Interpolation Properties and SAT-based Model Checking

Arie Gurfinkel[1], Simone Fulvio Rollini[2], and Natasha Sharygina[2]

[1] Software Engineering Institute, CMU
arie@cmu.edu
[2] Formal Verification Lab, University of Lugano
{simone.fulvio.rollini, natasha.sharygina}@usi.ch

Abstract Craig interpolation is a widespread method in verification, with important applications such as Predicate Abstraction, CounterExample Guided Abstraction Refinement and Lazy Abstraction With Interpolants. Most state-of-the-art model checking techniques based on interpolation require *collections* of interpolants to satisfy particular properties, to which we refer as "collectives"; they do not hold in general for all interpolation systems and have to be established for each particular system and verification environment. Nevertheless, no systematic approach exists that correlates the individual interpolation systems and compares the necessary collectives. This paper proposes a uniform framework, which encompasses (and generalizes) the most common collectives exploited in verification. We use it for a systematic study of the collectives and of the constraints they pose on propositional interpolation systems used in SAT-based model checking. Our results have practical applications to verification and provide a better theoretical overview of collectives in interpolation-based model checking.

1 Introduction

Craig interpolation is a popular approach in verification [13,12] with notable applications such as Predicate Abstraction [9], CounterExample Guided Abstraction Refinement (CEGAR) [6], and Lazy Abstraction With Interpolants (LAWI) [14].

Formally, given two formulae A and B such that $A \wedge B$ is unsatisfiable, a *Craig interpolant* is a formula I such that A implies I, I is inconsistent with B and I is defined over the atoms (i.e., propositional variables) common to A and B. It can be seen as an over-approximation of A that is still inconsistent with B[3]. In model checking applications, A typically encodes some finite program traces, and B denotes error locations. In this case, an interpolant I represents a set of *safe* states that over-approximate the states reachable in A.

In most verification tasks, a single interpolant, i.e., a single subdivision of constraints into two groups A and B, is not sufficient. For example, consider

[3] We write $Itp(A \mid B)$ for an interpolant of A and B, and I_A when B is clear from the context.

the refinement problem in CEGAR: given a spurious error trace $\pi = \tau_1, \ldots, \tau_n$, where τ_i is a program statement, find a set of formulae $X_0, \ldots, X_n$ such that $X_0 = \top$, $X_n = \bot$, and for $1 \leq i \leq n$, the Hoare triples $\{X_{i-1}\}\ \tau_i\ \{X_i\}$ are valid. The sequence $\{X_i\}$ justifies that the error trace is infeasible and is used to refine the abstraction. The solution is a *sequence* of interpolants $\{I_i\}_{i=1}^n$ such that: $I_i = Itp(\tau_1 \ldots \tau_i \mid \tau_{i+1} \ldots \tau_n)$ and $I_{i-1} \wedge \tau_i \implies I_i$. That is, in addition to requiring that each I_i is an interpolant between the prefix (statements up to position i in the trace) and the suffix (statements following position i), the sequence $\{I_i\}$ of interpolants must be inductive: this property is known as the *path interpolation property* [17].

Other properties (e.g., simultaneous abstraction, interpolation sequence, path-, symmetric-, and tree-interpolation) are used in existing verification frameworks such as IMPACT [14], Whale [1], FunFrog [19] and eVolCheck [20], which implement instances of Predicate Abstraction [8], Lazy Abstraction with Interpolation [14], Interpolation-based Function Summarization [19] and Upgrade Checking [20]. These properties, to which we refer as *collectives* since they concern collections of interpolants, are not satisfied by arbitrary sequences of Craig interpolants and must be established for each interpolation algorithm and verification technique.

This paper performs a systematic study of collectives in verification and identifies the particular constraints they pose on propositional interpolation systems used in SAT-based model checking. The SAT-based approach provides bit-precise reasoning which is essential both in software and hardware applications, e.g., when dealing with pointer arithmetic and overflow. To-date, there exist successful tools which perform SAT-based model checking (such as CBMC[4] and SATABS[5]), and which integrate it with interpolation (for example, eVolCheck and FunFrog). However, there is no a framework which would correlate the existing interpolation systems and compare the various collectives. This work addresses the problem and contributes as follows:

Contribution 1: This paper, for the first time, collects, identifies, and uniformly presents the most common collectives imposed on interpolation by existing verification approaches (see §2).

In addition to the issues related to a diversity of interpolation properties, it is often desirable to have flexibility in choosing different algorithms for computing different interpolants in a sequence $\{I_i\}$, rather than using a single interpolation algorithm (or *interpolation system*) Itp_S, as assumed in the path interpolation example above. To guarantee such a flexibility, this paper presents a framework which generalizes the traditional setting consisting of a single interpolation system to allow for sequences, or *families*, of interpolation systems. For example, given a family of systems $\mathcal{F} = \{Itp_{S_i}\}_{i=1}^n$, let $I_i = Itp_{S_i}(\tau_1, \ldots \tau_i \mid \tau_{i+1} \ldots \tau_n)$. If the resulting sequence of interpolants $\{I_i\}$ satisfies the condition of path interpolation, we say that the family $\mathcal{F}$ has the path interpolation property.

[4] http://www.cprover.org/cbmc
[5] http://www.cprover.org/satabs

Families find practical applicability in several contexts[6]. One example is LAWI-style verification, where it is desirable to obtain a path interpolant $\{I_i\}$ with weak interpolants at the beginning (i.e., $I_1, I_2, \ldots$) and strong interpolants at the end (i.e., $\ldots, I_{n-1}, I_n$). This would increase the likelihood of the sequence to be inductive and can be achieved by using a family of systems of different strength. Another example is software Upgrade Checking, where function summaries are computed by interpolation. Different functions in a program could require different levels of abstraction by means of interpolation. A system that generates stronger interpolants can yield a tighter abstraction, more closely reflecting the behavior of the corresponding function. On the other hand, a system that generates weaker interpolants would give an abstraction which is more "tolerant" and is more likely to remain valid when the function is updated.

Contribution 2: This paper systematically studies the collectives and the relationships among them; in particular, it shows that for families of interpolation systems the collectives form a hierarchy, whereas for a single system all but two (i.e., path interpolation and simultaneous abstraction) are equivalent (see §3).

Another issue which this paper deals with is the fact that there exist different approaches for generating interpolants. One is to use specialized algorithms: examples are procedures based on constraint solving (e.g., [18]), machine learning (e.g., [21]), and, even, pure verification algorithms like IC3 [2] and PDR [4] that can be viewed as computing a path interpolation sequence. A second, well-known approach is to extract an interpolant of $A \wedge B$ from a resolution proof of unsatisfiability of $A \wedge B$. Examples are the algorithm by Pudlák [16] (also independently proposed by Huang [7] and by Krajíček [10]), the algorithm by McMillan [11], and the Labeled Interpolation Systems (*LISs*) of D'Silva et al. [3], the latter being the most general version of this approach.

The variety of interpolation algorithms makes it difficult to reason about their properties in a systematic manner. At a low level of representation, the challenge is determined by the complexity of individual algorithms and by the diversity among them, which makes it hard to study them uniformly. On the other hand, at a high level, where the details are hidden, not many interesting results can be obtained. For this reason, this paper adopts a twofold approach, working both at a high and at a low level of representation: at the high level, we give a global view of the entire collection of properties and of their relationships and hierarchy; at the low level, we obtain additional stronger results for concrete interpolation systems. In particular, we first investigate the properties of interpolation systems treating them as black boxes, and then focus on the propositional LISs. In the paper, the results of §3 apply to arbitrary interpolation algorithms, while those of §4 apply to LISs.

Contribution 3: For the first time, this paper gives both sufficient and necessary conditions for a family of LISs and for a single LIS to enjoy each of the collectives. In particular, we show that in case of a single system path interpo-

[6] The notion of families is additionally a useful technical tool to make the discussion and the results more general and easier to compare with the prior work of CAV'12 [17] (which formally defined families for the first time).

lation is common to all LISs, while simultaneous abstraction is as strong as all other properties. Concrete applications of our results are also discussed (see §4).

Contribution 4. We developed an interpolating prover, PeRIPLO[7], implementing the proposed framework as discussed in §5; PeRIPLO is currently employed for solving and interpolation by the FunFrog and eVolcheck tools.

Related Work. To our knowledge, despite interpolation being an important component of verification, no systematic investigation of verification-related requirements for interpolants has been done prior to this paper. One exception is the work by the first two authors [17], that studies a subset of the properties in the context of LISs. This paper significantly extends the results of that work by considering the most common collectives used in verification, at the same time addressing a wider class of interpolation systems. Moreover, for LISs, it provides both the *necessary* and *sufficient* conditions for each property.

2 Interpolation Systems

In this section we introduce the basic notions of interpolation, and then proceed to discuss the collectives. Among several possible styles of presentation, we chose the one that highlights te use of collectives in the context of model checking. We employ the standard convention of identifying conjunctions of formulae with sets of formulae and concatenation with conjunction, whenever convenient. For example, we interchangeably use $\{\phi_1, \ldots, \phi_n\}$ and $\phi_1 \cdots \phi_n$ for $\phi_1 \wedge \ldots \wedge \phi_n$.

Interpolation System. An *interpolation system* Itp_S is a function that, given an inconsistent $\Phi = \{\phi_1, \phi_2\}$, returns a *Craig's interpolant*, that is a formula $I_{\phi_1,S} = Itp_S(\phi_1 \mid \phi_2)$ such that:

$$\phi_1 \implies I_{\phi_1,S} \qquad I_{\phi_1,S} \wedge \phi_2 \implies \bot \qquad \mathcal{L}_{I_{\phi_1,S}} \subseteq \mathcal{L}_{\phi_1} \cap \mathcal{L}_{\phi_2}$$

where $\mathcal{L}_\phi$ denotes the atoms of a formula ϕ. That is, $I_{\phi_1,S}$ is implied by ϕ_1, is inconsistent with ϕ_2 and is defined over the common language of ϕ_1 and ϕ_2.

For $\Phi = \{\phi_1, \ldots, \phi_n\}$, we write $I_{\phi_1 \cdots \phi_i,S}$ to denote $Itp_S(\phi_1 \cdots \phi_i \mid \phi_{i+1} \cdots \phi_n)$. W.l.o.g., we assume that, for any Itp_S and any formula ϕ, $Itp_S(\top \mid \phi) = \top$ and $Itp_S(\phi \mid \top) = \bot$, where we equate the constant true $\top$ with the empty formula. We omit S whenever clear from the context.

An interpolation system Itp is called *symmetric* if for any inconsistent $\Phi = \{\phi_1, \phi_2\}$: $Itp(\phi_1 \mid \phi_2) \iff \overline{Itp(\phi_2 \mid \phi_1)}$ (we use the notation $\overline{\phi}$ for the negation of a formula ϕ).

A sequence $\mathcal{F} = \{Itp_{S_1}, \ldots, Itp_{S_n}\}$ of interpolation systems is called a *family*.

Collectives. In the following, we formulate the properties of interpolation systems that are required by existing verification algorithms. Furthermore, we generalize the collectives by presenting them over families of interpolation systems (i.e., we allow the use different systems to generate different interpolants in a sequence). Later, we restrict the properties to the more traditional setting of the singleton families.

***n*-Path Interpolation (PI)** was first defined in [8], where it is employed in the refinement phase of CEGAR-based predicate abstraction. It has also appeared

[7] http://verify.inf.usi.ch/periplo.html

in [22] under the name *interpolation-sequence*, where it is used for a specialized interpolation-based hardware verification algorithm.

Formally, a family of $n+1$ interpolation systems $\{Itp_{S_0}, \ldots, Itp_{S_n}\}$ has the *n-path interpolation* property (n-PI) iff for any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$ and for $0 \leq i \leq n-1$ (recall that $I_\top = \top$ and $I_\Phi = \bot$):

$$(I_{\phi_1 \ldots \phi_i, S_i} \wedge \phi_{i+1}) \implies I_{\phi_1 \ldots \phi_{i+1}, S_{i+1}}$$

n-Generalized Simultaneous Abstraction (GSA) is the generalization of *simultaneous abstraction*, a property that first appeared, under the name *symmetric interpolation*, in [9], where it is used for approximation of a transition relation for predicate abstraction. We changed the name to avoid confusion with the notion of *symmetric interpolation system* (see above). The reason for generalizing the property will be apparent later.

