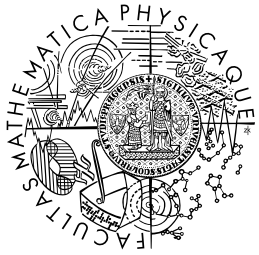

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Martin Blicha

Methods for reduction of Craig's interpolant size using partial variable assignment

Department of Distributed and Dependable Systems

Supervisor of the master thesis:	RNDr. Jan Kofroň, Ph.D.
Study programme:	Computer Science
Study branch:	Theoretical Computer Science

Prague 2016

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date .....

Title: Methods for reduction of Craig's interpolant size using partial variable assignment

Author: Martin Blicha

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D., Department of Distributed and Dependable Systems

Abstract: Since the introduction of interpolants to the field of symbolic model checking, interpolation-based methods have been successfully used in both hardware and software model checking. Recently, variable assignments have been introduced to the computation of interpolants. In the context of abstract reachability graphs, variable assignment can be used not only to prevent out-of-scope variables from appearing in interpolants, but also to reduce the size of the interpolant significantly. We further extend the framework for computing interpolants under variable assignment, prove the correctness of the system and show that it has potential to further decrease the size of the computed interpolants. At the end we analyze under which conditions the computed interpolants will still have the path interpolation property, a desired property in many interpolation-based techniques.

Keywords: Craig interpolation, partial variable assignment, Tseitin's encoding, symbolic model checking

I would like to thank my supervisor, RNDr. Jan Kofroň, Ph.D., for his patience, understanding and encouragement. Most of all I would like to thank him for showing me the application of logic I was looking for.

Contents

Introduction

In recent years, a massive technological development has been achieved and computer systems have penetrated everyday lives. Automation has reached a lot of new, sometimes critical, areas. The reliability of such systems is in many cases of particular importance and a lot of attention has been devoted to the research of methods for establishing their correctness. Model checking is an approach to formal verification that automatically checks whether a model of a system satisfies certain formally specified property. A system is usually represented as a set of possible states in which the system can be and a property is checked with respect to this state representation. At the beginning, an explicit-state model checking was used, which traverses the set of states explicitly, by constructing and examining one state at a time. However, this approach works only for a really small systems and does not scale to real-life designs. To tackle the state explosion problem, methods that can represent and operate with many states simultaneously were being considered. Such methods are called *symbolic*.

Binary decision diagrams (BDDs) were used in the first symbolic model checking method and they manage to solve much larger problems than explicit-state model checking methods. However, even they quickly reached their limit, so other possibilities were examined. The great development and success of SAT solvers led to their widespread use in symbolic model checking. Bounded model checking (BMC, [**?**]) pushed the capabilities of model checking even further. In BMC, the initial condition, the transition relation and the safety property are encoded as propositional formulas and then the SAT solver is used to determine whether or not a state where the safety property does not hold is reachable from the initial state in k steps. This process can be repeated iteratively, gradually increasing k. BMC proved to be very successful in finding counterexamples, however an upper bound on the depth of the state space is needed for BMC to be a complete method. Such bound are often hard to obtain. Several methods have been developed to address this issue with various success.

One of the approaches to extend BMC to a full verification method was introduced by McMillan in [**?**]. It combines bounded model checking with computing Craig interpolants [**?**], resulting in a fully SAT-based *unbounded* model checking method. Since then, more approaches using interpolants in verification have appeared, e.g. [**?**] takes a similar approach as McMillan but uses interpolation sequences instead of single interpolants,[**?**] uses interpolation to refine the labels of nodes of an abstract reachability graph (ARG).

In the state space, Boolean formulas represent set of states and an unsatisfiable pair of formulas (A, B) represent two disjoint sets of states. An interpolant I for (A, B) is then an *over-approximation*, i.e. a superset, of the set represented by A which is still disjoint with set represented by B. This is depicted in Fig. 1.

Since only variables common to both A and B can appear in the interpolant, the representation of the over-approximation can be significantly smaller than the representation of A.

An interpolant for an unsatisfiable pair of formulas is not unique. In [**?**], authors present Labeled Interpolation System (LIS), a framework that is able to compute different interpolants from the same refutation proof. In [**?**], the frame-

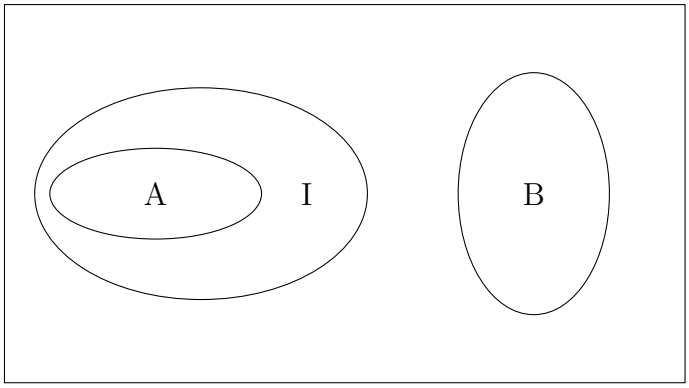

Figure 1: Interpolant in state space

work is further analyzed and a way to compute interpolants comparable in terms of logical strength is presented.

Another way to compare interpolants is in terms of their size. Computing smaller interpolants in one stage can lead to a potential speed-up in further stages of verification tools. Labeled Partial Assignment Interpolation System (LPAIS, [**?**]) incorporates partial variable assignment sometimes provided in verification tools from previous stages into the framework of LIS. If the interpolation problem represents the abstract reachability graph of a program, then using partial assignments can not only reduce the size of the interpolant, but also solve the problem of out-of-scope variables. Moreover, interpolants produced by LPAIS are quantifier-free.

In this thesis, we point out a possible drawback that can prevent LPAIS from fully exploiting the partial assignment and propose a new system extending LPAIS. Moreover, we show that the proposed system fully exploits the partial assignment, prove its correctness, and show that we can guarantee the path interpolation property [**?**] under additional constraints, which, however does not prevent its intended use in the context of ARG.

1. Preliminaries

Much of this thesis is based on the work done in [**?**], so we try to use the same terminology as presented there.

In propositional logic a *literal* is a propositional variable or its negation. A *clause* is a disjunction of zero or more literals. *Empty* clause has zero literals and is equivalent to $\bot$. We say a propositional formula is in *negation normal form* (NNF) if it contains only $\neg$, $\wedge$ and $\vee$ as connectives and negation can occur only in front of variables. *Conjunctive normal form* (CNF) is a special case of NNF. Formula is in CNF if it is a conjunction of clauses. Instead of conjunction of clauses, it is common to use the term "set of clauses", as the semantics of a set of formulas can be define to be their conjunction. We use Θ to denote a set of literals and $\langle\Theta\rangle$ to denote the clause built from the literals in Θ. For a literal l, its variable is denoted by $var(l)$ and the dual literal is denoted by $\bar{l}$. The *resolution* of two clauses $\langle\Theta, p\rangle$ and $\langle\Theta', \bar{p}\rangle$, called the *antecedents*, is a clause $\langle\Theta, \Theta'\rangle$, called *resolvent*. The literals p and $\bar{p}$ are called *pivots* and their variable is the *pivot variable*.

Definition 1.1 (Derivation tree)**.** A *derivation tree* in propositional logic is a tree whose nodes are labeled with propositions according to the following rules:

- Leaves (and only leaves) are labeled with propositional variables.
- If a node has label $\neg\varphi$, then it has a single child with label φ.
- If a node has label $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, then it has two children, the left child has label φ and the right child has label ψ.

A derivation tree for a proposition φ is a derivation tree with root labeled φ.

Derivation tree represents the structure of a given formula. Notice that in each node, the label is a subformula of the represented formula. Moreover, the subformula in each inner node is split according to its main connective and the main connective is uniquely determined. Therefore the labels can be simplified such that the inner nodes are labeled with this main connective instead of the whole subformula.

SAT solver is an algorithm that decides the satisfiability of a propositional formula. However, modern SAT solvers are optimized to decide the satisfiability of a formula in CNF, thus a general formula is usually preprocessed before it is fed to the solver. There is an algorithm that transforms a general formula to an equivalent formula in CNF. However, such formula can have length exponential in the length of the original formula. *Tseitin's encoding* [**?**] is an algorithm that constructs *equisatisfiable* formula whose length is only linear in the length of the original formula. A form of Tseitin's encoding is employed by most modern SAT solvers.

Given a formula φ, the original encoding works as follows:

1. Build a derivation tree of φ and label inner nodes by the main connective of the corresponding subformula.

2. Introduce new encoding variable for each inner node of the tree. We will use suggestive notation, e.g. $e^{\vee}$ to stand for an encoding variable representing a node with disjunction.

