downward-closedness.*

**Proposition 18.** *Let $\mathcal{V}$ be a downward-closed topology for $(\mathcal{P}, \mathcal{Q})$ and $\Phi$ a witness for downward-closedness. Then $\Phi$ is a cut-off set for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$.*

*Proof.* In order to prove that $\Phi$ is a cut-off set for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$, it is sufficient to show that if $[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\phi$ for all $\phi \in \Phi$, then $[\![\mathcal{P}]\!]_\psi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\psi$ for all $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$, because the opposite implication is trivial. Hence, let us assume that $[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\phi$ for all $\phi \in \Phi$ and let $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$. Since $\Phi$ is a witness for downward-closedness, for every $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$, there is a valuation $\phi_P \in \Phi$ and a sortwise injection $g_P$ from $\mathrm{Im}(\phi_P)$ such that $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, g_P(\phi_P))$ and $g_P(\phi_P)$ is a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))$-subvaluation of $\psi$. By the third item of Lemma 14, $[\![\mathcal{P} \parallel \mathcal{Q}]\!]_{g_P(\phi_P)}$ is smaller than $[\![\mathcal{P} \parallel \mathcal{Q}]\!]_\psi$ for all $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$, which implies that $\mathrm{comp}(\mathcal{P} \| \mathcal{Q}, \psi) = \bigcup_{P \in \mathrm{comp}(\mathcal{P} \| \mathcal{Q}, \psi)} \mathrm{comp}(\mathcal{P} \| \mathcal{Q}, g_P(\phi_P))$. Since $[\![\mathcal{P}]\!]_\phi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\phi$ for any $\phi \in \Phi$, we know that $[\![\mathcal{P}]\!]_{g(\phi)} \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_{g(\phi)}$ for all sortwise injections $g$ such that $\mathrm{dom}(g) = \mathrm{Im}(\phi)$, too. Hence, $[\![\mathcal{P}]\!]_{g_P(\phi_P)} \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_{g_P(\phi_P)}$ for all $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$. By Prop. 15, it implies that $[\![\mathcal{P}]\!]_\psi \preceq_{\mathrm{tr}} [\![\mathcal{Q}]\!]_\psi$. Therefore, $\Phi$ is a cut-off set. □

Prop. 18 gives a sufficient condition for a cut-off set, but it does not clearly say which valuations should be included in the set. That is why we will transform the condition of downward-closedness into a first-order formula, the satisfaction of which can be, in decidable cases, analysed by using existing tools. For that purpose, we introduce Lemmata 20 and 22, which tell how the tests for $P \in \mathrm{comp}(\mathcal{P}, \phi)$ and $g(\phi) \subseteq^{\Pi}_{\Xi} \psi$ occurring in the condition are converted in FOL. In order to present the results, we assume that $\mathbb{X}$ is partitioned into sets $\mathbb{X}^0$, $\mathbb{X}'$, and $\mathbb{X}''$, each containing infinitely many variables for each sort. Moreover, we assume that only the variables in $\mathbb{X}^0$ are used in PLTSs and topology formulae and for each atom $a \in \mathbb{A}$ there is a unique variable $x''_a \in \mathbb{X}''$ such that $T_a = T_{x''_a}$.

**Definition 19 (Branch).** *The set of the* branches *of a PLTS $\mathcal{P}$, denoted* $\mathrm{br}(\mathcal{P})$*, is defined inductively as follows:*

1. $\mathrm{br}(L(x_1, \ldots, x_n)) = \{\top\}$,
2. $\mathrm{br}([G]\mathcal{P}') = \{G \wedge \mathcal{B} \mid \mathcal{B} \in \mathrm{br}(\mathcal{P}')\}$,
3. $\mathrm{br}(\mathcal{P}_1 \parallel \mathcal{P}_2) = \mathrm{br}(\mathcal{P}_1) \cup \mathrm{br}(\mathcal{P}_2)$,
4. $\mathrm{br}(\|_x \mathcal{P}') = \{\exists x.(x' = x \wedge \mathcal{B}) \mid \mathcal{B} \in \mathrm{br}(\mathcal{P}')\}$, *where $x' \in \mathbb{X}'$ is a variable of the sort $T_x$ not occurring in* $\mathrm{br}(\mathcal{P}')$*, and*
5. $\mathrm{br}(\mathcal{P}' \setminus C) = \mathrm{br}(\mathcal{P}')$.