Formally, a family of $n+1$ interpolation systems $\{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ has the *n-generalized simultaneous abstraction* property (n-GSA) iff for any inconsistent $\Phi = \{\phi_1, \ldots, \phi_{n+1}\}$:

$$\bigwedge_{i=1}^{n} I_{\phi_i, S_i} \implies I_{\phi_1 \ldots \phi_n, S_{n+1}}$$

The case $n = 2$ is called *Binary GSA (BGSA)*: $I_{\phi_1, S_1} \wedge I_{\phi_2, S_2} \implies I_{\phi_1 \phi_2, S_3}$.
If $\phi_{n+1} = \top$, the property is called *n-simultaneous abstraction* (n-SA): $\bigwedge_{i=1}^{n} I_{\phi_i, S_i} \implies \bot (= I_{\phi_1 \ldots \phi_n, S_{n+1}})$ and, if $n = 2$, *binary SA (BSA)*. In n-SA $Itp_{S_{n+1}}$ is irrelevant and is often omitted.

n-State-Transition Interpolation (STI) is defined as a combination of PI and SA in a single family of systems. It was introduced in [1] as part of the interprocedural verification algorithm WHALE. Intuitively, the "state" interpolants over-approximate the set of reachable states, and the "transition" interpolants summarize the transition relations (or function bodies). The STI requirement ensures that state over-approximation is "compatible" with the summarization. That is, $\{I_{\phi_1 \cdots \phi_i, S_i}\} I_{\phi_{i+1}, T_{i+1}} \{I_{\phi_1 \cdots \phi_{i+1}, S_{i+1}}\}$ is a valid Hoare triple for each i.

Formally, a family of interpolation systems $\{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ has the *n-state-transition interpolation* property (n-STI) iff for any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$ and for $0 \leq i \leq n-1$:

$$(I_{\phi_1 \ldots \phi_i, S_i} \wedge I_{\phi_{i+1}, T_{i+1}}) \implies I_{\phi_1 \ldots \phi_{i+1}, S_{i+1}}$$

T-Tree Interpolation (TI) is a generalization of classical interpolation used in model checking applications, in which partitions of an unsatisfiable formula naturally correspond to a tree structure such as call tree or program unwinding. The collective was first introduced by McMillan and Rybalchenko for computing post-fixpoints of a system of Horn clauses (e.g., used in analysis of recursive programs) [15], and is equivalent to the nested-interpolants of [5].

Formally, let $T = (V, E)$ be a tree with n nodes $V = [1, \ldots, n]$. A family of n interpolation systems $\{Itp_{S_1}, \ldots, Itp_{S_n}\}$ has the *T-tree interpolation property* (T-TI) iff for any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$:

$$\bigwedge_{(i,j) \in E} I_{F_j, S_j} \wedge \phi_i \implies I_{F_i, S_i}$$

where $F_i = \{\phi_j \mid i \sqsubseteq j\}$, and $i \sqsubseteq j$ iff node j is a descendant of node i in T. Notice that for the root i of T, $F_i = \Phi$ and $I_{F_i,S_i} = \bot$.

An interpolation system Itp_S is said to *have a property* P (or, simply, to have P), where P is one of the properties defined above, if every family induced by Itp_S has P. For example, Itp_S has GSA iff for every k the family $\{Itp_{S_1}, \ldots, Itp_{S_k}\}$, where $Itp_{S_i} = Itp_S$ for all i, has k-GSA.

3 Collectives of Interpolation Systems

In this section, we study collectives of general interpolation systems, that is, we treat interpolation systems as black-boxes. In section §4 we will extend the study to the implementation-level details of the LISs.

Collectives of Single Systems. We begin by studying the relationships among the various collectives of single interpolation systems.

Theorem 1. *Let Itp_S be an interpolation system. The following are equivalent: Itp_S has BGSA (1), Itp_S has GSA (2), Itp_S has TI (3), Itp_S has STI (4).*

Proof. We show that $1 \to 2$, $2 \to 3$, $3 \to 4$, $4 \to 1$.

$(1 \to 2)$ Assume Itp_S has BGSA. Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_{n+1}\}$. Then, for $2 \leq i \leq n$: $(I_{\phi_1 \cdots \phi_{i-1}} \wedge I_{\phi_i}) \Rightarrow I_{\phi_1 \cdots \phi_i}$, which together yield $(\bigwedge_{i=1}^{n} I_{\phi_i}) \Rightarrow I_{\phi_1 \ldots \phi_n}$. Hence, Itp_S has GSA.

$(2 \to 3)$ Let $T = ([1, \ldots, n], E)$, take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. Since Itp_S has GSA: $(\bigwedge_{(i,j) \in E} I_{F_j} \wedge I_{\phi_i}) \Rightarrow I_{F_i}$, and, from the definition of Craig interpolation, $\phi_i \Rightarrow I_{\phi_i}$. Hence, Itp_S has T-TI.

$(3 \to 4)$ Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$ and extend it to a Φ' by adding n copies of $\top$ at the end. Define a tree $T_{STI} = ([1, \ldots, 2n], E)$ s.t.: $E = \{(n+i, i) \mid 1 \leq i \leq n\} \cup \{(n+i, n+i-1) \mid 1 \leq i \leq n\}$. Then, for $1 \leq i \leq n$, $F_i = \{\phi_i\}$ and $F_{n+i} = \{\phi_1, \ldots, \phi_i\}$, where F_i is as in the definition of T-TI. By the T-TI property: $(I_{F_{n+i}} \wedge I_{F_{i+1}} \wedge \top) \Rightarrow I_{F_{n+i+1}}$, which is equivalent to STI.

$(4 \to 1)$ Follows from STI being syntactically equivalent to BGSA for $i = 1$.

Theorem 1 has a few simple extensions. First, GSA implies SA directly from the definitions. Similarly, since $\phi \Rightarrow I_\phi$, STI implies PI. Finally, we conjecture that both SA and PI are strictly weaker than the rest. In §4 (Theorem 16), we show that for LISs, PI is strictly weaker than SA. As for SA, we show that it is equivalent to BGSA in symmetric interpolation systems (Proposition 1 in the appendix). But, in the general case, the conjecture remains open.

These results define a hierarchy of collectives which is summarized in Fig. 1, where the edges indicate implications among the collectives. Note that $SA \to GSA$ holds only for symmetric systems.

In summary, the main contribution in the setting of a single system is the proof that almost all collectives are equivalent and the hierarchy of the collectives collapses. From a practical perspective, this means that McMillan's interpolation system (implemented by most interpolating SMT-solvers) has all of the collective properties, including the recently introduced TI.

Collectives of Families of Systems. Here, we study collectives of families of interpolation systems. We first show that the collectives introduced in §2 directly extend from families to sub-families. Second, we examine the hierarchy of the relationships among the properties. Finally, we conclude by discussing the practical implications of these results.

Collectives of Sub-families. If a family of interpolation systems $\mathcal{F}$ has a property P, then sub-families of $\mathcal{F}$ have P as well. We state this formally for k-STI (since we use it in the proof of Theorem 11); similar statements for the other collectives are discussed in the appendix[8].

Theorem 2. *A family* $\{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-STI iff for all* $k \leq n$ *the sub-family* $\{Itp_{S_0}, \ldots, Itp_{S_k}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_k}\}$ *has k-STI.*

Relationships Among Collectives. We now show the relationships among collectives. First, we note that n-SA and BGSA are equivalent for symmetric interpolation systems. Whenever a family $\mathcal{F} = \{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ has $(n+1)$-SA and $Itp_{S_{n+1}}$ is symmetric, then $\mathcal{F}$ has n-GSA (Proposition 2 in the appendix, which is the analogue of Proposition 1 for single systems).

In the rest of the section, we delineate the hierarchy of collectives. In particular, we show that T-TI is the most general collective, immediately followed by n-GSA, which is followed by $BGSA$ and n-STI, which are equivalent, and at last by n-SA and n-PI. The first result is that the n-STI property implies both the n-PI and n-SA properties separately:

Theorem 3. *If a family* $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-STI then (1)* $\{Itp_{S_0}, \ldots, Itp_{S_n}\}$ *has n-PI and (2)* $\{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-SA.*

A natural question to ask is whether the converse of Theorem 3 is true. That is, whether the family $\mathcal{F}_1 \cup \mathcal{F}_2$ that combines two arbitrary families $\mathcal{F}_1$ and $\mathcal{F}_2$ that independently enjoy n-PI and n-SA, respectively, has n-STI. We show in §4, Theorem 11, that this is not the case.

As for BGSA, the n-STI property is closely related to it: deciding whether a family $\mathcal{F}$ has n-STI is in fact reducible to deciding whether a collection of sub-families of $\mathcal{F}$ has BGSA.

Theorem 4. *A family* $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-STI iff* $\{Itp_{S_i}, Itp_{T_{i+1}}, Itp_{S_{i+1}}\}$ *has BGSA for all* $0 \leq i \leq n-1$.

From Theorem 4 and Theorem 3 we derive:

Corollary 1. *If there exists a family* $\{Itp_{S_0}, \ldots, Itp_{S_n}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *s.t.* $\{Itp_{S_i}, Itp_{T_{i+1}}, Itp_{S_{i+1}}\}$ *has BGSA for all* $0 \leq i \leq n-1$*, then* $\{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-SA.*

We now relate T-TI and n-GSA. Note that the need for two theorems with different statements arises from the asymmetry between the two properties: all ϕ_i are abstracted by interpolation in n-GSA, whereas in T-TI a formula is not abstracted, when considering the correspondent parent together with its children.

[8] All proofs can be found in the appendix.

Theorem 5. *Given a tree $T = (V, E)$ if a family $\mathcal{F} = \{Itp_{S_i}\}_{i \in V}$ has T-TI, then, for every parent i_{k+1} and its children $i_1, \ldots, i_k$:*

1. *If i_{k+1} is the root, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}\}$ has k-SA.*
2. *Otherwise, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}, Itp_{S_{i_{k+1}}}\}$ has k-GSA.*

Theorem 6. *Given a tree $T = (V, E)$, a family $\mathcal{F} = \{Itp_{S_i}\}_{i \in V}$ has T-TI if, for every node i_{k+1} and its children $i_1, \ldots, i_k$, there exists $T_{i_{k+1}}$ such that:*

1. *If i_{k+1} is the root, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}, Itp_{T_{i_{k+1}}}\}$ has $(k+1)$-SA.*
2. *Otherwise, $\{Itp_{S_{i_1}}, \ldots, Itp_{T_{i_{k+1}}}, Itp_{S_{i_{k+1}}}\}$ has $(k+1)$-GSA.*

An important observation is that the T-TI property is the most general, in the sense that it realizes any of the other properties, given an appropriate choice of the tree T. We state here (and prove in the appendix) that n-GSA and n-STI can be implemented by T-TI for some T^n_{GSA} and T^n_{STI}; the remaining cases can be derived in a similar manner. Note that the converse implications are not necessarily true in general, since the tree interpolation requirement is stronger.

Theorem 7. *If a family $\mathcal{F} = \{Itp_{S_{n+1}}, Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ has T^n_{GSA}-TI, then $\{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ has n-GSA.*

Theorem 8. *If a family $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_n}\}$ has T^n_{STI}-TI, then it has n-STI.*

The results of so far (including Theorem 11 of §4) define a hierarchy of collectives which is summarized in Fig. 2. The solid edges indicate direct implication between properties; $SA \to GSA$ requires symmetry, while $GSA \to TI$ requires the existence of an additional set of interpolation systems. The dashed edges represent the ability of TI to realize all the other properties for an appropriate tree; only the edges to STI and GSA are shown, the other ones are implicit. The dash-dotted edges represent the sub-family properties.

An immediate application of our results is that they show how to overcome limitations of existing implementations. For example, they enable the trivial construction of tree interpolants in MathSat[9] (currently only available in iZ3) – thus enabling its usability for Upgrade Checking [20] – by reusing existing BGSA-interpolation implementation of MathSat. Similarly, our results enable construction of BGSA and GSA interpolants in iZ3 (currently only available in MathSat) – thus enabling the use of iZ3 in Whale.

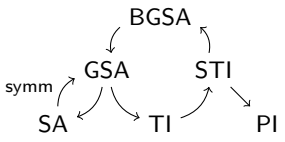

Figure 1: Single systems collectives.

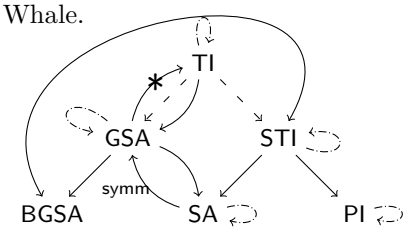

Figure 2: Families of systems collectives.