3. Introduce equations to capture the meaning of encoding variables: Let e^{*} ($*$ representing logical connective) be an encoding variable representing an inner node with children c_1 and c_2 (or only c_1 in case of negation). Then the following equations captures the meaning of the encoding variables: $e^{\vee} \leftrightarrow c_1 \vee c_2$, $e^{\wedge} \leftrightarrow c_1 \wedge c_2$, $e^{\rightarrow} \leftrightarrow c_1 \rightarrow c_2$, $e^{\neg} \leftrightarrow \neg c_1$.

4. Convert these equations into sets of clauses. For example in case of disjunction the conversion looks like this: $e^{\vee} \leftrightarrow c_1 \vee c_2 \implies (e^{\vee} \rightarrow c_1 \vee c_2) \wedge (c_1 \vee c_2 \rightarrow e^{\vee}) \implies (\neg e^{\vee} \vee c_1 \vee c_2) \wedge (\neg c_1 \vee e^{\vee}) \wedge (\neg c_2 \vee e^{\vee})$.

5. Add r, the encoding variable of the root of the derivation tree, to the set of clauses obtained in the previous step.

The result of Tseitin's encoding on a formula φ is a set of clauses that is equisatisfiable with φ and the result is denoted by $\tau(\varphi)$.

Tseitin's encoding can be used to transform *any* formula to equisatisfiable formula in conjunctive normal form. However, if the input formula is already in *negation normal form* then the encoding can be more concise. If the input formula is in NNF, we can use *one-sided* Tseitin's encoding. It differs from the original encoding in step 3 where instead of using equivalences to capture the meaning of encoding variables only implications from left to right are used. Since in NNF the formula contains only conjunctions, disjunctions and negations, only the following implications are used: $e^{\vee} \rightarrow c_1 \vee c_2$, $e^{\wedge} \rightarrow c_1 \wedge c_2$, $e^{\neg} \rightarrow \neg c_1$.

One more simplification is available. Since the negation is only in front of propositional variables, we can relax the condition on the derivation tree that we stop only at variables and, instead, stop already at literals. As a result we save one encoding variable for each negative literal and there will be encoding variables only for conjunctions and disjunctions. Since this encoding has favorable properties, which we will use later, from now on when referring to Tseitin's encoding we assume this compact version. Note that further savings are possible, for example by allowing n-ary conjunctions and disjunctions, but we do not consider them now, since the fact that the tree is binary is often useful.

Definition 1.2 (Resolution tree)**.** Resolution tree ρ is a directed tree (V, E) with functions piv, cl, and a special vertex s where $Leaves(\rho)$ denotes the set of leaves, i.e. vertices with in-degree 0 and the sink vertex s is the only vertex with out-degree 0. All vertices except the sink has out-degree 1 and all vertices except leaves have in-degree 2. These are called *inner vertices*. The pivot function piv maps the inner vertices to variables. The clause function cl maps the inner vertices to clauses. For each internal vertex v and $(v_1, v), (v_2, v) \in E$, $cl(v) = Res(cl(v_1), cl(v_2), piv(v))$, where $Res(C, D, p)$ denotes the resolution of clauses C and D over a pivot variable p.

Definition 1.3 (Refutation tree)**.** Refutation tree for and unsatisfiable set of clauses S is a resolution tree such that $cl(s)$ is the empty clause and for each leaf v $cl(v)$ is from S.

It is more common to treat the refutation proof as DAG, see e.g. [**?**]. However, for us it is more convenient to work with refutation *tree*. This is not a restriction since every refutation proof in form of a DAG can be unpacked to the full tree.

Refutation tree can be used to produce an interpolant I for A and B, a division of the input set of clauses into two parts. *Interpolant* for a pair of formulas A and B is a formula I such that $A \to I$, $B \to \neg I$ and $Var(I) \subseteq Var(A) \cap Var(B)$. *Interpolation system* is an algorithm for constructing interpolants from a refutation proof for (A, B). Interpolation system usually works by computing partial interpolant for each vertex in the refutation proof. This is done inductively. First, a set of rules determines the partial interpolants for leaves, then another set of rules defines how a partial interpolant for a vertex is computed from partial interpolants of its parents. Finally, the partial interpolant for the sink is the computed interpolant. For an unsatisfiable sequence of formulas $A_1, \ldots, A_n$ an *interpolant sequence* $I_1, \ldots, I_{n+1}$ is defined as follows: $I_1 = \top$, $I_{n+1} = \bot$, $\forall i$: $I_i \wedge A_i \to I_{i+1}$, variables of I_i are among those shared by the sequence $A_1, \ldots, A_{i-1}$ and the sequence $A_i, \ldots, A_n$. If an interpolation system is able to compute interpolation sequence for any unsatisfiable sequence of formulas, we say it has the *path interpolation* property.

In our work with derivation trees and refutation trees we use the following notions. When working with a derivation tree we consistently use the term "node" of a tree, while in a refutation tree we use the term "vertex" of a tree. We use $e^\wedge$ and $e^\vee$ to denote encoding variables obtained from inner nodes labeled with conjunction and disjunction, respectively. Since for every inner node a new encoding variable is introduced, we can use an encoding variable not only to refer to the "real" variable in $\tau(\varphi)$, but also to the node of the derivation tree that it originated from.

Sometimes it is useful to consider the edges in the derivation tree as having an orientation from root to leaves, i.e. going from parent node to child node. It is then possible to refer to an edge as an order pair of nodes: $\mathbf{e} = (n_1, n_2)$ where n_1 as a parent node of n_2.

Definition 1.4 (Parent edge, child edge, incident edge)**.** For every encoding variable e with parent p and children c_1, c_2, we use terms *parent edge* of e for (p, e) and *child edges* for (e, c_1) and (e, c_2). Together we speak of *incident* edges of e. For a node n in the derivation tree let $\mathbf{e}^\uparrow(n)$ stand for the parent edge of n.

Note that there is a relation between the clauses of $\tau(\varphi)$ and the edges in the derivation tree of φ. This can be seen in steps 3 and 4 of the Tseitin's encoding, where the introduced implications capture the relation of the encoding variable (the parent node) to its child nodes. Consider the situation where e has two children c_1 and c_2. If $e = e^\wedge$ then the implication $e^\wedge \to (c_1 \wedge c_2)$ is converted to two clauses $\neg e^\wedge \vee c_1$ and $\neg e^\wedge \vee c_2$, each containing a single child, while if $e = e^\vee$ then the implication $e^\vee \to (c_1 \vee c_2)$ is converted to a single clause $\neg e^\wedge \vee c_1 \vee c_2$ containing both children. We can define a map from edges to clauses in the following way:

Definition 1.5 (Mapping edges to clauses)**.** Let $\Theta \colon E \longrightarrow \tau(\varphi)$ be a function, where E is the set of edges in the derivation tree of φ, such that:

$$\Theta(\mathbf{e}) = \begin{cases} \neg e^\wedge \vee c & \text{if } \mathbf{e} = (e^\wedge; c) \\ \neg e^\vee \vee c_1 \vee c_2 & \text{if } \mathbf{e} = (e^\vee; c_1) \text{ and } c_2 \text{ is the second child of } e^\vee \end{cases}$$

Note that each clause in $\tau(\varphi)$ is obtained from some edge in the derivation tree with the exception of the one-literal clause added in step 5 of the encoding. This clause corresponds to the root r of the tree. We add an artificial edge $\mathbf{e}_r$, which will take the role of a parent edge of the root, to the tree. This enables us to treat all encoding variables equally, as every encoding variable will have a parent edge and two child edges in the derivation tree. Moreover, this enables extending Θ such that $\Theta(\mathbf{e}_r) = \langle r \rangle$. In this way it holds that every clause in $\tau(\varphi)$ is an image of some edge in the mapping Θ, i.e. Θ is surjective.

Observation 1.6. *For every encoding variable e it holds that if $e \in Var(\Theta(\mathbf{e}))$ (e is amongst the variables of clause $\Theta(\mathbf{e})$) then $\mathbf{e}$ is an edge incident with e in the derivation tree. Moreover e has positive occurrence in $\Theta(\mathbf{e})$ if $\mathbf{e}$ is a child edge of e and negative occurrence if it is the parent edge.*

Suppose a partial assignment σ of the variables from $Var(\varphi)$ is given. Under σ, some subformulas of φ are satisfied. These satisfied subformulas correspond to satisfied subtrees in the derivation tree of φ.

Definition 1.7 (Satisfied node)**.** We say that a node in the derivation tree is satisfied if it is the root of a satisfied subtree.

Definition 1.8 (Branch in a refutation tree)**.** A *branch* in a refutation tree is a sequence of vertices starting with a leaf and ending with the sink such that each vertex except the first is the only child of the previous vertex.