**Lemma 20.** *Let $\mathcal{P}$ be a PLTS and $\phi$ and $\psi$ compatible valuations. There is $P \in \mathrm{comp}(\mathcal{P}, \phi) \cap \mathrm{comp}(\mathcal{P}, \psi)$ if and only if there are $\mathcal{B} \in \mathrm{br}(\mathcal{P})$, $\phi' \in \mathrm{ext}(\phi, \mathrm{par}_{\mathbb{X}'}(\mathcal{B}))$ and $\psi' \in \mathrm{ext}(\psi, \mathrm{par}_{\mathbb{X}'}(\mathcal{B}))$ such that $[\![\mathcal{B}]\!]_{\phi'}$ and $[\![\mathcal{B}]\!]_{\psi'}$ are true and $\psi'|_{\mathbb{X}} = \phi'|_{\mathbb{X}}$.*

*Proof.* By induction on the structure of $\mathcal{P}$ by using the lemma as an induction hypothesis. □

*Example 21.* Recall the Raft specification *Spec* and the valuation $\theta$ from Ex. 11. Since only the parameterised version of the parallel composition is used it *Spec*, there is a single branch

$$\exists x_0.(x'_0 = x_0 \wedge \exists x_1.(x'_1 = x_1 \wedge \exists x_2.(x'_2 = x_2 \wedge \exists y.(y' = y \wedge Q_S(x_0, y, x_2) \wedge Q_S(x_1, y, x_2) \wedge \top)))) .$$

For every $(s_i, (s_j, (s_k, (t_l, \mathit{Spec2}(s_i, s_j, t_l)))))) \in \mathrm{comp}(\mathit{Spec}, \theta)$, there is an extension $\theta'$ of $\theta$ to $\{x'_0, x'_1, x'_2, y'\}$ such that $\theta'(x'_0) = s_i$, $\theta'(x'_1) = s_j$, $\theta'(x'_2) = s_k$, and $\theta'(y') = t_l$. It is also easy to see that $\theta'$ satisfies the branch above. □

**Lemma 22.** *Let $\phi$ and $\psi$ be valuations with the same domain, $\Pi, \Xi \subseteq \mathrm{dom}_{\mathbb{F}}(\phi)$ sets of predicates, $g : \mathrm{Im}(\phi) \to \mathrm{Im}(\psi)$ a sortwise function, and $\psi_g$ a valuation in $\mathrm{ext}(\psi, \{x''_a \mid a \in \mathrm{Im}(\phi)\})$ such that $\psi_g(x''_a) = g(a)$ for all $a \in \mathrm{Im}(\phi)$. Then $g$ is an injection and $g(\phi) \subseteq^{\Pi}_{\Xi} \psi$, if $[\![\mathit{NoSval}(\phi, \Pi, \Xi)]\!]_{\psi_g}$ is false, where*

$$\mathit{NoSval}(\phi, \Pi, \Xi) := \Big( \bigvee_{\substack{\{a,b\} \subseteq \mathrm{Im}(\phi) \\ a \neq b}} x''_a = x''_b \Big) \vee \Big( \bigvee_{x \in \mathrm{dom}_{\mathbb{X}}(\phi)} x''_{\phi(x)} \neq x \Big) \vee$$
$$\Big( \bigvee_{F \in \Pi} \bigvee_{(a_1,\ldots,a_n) \in \phi(F)} \neg F(x''_{a_1}, \ldots, x''_{a_n}) \Big) \vee \Big( \bigvee_{F \in \Xi} \bigvee_{(a_1,\ldots,a_n) \in \overline{\phi(F)}} F(x''_{a_1}, \ldots, x''_{a_n}) \Big) .$$

*Proof.* Let us assume that $[\![\mathit{NoSval}(\phi, \Pi, \Xi)]\!]_{\psi_g}$ is false. Because the first big disjunction is false, it implies that the variables $x''_a$, where $a \in \mathrm{Im}(\phi)$, representing the image of $g$ are mapped to different values. Hence, $g$ is an injection.