[9] http://mathsat.fbk.eu/

4 Collectives of Labeled Interpolation Systems

In this section, we move from the abstract level of general interpolation systems to the implementation level of the propositional Labeled Interpolation Systems. After introducing and defining LISs, we study collectives of families, then summarize the results for single LISs, also answering the questions left open in §3. The key results are in Lemmas $1-4$. Unfortunately, the proofs are quite technical. For readability, we focus on the main results and their significance and refer the reader to the appendix for full details.

There are several state-of-the art approaches for automatically computing interpolants. The most successful techniques derive an interpolant for $A \wedge B$ from a resolution proof of the unsatisfiability of the conjunction. Noteworthy examples are the algorithm independently developed by Pudlák [16], Huang [7] and Krajíček [10], and the one by McMillan [11]. These algorithms are implemented recursively by initially computing *partial interpolants* for the axioms (leaves of the proof), and, then, following the proof structure, by computing a partial interpolant for each conclusion from those of the premises. The partial interpolant of the root of the proof is the interpolant for the formula. In this section, we review these algorithms following the framework of D'Silva et al. [3].

Resolution Proofs. We assume a *countable set* of propositional variables. A *literal* is a variable, either with positive (p) or negative ($\overline{p}$) polarity. A *clause* C is a finite disjunction of literals; a formula Φ in conjunctive normal form (CNF) is a finite conjunction of clauses. A *resolution proof of unsatisfiability* (or *refutation*) of a formula Φ in CNF is a tree such that the leaves are the clauses of Φ, the root is the empty clause $\bot$ and the inner nodes are clauses generated via the *resolution rule* (where $C^+ \vee p$ and $C^- \vee \overline{p}$ are the *antecedents*, $C^+ \vee C^-$ the *resolvent*, and p is the *pivot*):

$$\frac{C^+ \vee p \qquad C^- \vee \overline{p}}{C^+ \vee C^-}$$

Labelings and Interpolant Strength. D'Silva et al. [3] generalize the algorithms by Pudlák [16] and McMillan [11] for propositional resolution systems by introducing the notion of *Labeled Interpolation System* (LIS), focusing on the concept of *interpolant strength* (a formula ϕ is stronger than ψ when $\phi \Longrightarrow \psi$).

Given a refutation of a formula $A \wedge B$, a variable p can appear as a literal only in A, only in B or in both; p is respectively said to have *class* A, B or AB. A *labeling* L is a mapping that assigns a *label* among $\{a, b, ab\}$ independently to each variable in each clause; we assume that no clause has both a literal and its negation, so assigning a label to variables or literals is equivalent. The set of possible labelings is restricted by ensuring that class A variables have label a and class B variables label b; AB variables can be labeled either a, b or ab.

In [3], a *Labeled Interpolation System* (LIS) is defined as a procedure Itp_L (shown in Fig. 3) that, given A, B, a refutation R of $A \wedge B$ and a labeling L, outputs a partial interpolant $I_{A,L}(C) = Itp_L(A \mid B)(C)$ for any clause C in R; this depends on the clause being in A or B (if leaf) and on the label of the pivot associated with the resolution step (if inner node). $I_{A,L} = Itp_L(A \mid B)$ represents the interpolant for $A \wedge B$, that is $Itp_L(A \mid B)(\bot)$. We omit the parameters whenever clear from the context.

Leaf: $C\,[I]$	Inner node: $\dfrac{C^+ \vee p : \alpha\,[I^+] \qquad C^- \vee \overline{p} : \beta\,[I^-]}{C^+ \vee C^-\,[I]}$
$I = \begin{cases} C\downarrow b & \text{if } C \in A \\ \neg(C\downarrow a) & \text{if } C \in B \end{cases}$	$I = \begin{cases} I^+ \vee I^- & \text{if } \alpha \sqcup \beta = a \\ I^+ \wedge I^- & \text{if } \alpha \sqcup \beta = b \\ (I^+ \vee p) \wedge (I^- \vee \overline{p}) & \text{if } \alpha \sqcup \beta = ab \end{cases}$

Figure 3: Labeled Interpolation System Itp_L.

In Fig. 3, $C\downarrow \alpha$ denotes the restriction of a clause C to the variables with label α. $p : \alpha$ indicates that variable p has label $\alpha \in \{a, b, ab\}$. By $C[I]$ we represent that clause C has a partial interpolant I. I^+, I^- and I are the partial interpolants respectively associated with the two antecedents and the resolvent of a resolution step: $I^+ = Itp_L(C^+ \vee p)$, $I^- = Itp_L(C^- \vee \overline{p})$, $I = Itp_L(C^+ \vee C^-)$.

A join operator $\sqcup$ allows to determine the label of a pivot p, taking into account that p might have different labels α and β in the two antecedents: $\sqcup$ is defined by $a \sqcup b = ab$, $a \sqcup ab = ab$, $b \sqcup ab = ab$.

The systems corresponding to McMillan and Pudlák's interpolation algorithms are referred to as Itp_M and Itp_P; the system dual to McMillan's is $Itp_{M'}$. Itp_M, Itp_P and $Itp_{M'}$ are obtained as special cases of Itp_L by labeling all the occurrences of AB variables with b, ab and a, respectively (see [3] and [17]).

A total order $\preceq$ is defined over labels as $b \preceq ab \preceq a$, and pointwise extended to a partial order over labelings: $L \preceq L'$ if, for every clause C and variable p in C, $L(p, C) \preceq L'(p, C)$. This allows to directly compare the logical strength of the interpolants produced by two systems. In fact, for any refutation R of a formula $A \wedge B$ and labelings L, L' such that $L \preceq L'$, we have $Itp_L(A, B, R) \implies Itp_{L'}(A, B, R)$ and we say that Itp_L is *stronger* than $Itp_{L'}$ [3].

Since a labeled system Itp_L is uniquely determined by the labeling L, when discussing a family of LISs $\{Itp_{L_1}, \ldots, Itp_{L_n}\}$ we will refer to the correspondent *family of labelings* as $\{L_1, \ldots, L_n\}$.

Labeling Notation. In the previous sections, we saw how the various collectives involve the generation of multiple interpolants from a single inconsistent formula $\Phi = \{\phi_1, \ldots, \phi_n\}$ for different subdivisions of Φ into an A and a B parts; we refer to these ways of splitting Φ as *configurations*. Remember that a labeling L has freedom in assigning labels only to occurrences of variables of class AB; each configuration identifies these variables.

Since we deal with several configurations at a time, it is useful to separate the variables into *partitions* of Φ depending on whether the variables are local to a ϕ_i or shared, taking into account all possible combinations. For example, Table 1 is the *labeling table* that characterizes 3-SA. Recall that in 3-SA we are given an inconsistent $\Phi = \{\phi_1, \phi_2, \phi_3\}$ and a family of labelings $\{L_1, L_2, L_3\}$ and generate three interpolants I_{ϕ_1, L_1}, I_{ϕ_2, L_2}, I_{ϕ_3, L_3}. The labeling L_i is associated with the ith configuration. For example, the table shows that L_1 can independently assign a label from $\{a, b, ab\}$ to each occurrence of each variable shared between ϕ_1 and ϕ_2, ϕ_1 and ϕ_3 or ϕ_1, ϕ_2 and ϕ_3 (as indicated by the presence of $\alpha_1, \gamma_1, \delta_1$).

When talking about an occurrence of a variable p in a certain partition $\phi_{i_1} \cdots \phi_{i_k}$, it is convenient to associate to p and the partition a *labeling vector* $(\eta_{i_1}, \ldots, \eta_{i_k})$, representing the labels assigned to p by $L_{i_1}, \ldots, L_{i_k}$ in configuration $i_1, \ldots, i_k$ (all other labels are fixed). Strength of labeling vectors is compared pointwise, extending the linear order $b \preceq ab \preceq a$ as described earlier.

We reduce the problem of deciding whether a family $\mathcal{F} = \{Itp_{L_1}, \ldots, Itp_{L_n}\}$ has an interpolation property P to showing that all labeling vectors of $\{L_1, \ldots, L_n\}$

p in ?	Variable *class*, *label*		
	$\phi_1 \mid \phi_2\phi_3$	$\phi_2 \mid \phi_1\phi_3$	$\phi_3 \mid \phi_1\phi_2$
ϕ_1	A, a	B, b	B, b
ϕ_2	B, b	A, a	B, b
ϕ_3	B, b	B, b	A, a
$\phi_1\phi_2$	AB, α_1	AB, α_2	B, b
$\phi_2\phi_3$	B, b	AB, β_2	AB, β_3
$\phi_1\phi_3$	AB, γ_1	B, b	AB, γ_3
$\phi_1\phi_2\phi_3$	AB, δ_1	AB, δ_2	AB, δ_3

Table 1: 3-SA.

p in ?	Variable *class*, *label*		
	$\phi_1 \mid \phi_2\phi_3$	$\phi_2 \mid \phi_1\phi_3$	$\phi_1\phi_2 \mid \phi_3$
ϕ_1	A, a	B, b	A, a
ϕ_2	B, b	A, a	A, a
ϕ_3	B, b	B, b	B, b
$\phi_1\phi_2$	AB, α_1	AB, α_2	A, a
$\phi_2\phi_3$	B, b	AB, β_2	AB, β_3
$\phi_1\phi_3$	AB, γ_1	B, b	AB, γ_3
$\phi_1\phi_2\phi_3$	AB, δ_1	AB, δ_2	AB, δ_3

Table 2: BGSA.

satisfy a certain set of *labeling constraints*. For simplicity of presentation, in the rest of the paper we assume that all occurrences of a variable are labeled uniformly. The extension to differently labeled occurrences is straightforward.

Collectives of LISs Families. We derive in the following both *necessary* and *sufficient* conditions for the collectives to hold in the context of LISs families. The practical significance of our results is to identify which LISs satisfy which collectives. In particular, for the first time, we show that not all LISs identified by D'Silva et al. satisfy all collectives. This work provides an essential guide for using interpolant strength results when collectives are required (such as in Upgrade Checking).

We proceed as follows. First, we identify necessary and sufficient labeling constraints to characterize BGSA. Second, we extend them to n-GSA and to n-SA. Third, we exploit the connections between BGSA and n-GSA on one side, and n-STI and T-TI on the other (Theorem 4, Lemma 5, Lemma 6) to derive the labeling constraints both for n-STI and T-TI, thus completing the picture.

BGSA. Let $\Phi = \{\phi_1, \phi_2, \phi_3\}$ be an unsatisfiable formula in CNF, and $\mathcal{F} = \{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ a family of LISs. We want to identify the restrictions on the labeling vectors of $\{L_1, L_2, L_3\}$ for which $\mathcal{F}$ has BGSA, i.e., $I_{\phi_1,L_1} \wedge I_{\phi_2,L_2} \implies I_{\phi_1\phi_2,L_3}$. We define a set of *BGSA constraints* CC_{BGSA} on labelings as follows. A family of labelings $\{L_1, L_2, L_3\}$ satisfies CC_{BGSA} iff:

$$(\alpha_1, \alpha_2), (\delta_1, \delta_2) \preceq \{(ab, ab), (b, a), (a, b)\}, \beta_2 \preceq \beta_3, \gamma_1 \preceq \gamma_3, \delta_1 \preceq \delta_3, \delta_2 \preceq \delta_3$$

hold for all variables, where α_i, β_i, γ_i and δ_i are as shown in Table 2, the labeling table for *BGSA*. $* \preceq \{*_1, *_2\}$ denotes that $* \preceq *_1$ or $* \preceq *_2$ (both can be true).

We aim to prove that CC_{BGSA} is necessary and sufficient for a family of LISs to have BGSA. On one hand, we claim that, if $\{L_1, L_2, L_3\}$ satisfies CC_{BGSA}, then $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ has BGSA. It is sufficient to prove the thesis for a set of *restricted BGSA constraints* CC^*_{BGSA}, defined as follows:

$$(\alpha_1, \alpha_2), (\delta_1, \delta_2) \in \{(ab, ab), (b, a), (a, b)\}, \beta_2 = \beta_3, \gamma_1 = \gamma_3, \delta_3 = \max\{\delta_1, \delta_2\}$$

Lemma 1. *If* $\{L_1, L_2, L_3\}$ *satisfies* CC^*_{BGSA}*, then* $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ *has BGSA.*

The CC^*_{BGSA} constraints can be relaxed to CC_{BGSA} as shown in [17] (Theorem 2, Lemma 3), due to the connection between partial order on labelings and LISs and strength of the generated interpolants. For example, the constraint $\delta_3 = \max(\delta_1, \delta_2)$ can be relaxed to $\delta_3 \succeq \delta_1$, $\delta_3 \succeq \delta_2$. This leads to:

Corollary 2. *If* $\{L_1, L_2, L_3\}$ *satisfies* CC_{BGSA}*, then* $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ *has BGSA.*

On the other hand, it holds that the satisfaction of the CC_{BGSA} constraints is necessary for BGSA:

Lemma 2. *If* $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ *has BGSA, then* $\{L_1, L_2, L_3\}$ *satisfies* CC_{BGSA}.