Every leaf determines a branch and every branch is determined by its leaf. We write $\beta(\Theta)$ to denote a branch that starts in a leaf with clause Θ. For each pair of branches β and β' there is a single vertex these two branches meet (the first vertex common to both branches) and we write $\beta \wedge \beta'$ to denote this vertex. Moreover we can use the tree ordering to capture the intuitive meaning of a phrase "branch β_1 meets branch β_2 before it meets branch β_3": $\beta_1 \wedge \beta_2 \leq \beta_1 \wedge \beta_3$ means that β_1 meets β_2 before or at the same vertex as it meets β_3.

2. Structure-aware labeled interpolation system

In this chapter we show that if Tseitin's encoding is used to transform the input to an equisatisfiable formula in CNF, then it may happen that LPAIS [**?**] does not fully exploit a given partial assignment. We then propose a modification of the framework that deals with this problem and show the correctness of the new framework.

Consider a simple formula $\neg a \vee (b \wedge c)$ and a partial assignment σ such that $\sigma \models \neg a$. Under σ the whole formula is satisfied and as such should be removed from the problem.

However, after applying Tseitin's encoding, the following set of clauses is obtained from the formula: $\{h_1, \neg h_1 \vee \neg a \vee h_2, \neg h_2 \vee b, \neg h_2 \vee c\}$. In this set the only clause satisfied by σ is $\neg h_1 \vee \neg a \vee h_2$. In this case literals b and c are not excluded from the subproblem as they have should been. This example demonstrates that there is a difference in the notion of a subproblem for a partial assignment in these two approaches.

Suppose the input is given as a set of formulas divided into two parts (A and B) together with a partial variable assignment. The first approach transforms the input set into a set of clauses and then uses partial assignment to narrow the problem to a subproblem by filtering out satisfied clauses (and falsified literals). The second approach is to apply the assignment *first* and only then transform the simplified input to a set of clauses. The result could be a smaller subproblem, as can be seen in the example above.

However, in the context of abstract reachability graph [**?**], where a sequence of interpolants, one for each node, is computed from a single refutation proof for the whole graph, the first approach has the advantage of working with a single refutation proof, while the second approach has to refute each subproblem separately. As a result, the second approach is able to guarantee the path interpolation property for the computed sequence (see [**?**] for the proof), while the first is not.

We want to keep the advantage of the single call to SAT solver while at the same time we would like to fully exploit the assignment to focus on the smallest subproblem.

2.1 Introducing new label

LPAIS introduced a new label into the framework of LIS to keep track of satisfied literals in the refutation tree. When a satisfied literal is a pivot of a resolution, the partial interpolant is copied from the other vertex. This can be seen as if the branch with the satisfied literal is cut off at this vertex. And it makes sense, because it simulates the situation in which the input is filtered out by the assignment before the refutation tree is constructed. The leaf with the satisfied clause should no longer be present in the tree and the falsified pivot in the other branch would also have been filtered out, so no resolution would happen at this point.

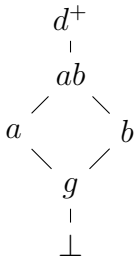

Figure 2.1: Lattice of labels (according $\sqsubseteq$)

However, this reasoning can go one step further. Suppose that at this vertex, where a resolution with a satisfied pivot takes place, there is a literal that was present only in the predecessor with the satisfied pivot. In LPAIS, such a literal is treated as any other at this point. However, in the reasoning above, that branch would no longer be in the tree, so such literal would never appear on the remaining branch. Therefore, we propose to tag such literals and treat them as "ghost" literals, i.e. literals that would not be present in the tree should the assignment be applied beforehand. For this purpose we add a new label g to the system.

Unfortunately, to incorporate this new label to framework, we have to tweak the definitions of its components a little bit. For example, no literal is labeled g in the leaves of the tree. Instead, some literals may acquire it later, under special circumstances. As a consequence, a label of a literal in inner vertex cannot be simply taken to be the supremum of its labels in the predecessor vertices anymore.

To overcome this obstacle, we adopt the approach that labeling function should define only labels of literals in the leaves of the refutation tree. Labels in inner vertices are then computed based on the type of the resolution in the given vertex.

We work with the lattice of labels L depicted in Fig. 2.1.

Definition 2.1 (Labeling function)**.** Let L be a lattice of labels as depicted in Fig. 2.1. Given a refutation tree ρ and a set of literals Lit occurring in ρ, we say that Lab: $Lit \times Leaves(\rho) \longrightarrow L$ is a labeling function if

- $Lab(l, v) = \bot$ iff $l \notin cl(v)$
- $Lab(l, v) \neq g$

The labeling functions as defined above label only literals in leaves. To derive the labels of inner vertices, we define rules for propagating labels from leaves to the sink.

Definition 2.2 (Resolution types and propagating labels)**.** Suppose v is an inner vertex with parents v_1 and v_2 and all literals in the parents are already labeled. Let p denote the pivot in v_1 and $\bar{p}$ denote the pivot in v_2. Firstly, a type of resolution for v is determined by checking the following rules.

- if $Lab(p, v_1) = g$ or $Lab(\bar{p}, v_2) = g$ it is a resolution of type Res-g
- if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2,) = d^+$ it is a resolution of type Res-d^+

- if $Lab(p, v_1,) \sqcup Lab(\bar{p}, v_2) = a$ it is a resolution of type Res-a
- if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2) = b$ it is a resolution of type Res-b
- if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2) = ab$ it is a resolution of type Res-ab

We sometimes use terms "a-resolution" or simple "Res-a" instead of the whole phrase "resolution of type Res-a"

Note that it may happen that two rules are applicable at the same time (if pivot has label g in one parent and any other label in the other parent). In such case the first rule has priority over others.

Next, the labels of literals in v are determined based on the type of the resolution. We use the term *supremum rule* if label of a literal is computed as the supremum of its labels in parent vertices.

- Res-a, Res-b, Res-ab: The supremum rule is used to label literals.
- Res-d^+: Assume the pivot in the first predecessor has label d^+ (the second case is symmetric). Then first, all literals from v_1 are considered as having label g, and then the supremum rule is used to determine the labels in v.
- Res-g: Assume the pivot in the first predecessor has label g (the second case is symmetric). Then first, all literals from v_2 are considered as having label g, and then the supremum rule is used to determine the labels in v. Actually, it may happen that both pivots have label g. It is important to apply this rule only once in such case. Therefore, we arbitrarily decided to use the rule as if only the first pivot is labeled g.

Example 2.3. Consider a resolution $\frac{v_1:\langle p,q\rangle \quad v_2:\langle \bar{p},q,r\rangle}{v:\langle q,r\rangle}$ with labeling function Lab such that $Lab(p, v_1) = g$, $Lab(q, v_1) = a$, $Lab(\bar{p}, v_2) = Lab(q, v_2) = Lab(r, v_2) = b$. Then it is a g-resolution and the labels of q and r in v are determined as follows: First, the labels of q and r in v_2 are considered to be g. Then the supremum rule is used resulting in $Lab(q, v) = a \sqcup g = a$ and $Lab(r, v) = \bot \sqcup g = g$.

2.2 Locality-preserving labeling functions

We have defined a labeling function for a resolution proof and showed how labeling is propagated from leaves to the sink. However, if a labeling function is to be used to compute interpolant from the resolution proof, additional conditions must be satisfied. For this purpose, a notion of locality-preserving labeling function is introduced.

Note that in LIS [**?**], the locality was simple to define. If a variable occurs only in the A-part (B-part) of the problem, it is A-local (B-local). The locality constraints put on a labeling function then simply demand the literals with A-local (B-local) variable to be labeled a (b).

With the introduction of the partial variable assignment in [**?**], the notion of locality was restricted to the subproblem defined by the assignment. That is, a variable is A-local if it occurs only in the $A_{\bar{\pi}}$ part of the problem, where $A_{\bar{\pi}}$ is the set of clauses from A that are not satisfied by π. Similarly for B. Note that here A and B are assumed to be sets of clauses. This restriction (next to some others)

was necessary to ensure that the computed interpolant contained only variables that were common to both parts of the *subproblem*.

We have already shown that if a partial variable assignment is applied before Tseitin's encoding, the resulting subproblem may be smaller. Especially, there may be some variables local with respect to the narrower notion of the subproblem that are *not* local in the subproblem in the previous case. We formalize the subproblem in the following way: Let $\tau(\varphi)$ stand for the set of clauses that is the result of (one-sided) Tseitin's encoding of φ and let $\varphi[\pi]$ denote the formula obtained from φ by substituting the assigned variables by their corresponding values given by the assignment and simplifying the result. Let A be a set of formulas. Then $A_\pi = \{C \in \tau(A) \mid C[\pi] = \top\}$ and $A_{\bar{\pi}} = \tau(A) \setminus A_\pi$. This is the partitioning to satisfied and unsatisfied clauses. We introduce an additional partitioning. Let $A^+_\pi \subseteq \tau(A)$ denote the set of clauses from satisfied subtrees (under π) of the derivation trees of A and let $A^-_\pi = \tau(A) \setminus A^+_\pi$. Notice that $A^-_\pi \subseteq A_{\bar{\pi}}$ and that A^-_π is isomorphic (same up to renaming of the auxiliary variables) to $\tau(A[\pi])$. Given (A, B, π), the pair (A^-_π, B^-_π) defines the subproblem, which is smaller or equal to the subproblem defined by $(A_{\bar{\pi}}, B_{\bar{\pi}})$.