In order to prove that $g(\phi) \subseteq^{\Pi}_{\Xi} \psi$, we will show that the conditions (1)–(5) of Def. 12 are met. (1) By the assumption, $\mathrm{dom}(g(\phi)) = \mathrm{dom}(\phi) = \mathrm{dom}(\psi)$. (2) Because $g$ is a sortwise function: $\mathrm{Im}(\phi) \to \mathrm{Im}(\psi)$, $(g(\phi))(T) \subseteq \psi(T)$ for all sorts $T \in \mathrm{dom}_{\mathbb{T}}(\phi)$. (3) Since the second big disjunction is false, $(g(\phi))(x) = \psi_g(x''_{\phi(x)}) = \psi_g(x) = \psi(x)$ for all $x \in \mathrm{dom}(\phi)$, (4) If $F \in \Pi$ and $(a_1, \ldots, a_n) \in (g(\phi))(F)$, then $(g^{-1}(a_1), \ldots, g^{-1}(a_n)) \in \phi(F)$. Because the third big disjunction is false, it implies that $(\psi_g(x''_{g^{-1}(a_1)}), \ldots, \psi_g(x''_{g^{-1}(a_n)})) \in \psi_g(F)$. Since $(\psi_g(x''_{g^{-1}(a_1)}), \ldots, \psi_g(x''_{g^{-1}(a_n)})) = (g(g^{-1}(a_1)), \ldots, g(g^{-1}(a_n))) = (a_1, \ldots, a_n)$ and $\psi_g(F) = \psi(F)$, it means that $(a_1, \ldots, a_n) \in \psi(F)$. (5) Similar to (4). □

By combining Prop. 18 and Lemmata 20 and 22, a sufficient condition for a cut-off set can be converted into the unsatisfiability of a first-order formula.

**Theorem 23 (Cut-off theorem).** *Let $\mathcal{P}$ be an implementation PLTS, $\mathcal{Q}$ a specification PLTS, $\mathcal{V}$ a topology formula, and $\Phi \subseteq \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ a finite set of valuations. The set $\Phi$ is a cut-off set for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$, if the first-order formula*

$$\mathcal{V} \wedge \mathcal{B} \wedge \bigwedge_{\phi \in \Phi_{\mathcal{B}}} \big( \forall x''_{a_1} . \cdots . \forall x''_{a_n} . \mathit{NoSval}(\phi, \mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q})) \big) \qquad (1)$$

*is unsatisfiable for all $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$, where $\Phi_{\mathcal{B}} = \{\phi' \in \mathrm{ext}(\phi, \mathrm{par}_{\mathbb{X}'}(\mathcal{B})) \mid \phi \in \Phi, [\![\mathcal{B}]\!]_{\phi'}\}$ is the set of the extensions of the valuations in $\Phi$ to $\mathrm{par}_{\mathbb{X}'}(\mathcal{B})$ satisfying $\mathcal{B}$ and for every $\phi \in \Phi_{\mathcal{B}}$, $a_1, \ldots, a_n$ are the atoms in $\mathrm{Im}(\phi)$.*

*Proof.* We will show that if $\Phi$ is not a witness for downward-closedness, then Formula 1 is satisfiable for some branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$, which by Prop. 18 implies the theorem.

By Def. 17, $\Phi$ is not a witness for downward-closedness if the following condition holds: There is a valuation $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ and $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$

such that for every valuation $\phi \in \Phi$ and a sortwise injection $g$ from $\mathrm{Im}(\phi)$, if $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, g(\phi))$ then $g(\phi)$ is not a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))$-subvaluation of $\psi$.