Having proved that CC_{BGSA} is both sufficient and necessary, we conclude:

Theorem 9. *A family* $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ *has BGSA if and only if* $\{L_1, L_2, L_3\}$ *satisfies* CC_{BGSA}.

n-GSA. After addressing the binary case, we move to defining necessary and sufficient conditions for n-GSA. A family of LISs $\{Itp_{L_1}, \ldots, Itp_{L_{n+1}}\}$ has n-GSA if, for any $\Phi = \{\phi_1, \ldots, \phi_{n+1}\}$, $I_{\Phi_1, L_1} \wedge \cdots \wedge I_{\phi_n, L_n} \implies I_{\phi_1 \ldots \phi_n, L_{n+1}}$, provided Φ is inconsistent. As we defined a set of labeling constraints for BGSA, we now introduce *n-GSA constraints* (CC_{nGSA}) on a family of labelings $\{L_1, \ldots, L_{n+1}\}$; for every variable with labeling vector $(\alpha_{i_1}, \ldots, \alpha_{i_{k+1}})$, $1 \leq k \leq n$, letting $m = i_{k+1}$ if $i_{k+1} \neq n+1$, $m = i_k$ otherwise:

(1) $(\exists j \in \{i_1, \ldots, i_m\}\, \alpha_j = a) \implies (\forall h \in \{i_1, \ldots, i_m\}\, h \neq j \implies \alpha_h = b)$

(2) Moreover, if $i_{k+1} = n+1 : \forall j \in \{i_1, \ldots, i_k\}, \alpha_j \preceq \alpha_{i_{k+1}}$

That is, if a variable is not shared with ϕ_{n+1}, then, if one of the labels is a, all the others must be b; if the variable is shared with ϕ_{n+1}, condition (1) still holds for $(\alpha_{i_1}, \ldots, \alpha_{i_{k-1}})$, and all these labels must be stronger or equal than $\alpha_{i_{k+1}} = \alpha_{n+1}$. We can prove that these constraints are necessary and sufficient for a family of LIS to have n-GSA:

Theorem 10. *A family* $\mathcal{F} = \{Itp_{L_1}, \ldots, Itp_{L_{n+1}}\}$ *has n-GSA if and only if* $\{L_1, \ldots, L_{n+1}\}$ *satisfies* CC_{nGSA}.

In [17] (see Setting 1) it is proved that n-SA holds for any family of LISs stronger than Pudlák. Theorem 10 is strictly more general, since it allows for tuples of labels (e.g., $(\alpha_1, \alpha_2) = (a, b)$ or $(\delta_1, \delta_3, \delta_2) = (a, b, b)$) that were not considered in [17]. The constraints for n-SA follow as a special case of CC_{nGSA}:

Corollary 3. *A family* $\mathcal{F} = \{Itp_{L_1}, \ldots, Itp_{L_n}\}$ *has n-SA if and only if* $\{L_1, \ldots, L_n\}$ *satisfies the following constraints: for every variable with labeling vector* $(\alpha_{i_1}, \ldots, \alpha_{i_k})$, *for* $2 \leq k \leq n$: $(\exists j \in \{i_1, \ldots, i_k\}\, \alpha_j = a) \implies (\forall h \in \{i_1, \ldots, i_k\}\, h \neq j \implies \alpha_h = b)$.

Moreover, a family that has $(n+1)$-SA also has n-GSA if the last member of the family is Pudlák's system. In fact, from Proposition 2 and Pudlák's system being symmetric (as shown in [7]), it follows that *if a family* $\{Itp_{L_1}, \ldots, Itp_{L_n}, Itp_P\}$ *has* $(n+1)$*-SA, then it has n-GSA.*

After investigating n-GSA and n-SA, we address two questions which were left open in §3: do n-SA and n-PI imply n-STI? Is the requirement of additional interpolation systems necessary to obtain T-TI from n-GSA? We show here that n-SA and n-PI do not necessarily imply n-STI, and that, for LISs, n-GSA and T-TI are equivalent.

n-STI. Theorem 3 shows that if a family has n-STI, then it has both n-SA and n-PI. We prove that the converse is not necessarily true. First, it is not difficult to show that any family $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}\}$ has 2-PI (Proposition 3 in the appendix); a second result is that:

Lemma 5. *There exists a family* $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}\}$ *that has* 2*-PI and a family* $\{Itp_{L'_1}, Itp_{L'_2}\}$ *that has* 2*-SA, but the family* $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}, Itp_{L'_1}, Itp_{L'_2}\}$ *does not have* 2*-STI.*

We obtain the main result applying the STI sub-family property (Theorem 2):

Theorem 11. *There exists a family* $\{Itp_{S_0}, \ldots, Itp_{S_n}\}$ *that has* n*-PI, and a family* $\{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *that has* n*-SA, but the family* $\{Itp_{S_0}, \ldots, Itp_{S_n}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *does not have* n*-STI.*

T-TI. The last collective to be studied is T-TI. Theorem 6 shows how T-TI can be obtained by multiple applications of GSA at the level of each parent and its children, provided that we can find an appropriate labeling to generate an interpolant for the parent. We prove here that, in the case of LISs, this requirement is not needed, and derive explicit constraints on labelings for T-TI.

Let us define n*-GSA strengthening* any property derived from n-GSA by not abstracting any of the subformulae ϕ_i, for example $I_{\phi_1,L_1} \wedge \ldots \wedge I_{\phi_{n-1},L_{n-1}} \wedge \phi_n \implies I_{\phi_1 \ldots \phi_n, L_{n+1}}$; it can be proved that:

Lemma 6. *The set of labeling constraints of any* n*-GSA strengthening is a subset of constraints of* n*-GSA.*

From Theorem 6 and Lemma 6, it follows that:

Lemma 7. *Given a tree* $T = (V, E)$ *a family* $\{Itp_{S_i}\}_{i \in V}$ *has* T*-TI if, for every parent* i_{k+1} *and its children* $i_1, \ldots, i_k$*, the family of labelings of the* $(k+1)$*-GSA strengthening obtained by non abstracting the parent satisfies the correspondent subset of* $(k+1)$*-GSA constraints.*

Note that, in contrast to Theorem 6, in the case of LISs we do not need to ensure the existence of an additional set of interpolation systems to abstract the parents. The symmetry between the necessary and sufficient conditions given by Theorem 6 and Theorem 5 is restored, and we establish:

Theorem 12. *Given a tree* $T = (V, E)$ *a family* $\{Itp_{S_i}\}_{i \in V}$ *has* T*-TI if and only if for every parent* i_{k+1} *and its children* $i_1, \ldots, i_k$*, the family of labelings of the* $(k+1)$*-GSA strengthening obtained by non abstracting the parent satisfies the correspondent subset of* $(k+1)$*-GSA constraints.*

Alternatively, in the case of LISs, the additional interpolation systems can be constructed explicitly:

Theorem 13. *Any* $\mathcal{F} = \{Itp_{L_{i_1}}, \ldots, Itp_{L_{i_k}}, Itp_{L_{n+1}}\}$ *s.t.* $k < n$ *that has an* n*-GSA strengthening property can be extended to a family that has* n*-GSA.*

Collectives of Single LISs. In the following, we highlight the fundamental results in the context of single LISs, which represent the most common application of the framework of D'Silva et al. to SAT-based model checking.

First, importantly for practical applications, any LIS satisfies PI:

Theorem 14. *PI holds for all single LISs.*

Second, recall that in §3 we proved that BGSA, STI, TI, GSA are equivalent for single interpolation systems, and that SA $\rightarrow$ BGSA for symmetric ones. We now show that for a single LIS, SA is equivalent to BGSA and that PI is not.

Theorem 15. *If a LIS has SA, then it has BGSA.*

Proof. We show that, for any L, the labeling constraints of SA imply those of BGSA. Refer to Table 2, Table 1, Theorem 10 and Corollary 3. In case of a family $\{L_1, L_2, L_3\}$, the constraints for 3-SA are:

$$(\alpha_1, \alpha_2), (\beta_2, \beta_3), (\gamma_1, \gamma_3) \preceq \{(ab, ab), (b, a), (a, b)\}$$
$$(\delta_1, \delta_2, \delta_3) \preceq \{(ab, ab, ab), (a, b, b), (b, a, b), (b, b, a)\}$$

When $L_1 = L_2 = L_3$, they simplify to $\alpha, \beta, \gamma, \delta \in \{ab, b\}$; this means that, in case of a single LIS, only Pudlák's or stronger systems are allowed. In case of a family $\{L_1, L_2, L_3\}$, the constraints for BGSA are:

$$(\alpha_1, \alpha_2), (\delta_1, \delta_2) \preceq \{(ab, ab), (b, a), (a, b)\}, \beta_2 \preceq \beta_3, \gamma_1 \preceq \gamma_3, \delta_1 \preceq \delta_3, \delta_2 \preceq \delta_3$$

When $L_1 = L_2 = L_3$, they simplify to $\alpha, \delta \in \{ab, b\}$; clearly, the constraints for 3-SA imply those for BGSA, but not vice versa.
Finally, Theorem 14 and Theorem 15 yield:

Theorem 16. *The system $Itp_{M'}$ has PI but does not have BGSA.*

Proof. From the *proof* of Theorem 15: a LIS has the BGSA property iff it is stronger or equal than Pudlák's system. $Itp_{M'}$ is strictly weaker than Itp_P. Thus, it does not have BGSA.

Note that the necessary and sufficient conditions for LISs to support each of the collectives simplify implementing procedures with a given property, or, more importantly from a practical perspective, determine which implementation supports which property.

5 Implementation

We developed an interpolating prover, PeRIPLO[10], which implements the proposed framework. PeRIPLO is, to the best of our knowledge, the first SAT-solver built on MiniSAT 2.2.0 that realizes the Labeled Interpolation Systems of [3] and allows to perform interpolation, path interpolation, generalized simultaneous abstraction, state-transition interpolation and tree interpolation; it also offers proof logging and manipulation routines. The tool has been integrated within the FunFrog and eVolCheck verification frameworks, which make use of its solving and interpolation features for SAT-based model checking. In theory, using different partitions of the same formula and different labelings with each partition does not change the algorithmic complexity of LISs (see appendix C). In our experience, there is no overhead in practice as well.

6 Conclusions

Craig interpolation is a widely used approach in abstraction-based model checking. This paper conducts a systematic investigation of the most common interpolation properties exploited in verification, focusing on the constraints they pose on propositional interpolation systems used in SAT-based model checking.

[10] An executable of PeRIPLO is available to reviewers for experimentation at http://www.inf.usi.ch/phd/rollini/ATVA2013.tar.gz

The paper makes the following contributions. It systematizes and unifies various properties imposed on interpolation by existing verification approaches and proves that for families of interpolation systems the properties form a hierarchy, whereas for a single system all properties except path interpolation and simultaneous abstraction are in fact equivalent. Additionally, it defines and proves both sufficient and necessary conditions for a family of Labeled Interpolation Systems. In particular, it demonstrates that in case of a single system path interpolation is common to all LISs, while simultaneous abstraction is as strong as all other more complex properties. Extending our framework to address interpolation in first order theories is an interesting open problem, and is part of our future work.