Recall the categorizing of unassigned variables from [**?**]: We say that an unassigned variable x is

- $A_{\bar{\pi}}$-local if $x \in Var(A_{\bar{\pi}})$ and $x \notin Var(B_{\bar{\pi}})$,
- $B_{\bar{\pi}}$-local if $x \notin Var(A_{\bar{\pi}})$ and $x \in Var(B_{\bar{\pi}})$,
- $A_{\bar{\pi}}B_{\bar{\pi}}$-shared if $x \in Var(A_{\bar{\pi}})$ and $x \in Var(B_{\bar{\pi}})$,
- $A_{\bar{\pi}}B_{\bar{\pi}}$-clean if $x \notin Var(A_{\bar{\pi}})$ and $x \notin Var(B_{\bar{\pi}})$.

The same categorization of variables can be done with respect to (A^-_π, B^-_π), but we will need that only later.

We begin with the same constraints on labeling functions as in [**?**] and show that even with the new label g, valid interpolants are still computed. We then propose more restrictions on the labeling functions to reflect the notion of the narrower subproblem and prove a stronger claim for the restricted class of labeling functions.

Definition 2.4 (Locality-preserving labeling [**?**])**.** Let A, B be an unsatisfiable pair of sets of clauses and let ρ be a refutation tree for (A, B). A labeling function Lab for a (A, B, π, ρ), is *locality-preserving* iff $\forall v \in Leaves(\rho), \forall l \in cl(v)$:

1. $Lab(l, v) = d^+ \iff \pi \models l$
2. $var(l)$ is unassigned and $A_{\bar{\pi}}$-local $\implies Lab(l, v) = a$
3. $var(l)$ is unassigned and $B_{\bar{\pi}}$-local $\implies Lab(l, v) = b$
4. $var(l)$ is unassigned and $A_{\bar{\pi}}B_{\bar{\pi}}$-clean $\implies$ l is consistently labeled a or b.

Here "l is consistently labeled a or b" means that either l is labeled a in all leaves (if present) or in all leaves it is labeled b. Formally $(\forall v \in Leaves(R) : (l \in Var(cl(v)) \rightarrow Lab(l, v) = a)) \vee (\forall v \in Leaves(R) : (l \in Var(cl(v)) \rightarrow Lab(l, v) = b))$

Definition 2.5 (Clause filters [**?**])**.** For a clause $\langle\Theta\rangle$, labeling function Lab, vertex v, label c and partial assignment σ we define

- match filter $|$ as $\langle\Theta\rangle|_{c,v,Lab} = \{l \in \Theta \mid Lab(l,v) = c\}$
- upward filter $\restriction$ as $\langle\Theta\rangle\restriction_{c,v,Lab} = \{l \in \Theta \mid c \sqsubseteq Lab(l,v)\}$
- assignment filter $[\sigma]$ as $\langle\Theta\rangle[\sigma] = \{l \in \Theta \mid \ var(l) \text{ is not assigned by } \sigma\}$

Definition 2.6 (Structure-aware Labeled Interpolation System)**.** Let A, B be an unsatisfiable pair of sets of clauses, let π be a partial variable assignment of variables in $A \cup B$, and let ρ be a refutation tree for (A, B) Let Lab be a locality-preserving labeling function for (A, B, π, ρ).

The *Structure-aware Labeled Interpolation System* $SALIS(A, B, \pi, \rho, Lab)$ is defined in Table 2.1

Leaf v:		$\langle\Theta\rangle, [I]$	
$I =$	$\langle\Theta\rangle[\pi]\rVert_{b,v,Lab}$	if $\langle\Theta\rangle \in A_{\bar{\pi}}$	Hyp-$A_{\bar{\pi}}$
$\neg$	$\langle\Theta\rangle[\pi]\rVert_{a,v,Lab}$	if $\langle\Theta\rangle \in B_{\bar{\pi}}$	Hyp-$B_{\bar{\pi}}$
	$\top$	if $\langle\Theta\rangle \in A_\pi \cup B_\pi$	Hyp-A_π, Hyp-B_π
Inner vertex v:		$\dfrac{v_1 : \langle p, \Theta_1\rangle, [I_1] \qquad v_2 : \langle \bar{p}, \Theta_2\rangle, [I_2]}{\langle\Theta_1, \Theta_2\rangle, [I]}$	
$I =$	I_1	if $Lab(p, v_1) = g$	Res-g
	I_2	if $Lab(\bar{p}, v_2) = g$	Res-g
	I_2	if $Lab(p, v_1) = d^+$	Res-d^+
	I_1	if $Lab(\bar{p}, v_2) = d^+$	Res-d^+
	$I_1 \vee I_2$	if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2) = a$	Res-a
	$I_1 \wedge I_2$	if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2) = b$	Res-b
	$(I_1 \vee p) \wedge (I_2 \vee \bar{p})$	if $Lab(p, v_1) \sqcup Lab(\bar{p}, v_2) = ab$	Res-ab

Table 2.1: Partial interpolants in SALIS

Note that we have resolved any possible ambiguity about which rule to apply in a given vertex when discussing propagating labels in Definition 2.2. We have determined the type of resolution for each vertex; for resolution of type Res-g with both pivots labeled g, we have determined which one has priority over the other.

Theorem 2.7 (Weak correctness)**.** *Let (A, B) be an unsatisfiable pair of sets of clauses, let ρ be a refutation tree for $A \cup B$ and let π be a partial variable assignment. Let Lab be a locality-preserving labeling function for (A, B, π, ρ). Then for the formula I, produced by SALIS for (A, B, π, ρ, Lab), the following holds:*

1. $\pi \models A \to I$
2. $\pi \models B \to \neg I$
3. $Var(I) \subseteq Var(A_{\bar{\pi}}) \cap Var(B_{\bar{\pi}})$
4. $Var(I) \cap Var(\pi) = \emptyset$

The proof is based on the correctness of LPAIS (see [**?**]) and we will modify their proof to deal with the new label g. Note that the weak correctness is formulated for input in CNF. We later prove the strong correctness for input as a set of NNF formulas where the interpolant will contain only variables common to the narrower subproblem.

Proof. The system described here is very similar to the original LPAIS, the difference is only the new label g which introduces slight changes to the labeling function and also a new case of resolution in inner nodes of the refutation proof. By going through the proof in [**?**] we see that we can reuse much of the proof, therefore here we only stress the differences.

We will be proving the same invariants as in [**?**], i.e. for every vertex v, its clause Θ and partial interpolant I_v it holds that

- $\pi \models A \wedge \neg\langle\Theta\rangle\restriction_{a,v,Lab} \rightarrow I_v$
- $\pi \models B \wedge \neg\langle\Theta\rangle\restriction_{b,v,Lab} \rightarrow \neg I_v$
- $Var(I_v) \subseteq Var(A_{\bar{\pi}}) \cap Var(B_{\bar{\pi}})$ and $Var(I_v) \cap Var(\pi) = \emptyset$

We proceed by induction in the refutation tree.

Leaves. Since label g cannot occur in the leaves and the rules for partial interpolants in leaves are the same as in LPAIS, the invariants in leaves are proven in exactly the same way as in [**?**]. We do not repeat them here.

Inner vertices. The situation in inner vertices is slightly different than in [**?**]. Most notably, it may happen that $Lab(p, v) \sqsubset Lab(p, v')$ for some v a child of v'. However, this happens only when a literal obtains label g and thus it is possible only in the case of g- or d^+-resolution. In the case of a-, b- and ab-resolution, it still holds that $Lab(p, v') \sqsubseteq Lab(p, v)$ for v a child of v'. As a result, these three cases are completely analogous to their counterparts in [**?**]. We show the case of a-resolution to illustrate the technique and for the other two cases we refer the reader to [**?**]. The case of Res-d^+ resolution is slightly different and the case of Res-g is new, so we cover them here although the reasoning is again very similar.