By Lemma 20, the condition can be converted into the form: There is a valuation $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$, a branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$, and $\psi' \in \mathrm{ext}(\psi, \mathrm{par}_{\mathbb{X}'}(\mathcal{B}))$ such that $[\![\mathcal{B}]\!]_{\psi'}$ and for every valuation $\phi \in \Phi$ and a sortwise injection $g$ from $\mathrm{Im}(\phi)$, if $[\![\mathcal{B}]\!]_{\phi'}$ for some $\phi' \in \mathrm{ext}(g(\phi), \mathrm{par}_{\mathbb{X}'}(\mathcal{B}))$ with $\psi'|_{\mathbb{X}} = \phi'|_{\mathbb{X}}$ then $g(\phi)$ is not a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))$-subvaluation of $\psi$.

After simplification, the condition can be put as follows: There is a branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$ and a valuation $\psi' \in \mathrm{va}(\mathcal{V} \wedge \mathcal{B} \mid \mathcal{P}, \mathcal{Q})$ such that for every valuation $\phi' \in \Phi_{\mathcal{B}}$ and for every sortwise function $g : \mathrm{Im}(\phi') \to \mathrm{Im}(\psi')$, $g$ is not an injection or $g(\phi')$ is not a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))$-subvaluation of $\psi'$.

Next, we apply Lemma 22 and convert the condition into the form: There is a branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$ and a valuation $\psi' \in \mathrm{va}(\mathcal{V} \wedge \mathcal{B} \mid \mathcal{P}, \mathcal{Q})$ such that for all valuations $\phi' \in \Phi_{\mathcal{B}}$ and for all sortwise functions $g : \mathrm{Im}(\phi') \to \mathrm{Im}(\psi')$, $[\![\mathit{NoSval}(\phi', \mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))]\!]_{\psi'_g}$ is true.

Since $\Phi$ is finite and each valuation only has finitely many extensions to $\mathrm{par}_{\mathbb{X}'}(\mathcal{B})$, universal quantification over the valuations in $\Phi_{\mathcal{B}}$ can be substituted by a finite conjunction. Universal quantification over sortwise functions is, in general, a second-order construct. However, since the image of each function $g : \mathrm{Im}(\phi') \to \mathrm{Im}(\psi')$ is finite and represented by the variables in $\{x''_a \mid a \in \mathrm{Im}(\phi')\}$, the universal quantification over sortwise functions can be replaced by the universal quantification over these variables. Hence, the condition gets the form: There is a branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$ and a valuation $\psi' \in \mathrm{va}(\mathcal{V} \wedge \mathcal{B} \mid \mathcal{P}, \mathcal{Q})$ such that $[\![\bigwedge_{\phi' \in \Phi_{\mathcal{B}}} \left(\forall x''_{a_1}. \cdots . \forall x''_{a_n}. \mathit{NoSval}(\phi', \mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \parallel \mathcal{Q}))\right)]\!]_{\psi'}$ is true.

Finally, the existence of $\psi' \in \mathrm{va}(\mathcal{V} \wedge \mathcal{B} \mid \mathcal{P}, \mathcal{Q})$ can be simply encoded as the satisfaction of the formula $\mathcal{V} \wedge \mathcal{B}$. This means that $\Phi$ is not a witness for downward-closedness, if there is a branch $\mathcal{B} \in \mathrm{br}(\mathcal{P} \parallel \mathcal{Q})$ such that Formula 1 is satisfiable, which by Prop. 18 implies the theorem. Obviously, the formula is also in FOL since it only involves first-order constructs. □

The iterative application of Thm 23 results in Algorithm 1 that allows us to determine a cut-off set for systems with a downward-closed topology by starting from the empty set and appending the set with non-isomorphic valuations of increasing size until the Formula 1 becomes unsatisfiable for all branches. The removal of isomorphs is implemented by converting the valuations into coloured graphs as described in [19] and by using the nauty library [16] to compute a canonical form for each graph. Finally, the valuations with redundant canonical graph representations are removed.

There are two points in the algorithm that are critical for termination. The one is the condition of the while loop, Formula 1. The formula consists of several conjuncts: the topology $\mathcal{V}$, a branch $\mathcal{B}$ involving only conjunctions and existential quantification, and universally quantified conjuncts. Since quantification (over non-empty sets) can be pushed outside the conjunctions, the condition is of the form $\mathcal{V} \wedge \mathcal{U}$, where $\mathcal{U}$ is in the $\exists^*\forall^*$ fragment. Therefore, the whole condition is within the decidable $\exists^*\forall^*$ fragment if the topology $\mathcal{V}$ is.