References

1. A. Albarghouthi, A. Gurfinkel, and M. Chechik. Whale: An Interpolation-Based Algorithm for Inter-procedural Verification. In *VMCAI'12*, pages 39–55.
2. A. R. Bradley. SAT-Based Model Checking without Unrolling. In *VMCAI'11*.
3. V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant Strength. In *VMCAI'10*, pages 129–145.
4. N. Een, A. Mishchenko, and R. Brayton. Efficient Implementation of Property-Directed Reachability. In *FMCAD'11*.
5. M. Heizmann, J. Hoenicke, and A. Podelski. Nested Interpolants. In *POPL'10*.
6. T. Henzinger, R. Jhala, R. Majumdar, and K. McMillan. Abstractions from Proofs. In *POPL'04*, pages 232–244.
7. G. Huang. Constructing Craig Interpolation Formulas. In *COCOON'95*.
8. R. Jhala and K. McMillan. A Practical and Complete Approach to Predicate Refinement. In *TACAS'06*, pages 459–473.
9. R. Jhala and K. McMillan. Interpolant-Based Transition Relation Approximation. In *CAV'05*, pages 39–51.
10. J. Krajícek. Interpolation Theorems, Lower Bounds for Proof Systems, and Independence Results for Bounded Arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997.
11. K. McMillan. An Interpolating Theorem Prover. In *TACAS'04*, pages 16–30.
12. K. McMillan. Applications of Craig Interpolation to Model Checking. In *CSL'04*.
13. K. McMillan. Interpolation and SAT-Based Model Checking. In *CAV'03*.
14. K. McMillan. Lazy Abstraction with Interpolants. In *CAV'06*, pages 123–136.
15. K. McMillan and A. Rybalchenko. Solving Constrained Horn Clauses Using Interpolation. Technical Report MSR-TR-2013-6, Microsoft Research, 2013.
16. P. Pudlák. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *J. Symb. Log.*, 62(3):981–998, 1997.
17. S. Rollini, O. Sery, and N. Sharygina. Leveraging Interpolant Strength in Model Checking. In *CAV'12*.
18. A. Rybalchenko and V. Sofronie-Stokkermans. Constraint Solving for Interpolation. In *VMCAI'07*, pages 346–362.
19. O. Sery, G. Fedyukovich, and N. Sharygina. FunFrog: Bounded Model Checking with Interpolation-based Function Summarization. In *ATVA'12*.
20. O. Sery, G. Fedyukovich, and N. Sharygina. Incremental Upgrade Checking by Means of Interpolation-based Function Summaries. In *FMCAD'12*.
21. R. Sharma, A. V. Nori, and A. Aiken. Interpolants as Classifiers. In *CAV'12*.
22. Y. Vizel and O. Grumberg. Interpolation-Sequence Based Model Checking. In *FMCAD'09*, pages 1–8.

A Properties of Sub-families

Theorem 2. *A family* $\{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has n-STI iff for all* $k \leq n$ *the subfamily* $\{Itp_{S_0}, \ldots, Itp_{S_k}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_k}\}$ *has k-STI.*

Proof. $\rightarrow$) Assume an inconsistent $\Phi \triangleq \{\phi_1, \ldots, \phi_k\}$. We can extend it to a $\Phi' \triangleq \{\phi'_1, \ldots, \phi'_n\}$ such that $\phi'_i \equiv \phi_i$, by adding $n-k$ empty formulae $\top$. If $\mathcal{F}$ has the n-STI property, for $0 \leq j \leq k-1$

$$I_{\phi_1 \ldots \phi_j, S_j} \wedge I_{\phi_{j+1}, T_{j+1}} \rightarrow I_{\phi_1 \ldots \phi_{j+1}, S_{j+1}}$$

$\leftarrow$) Follows from $k = n$.

Theorem 17. *A family* $\mathcal{F} = \{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ *has n-GSA iff for all* $k \leq n$ *all the subfamilies* $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_{k+1}}}\}$ *have k-GSA.*

Proof. $(\rightarrow)$ Let n be a natural number. Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_{k+1}\}$ such that $k \leq n$. Let $\{i_1, \ldots, i_{k+1}\}$ be a subset of $\{1, \ldots, n+1\}$. Extend Φ to a $\Phi' = \{\phi'_1, \ldots, \phi'_{n+1}\}$ by adding $(n-k)$ copies of $\top$, so that $\phi'_{i_1} = \phi_1, \ldots, \phi'_{i_k} = \phi_k$, $\phi'_{i_{k+1}} = \phi_{n+1}$. Since $\mathcal{F}$ has n-GSA:

$$\bigwedge_{j=1}^{n} I_{\phi'_j, S_j} \implies I_{\phi'_1 \ldots \phi'_n, S_{n+1}}$$

and, since $\phi'_j = \top$ for $j \notin \{i_1, \ldots, i_k\}$:

$$\bigwedge_{j \in \{i_1 \ldots i_k\}} I_{\phi_j, S_j} \implies I_{\phi_{i_1 \ldots i_k}, S_{i_{k+1}}}$$

$(\leftarrow)$ Follows from $k = n$.

It is easy to see that the technique used in the proof of Theorem 17, i.e., extending an unsatisfiable formula with $\top$ conjuncts, applies to the other properties as well.

Theorem 18. *A family* $\{Itp_{S_1}, \ldots, Itp_{S_n}\}$ *has n-SA iff for all* $k \leq n$ *all the subfamilies* $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}\}$ *have k-SA.*

Proof. The proof works as in Theorem 17.

Theorem 19. *A family* $\{Itp_{S_0}, \ldots, Itp_{S_n}\}$ *has n-PI iff for all* $k \leq n$ *the subfamily* $\{Itp_{S_0}, \ldots, Itp_{S_k}\}$ *has k-PI.*

Proof. The proof works as in Theorem 2.

Theorem 20. *For a given tree* $T = (V, E)$*, a family* $\{Itp_{S_i}\}_{i \in V}$ *has T-TI iff for every subtree* $T' = (V', E')$ *of* T*, the family* $\{Itp_{S_j}\}_{j \in V'}$ *has* T'*-TI.*

Proof. $\rightarrow$). Assume an inconsistent $\Phi \triangleq \{\phi_{i_1}, \ldots, \phi_{i_k}\}$ decorating T'. We can extend Φ with $|V'| - |V|$ empty formulae $\top$ to $\Phi' \triangleq \{\phi'_1, \ldots, \phi'_n\}$ decorating T. If $\{Itp_{S_i}\}_{v_i \in V}$ has the T-TI property, for all v'_i in V and in particular for all v_i in V'

$$\bigwedge_{(v_i,v_j)\in E'} I_{F_j,S_j} \wedge \phi_i \rightarrow I_{F_i,S_i}$$

$\leftarrow$). Follows from $T' \equiv T$.

B Other Proofs

Proposition 1. *SA implies BGSA in symmetric interpolation systems.*

Proof. Take any inconsistent $\Phi = \{\phi_1, \phi_2, \phi_3\}$. If an interpolation system has SA, then:

$$I_{\phi_1} \wedge I_{\phi_2} \wedge I_{\phi_3} \implies \bot$$

Equivalently,

$$I_{\phi_1} \wedge I_{\phi_2} \implies \overline{I_{\phi_3}}$$

For a symmetric system, $\overline{I_{\phi_3}} = I_{\phi_1\phi_2}$.

Proposition 2. *If a family $\mathcal{F} = \{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ has $(n+1)$-SA and $Itp_{S_{n+1}}$ is symmetric, then $\mathcal{F}$ has n-GSA.*

Proof. Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. Since $\mathcal{F}$ has $(n+1)$-SA, then $I_{\phi_1,S_1} \wedge \cdots \wedge I_{\phi_{n+1},S_{n+1}} \implies \bot$. Assuming $Itp_{S_{n+1}}$ is symmetric, $\overline{I_{\phi_{n+1},S_{n+1}}} = I_{\phi_1,\ldots,\phi_n,S_{n+1}}$ and the thesis is proved.

Theorem 3. *If a family $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ has n-STI then (1) $\{Itp_{S_0}, \ldots, Itp_{S_n}\}$ has n-PI and (2) $\{Itp_{T_1}, \ldots, Itp_{T_n}\}$ has n-SA.*

Proof. (1) It follows from $\phi_i \implies I_{\phi_i,S_i}$ for every i.

(2). Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. If $\mathcal{F}$ has n-STI, then, for $0 \leq i \leq n-1$:

$$I_{\phi_1\cdots\phi_i,S_i} \wedge I_{\phi_{i+1},T_{i+1}} \implies I_{\phi_1\cdots\phi_{i+1},S_{i+1}}$$

Since $I_{\phi_1\cdots\phi_n} = \bot$, we get $I_{\phi_1,T_1} \wedge \cdots \wedge I_{\phi_n,T_n} \implies \bot$.

Theorem 4. *A family $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}, Itp_{T_1}, \ldots, Itp_{T_n}\}$ has n-STI iff $\{Itp_{S_i}, Itp_{T_{i+1}}, Itp_{S_{i+1}}\}$ has BGSA for all $0 \leq i \leq n-1$.*

Proof. ($\rightarrow$). Take any inconsistent $\Phi = \{\phi_1, \phi_2, \phi_3\}$. For $0 \leq i \leq n-1$, extend Φ to a $\Phi' = \{\phi'_1, \ldots, \phi'_n\}$ by adding $(n-3)$ copies of $\top$, so that $\phi'_i = \phi_1$, $\phi'_{i+1} = \phi_2$, $\phi'_{i+2} = \phi_3$. Since $\mathcal{F}$ has n-STI:

$$I_{\phi'_1\cdots\phi'_i,S_i} \wedge I_{\phi'_{i+1},T_{i+1}} \implies I_{\phi'_1\cdots\phi'_{i+1},S_{i+1}}$$

Hence, by construction:

$$I_{\phi_1,S_i} \wedge I_{\phi_2,T_{i+1}} \implies I_{\phi_1\phi_2,S_{i+1}}$$

($\leftarrow$) Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. Since $\{Itp_{S_i}, Itp_{T_{i+1}}, Itp_{S_{i+1}}\}$ has BGSA, it follows that for $\{\phi'_1, \phi'_2, \phi'_3\}$, where $\phi'_1 = \phi_1 \wedge \cdots \wedge \phi_i$, $\phi'_2 = \phi_{i+1}$, $\phi'_3 = \phi_{i+2} \wedge \cdots \wedge \phi_n$:

$$I_{\phi'_1,S_i} \wedge I_{\phi'_2,T_{i+1}} \implies I_{\phi'_1\phi'_2,S_{i+1}}$$

Hence, by construction:

$$I_{\phi_1\ldots\phi_i,S_i} \wedge I_{\phi_{i+1},T_{i+1}} \implies I_{\phi_1\ldots\phi_{i+1},S_{i+1}}$$

Theorem 5. *Given a tree $T = (V, E)$ if a family $\mathcal{F} = \{Itp_{S_i}\}_{i\in V}$ has T-TI, then, for every parent i_{k+1} and its children $i_1, \ldots, i_k$:*

1. *If i_{k+1} is the root, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}\}$ has k-SA.*
2. *Otherwise, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}, Itp_{S_{i_{k+1}}}\}$ has k-GSA.*

Proof. Take any inconsistent $\Phi = \{\phi_{i_1}, \ldots, \phi_{i_{k+1}}\}$. Consider a parent i_{k+1} and its children $i_1, \ldots, i_k$. If i_{k+1} is not the root, extend Φ to a Φ' in such a way that: the children are decorated with $\phi_{i_1}, \ldots, \phi_{i_k}$, all their descendants and i_{k+1} with $\top$, all the nodes external to the subtree rooted in i_{k+1} with ϕ_{n+1}. Since $\mathcal{F}$ has T-TI, then at node i_{k+1}:

$$\bigwedge_{(i_{k+1},j)\in E} I_{F_j,S_j} \wedge \phi_{i_{k+1}} \implies I_{F_{i_{k+1}},S_{i_{k+1}}}$$

that is:

$$\bigwedge_{i\in\{i_1\ldots i_k\}} I_{\phi_i,S_i} \wedge \top \implies I_{\phi_{i_1}\cdots\phi_{i_k},S_{k+1}}$$

If i_{k+1} is the root, the proof simply ignores the presence of $\phi_{i_{k+1}}$ and $S_{i_{k+1}}$.