In the rest of the proof, we assume that v is an inner vertex of the resolution tree with a clause $\langle\Theta_1, \Theta_2\rangle$ and a partial interpolant I such that its parents are vertex v_1 with clause $\langle p, \Theta_1\rangle$ and interpolant I_1, and vertex v_2 with clause $\langle \bar{p}, \Theta_2\rangle$ and interpolant I_2. By the induction hypothesis we know that the following is true:

$$\begin{aligned}
&\pi \models A \wedge \neg\langle p, \Theta_1\rangle\restriction_{a,v_1} \rightarrow I_1\\
&\pi \models B \wedge \neg\langle p, \Theta_1\rangle\restriction_{b,v_1} \rightarrow \neg I_1\\
&\pi \models A \wedge \neg\langle \bar{p}, \Theta_2\rangle\restriction_{a,v_2} \rightarrow I_2\\
&\pi \models B \wedge \neg\langle \bar{p}, \Theta_2\rangle\restriction_{b,v_2} \rightarrow \neg I_2
\end{aligned}$$

Res-a: Assume the resolution is of type Res-a, thus both pivots p and $\bar{p}$ are labeled a and $I = I_1 \vee I_2$.

For the first invariant, reason as follows: Our assumptions are π, A and $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v}$ and we want to derive $I_1 \vee I_2$. From our assumption we can derive $\neg\langle\Theta_1\rangle\restriction_{a,v_1}$, as no literal in Θ_1 that was not filtered out in v_1 can be filtered out in v. This holds because labels cannot decrease (in ordering $\sqsubseteq$) during resolution of type Res-a. In the presence of additional assumption $\bar{p}$ we could derive $\neg\langle p, \Theta_1\rangle\restriction_{a,v_1}$ and thus I_1 by the induction hypothesis. In the same way, we can derive $\neg\langle\Theta_2\rangle\restriction_{a,v_2}$ from our assumptions and with the help of an additional assumption p we could derive $\neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2}$ and consequently I_2 by the induction hypothesis. However, we can always assume $p \vee \bar{p}$ and therefore derive $I_1 \vee I_2$ from our assumptions, as desired. This reasoning is formalized below:

$$\begin{aligned}\pi \models \bar{p} \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies\ & \neg\langle p\rangle\restriction_{a,v_1} \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies \\ & A \wedge \neg\langle p, \Theta_1\rangle\restriction_{a,v_1} \implies\ I_1\end{aligned}$$

$$\begin{aligned}\pi \models p \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies\ & \neg\langle\bar{p}\rangle\restriction_{a,v_2} \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies \\ & A \wedge \neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2} \implies\ I_2\end{aligned}$$

Combining these facts we obtain the required result:

$$\begin{aligned}\pi \models A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies\ & (p \vee \bar{p}) \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} \implies \\ & (p \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v}) \vee (\bar{p} \wedge A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v}) \implies\ I_1 \vee I_2\end{aligned}$$

For the second invariant, we reason as follows: As both pivots are labeled a in parent vertices they are filtered out by the filter $\restriction_b$. Moreover the labels of literals cannot weaken in resolution of type Res-a. Thus it holds that $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} \implies \neg\langle p, \Theta_1\rangle\restriction_{b,v_1}$ and also $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} \implies \neg\langle\bar{p}, \Theta_2\rangle\restriction_{b,v_2}$. Both I_1 and I_2 can be thus obtained from the assumptions π, B and $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v}$ by the induction hypothesis. Formally:

$$\begin{aligned}\pi \models B \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} \implies\ & B \wedge \neg\langle\Theta_1\rangle\restriction_{b,v_1} \implies\ B \wedge \neg\langle p, \Theta_1\rangle\restriction_{b,v_1} \implies\ \neg I_1 \\ \pi \models B \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} \implies\ & B \wedge \neg\langle\Theta_2\rangle\restriction_{b,v_2} \implies\ B \wedge \neg\langle\bar{p}, \Theta_2\rangle\restriction_{b,v_2} \implies\ \neg I_2\end{aligned}$$

Combining these facts we obtain the required result

$$\pi \models B \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} \implies\ \neg I_1 \wedge \neg I_2 \implies\ \neg(I_1 \vee I_2).$$

The third invariant holds trivially because no new variables were added to the interpolant.

Res-b, Res-ab: As said before, the cases of Res-b and Res-ab resolutions are proved using similar reasoning and the proof can be found in [**?**].

Res-d^+: We cannot use exactly the proof as in [?] as here the labels of some literals can decrease on the way from parent to child vertex. However, after examining the rules for labeling, we see this can happen only if these literals came solely from parent whose interpolant is cut off. Suppose that literal p in v_1 is labeled d^+, so $\pi \models p$ and $I = I_2$. The other case is symmetric.

For the first invariant, we reason as follows: Our assumptions are π, A and $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v}$. We want to derive I_2, so by the induction hypothesis we only need to show we can derive $\neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2}$. We show that $\neg\langle\Theta_2\rangle\restriction_{a,v} \implies \neg\langle\Theta_2\rangle\restriction_{a,v_2}$. Suppose $l \in \langle\Theta_2\rangle\restriction_{a,v_2}$. Then according to the labeling rules for d^+-resolution, the supremum rule is used to determine the label of l in v after possibly reconsidering the label of l in v_1 as having label g. Since for any labels c, c' it holds that $c \in \{a, ab, d^+\} \implies c \sqcup c' \in \{a, ab, d^+\}$, we get that $l \in \langle\Theta_2\rangle\restriction_{a,v}$.

Moreover $\pi \models \neg\langle\bar{p}\rangle\restriction_{a,v_2}$ because $\bar{p}$ is either filtered out, in which case the expression evaluates to $\top$, or not, in which case it evaluates to p which we know is satisfied under π. So from our assumptions we can derive the required result. Formally:

$$\begin{aligned}\pi \models A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} &\implies A \wedge \neg\langle\Theta_2\rangle\restriction_{a,v} \implies A \wedge \neg\langle\Theta_2\rangle\restriction_{a,v_2} \\ &\implies A \wedge \neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2} \implies I_2\end{aligned}$$

The second invariant is proven using the exact same reasoning:

$$\begin{aligned}\pi \models B \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{b,v} &\implies B \wedge \neg\langle\Theta_2\rangle\restriction_{b,v} \implies B \wedge \neg\langle\Theta_2\rangle\restriction_{b,v_2} \\ &\implies B \wedge \neg\langle\bar{p}, \Theta_2\rangle\restriction_{b,v_2} \implies \neg I_2\end{aligned}$$

The third invariant holds trivially because no new variables were added to the interpolant.

Res-g: This case is almost identical to Res-d^+. Assume that $\bar{p}$ is labeled g in v_2 and the first branch is cut off, so $I = I_2$. For the first invariant, we reason as follows: Our assumptions are π, A and $\neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v}$. To derive I_2 we only need to derive $\neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2}$ and apply the induction hypothesis. We show that $\neg\langle\Theta_2\rangle\restriction_{a,v} \implies \neg\langle\Theta_2\rangle\restriction_{a,v_2}$. Suppose $l \in \langle\Theta_2\rangle\restriction_{a,v_2}$. Then according to the labeling rules for g-resolution, the supremum rule is used to determine the label of l in v after possibly reconsidering the label of l in v_1 as having label g. Since for any labels c, c' it holds that $c \in \{a, ab, d^+\} \implies c \sqcup c' \in \{a, ab, d^+\}$, we get that $l \in \langle\Theta_2\rangle\restriction_{a,v}$.

Moreover, $\neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2}$ is equivalent to $\neg\langle\Theta_2\rangle\restriction_{a,v_2}$ because $\bar{p}$ is labeled g in v_2, so it is filtered out by the filter. Formally:

$$\begin{aligned}\pi \models A \wedge \neg\langle\Theta_1, \Theta_2\rangle\restriction_{a,v} &\implies A \wedge \neg\langle\Theta_2\rangle\restriction_{a,v} \implies A \wedge \neg\langle\Theta_2\rangle\restriction_{a,v_2} \\ &\implies A \wedge \neg\langle\bar{p}, \Theta_2\rangle\restriction_{a,v_2} \implies I_2\end{aligned}$$

The second invariant is proven using the exact same reasoning and the third invariant holds because no new variables were added to the interpolant.

Conclusion. The final interpolant is the partial interpolant of the sink of the resolution tree, which has an empty clause. The proven invariants for the sink are exactly the conditions ensuring the produced formula is indeed a partial variable interpolant for (A, B, π): $\pi \models A \to I$, $\pi \models B \to \neg I$, $Var(I) \subseteq Var(A_{\bar{\pi}}) \cap Var(B_{\bar{\pi}})$ and $Var(I) \cap Var(\pi) = \emptyset$ □

We have successfully incorporated new label g to LPAIS without losing the correctness of the system. If the input is given as a set of clauses then focusing to the subproblem defined by unsatisfied clauses is the best we can do. However, if the input is given as a formula only in NNF and needs to be encoded to CNF, then the subproblem defined by applying partial assignment directly to the input formula, rather than to the equisatisfiable formula in CNF, can result in a smaller subproblem.