The other critical point is querying a sort from an oracle and incrementing the cut-off, because the cut-offs heavily affect the length of Formula 1. In the worst case, when predicates are involved, the formula grows exponentially in the size of the cut-offs, and even when predicates are not used, the length of the formula is quadratic in the size of the cut-offs. Of course, a non-deterministic oracle can guess the optimal order, but also in practice, the algorithm will always compute some (not necessarily an optimal) cut-off set as long as incrementing is done fairly, i.e., the incrementation of a sort cannot be omitted infinitely many times consecutively. Hence, the algorithm will always terminate for topologies expressible in the $\exists^*\forall^*$ fragment of FOL, because as stated in Lemma 24, the fragment is decidable and such topologies are downward-closed. The algorithm also enables us to consider systems with downward-closed topologies beyond this fragment, but termination depends on the capabilities of the used SMT solver. If the solver is unable to decide the satisfiability of the equation and consequently, returns the unknown value, it is always safe to consider the equation satisfiable. This may lead to an infinite execution loop but guarantees the correctness of the algorithm.

**input :** implementation PLTS $\mathcal{P}$, specification PLTS $\mathcal{Q}$, topology formula $\mathcal{V}$
**output:** cut-off set $\Phi$ for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$

**foreach** $T \in \mathrm{par}_{\mathbb{T}}(\mathcal{P} \parallel \mathcal{Q}) \cup \mathrm{par}_{\mathbb{T}}(\mathcal{V})$ **do** $\mathit{cutoff}_T \leftarrow 1$;
$\Phi \leftarrow$ the set of all non-isomorphic valuations $\phi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ such that $|\phi(T)| = 1$ for all $T \in \mathrm{dom}_{\mathbb{T}}(\phi)$;
**foreach** $\mathcal{B} \in \mathrm{br}(\mathcal{P} \| \mathcal{Q})$ **do**
  **while** *Formula 1 of Thm 23 is satisfiable* **do**
    get a sort $T_o \in \mathrm{par}_{\mathbb{T}}(\mathcal{P} \parallel \mathcal{Q}) \cup \mathrm{par}_{\mathbb{T}}(\mathcal{V})$ from an oracle and increment $\mathit{cutoff}_{T_o}$;
    append the set $\Phi$ with all non-isomorphic valuations $\phi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$ such that $|\phi(T_o)| = \mathit{cutoff}_{T_o}$ and $|\phi(T)| \leq \mathit{cutoff}_T$ for all $T \in \mathrm{dom}_{\mathbb{T}}(\phi) \setminus \{T_o\}$;
  **end**
**end**

**Algorithm 1:** Dynamic cut-off algorithm

**Lemma 24.** *Let $\mathcal{V}$ be a formula in the $\exists^*\forall^*$ fragment of FOL. Then the satisfiability of $\mathcal{V}$ is decidable and the topology $\mathcal{V}$ of any implementation-specification pair is downward-closed.*

*Proof.* The decidability of the $\exists^*\forall^*$ fragment is well-known [8, 1].

Let us then consider a topology $\mathcal{V} := \exists x_1.\cdots.\exists x_n.\mathcal{U}$, where $\mathcal{U}$ is in the $\forall^*$ fragment. Without the loss of generality, we may assume that the variables $x_1, \ldots, x_n$ do not occur in $\mathcal{P}$ and $\mathcal{Q}$. By Thm 50 in [24], we know that there is a witness $\Phi'$ for downward-closedness for the topology $\mathcal{U}$ of $(\mathcal{P}, \mathcal{Q})$.