Theorem 6. *Given a tree $T = (V, E)$, a family $\mathcal{F} = \{Itp_{S_i}\}_{i\in V}$ has T-TI if, for every node i_{k+1} and its children $i_1, \ldots, i_k$, there exists $T_{i_{k+1}}$ such that:*

1. *If i_{k+1} is the root, $\{Itp_{S_{i_1}}, \ldots, Itp_{S_{i_k}}, Itp_{T_{i_{k+1}}}\}$ has $(k+1)$-SA.*
2. *Otherwise, $\{Itp_{S_{i_1}}, \ldots, Itp_{T_{i_{k+1}}}, Itp_{S_{i_{k+1}}}\}$ has $(k+1)$-GSA.*

Proof. Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. Consider a parent i_{k+1} different from the root and its children $i_1, \ldots, i_k$.
If $\{Itp_{S_{i_1}}, \ldots, Itp_{T_{i_{k+1}}}, Itp_{S_{i_{k+1}}}\}$ has k-GSA, for $\{F_{i_1}, \ldots, F_{i_k}, \phi_{i_{k+1}}, \Phi\setminus(\bigcup F_{i_j} \cup \{\phi_{i_{k+1}}\})\}$:

$$\bigwedge_{i\in\{i_1\ldots i_k\}} I_{F_i,S_i} \wedge I_{\phi_{i_{k+1}},T_{i_{k+1}}} \implies I_{F_{i_{k+1}},S_{i_{k+1}}}$$

The thesis follows since $\phi_{i_{k+1}} \implies I_{\phi_{i_{k+1}},T_{i_{k+1}}}$. If i_{k+1} is the root, $I_{F_{i_{k+1}},S_{i_{k+1}}} = \bot$ and $S_{i_{k+1}}$ is superfluous.

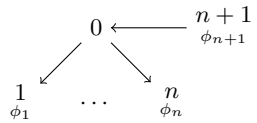

Figure 4: T^n_{GSA}.

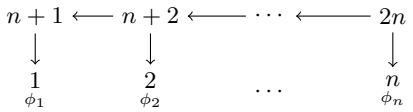

Figure 5: T^n_{STI}.

Theorem 7. *If a family* $\mathcal{F} = \{Itp_{S_{n+1}}, Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ *has* T^n_{GSA}*-TI, then* $\{Itp_{S_1}, \ldots, Itp_{S_{n+1}}\}$ *has n-GSA.*

Proof. Let $T^n_{GSA} = (V, E)$ be the tree shown in Fig. 4, where $V = \{0, \ldots, n+1\}$ and $E = \{(0, i) \mid 1 \leq i \leq n\} \cup \{(n+1, 0)\}$.

Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_{n+1}\}$. We decorate node 0 with $\top$, all other nodes i with ϕ_i, for $1 \leq i \leq n+1$. Since $\mathcal{F}$ has T-TI, then at node 0:

$$\bigwedge_{(0,j)\in E} I_{F_j,S_j} \wedge \top \implies I_{F_0,S_{n+1}}$$

Hence, by construction:

$$\bigwedge_{i=1}^{n} I_{\phi_i,S_i} \implies I_{\phi_1 \ldots \phi_n, S_{n+1}}$$

Theorem 8. *If a family* $\mathcal{F} = \{Itp_{S_0}, \ldots, Itp_{S_n}\} \cup \{Itp_{T_1}, \ldots, Itp_{T_n}\}$ *has* T^n_{STI}*-TI, then it has n-STI.*

Proof. Let $T^n_{STI} = (V, E)$ be the tree shown in Fig. 5, where $V = \{1, \ldots, 2n\}$ and $E = \{(n+i, i) \mid 1 \leq i \leq n\} \cup \{(n+i, n+i-1) \mid 1 \leq i \leq n\}$.

Take any inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$. For $1 \leq i \leq n$, we decorate i with ϕ_i, $n+i$ with $\top$; similarly we associate i with Itp_{T_i} and $n+i$ with Itp_{S_i}. Since $\mathcal{F}$ has T-TI, then at every node $n+i+1$, for $0 \leq i \leq n-1$:

$$(I_{F_{n+i},S_i} \wedge I_{F_{i+1},T_{i+1}}) \wedge \top \implies I_{F_{n+i+1},S_{i+1}}$$

Hence, by construction,

$$I_{\phi_1 \ldots \phi_i, S_i} \wedge I_{\phi_{i+1},T_{i+1}} \implies I_{\phi_1 \ldots \phi_{i+1}, S_{i+1}}$$

Lemma 1. *If* $\{L_1, L_2, L_3\}$ *satisfies* CC^*_{BGSA}*, then* $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ *has BGSA.*

Proof (by structural induction). We remind here the *restricted BGSA constraints* CC^*_{BGSA}:

$$(\alpha_1, \alpha_2), (\delta_1, \delta_2) \in \{(ab, ab), (b, a), (a, b)\}, \beta_2 = \beta_3, \gamma_1 = \gamma_3, \delta_3 = \max\{\delta_1, \delta_2\}$$

The reader can verify that the conditions on the δ_i are equivalent to $(\delta_1, \delta_2, \delta_3) \in \{(ab, ab, ab), (b, a, a), (a, b, a)\}$.

We show that, given a refutation of Φ, for any clause C in the refutation the partial interpolants satisfy $I_{\phi_1,L_1}(C) \wedge I_{\phi_2,L_2}(C) \implies I_{\phi_1\phi_2,L_3}(C)$, that is $I_{\phi_1,L_1}(C) \wedge I_{\phi_2,L_2}(C) \wedge \overline{I_{\phi_1\phi_2,L_3}(C)} \implies \bot$.

For simplicity, we write I_1, I_2, I_3 to refer to the three partial interpolants for C and, if C has antecedents, we denote their partial interpolants with I_1^+, I_2^+, I_3^+ and I_1^-, I_2^-, I_3^-.

Base case (leaf). Case splitting on C (refer to Table 2):

$$\begin{array}{lllll} C \in \phi_1 : & I_1 = C\lfloor_{1,b} & I_2 = \overline{C\lfloor_{2,a}} & \overline{I_3} = \overline{C\lfloor_{3,b}} \\ C \in \phi_2 : & I_1 = \overline{C\lfloor_{1,a}} & I_2 = C\lfloor_{2,b} & \overline{I_3} = \overline{C\lfloor_{3,b}} \\ C \in \phi_3 : & I_1 = \overline{C\lfloor_{1,a}} & I_2 = \overline{C\lfloor_{2,a}} & \overline{I_3} = C\lfloor_{3,a} \end{array}$$

The goal is to show that in each case $I_1 \wedge I_2 \wedge \overline{I_3} \implies \bot$. Representing C by grouping variables into the different partitions, with overbraces to show the label assigned to each variable, we have:

$C \in \phi_1$:

$$C\lfloor_{1,b} = \overbrace{C_{\phi_1}\lfloor_b}^{a} \vee \overbrace{C_{\phi_1\phi_2}\lfloor_b}^{\alpha_1} \vee \overbrace{C_{\phi_1\phi_3}\lfloor_b}^{\gamma_1} \vee \overbrace{C_{\phi_1\phi_2\phi_3}\lfloor_b}^{\delta_1}$$

$$\overline{C\lfloor_{2,a}} = \overbrace{\overline{C_{\phi_1}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_2}\lfloor_a}}^{\alpha_2} \wedge \overbrace{\overline{C_{\phi_1\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_2}$$

$$\overline{C\lfloor_{3,b}} = \overbrace{\overline{C_{\phi_1}\lfloor_b}}^{a} \wedge \overbrace{\overline{C_{\phi_1\phi_2}\lfloor_b}}^{a} \wedge \overbrace{\overline{C_{\phi_1\phi_3}\lfloor_b}}^{\gamma_3} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_b}}^{\delta_3}$$

$C \in \phi_2$:

$$\overline{C\lfloor_{1,a}} = \overbrace{\overline{C_{\phi_2}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_2}\lfloor_a}}^{\alpha_1} \wedge \overbrace{\overline{C_{\phi_2\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_1}$$

$$C\lfloor_{2,b} = \overbrace{C_{\phi_2}\lfloor_b}^{a} \vee \overbrace{C_{\phi_1\phi_2}\lfloor_b}^{\alpha_2} \vee \overbrace{C_{\phi_2\phi_3}\lfloor_b}^{\beta_2} \vee \overbrace{C_{\phi_1\phi_2\phi_3}\lfloor_b}^{\delta_2}$$

$$\overline{C\lfloor_{3,b}} = \overbrace{\overline{C_{\phi_2}\lfloor_b}}^{a} \wedge \overbrace{\overline{C_{\phi_1\phi_2}\lfloor_b}}^{a} \wedge \overbrace{\overline{C_{\phi_2\phi_3}\lfloor_b}}^{\beta_3} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_b}}^{\delta_3}$$

$C \in \phi_3$:

$$\overline{C\lfloor_{1,a}} = \overbrace{\overline{C_{\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_2\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_3}\lfloor_a}}^{\gamma_1} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_1}$$

$$\overline{C\lfloor_{2,a}} = \overbrace{\overline{C_{\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_2\phi_3}\lfloor_a}}^{\beta_2} \wedge \overbrace{\overline{C_{\phi_1\phi_3}\lfloor_a}}^{b} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_2}$$

$$C\lfloor_{3,a} = \overbrace{C_{\phi_3}\lfloor_a}^{b} \vee \overbrace{C_{\phi_2\phi_3}\lfloor_a}^{\beta_3} \vee \overbrace{C_{\phi_1\phi_3}\lfloor_a}^{\gamma_3} \vee \overbrace{C_{\phi_1\phi_2\phi_3}\lfloor_a}^{\delta_3}$$

We can carry out some simplifications, due to the equality constraints in CC^*_{BGSA} and the fact that variables with label a restricted w.r.t. b (and vice versa) are removed, leading (with the help of the resolution rule) to the constraints:

$$(\overbrace{C_{\phi_1\phi_2}\lfloor_b}^{\alpha_1} \vee \overbrace{C_{\phi_1\phi_2\phi_3}\lfloor_b}^{\delta_1}) \wedge \overbrace{\overline{C_{\phi_1\phi_2}\lfloor_a}}^{\alpha_2} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_2} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_b}}^{\delta_3} \implies \bot$$

$$\overbrace{\overline{C_{\phi_1\phi_2}\lfloor_a}}^{\alpha_1} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_1} \wedge (\overbrace{\overline{C_{\phi_1\phi_2}\lfloor_b}}^{\alpha_2} \vee \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_b}}^{\delta_2}) \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_b}}^{\delta_3} \implies \bot$$

$$\overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_1} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_2} \wedge \overbrace{\overline{C_{\phi_1\phi_2\phi_3}\lfloor_a}}^{\delta_3} \implies \bot$$

Finally, the constraints on (α_1, α_2) and $(\delta_1, \delta_2, \delta_3)$ guarantee that the remaining variables are simplified away, proving the base case.

Inductive step (inner node). The inductive hypothesis (i.h.) consists of $I_1^+ \wedge I_2^+ \wedge I_3^+ \implies \bot$, $I_1^- \wedge I_2^- \wedge \overline{I_3^-} \implies \bot$. We do a case splitting on the pivot p:

Case 1 (p in ϕ_1).

$$\begin{aligned}
I_1 \wedge I_2 \wedge \overline{I_3} &\iff \\
(I_1^+ \vee I_1^-) \wedge (I_2^+ \wedge I_2^-) \wedge \overline{(I_3^+ \vee I_3^-)} &\iff \\
(I_1^+ \vee I_1^-) \wedge I_2^+ \wedge I_2^- \wedge \overline{I_3^+} \wedge \overline{I_3^-} &\implies \\
(I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee (I_1^- \wedge I_2^- \wedge \overline{I_3^-}) &\implies^{\text{i.h.}} \bot
\end{aligned}$$

Case 2 (p in ϕ_2).

$$\begin{aligned}
I_1 \wedge I_2 \wedge \overline{I_3} &\iff \\
(I_1^+ \wedge I_1^-) \wedge (I_2^+ \vee I_2^-) \wedge \overline{(I_3^+ \vee I_3^-)} &\iff \\
I_1^+ \wedge I_1^- \wedge (I_2^+ \vee I_2^-) \wedge \overline{I_3^+} \wedge \overline{I_3^-} &\implies \\
(I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee (I_1^- \wedge I_2^- \wedge \overline{I_3^-}) &\implies^{\text{i.h.}} \bot
\end{aligned}$$

Case 3 (p in ϕ_3).

$$\begin{aligned}
I_1 \wedge I_2 \wedge \overline{I_3} &\iff \\
(I_1^+ \wedge I_1^-) \wedge (I_2^+ \wedge I_2^-) \wedge \overline{(I_3^+ \wedge I_3^-)} &\iff \\
I_1^+ \wedge I_1^- \wedge I_2^+ \wedge I_2^- \wedge (\overline{I_3^+} \vee \overline{I_3^-}) &\implies \\
(I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee (I_1^- \wedge I_2^- \wedge \overline{I_3^-}) &\implies^{\text{i.h.}} \bot
\end{aligned}$$