We claim that the label g and its rules were designed such that the resulting interpolant contains only variables common to both parts of the *narrower* subproblem. To show this, we need to examine the relation between derivation tree of the input formula and the resolution tree more closely.

2.3 Relation between derivation and refutation tree

The goal of this section is to show that we can formulate stronger constraints on the variables occurring in the interpolants computed by SALIS. We show that branches starting with clauses from satisfied subtrees of the derivation tree encounter at some vertex a resolution that will cut off their current partial interpolant. We use it to expand the set of variables that cannot occur in the final interpolant.

In this section we assume that an unsatisfiable formula φ in NNF is given and (one-sided) Tseitin's encoding was used to produce an equisatisfiable set of clauses which was then refuted by a refutation tree ρ. Recall also the mapping Θ from edges of derivation tree to the set of clauses.

Observation 2.8. *For an encoding variable e, if a branch starting at leaf with clause Θ_1, which is an image of a parent (child) edge of e, is present in ρ-tree, then in ρ there is also a branch starting at leaf with a clause that is an image of a child (parent) edge of e such that these branches meet on a vertex v with pivot variable e. Moreover this is the first vertex for both branches where the literal with variable e that is present in the clause of the leaf is resolved away.*

Proof. Follows from Observation 1.6. □

Recall that $\mathbf{e}^{\uparrow}(n)$ stands for the parent edge of node n in the derivation tree. The clause corresponding to this edge is denoted as $\Theta(\mathbf{e}^{\uparrow}(n))$. For the sake of brevity we will write Θ^n instead of $\Theta(\mathbf{e}^{\uparrow}(n))$.

Definition 2.9 (Cutting resolution)**.** Let β be a branch in the refutation tree, and let v be a vertex on β where β meets some other branch β'. We say that the current partial interpolant of the branch β is cut off at vertex v if one of the following is true:

(i) the resolution is of type Res-d^+ and the pivot from branch β is labeled d^+

(ii) the resolution is of type Res-g, the pivot from β' is labeled g and the pivot from β is not labeled g

(iii) the resolution is of type Res-g, both pivots have label g, but the partial interpolant in v is chosen to be the interpolant from the branch β'

Lemma 2.10. *Assume l is a satisfied literal and Θ^l is the clause corresponding to its parent edge. Then every branch starting at leaf with clause Θ^l encounters a resolution where its current partial interpolant is cut off.*

Proof. Satisfied literal l appears positively in Θ^l (meaning that l, not $\bar{l}$, is present in Θ^l), so it is labeled d^+ in any leaf with Θ^l. This literal has to appear as a pivot at some vertex v along $\beta(\Theta^l)$, thus the resolution at v satisfies condition (i) of Definition 2.9 and the partial interpolant of Θ^l is cut off at v. □

Lemma 2.11. *Let n be a satisfied node of the derivation tree and let Θ^n be the clause corresponding to its parent edge. Then every branch starting at leaf with clause Θ^n encounters a resolution where its current partial interpolant is cut off.*

Proof. Let $d(n)$ denote the maximal distance from n to a leaf in the subtree of n. We prove the claim by induction over $d(n)$.

If $d(n) = 0$ then n is a satisfied leaf and the claim for this case has been proved in Lemma 2.10.

If $d(n) > 0$, then n is an inner node. Denote its children as c_1 and c_2. The proof is slightly different depending on whether n stands for conjunction or disjunction; however, for now we can cover both cases. Without loss of generality, assume c_1 is also a satisfied node (at least one of the children has to be). By Observation 2.8, there is a branch starting at leaf with clause Θ^{c_1} such that $\beta(\Theta^n)$ meets $\beta(\Theta^{c_1})$ at vertex which is the first vertex on $\beta(\Theta^n)$ with n as the pivot variable. Denote this vertex as w. The situation is depicted in Fig. 2.2.

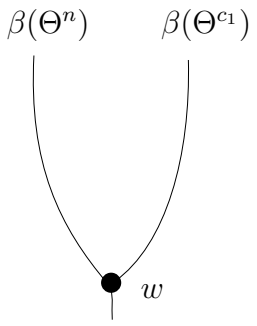

Figure 2.2: Two branches in the refutation tree

Since $d(n) > d(c_1)$, by induction hypothesis there is a vertex v on $\beta(\Theta^{c_1})$ such that the current partial interpolant of $\beta(\Theta^{c_1})$ is cut off at v. We discuss the three possibilities how v and w can be ordered on branch $\beta(\Theta^{c_1})$.

- If $v > w$ then the current partial interpolant of $\beta(\Theta^n)$ is also cut off at v.
- If $v = w$ then the cutting is not due to a resolution of type Res-d^+ because n is an inner node, therefore an encoding variable, and the assignment assigns only the original variables. As a result, the resolution is of type Res-g such that the pivot from branch $\beta(\Theta^n)$ has label g. However, the pivot is n, which is present in the clause of all vertices on the branch up to this vertex. Since it did not have label g at the leaf, it had to obtain the label somewhere

along the branch $\beta(\Theta^n)$. But the only possible way a literal can obtain label g along a branch is if at some vertex at that branch, the current partial interpolant of the branch is cut off according to Definition 2.9. This is a direct consequence of rules for propagating labels in inner vertices (see Definition 2.2). As a consequence, we have proved that the current partial interpolant of branch $\beta(\Theta^n)$ is cut off at some vertex.

- If $v < w$ then the situation is a little more complicated. As Θ^{c_1} is a clause corresponding to the child edge of n, it contains n negatively, i.e. $\bar{n}$ is present in Θ^{c_1}. We also know that $\bar{n}$ is present in the clause along $\beta(\Theta^{c_1})$ until vertex w where it is the pivot of resolution. Therefore, it is present in the clause of $\beta(\Theta^{c_1})$ at vertex v, when the current partial interpolant of $\beta(\Theta^{c_1})$ is cut off. At this point, $\bar{n}$ may obtain label g according to the rules of propagating labels in case Res-d^+ or Res-g. Suppose that $\bar{n}$ indeed successfully obtained label g and kept the label until vertex w. Then at w either the pivot from $\beta(\Theta^n)$ does not have label g, which means that the current partial interpolant of $\beta(\Theta^n)$ is cut off at w, or the pivot from $\beta(\Theta^n)$, too, has label g. But as we have shown in the previous case, in such situation the current partial interpolant of $\beta(\Theta^n)$ had to be cut off somewhere before w.

  However, it may happen that $\bar{n}$ does not obtain g during resolution at v or that it loses it somewhere between v and w after successfully obtaining it at v. According to the labeling rules, this is possible only if $\bar{n}$ is also present in the clause from the other branch and it does not have label g there. Suppose that this indeed happened, so there is some branch β' such that $v \leq u = \beta' \wedge \beta(\Theta^{c_1}) < w$ and at vertex u $\bar{n}$ is also present in the clause from β' and it does not have label g. Moreover, we can assume that β' is such a branch that $\bar{n}$ is already present at its leaf and is not resolved away until w. As a consequence, the clause at the leaf of β' corresponds to one of the child edges of n. We show that the induction hypothesis can be again applied to β'. Either $\beta' = \beta(\Theta^{c_1})$, in which case we know induction hypothesis can be applied, or $\beta' = \beta(\Theta^{c_2})$. If n represents disjunction then $\Theta^{c_1} = \Theta^{c_2}$, so again the induction hypothesis can be applied. If n represents conjunction then also c_2 has to be a satisfied node and as $d(n) > d(c_2)$, the induction hypothesis can be applied to $\beta(\Theta^{c_2})$. In either case, the induction hypothesis can be applied to get a vertex v' on branch β' where the current partial interpolant of β' is cut off. However, this is the same situation as at the beginning with the first $\beta(\Theta^{c_1})$. We again can discuss the relative order of vertices v' and w on branch β' to conclude that either we can found a vertex where the current partial interpolant of $\beta(\Theta^n)$ is cut off, or yet another branch, β'', must exist that is in the same relation to β' as is β' to $\beta(\Theta^{c_1})$. The same reasoning is applied again to β'' if necessary. As the number of branches is finite, there is some last branch in this sequence which gives us the vertex where the current partial interpolant of $\beta(\Theta^n)$ is cut off.

□

We now move from satisfied nodes to nodes in the satisfied subtrees that are not themselves satisfied.