Next, we will show that $\Phi := \{\phi'|_{\mathrm{dom}(\phi') \setminus \{x_1,\ldots,x_n\}} \mid \phi' \in \Phi'\}$ is a witness for downward-closedness for the topology $\mathcal{V}$ of $(\mathcal{P}, \mathcal{Q})$. Let $\psi \in \mathrm{va}(\mathcal{V} \mid \mathcal{P}, \mathcal{Q})$

and $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi)$. Then there is $\psi' \in \mathrm{ext}(\psi, \{x_1, \ldots, x_n\})$ such that $\psi' \in \mathrm{va}(\mathcal{U} \mid \mathcal{P}, \mathcal{Q})$ and $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, \psi')$. Since $\Phi'$ is a witness for downward-closedness for $\mathcal{U}$, there is a valuation $\phi' \in \Phi'$ and a sortwise injection $g$ such that $P \in \mathrm{comp}(\mathcal{P} \| \mathcal{Q}, g(\phi'))$ and $g(\phi')$ is a $(\mathrm{pr}^+(\mathcal{P} \| \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \| \mathcal{Q}))$-subvaluation of $\psi'$. Because $x_1, \ldots, x_n$ do not occur in $\mathcal{P} \| \mathcal{Q}$, it implies that $\phi := \phi'|_{\mathrm{dom}(\phi') \setminus \{x_1, \ldots, x_n\}}$ is a valuation in $\Phi$ such that $P \in \mathrm{comp}(\mathcal{P} \parallel \mathcal{Q}, g(\phi))$ and $g(\phi)$ is a $(\mathrm{pr}^+(\mathcal{P} \parallel \mathcal{Q}), \mathrm{pr}^-(\mathcal{P} \| \mathcal{Q}))$-subvaluation of $\psi$. Hence, $\Phi$ is a witness for downward-closedness for the topology $\mathcal{V}$. □

**Corollary 25 (Soundness and completeness).** *Let $\mathcal{P}$ be an implementation PLTS, $\mathcal{Q}$ a specification PLTS, and $\mathcal{V}$ a topology formula.*

1. *If Algorithm 1 terminates with $\Phi$, then $\Phi$ is a cut-off set for $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{V}$.*
2. *If $\mathcal{V}$ is downward-closed and there is a decision procedure for formulas of the form $\mathcal{V} \wedge \mathcal{U}$ (where $\mathcal{U}$ is in the $\exists^*\forall^*$ fragment), then Algorithm 1 terminates.*
3. *If $\mathcal{V}$ is in the $\exists^*\forall^*$ fragment, then Algorithm 1 terminates.*

*Proof.* (1) Follows from Thm 23. (2) Follows from the facts that downward-closed topologies have a witness and the algorithm will eventually compute some witness. (3) Follows from Lemma 24 and (2). □

We have implemented a prototype version of Algorithm 1 in Bounds tool [20]. The tool uses the Z3 SMT solver [3] for testing satisfiability and the oracle is implemented as a fair heuristic. The heuristic uses the satisfying valuation provided by Z3 and favours the incrementation of sorts which the valuation maps to a set larger than the current cut-off. The tool takes a parameterised verification task as an input, computes a cut-off set by using the dynamic cut-off algorithm, and provided the algorithm terminates, produces an LTS refinement checking task for each valuation in the cut-off set. After that, the verification is completed by refinement checking the instances by using FDR [7]. Once we have proved that a system implementation refines its specification, we can use the specification, which is usually much smaller, in place of the system implementation in further verification efforts. This is possible since our PLTS formalism is compositional.

*Example 26.* We have applied Bounds to several system models by using both the dynamic and static cut-off algorithms (Table 1). The Raft model for the static algorithm uses specific *quorum function variables* [21] which in our formalism are modelled in FOL. The tree topology of taDOM2+ and the lower bound for the number of transactions are naturally modelled by using existential quantification but for the static algorithm, which does not support it, they are modelled by using free variables. Otherwise the models are identical. In each case, the dynamic algorithm performs at least as well as the static one in terms of running time and the size of a cut-off set. In the case of Raft and tree-based taDOM2+, the implementation based on static cut-offs ran out of memory, whereas the dynamic one terminated with very small cut-offs, which we claim to be optimal. Hence, the dynamic algorithm not only enables extending the application domain of static ones but also provides more compact cut-offs. All experiments were made on an 8-thread Intel i7-4790 with 16GB of memory running Ubuntu 16.04 LTS. An example run of Bounds on the Raft model is in the online appendix [22]. □