Case 4 (p in $\phi_1\phi_2$). If $(\alpha_1, \alpha_2) = (ab, ab)$:

$$\begin{aligned}
I_1 \wedge I_2 \wedge \overline{I_3} &\iff \\
(I_1^+ \vee p) \wedge (I_1^- \vee \overline{p}) \wedge (I_2^+ \vee p) \wedge (I_2^- \vee \overline{p}) \wedge \overline{(I_3^+ \vee I_3^-)} &\implies \\
(I_1^+ \vee p) \wedge (I_1^- \vee \overline{p}) \wedge (I_2^+ \vee p) \wedge (I_2^- \vee \overline{p}) \wedge (\overline{I_3^+} \vee p) \wedge (\overline{I_3^-} \vee \overline{p}) &\implies \\
((I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee p) \wedge ((I_1^- \wedge I_2^- \wedge \overline{I_3^-}) \vee \overline{p}) &\implies^{\text{resol}} \\
(I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee (I_1^- \wedge I_2^- \wedge \overline{I_3^-}) &\implies^{\text{i.h.}} \bot
\end{aligned}$$

Case 5 (p in $\phi_1\phi_2\phi_3$). If $(\delta_1, \delta_2, \delta_3) = (ab, ab, ab)$:

$$
\begin{aligned}
I_1 \wedge I_2 \wedge \overline{I_3} &\iff \\
(I_1^+ \vee p) \wedge (I_1^- \vee \overline{p}) \wedge (I_2^+ \vee p) \wedge (I_2^- \vee \overline{p}) \wedge \overline{((I_3^+ \vee p) \wedge (I_3^- \vee \overline{p}))} &\iff \\
(I_1^+ \vee p) \wedge (I_1^- \vee \overline{p}) \wedge (I_2^+ \vee p) \wedge (I_2^- \vee \overline{p}) \wedge ((\overline{I_3^+} \wedge \overline{p}) \vee (\overline{I_3^-} \wedge p)) &\implies \\
((I_1^+ \vee p) \wedge (I_2^+ \vee p) \wedge \overline{I_3^+} \wedge \overline{p}) \vee ((I_1^- \vee \overline{p}) \wedge (I_2^- \vee \overline{p}) \wedge \overline{I_3^-} \wedge p) &\implies^{\text{resol}} \\
(I_1^+ \wedge I_2^+ \wedge \overline{I_3^+}) \vee (I_1^- \wedge I_2^- \wedge \overline{I_3^-}) &\implies^{\text{i.h.}} \bot
\end{aligned}
$$

All the remaining cases are treated in a similar manner, to reach a point (possibly after a resolution step if some of the labels are ab) where the inductive hypothesis can be applied.

Lemma 2. *If $\{Itp_{L_1}, Itp_{L_2}, Itp_{L_3}\}$ has BGSA, then $\{L_1, L_2, L_3\}$ satisfies CC_{BGSA}.*

Proof (by contradiction). We remind here the *BGSA constraints* CC_{BGSA}:

$$(\alpha_1, \alpha_2), (\delta_1, \delta_2) \preceq \{(ab, ab), (b, a), (a, b)\}, \beta_2 \preceq \beta_3, \gamma_1 \preceq \gamma_3, \delta_1 \preceq \delta_3, \delta_2 \preceq \delta_3$$

We show that, if any of the CC_{BGSA} constraints is violated, there exist an unsatisfiable formula $\Phi = \{\phi_1, \phi_2, \phi_3\}$ and a refutation such that $I_{\phi_1, L_1} \wedge I_{\phi_2, L_2} \implies I_{\phi_1\phi_2, L_3}$. The possible violations for the CC_{BGSA} constraints consist of:

1. $(\alpha_1, \alpha_2), (\delta_1, \delta_2) \in \{(a, a), (ab, a), (a, ab)\}$
2. $(\beta_2, \beta_3), (\gamma_1, \gamma_3), (\delta_1, \delta_3), (\delta_2, \delta_3) \in \{(a, ab), (a, b), (ab, b)\}$

It is sufficient to take into account $(\alpha_1, \alpha_2) \in \{(a, a), (a, ab)\}$ and $(\beta_2, \beta_3) \in \{(a, ab), (a, b), (ab, b)\}$. The remaining cases follow by symmetry.

(1) $(\alpha_1, \alpha_2) = (a, a) : \phi_1 = (p \vee \overline{q}) \wedge r$, $\phi_2 = (\overline{p} \vee \overline{r}) \wedge q$, $\phi_3 = s$

$$
\begin{array}{l}
A = \phi_1 \quad B = \phi_2, \phi_3 \\
\dfrac{\dfrac{p \vee \overline{q}\,[\bot] \qquad \overline{p} \vee \overline{r}\,[p \wedge r]}{\overline{q} \vee \overline{r}\,[p \wedge r]} \qquad r\,[\bot]}{\overline{q}\,[p \wedge r]} \qquad q\,[\overline{q}]}{\bot\,[(p \wedge r) \vee \overline{q}]} \\
A = \phi_2 \quad B = \phi_1, \phi_3 \\
\dfrac{\dfrac{p \vee \overline{q}\,[\overline{p} \wedge q] \qquad \overline{p} \vee \overline{r}\,[\bot]}{\overline{q} \vee \overline{r}\,[\overline{p} \wedge q]} \qquad r\,[\overline{r}]}{\overline{q}\,[(\overline{p} \wedge q) \vee \overline{r}]} \qquad q\,[\bot]}{\bot\,[(\overline{p} \wedge q) \vee \overline{r}]}
\end{array}
$$

We have $I_{\phi_1, L_1} = (p \wedge r) \vee \overline{q}$, $I_{\phi_2, L_2} = (\overline{p} \wedge q) \vee \overline{r}$, $I_{\phi_1\phi_2, L_3} = \bot$ since s is absent from the proof. Then, $I_{\phi_1, L_1} \wedge I_{\phi_2, L_2} \not\implies I_{\phi_1\phi_2, L_3}$: a counter model is $\overline{q}, \overline{r}$.

(2) $(\alpha_1, \alpha_2) = (a, ab) : \phi_1 = (p \vee \overline{q}) \wedge r$, $\phi_2 = (\overline{p} \vee \overline{r}) \wedge q$, $\phi_3 = s$

$$\dfrac{\dfrac{\dfrac{p \vee \overline{q}\,[\top] \qquad \overline{p} \vee \overline{r}\,[\bot]}{\overline{q} \vee \overline{r}\,[\overline{p}]} \qquad q\,[\bot]}{\overline{r}\,[(\overline{p} \vee \overline{q}) \wedge q]} \qquad r\,[\top]}{\bot\,[((\overline{p} \vee \overline{q}) \wedge q) \vee \overline{r}]}$$

We have $I_{\phi_1,L_1} = (p \wedge r) \vee \overline{q}$ and $I_{\phi_1\phi_2,L_3} = \bot$ as in (1), while $I_{\phi_2,L_2} = ((\overline{p} \vee \overline{q}) \wedge q) \vee \overline{r}$. Then, $I_{\phi_1,L_1} \wedge I_{\phi_2,L_2} \not\Longrightarrow I_{\phi_1\phi_2,L_3}$: a counter model is $\overline{q}, \overline{r}$.

(3) $(\beta_2, \beta_3) = (a, b) : \phi_1 = s$, $\phi_2 = (\overline{p} \vee \overline{r}) \wedge q$, $\phi_3 = (p \vee \overline{q}) \wedge r$

$$\dfrac{\dfrac{\dfrac{p \vee \overline{q}\,[\top] \qquad \overline{p} \vee \overline{r}\,[\overline{p} \vee \overline{r}]}{\overline{q} \vee \overline{r}\,[\overline{p} \vee \overline{r}]} \qquad r\,[\top]}{\overline{q}\,[\overline{p} \vee \overline{r}]} \qquad q\,[q]}{\bot\,[(\overline{p} \vee \overline{r}) \wedge q]}$$

We have $I_{\phi_1,L_1} = \top$, since s is absent from the proof, while $I_{\phi_2,L_2} = (\overline{p} \wedge q) \vee \overline{r}$ as in (1); $I_{\phi_1\phi_2,L_3} = (\overline{p} \vee \overline{r}) \wedge q$. Then, $I_{\phi_1,L_1} \wedge I_{\phi_2,L_2} \not\Longrightarrow I_{\phi_1\phi_2,L_3}$: a counter model is $\overline{q}, \overline{r}$.

(4) $(\beta_2, \beta_3) = (a, ab) : \phi_1 = s$, $\phi_2 = (\overline{p} \vee \overline{r}) \wedge q$, $\phi_3 = (p \vee \overline{q}) \wedge r$

$$\dfrac{\dfrac{\dfrac{p \vee \overline{q}\,[\top] \qquad \overline{p} \vee \overline{r}\,[\bot]}{\overline{q} \vee \overline{r}\,[\overline{p}]} \qquad r\,[\top]}{\overline{q}\,[\overline{p} \vee \overline{r}]} \qquad q\,[\bot]}{\bot\,[(\overline{p} \vee \overline{r} \vee \overline{q}) \wedge q]}$$

$I_{\phi_1,L_1} = \top$ as in (3), $I_{\phi_2,L_2} = (\overline{p} \wedge q) \vee \overline{r}$ as in (1), $I_{\phi_1\phi_2,L_3} = (\overline{p} \vee \overline{r} \vee \overline{q}) \wedge q$. Then, $I_{\phi_1,L_1} \wedge I_{\phi_2,L_2} \not\Longrightarrow I_{\phi_1\phi_2,L_3}$: a counter model is $\overline{q}, \overline{r}$.

(5) $(\beta_2, \beta_3) = (ab, b) : \phi_1 = s$, $\phi_2 = (\overline{p} \vee \overline{r}) \wedge q$, $\phi_3 = (p \vee \overline{q}) \wedge r$

$I_{\phi_1,L_1} = \top$ as in (3), $I_{\phi_2,L_2} = ((\overline{p} \vee \overline{q}) \wedge q) \vee \overline{r}$ as in (2), $I_{\phi_1\phi_2,L_3} = (\overline{p} \vee \overline{r}) \wedge q$ as in (3). Then, $I_{\phi_1,L_1} \wedge I_{\phi_2,L_2} \not\Longrightarrow I_{\phi_1\phi_2,L_3}$: a counter model is $\overline{q}, \overline{r}$.

Lemma 3. *If* $\{L_1, \ldots, L_{n+1}\}$ *satisfies* CC_{nGSA}*, then the family* $\{Itp_{L_1}, \ldots, Itp_{L_{n+1}}\}$ *has n-GSA.*

Proof (by structural induction). We assume that the CC_{nGSA} constraints have been restricted in a similar manner to what shown in CC^*_{BGSA}. We prove that, given a refutation of Φ, for any clause C in the refutation the partial interpolants satisfy $I_{\phi_1,L_1}(C) \wedge \ldots \wedge I_{\phi_n,L_n}(C) \implies I_{\phi_1 \ldots \phi_n,L_{n+1}}(C)$, that is $I_{\phi_1,L_1}(C) \wedge \ldots \wedge I_{\phi_n,L_n}(C) \wedge \overline{I_{\phi_1 \ldots \phi_n,L_{n+1}}(C)} \implies \bot$.

Base case (leaf). Remember that, if $C \in \phi_i, i \neq n+1$, C has class A in configuration i (hence the partial interpolant is $C\lfloor_{i,b}$) and in configuration $n+1$ ($\overline{C\lfloor_{n+1,b}}$) and class B in all the other configurations $j \neq i, n+1$ ($\overline{C\lfloor_{j,a}}$). If $C \in \phi_{n+1}$, it has class B in all configurations ($C\lfloor_{n+1,a}$ in configuration $n+1$, $\overline{C\lfloor_{i,a}}$ everywhere else). So we need to prove:

$$\overline{C\lfloor_{1,a}} \wedge \ldots \wedge \overline{C\lfloor_{i-1,a}} \wedge C\lfloor_{i,b} \wedge \overline{C\lfloor_{i+1,a}} \wedge \ldots \wedge \overline{C\lfloor_{n,a}} \wedge \overline{C\lfloor_{n+1,b}} \implies \bot$$

$$\overline{C\lfloor_{1,a}} \wedge \ldots \wedge \overline{C\lfloor_{i-1,a}} \wedge \overline{C\lfloor_{i,a}} \wedge \overline{C\lfloor_{i+1,a}} \wedge \ldots \wedge \overline{C\lfloor_{n,a}} \wedge C\lfloor_{n+1,a} \implies \bot$$

respectively for $i \neq n+1$ and $i = n+1$.