Lemma 2.12. *Assume n is a child of a satisfied node p that is not itself satisfied, and Θ^n is the clause corresponding to its parent edge. Then every branch starting at leaf with clause Θ^n encounters a resolution where its current partial interpolant is cut off.*

Proof. If p is a satisfied node, but one of its children is not satisfied, then p has to represent disjunction. As a result, $\Theta^n = \Theta^{n'}$ where n' is the other, satisfied, child of p. Since Lemma 2.11 applies to n', we get that every branch starting at a leaf with clause Θ^n, which is a branch starting at a leaf with clause $\Theta^{n'}$, encounters a resolution where its current partial interpolant is cut off. □

Lemma 2.13. *Assume n is a node in the satisfied subtree that is not itself satisfied, and Θ^n is the clause corresponding to its parent edge. Then every branch starting at leaf with clause Θ^n encounters a resolution where its current partial interpolant is cut off.*

Proof. The reasoning is almost the same as in the proof of Lemma 2.11. Let $d_s(n)$ denote the distance from n to its closest satisfied ancestor. We prove the claim by induction over $d_s(n)$. If $d_s(n) = 1$ then the claim has been proved in Lemma 2.12. If $d_s(n) > 1$ then denote the parent of n as p. Note that p is not satisfied, but it is a node in a satisfied subtree and $d_s(p) < d_s(n)$, so induction hypothesis can be applied to p. Consider a branch in the refutation tree starting at leaf with clause Θ^n. Then by Observation 2.8, there is a branch starting at leaf with clause Θ^p such that $\beta(\Theta^p)$ meets $\beta(\Theta^n)$ at the first vertex on $\beta(\Theta^n)$ with p as the pivot variable. Denote this vertex as w. By induction hypothesis, there is a vertex v on $\beta(\Theta^p)$ such that the current partial interpolant of $\beta(\Theta^p)$ is cut off at v. Consider the three possible orderings of vertices v and w on $\beta(\Theta^p)$.

- If $v > w$ then the current partial interpolant of $\beta(\Theta^n)$ is also cut off at v.
- If $v = w$ then the cutting is not due to a resolution of type Res-d^+ because p is an inner node, therefore an encoding variable, and the assignment assigns only the original variables. As a result, the resolution is of type Res-g such that the pivot from branch $\beta(\Theta^n)$ has label g. However, the pivot is n, which is present in the clause of all vertices on the branch up to this vertex. Since it did not have label g at the leaf, it had to obtain the label somewhere along the branch $\beta(\Theta^n)$. But the only possible way a literal can obtain label g along a branch is if at some vertex at that branch, the current partial interpolant of the branch is cut off according to Definition 2.9. This is a direct consequence of rules for propagating labels in inner vertices (see Definition 2.2). As a consequence, we have proved that the current partial interpolant of branch $\beta(\Theta^n)$ is cut off at some vertex.
- If $v < w$ then take a look at literal p at branch $\beta(\Theta^p)$. This literal is present in the clause of all vertices on this branch up to w. Therefore, it is present in the clause of vertex immediately before v. We know that the current partial interpolant of $\beta(\Theta^p)$ is cut off at v, thus p may obtain g at his point, according to the rules of propagating labels. Suppose that p indeed obtained label g at v and kept it until w. Then either the current partial interpolant of $\beta(\Theta^n)$ is cut off at w because of the resolution of type

Res-g or at w the pivot from branch $\beta(\Theta^n)$ also has label g. But then the current partial interpolant of $\beta(\Theta^n)$ had to be cut off already somewhere before w, as we have shown in the previous case.

Now, if p did not obtained label g at v or lost it somewhere between v and w, then it had to happen because some other branch β' with p in its clause met with $\beta(\Theta^p)$ at vertex u ($v \leq u < w$) and p did not have label g when they met. Moreover, we may assume that β' is the branch which has literal p in the clause of all its vertices from the leaf to u. However, the only clause where p appears positively is Θ^p, thus branch β' starts at leaf with clause Θ^p. As a result, induction hypothesis can be applied to β'. The rest of the proof is the same as for Lemma 2.11: we either find a vertex where the current partial interpolant of $\beta(\Theta^n)$ is cut off, or yet another branch on which the induction hypothesis can be applied must exist in the refutation tree. This is repeated until no new branch that would prevent the existence of the cut-off vertex can be found (which must happen at some point). Therefore, even in this case the vertex where the current partial interpolant of $\beta(\Theta^n)$ is cut off has been found.

□

We now aim to use Lemma 2.13 to restrict the set of variables that can occur in the computed interpolant. Suppose that (A, B) is an unsatisfiable pair of NNF formulas.

Let $\hat{A} = \tau(A)$ and $\hat{B} = \tau(B)$, let σ be a partial variable assignment and let ρ be the refutation tree for $(\hat{A}, \hat{B})$. Recall the two divisions of $\hat{A}$ (and $\hat{B}$):

- A_σ and $A_{\bar{\sigma}}$ denotes the satisfied and unsatisfied clauses, respectively.
- $\hat{A}^+_\sigma$ denotes the clauses corresponding to edges from satisfied subtrees of the derivation tree of A and $\hat{A}^-_\sigma = \hat{A} \setminus \hat{A}^+_\sigma$

Recall as well the categorizing of unassigned variables as $\hat{A}_{\bar{\sigma}}$-local, $\hat{B}_{\bar{\sigma}}$-local, $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$-shared and $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$-clean. In the same way the unassigned variables can also be categorized as $\hat{A}^-_\sigma$-local, $\hat{B}^-_\sigma$-local, $\hat{A}^-_\sigma\hat{B}^-_\sigma$-shared and $\hat{A}^-_\sigma\hat{B}^-_\sigma$-clean. We want to show that SALIS can produce such interpolants that are focused on the smaller subproblem, i.e. only $\hat{A}^-_\sigma\hat{B}^-_\sigma$-shared variables appear in these interpolants. Lemma 2.13 is an important tool in achieving this goal, however it is not sufficient on its own. The constraints on the labeling function needs to strengthen a little. First notice the following relations between the two divisions of clauses and their corresponding categorizing of variables:

Observation 2.14.

(i) $\hat{A}_\sigma \subseteq \hat{A}^+_\sigma$, $\hat{B}_\sigma \subseteq \hat{B}^+_\sigma$

(ii) $\hat{A}^-_\sigma \subseteq \hat{A}_{\bar{\sigma}}$, $\hat{B}^-_\sigma \subseteq \hat{B}_{\bar{\sigma}}$

(iii) If a variable is $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$*-clean then it is also* $\hat{A}^-_\sigma\hat{B}^-_\sigma$*-clean.*

(iv) If a variable is $\hat{A}_{\bar{\sigma}}$*-local then it is either* $\hat{A}^-_\sigma$*-local or* $\hat{A}^-_\sigma\hat{B}^-_\sigma$*-clean.*

(v) If a variable is $\hat{B}_{\bar{\sigma}}$*-local then it is either* $\hat{B}^-_\sigma$*-local or* $\hat{A}^-_\sigma\hat{B}^-_\sigma$*-clean.*

(vi) If a variable is $\hat{A}^-_\sigma$-local then it is either $\hat{A}_{\bar{\sigma}}$-local or $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$-shared.

(vii) If a variable is $\hat{B}^-_\sigma$-local then it is either $\hat{B}_{\bar{\sigma}}$-local or $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$-shared.

Based on these observations, we can correctly strengthen the conditions on labeling functions:

Definition 2.15 (Structure-aware locality-preserving labeling)**.** Assume (A, B) is an unsatisfiable pair of NNF formulas, $\hat{A} = \tau(A)$, $\hat{B} = \tau(B)$, σ is a partial assignment of variables in $Var(A) \cup Var(B)$, ρ is a refutation tree for $(\hat{A}, \hat{B})$, and Lab is a labeling function for ρ. We say that a labeling function Lab is *structure-aware locality-preserving* for ρ and σ if it is locality-preserving and moreover $\forall v \in Leaves(\rho), \forall l \in cl(v)$

1. $var(l)$ is unassigned and $\hat{A}^-_\sigma$-local $\implies Lab(l, v) = a$
2. $var(l)$ is unassigned and $\hat{B}^-_\sigma$-local $\implies Lab(l, v) = b$
3. $var(l)$ is unassigned and $\hat{A}^-_\sigma\hat{B}^-_\sigma$-clean $\implies$ l is consistently labeled a or b

Note that by Observation 2.14 the additional constraints from Definition 2.15 are not in conflict with the original constraints from Definition 2.4.

Theorem 2.16 (Strong correctness)**.** *Let (A, B) be an unsatisfiable pair of NNF formulas, $\hat{A} = \tau(A)$, $\hat{B} = \tau(B)$, ρ be a refutation tree for $(\hat{A}, \hat{B})$, σ be a partial assignment of variables from $Var(A) \cup Var(B)$. If Lab is a structure-aware locality-preserving labeling function for ρ and σ then formula I produced by $SALIS(A, B, \sigma, \rho, Lab)$ satisfies the following conditions:*

1. $\sigma \models A \rightarrow I$
2. $\sigma \models B \rightarrow \neg I$
3. $Var(I) \subseteq Var(\hat{A}^-_\sigma) \cap Var(\hat{B}^-_\sigma)$
4. $Var(I) \cap Var(\sigma) = \emptyset$

Proof. Since every structure-aware locality-preserving labeling function is also locality-preserving, Theorem 2.7 applies to I. As a result, condition 4 holds for I.