**Table 1.** The performance of the dynamic cut-off algorithm with respect to the static ones, $|T|$ is the cut-off size for a sort $T$, $|\Phi|$ is the size of the cut-off set, and $t$ is the time taken (in seconds) by the computation of the cut-off set $\Phi$ plus refinement checking the $\phi$-instances for all $\phi \in \Phi$.

| **System** | **Parameters** | **Dynamic** | | **Static** | |
|---|---|---|---|---|---|
| | | **cut-offs** | **t(s)** | **cut-offs** | **t(s)** |
| Raft [21] | servers ($S$), terms ($T$), quorum topology | $\mathbf{\lvert S\rvert = 3}$, $\mathbf{\lvert T\rvert = 1}$, $\mathbf{\lvert\Phi\rvert = 20}$ | **7+2** | $\lvert S\rvert = 7, \lvert T\rvert = 1$, $\lvert\Phi\rvert > 10^5$ | dnf |
| Shared resources [24] | users ($U$), resources ($R$), forest topology | $\lvert U\rvert = 2, \lvert R\rvert = 3$, $\lvert\Phi\rvert = 6$ | 1+6 | $\lvert U\rvert = 2, \lvert R\rvert = 3$, $\lvert\Phi\rvert = 6$ | 1+6 |
| Shared resources [24] | users ($U$), resources ($R$), ring topology | $\lvert U\rvert = 4, \lvert R\rvert = 1$, $\lvert\Phi\rvert = 4$ | 1+1 | $\lvert U\rvert = 4, \lvert R\rvert = 1$, $\lvert\Phi\rvert = 4$ | 1+1 |
| taDOM2+ [24] | transactions ($T$), nodes ($N$), forest topology | $\lvert T\rvert = 2$, $\lvert N\rvert = 3$, $\lvert\Phi\rvert = 14$ | 1+ 1122 | $\lvert T\rvert = 2, \lvert N\rvert = 3$, $\lvert\Phi\rvert = 14$ | 1+ 1130 |
| taDOM2+ [24] | 2+ transactions ($T$), nodes ($N$), tree topology | $\mathbf{\lvert T\rvert = 2}, \mathbf{\lvert N\rvert = 3}$, $\mathbf{\lvert\Phi\rvert = 7}$ | **1+ 1121** | $\lvert T\rvert = 4, \lvert N\rvert = 4$, $\lvert\Phi\rvert = 45$ | 11+ dnf |
| Token ring [20] | users ($U$), ring topology | $\lvert U\rvert = 4$, $\lvert\Phi\rvert = 3$ | 1+1 | $\lvert U\rvert = 4$, $\lvert\Phi\rvert = 3$ | 1+1 |
| Ring with 2 tokens [24] | users ($U$), ring topology | $\lvert U\rvert = 5$, $\lvert\Phi\rvert = 30$ | **10+ 7** | $\lvert U\rvert = 5$, $\lvert\Phi\rvert = 30$ | 25+ 7 |

## 6 Conclusions and Future Work

We have provided a semi-algorithm for reducing a refinement checking task in the parameterised LTS formalism to a finite set of refinement checks between LTSs. The algorithm is implemented in a tool and applied to several system models. The novelty of the algorithm is in its generality; it not only combines existing static cut-off techniques but also extends their application domain beyond known decidable fragments. In future, we aim to extend the algorithm to other process algebraic formalisms such as modal interface automata [23].