We can divide the variables of $C \in \phi_i$ into partitions, obtaining $C = C_{\phi_i} \vee C_{\phi_i\phi_2} \vee \ldots \vee C_{\phi_1 \ldots \phi_n}$, leading to a system of constraints as shown for BGSA; the conjunction of:

$$\overline{(C_{\phi_i} \vee C_{\phi_i\phi_2} \vee \ldots \vee C_{\phi_1 \ldots \phi_n})\lfloor_{1,a}}$$

$$\vdots$$

$$(C_{\phi_i} \vee C_{\phi_i\phi_2} \vee \ldots \vee C_{\phi_1 \ldots \phi_n})\lfloor_{i,b}$$

$$\vdots$$

$$\overline{(C_{\phi_i} \vee C_{\phi_i\phi_2} \vee \ldots \vee C_{\phi_1 \ldots \phi_n})\lfloor_{n,a}}$$

must imply $\bot$ for every ϕ_i, $i \neq n+1$ (similarly for ϕ_{n+1}). All the simplifications are carried out in line with the proof of Lemma 1.

Inductive step (inner node). The proof is a again a direct generalization of the proof of Lemma 1.

Performing a case splitting on the pivot and on its labeling vector, the starting point is a conjunction of the partial interpolants $I_1 \wedge \ldots \wedge I_n \wedge \overline{I_{n+1}}$ of C, which is then expressed in terms of the partial interpolants for the antecedents. The goal is to reach a formula $\psi = (I_1^+ \wedge \ldots \wedge I_n^+ \wedge \overline{I_{n+1}^+}) \vee (I_1^- \wedge \ldots \wedge I_n^- \wedge \overline{I_{n+1}^-})$ where the inductive hypothesis can be applied.

The key observation is that the restricted CC_{nGSA} constraints give rise to a combination of boolean operators (after the dualization of the ones in $\overline{I_{n+1}}$ due to the negation) which makes it always possible to obtain the desired ψ, possibly with the help of the resolution rule.

Lemma 4. *If a family $\mathcal{F} = \{Itp_{L_1}, \ldots, Itp_{L_{n+1}}\}$ has n-GSA, then $\{L_1, \ldots, L_{n+1}\}$ satisfies CC_{nGSA}.*

Proof (by induction and contradiction). We prove the theorem by strong induction on $n \geq 2$.

Base Case ($n = 2$). Follows by Lemma 2.

Inductive Step. Assume the thesis holds for all $k \leq n-1$, we prove it for $k = n$. By Lemma 17, if a family $\mathcal{F} = \{Itp_{L_1}, \ldots, Itp_{L_{n+1}}\}$ has n-GSA, then any subfamily of size $k+1 \leq n$ has k-GSA. Combined with the inductive hypothesis, this implies that it is sufficient to establish the theorem for every variable p and labeling vectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{n+1})$ corresponding to partitions $\phi_1 \cdots \phi_n$ and $\phi_1 \cdots \phi_{n+1}$, respectively.

We only show the case of $\boldsymbol{\alpha}$. The proof for $\boldsymbol{\beta}$ is analogous. W.l.o.g., assume that there is a p such that $\boldsymbol{\alpha}$ violates CC_{nGSA} for $\alpha_1 = \alpha_2 = a$ (other cases are symmetric). Construct a family of labelings $\{L'_1, L'_2, L'_{n+1}\}$ from $\{L_1, \ldots, L_{n+1}\}$ by (1) taking all labelings of partitions involving only subsets of ϕ_1, ϕ_2 and ϕ_{n+1}. For example, vectors (η_3, η_4) and $(\eta_1, \eta_2, \eta_3, \eta_{n+1})$ would be discarded, while (η_1, η_2) and $(\eta_1, \eta_2, \eta_{n+1})$ would be kept; and (2) for p, set the labeling vector of partition $\phi_1\phi_2$ to $(\alpha_1, \alpha_2) = (a, a)$. By Lemma 2, $\{L'_1, L'_2, L'_{n+1}\}$ does not have BGSA. Let $\Phi' = \{\phi_1, \phi_2, \phi_{n+1}\}$ be such that $I_{\phi_1, L'_1} \wedge I_{\phi_2, L'_2} \implies I_{\phi_1\phi_2, L'_{n+1}}$, and let Π be the corresponding resolution refutation.

Construct $\Phi = \{\phi_1, \phi_2, p, \ldots, p, \phi_{n+1}\}$ by adding $(n-2)$ copies of p to Φ'. Φ is unsatisfiable, and Π is also a valid refutation for Φ. From this point, we assume that all interpolants are generated from Π.

Assume, by contradiction, that $\mathcal{F}$ has n-GSA. Then,

$$I_{\phi_1, L_1} \wedge \cdots \wedge I_{\phi_n, L_n} \implies I_{\phi_1 \cdots \phi_n, L_{n+1}}$$

But, because $\phi_3, \ldots, \phi_n$ do not contribute any clauses to Π, $I_{\phi_i, L_i} = \top$ for $3 \leq i \leq n$. Hence,

$$I_{\phi_1, L_1} \wedge I_{\phi_2, L_2} \implies I_{\phi_1\phi_2, L_{n+1}}$$

However, by construction:

$$I_{\phi_1, L_1} = I_{\phi_1, L'_1} \qquad I_{\phi_2, L_2} = I_{\phi_2, L'_2} \qquad I_{\phi_1\phi_2, L_{n+1}} = I_{\phi_1\phi_2, L'_{n+1}}$$

which leads to a contradiction. Hence $\boldsymbol{\alpha}$ must satisfy CC_{nGSA}.

Proposition 3. *Any family $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}\}$ has 2-PI.*

Proof. Recall that $I_{\top, L_0} = \top$ and $I_{\phi_1\phi_2, L_2} = \bot$ for any L_0, L_2. Hence, 2-PI reduces to the following two conditions: $\phi_1 \implies I_{\phi_1, L_1}$, $I_{\phi_1, L_1} \wedge \phi_2 \implies \bot$, which are true of any Craig interpolant.

Corollary 4. *A family $\{Itp_{L_1}, Itp_{L_2}\}$ has 2-SA if and only if $\{L_1, L_2\}$ satisfies $(\alpha_1, \alpha_2) \preceq \{(ab, ab), (a, b), (b, a)\}$*

Proof. Follows from Lemma 2 and Lemma 1.

Lemma 5. *There exists a family $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}\}$ that has 2-PI and a family $\{Itp_{L'_1}, Itp_{L'_2}\}$ that has 2-SA, but the family $\{Itp_{L_0}, Itp_{L_1}, Itp_{L_2}, Itp_{L'_1}, Itp_{L'_2}\}$ does not have 2-STI.*

Proof. By Theorem 4, a necessary condition for 2-STI is that $\{Itp_{L_1}, Itp_{L'_2}, Itp_{L_2}\}$ has BGSA. By Proposition 3, $\{L_0, L_1, L_2\}$ can be arbitrary. By Theorem 9 and Corollary 4, there exists $\{L'_1, L'_2\}$ such that $\{Itp_{L'_1}, Itp_{L'_2}\}$ has 2-SA, but $\{Itp_{L_1}, Itp_{L'_2}, Itp_{L_2}\}$ does not have BGSA.

Lemma 6. *The set of labeling constraints of any n-GSA strengthening is a subset of constraints of n-GSA.*

Proof. Assume w.l.o.g we strengthen the first subformula ϕ_1. Then any variable in any partition which does not involve ϕ_1 has the same labeling vector and its n-GSA labeling constraints are also the same. Instead, variables in any partition $\phi_1\phi_{i_2}\ldots\phi_{i_k}$ have now a labeling vector $(\alpha_{i_2}, \ldots, \alpha_{i_k})$, where the first component α_1 is missing. Referring to the definition of CC_{nGSA}, it is easy to verify that the set of the constraints for the strengthening are a subset of the constraints for n-GSA.

Theorem 13. *Any $\mathcal{F} = \{Itp_{L_{i_1}}, \ldots, Itp_{L_{i_k}}, Itp_{L_{n+1}}\}$ s.t. $k < n$ that has an n-GSA strengthening property can be extended to a family that has n-GSA.*

Proof. Refer to the definition of CC_{nGSA} and to Lemma 6. We can complete $\mathcal{F}$ for example by introducing $n-k$ instances of McMillan's system Itp_M. Both constraints (1) and (2) for n-GSA are satisfied, since Itp_M always assigns label b (recall the order $b \preceq ab \preceq a$). Note that Itp_M is not necessarily the only possible choice.

Theorem 14. *PI holds for all single LISs.*

Proof. In [17] we addressed n-PI for a family of LISs $\{Itp_{L_0}, \ldots, Itp_{L_n}\}$. Given an inconsistent $\Phi = \{\phi_1, \ldots, \phi_n\}$, Table 3 shows the labelings L_i, L_{i+1} for an arbitrary step $I_{\phi_1\ldots\phi_i, L_i} \wedge \phi_{i+1} \implies I_{\phi_1\ldots\phi_i\phi_{i+1}, L_{i+1}}$ ($\psi_1 = \phi_1 \wedge \ldots \wedge \phi_i$, $\psi_2 = \phi_{i+1}$, $\psi_3 = \phi_{i+2} \wedge \ldots \wedge \phi_n$):

Table 3: n-PI step.

p in ?	Variable *class*, *label*	
	$\psi_1 \mid \psi_2\psi_3$	$\psi_1\psi_2 \mid \psi_3$
ψ_1	A, a	A, a
ψ_2	B, b	A, a
ψ_3	B, b	B, b
$\psi_1\psi_2$	AB, α_1	A, a
$\psi_2\psi_3$	B, b	AB, β_2
$\psi_1\psi_3$	AB, γ_1	AB, γ_2
$\psi_1\psi_2\psi_3$	AB, δ_1	AB, δ_2

We identified a set of constraints for L_i, L_{i+1} as:

$$\gamma_1 \preceq \gamma_2 \qquad\qquad \delta_1 \preceq \delta_2$$

For a single LIS, $\gamma_1 = \gamma_2$ and $\delta_1 = \delta_2$, so all constraints are trivially satisfied for $0 \leq i \leq n-1$.

Leaf: $C\,[I]$	Inner node: $\dfrac{C^+ \vee p : \alpha\,[I^+] \qquad C^- \vee \overline{p} : \beta\,[I^-]}{C^+ \vee C^-\,[I]}$
$I = \begin{cases} C\downarrow b & \text{if } C \in A \\ \neg(C\downarrow a) & \text{if } C \in B \end{cases}$	$I = \begin{cases} I^+ \vee I^- & \text{if } \alpha \sqcup \beta = a \\ I^+ \wedge I^- & \text{if } \alpha \sqcup \beta = b \\ (I^+ \vee p) \wedge (I^- \vee \overline{p}) & \text{if } \alpha \sqcup \beta = ab \end{cases}$

Labeled interpolation system Itp_L.

C Complexity of the Labeled Interpolation Systems

We briefly examine here the complexity of a Labeled Interpolation System Itp_L. A simple realization of the interpolation algorithm of Fig. 3 (reported above) is based on a topological visit of the refutation DAG.

While visiting a leaf, the partial interpolant is computed by restricting the clause w.r.t. to a or b, given a labeling L for its shared variables. Note that it is not necessary to specify labels for local variables, since variables of class A can only have label a and variables of class B only label b.

While visiting an inner node, (i) the labels of the shared variables of the resolvent clause are updated based on the labels of the antecedent clauses, (ii) the label of the pivot is computed in the same way, and (iii) the partial interpolant is obtained by a boolean combination of the (already computed) partial interpolants of the antecedents, plus possibly two occurrences of the pivot.

We distinguish between the complexity of generating partial interpolants for leaves and inner nodes as follows.

Leaf. The cost of restricting a clause C is $|C|$. Checking whether a clause or a variable has class A, B, AB takes constant time.

Inner node. If C is the resolvent clause, (i) takes $|C|$, both (ii) and (iii) take constant time. We assume that, for each node, the labels of shared variables are encoded in a bit-vector-like data structure, so that retrieving the label of a variable takes constant time.

Assume the DAG has N nodes and the largest clause has size S, then the overall complexity is $O(NS)$. In practice, $S << N$ and the complexity is linear in the size of the DAG. The overhead introduced by the computations due to the use of a labeling is thus negligible.