## References

1. Abadi, A., Rabinovich, A., Sagiv, M.: Decidable fragments of many-sorted logic. J. Symb. Comput. **45**(2), 153–172 (2010)
2. Creese, S.J.: Data Independent Induction: CSP Model Checking of Arbitrary Sized Networks. Ph.D. thesis, Oxford University (2001)
3. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS '08. LNCS, vol. 4963, pp. 337–340. Springer (2008)
4. Emerson, E.A., Kahlon, V.: Reducing model checking of the many to the few. In: McAllester, D.A. (ed.) CADE-17, LNCS, vol. 1831, pp. 236–254. Springer (2000)
5. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. **256**(1), 63–92 (2001)

6. Gallier, J.H.: Logic for Computer Science: Foundations of Automatic Theorem Proving. Courier Dover Publications (2015)
7. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.W.: FDR3: A parallel refinement checker for CSP. STTT **18**(2), 149–167 (2016)
8. Gurevich, Y.: On the classical decision problem. In: Rozenberg, G., Salomaa, A. (eds.) Current Trends In Theoretical Computer Science: Essays and Tutorials, World Scientific Series in Computer Science, vol. 40, pp. 254–265. World Scientific (1993)
9. Hanna, Y., Samuelson, D., Basu, S., Rajan, H.: Automating cut-off for multi-parameterized systems. In: Dong, J.S., Zhu, H. (eds.) ICFEM '10, LNCS, vol. 6447, pp. 338–354. Springer (2010)
10. Haustein, M., Härder, T.: Optimizing lock protocols for native XML processing. Data Knowl. Eng. **65**(1), 147–173 (2008)
11. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1985)
12. Kaiser, A., Kroening, D., Wahl, T.: Dynamic cutoff detection in parameterized concurrent programs. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV '10, LNCS, vol. 6174, pp. 645–659. Springer (2010)
13. Lazić, R.S.: A Semantic Study of Data Independence with Applications to Model Checking. Ph.D. thesis, Oxford University (1999)
14. Lazić, R.S., Nowak, D.: A unifying approach to data-independence. In: Palamidessi, C. (ed.) CONCUR '00, LNCS, vol. 1877, pp. 581–595. Springer (2000)
15. Marić, O., Sprenger, C., Basin, D.: Cutoff bounds for consensus algorithms. In: Majumdar, R., Kunčak, V. (eds.) CAV '17, LNCS, vol. 10427, pp. 217–237. Springer (2017)
16. McKay, B.D., Piperno, A.: Practical graph isomorphism II. J. Symb. Comput. **60**, 94 – 112 (2014)
17. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: Gibson, G., Zeldovich, N. (eds.) USENIX ATC '14. pp. 305–320. USENIX Association (2014)
18. Roscoe, A.W.: Understanding Concurrent Systems. Springer (2010)
19. Siirtola, A.: Algorithmic Multiparameterised Verification of Safety Properties. Process Algebraic Approach. Ph.D. thesis, University of Oulu (2010)
20. Siirtola, A.: Bounds2: A tool for compositional multi-parametrised verification. In: Ábrahám, E., Havelund, K. (eds.) TACAS '14, LNCS, vol. 8413, pp. 599–604. Springer (2014)
21. Siirtola, A.: Refinement checking parameterised quorum systems. In: Legay, A., Schneider, K. (eds.) ACSD '17. pp. 39–48. IEEE (2017)
22. Siirtola, A., Heljanko, K.: Online appendix, `http://cc.oulu.fi/~asiirtol/papers/dyncutoffapp.pdf`
23. Siirtola, A., Heljanko, K.: Parametrised modal interface automata. ACM Trans. Embed. Comput. Syst. **14**(4), 65:1–65:25 (2015)
24. Siirtola, A., Kortelainen, J.: Multi-parameterised compositional verification of safety properties. Inform. Comput. **244**, 23–48 (2015)
25. Valmari, A., Tienari, M.: An improved failures equivalence for finite-state systems with a reduction algorithm. In: Jonsson, B., Parrow, J., Pehrson, B. (eds.) PSTV '91. pp. 3–18. North-Holland (1991)
26. Yang, Q., Li, M.: A cut-off approach for bounded verification of parameterized systems. In: Kramer, J., Bishop, J., Devanbu, P.T., Uchitel, S. (eds.) ICSE '10. pp. 345–354. ACM (2010)
27. Zuck, L., Pnueli, A.: Model checking and abstraction to the aid of parameterized systems (a survey). Comput. Lang. Syst. Str. **30**(3), 139–169 (2004)