Consider condition 1. From Theorem 2.7 we only get that $\sigma \models \hat{A} \rightarrow I$. We want to prove that $\sigma \models A \rightarrow I$. It is sufficient to show that for every complete assignment σ' such that σ' extends σ and $\sigma' \models A$ it also holds that $\sigma' \models I$. Let σ' be such an assignment. From the properties of Tseitin's encoding, we know there exists an assignment σ'' such that $\sigma'' \models \hat{A}$ and σ' and σ'' agree on variables of A. Clearly σ'' extends σ, therefore $\sigma'' \models I$. However I does not contain any variables from $Var(\hat{A}) \setminus Var(A)$, because these are encoding variables and are thus unique to $\hat{A}$. As a result $\sigma' \models A$.

Condition 2 is proved in the same way as condition 1 only using B instead of A.

Now consider condition 3. First recall how a variable can appear in the final interpolant. It can either be introduced to the partial interpolant after resolution of type Res-ab (and survive till sink) or it can be introduced to the partial

interpolant in the leaf (and survive till sink). A variable is introduced to the interpolant in the leaf if the clause of the leaf is from $\hat{A}$ and the literal with the corresponding variable has label b, or the clause is from $\hat{B}$ and the literal has label a.

Now, from Theorem 2.7 we already know that $Var(I) \subseteq Var(\hat{A}_{\bar{\sigma}}) \cap Var(\hat{B}_{\bar{\sigma}})$. Consider x to be a variable such that $x \in Var(\hat{A}_{\bar{\sigma}}) \cap Var(\hat{B}_{\bar{\sigma}})$ but $x \notin Var(\hat{A}^{-}_{\sigma}) \cap Var(\hat{B}^{-}_{\sigma})$. We need to prove that x will not be present in the produced interpolant. Note that from the assumption it follows that x is not an encoding variable ($x \in Var(A) \cup Var(B)$) and x is not $\hat{A}^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-shared. We analyze the possible category of x.

- If x is $\hat{A}^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-clean then by the constraint from Definition 2.15, x is consistently labeled a or b, therefore there is no resolution with pivot x of type Res-ab in the refutation tree. Consequently, x cannot be introduced to a partial interpolant in an inner vertex.

  Since x is $\hat{A}^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-clean, it appears only in clauses from satisfied subtrees of the derivation tree. Moreover, x appears only in clauses corresponding to parent edges of leaves of the derivation tree, because x is not an encoding variable. Suppose that x is introduced to the partial interpolant of some leaf in the refutation tree. Then the branch starting in this leaf satisfies the assumption of Lemma 2.13. As a result, the current partial interpolant of this branch is cut off at some point, hence the variable x on this branch does not survive to the final interpolant. However, this holds for all branches where x was introduced to the partial interpolant of the leaf. Since x cannot be introduced in inner vertices, as we have shown earlier, x cannot appear in the final interpolant.

- If x is $\hat{A}^{-}_{\sigma}$-local then by Definition 2.15 all literals with variable x are labeled a in the leaves. As a consequence, there is no resolution with pivot x of type Res-ab in the refutation tree, so x cannot be introduced to a partial interpolant in an inner vertex. Let v be a leaf in the resolution tree such that its clause Θ_v contains a literal with variable x. If Θ_v is a satisfied clause, then the partial interpolant of v does not contain x. If $\Theta_v \in \hat{A}$ then again, the partial interpolant of v does not contain x because x is labeled a in v. If $\Theta_v \in \hat{B}$ then Θ_v corresponds to an edge in a satisfied subtree, because x is $\hat{A}^{-}_{\sigma}$-local. As x is not an encoding variable, Θ_v corresponds to a parent edge of some leaf in the derivation tree corresponding to literal x or $\neg x$. As x is not assigned, this is not a satisfied leaf. As a consequence, Lemma 2.13 applies to the branch $\beta(\Theta_v)$ starting at the leaf v and its current partial interpolant is cut off at some vertex on $\beta(\Theta_v)$, hence the variable x on this branch does not survive to the final interpolant.

  We have shown that if x is present in the partial interpolant of some leaf in the refutation tree then the interpolant of this branch is cut off at some vertex. Moreover x cannot be introduced to the interpolant in an inner vertex. As a result, x cannot appear in the final interpolant.

- If x is $\hat{B}^{-}_{\sigma}$-local then the reasoning is symmetric to the previous case (only switching all $\hat{A}$s and as for $\hat{B}$s and bs and vice versa). Thus neither in this case can x appear in the final interpolant.

To conclude, we have already known that no variable that is not $\hat{A}_{\bar{\sigma}}\hat{B}_{\bar{\sigma}}$-shared can appear in the final interpolant. We have then extended this result to all variables that are not $\hat{A}^-_{\sigma}\hat{B}^-_{\sigma}$-shared. As a result, condition 3 holds for the produced interpolant I. □

3. Path interpolation property

Several verification techniques are based not only on computing interpolants for a given input, but actually on computing interpolation sequences. For example, this is an essential part of the UFO algorithm presented in [**?**]. In [**?**], the authors have shown that the framework of LIS [**?**] can be employed to compute interpolation sequences given a gradually weakening sequence of labeling functions. In [**?**], the authors successfully retained this property even in the presence of different partial variable assignments, given some additional conditions on the assignments.

Here, we examine SALIS with respect to the path interpolation property. We identify a potential problems in our system and present additional constraints that deal with such problems.

Most of this chapter is based on the proof of the path interpolation property presented in [**?**]. However, there are a few places where the new label g and cutting branches due to these labels causes new obstacles to arise.

3.1 Modified version of SALIS

We first show we need to modify the interpolation system presented in Chapter 2.

Recall that in LIS [**?**], a total order $\preceq$ of the set of labels $\{a, b, ab\}$ is defined as $b \preceq ab \preceq a$. This ordering reflects the logical strength of corresponding possibilities of computing a partial interpolant for an inner vertex of the resolution tree:

$$I_1 \wedge I_2 \implies (p \vee I_1) \wedge (\bar{p} \vee I_2) \implies I_1 \vee I_2$$

Moreover, it can be used to compare labeling functions by strength, and it has been also shown in [**?**] that stronger labeling functions produce stronger interpolants, in terms of logical strength. In [**?**] authors successfully incorporated their new label d^+ into this ordering as $d^+ \approx ab$, meaning it is of the same strength as ab, $ab \preceq d^+$ and $d^+ \preceq ab$. This is supported by the fact that if the pivot p is satisfied under a partial assignment of σ then $\sigma \models I_2 \iff (p \vee I_1) \wedge (\bar{p} \vee I_2)$.

Our new label g presents a problem for the totality of such pre-order. Label g cuts off partial interpolants similarly to label d^+, but the variables with label g are out of the scope of the partial variable assignment. If a pivot is labeled g then $I = I_1$ is the partial interpolant in the child vertex. However, there is no guarantee that $\sigma \models I_1 \iff (p \vee I_1) \wedge (\bar{p} \vee I_2)$ in this case, so label g is incomparable with ab and also with d^+. Fortunately, it can still be compared with labels a and b as $b \preceq g \preceq a$.

Another problem with the strength of g is that it breaks the nice property that if one labeling is stronger than another at leaves then it is stronger everywhere. To restore this property, additional constraints on assignments and label g are needed.

We deal with these problems one by one. First, we propose a restriction on labeling functions to gain some control over label g. The necessity of this restriction should be clear later from the proofs. With the path interpolation property as our goal, we assume that the input is now a set of NNF formulas. Moreover, we assume that the encoding step is applied separately to each formula from the input, resulting in a set of derivation trees.

Definition 3.1 (Labeling with restricted propagation)**.** We say that a labeling function has *restricted propagation*, if it uses the following conditions for propagating labels in case of resolution of types Res-d^+ and Res-g instead of the original ones (see Definition 2.2):

- Res-d^+: Assume the pivot in the first predecessor has label d^+ (the second case is symmetric). First, all literals from v_1, *such that their variable is an encoding variable from the same derivation tree as the satisfied pivot*, are considered as having label g, and then the supremum rule is used to determine the labels in v.

- Res-g: Assume the pivot in the first predecessor has label g (the second case is symmetric). First, all literals from v_2, *such that their variable is an encoding variable from the same derivation tree as the pivot*, are considered as having label g, and then the supremum rule is used to determine the labels in v.

If in case Res-g both pivots have label g, we resolve the situation as before. Notice that in case Res-g the pivot has to be an encoding variable, so both pivots come from the same derivation tree.

The effect of this modification is that the branches and their current partial interpolants are not cut off as often as before, but even when using labeling functions with restricted propagation, the system is still strongly correct.

Theorem 3.2 (Strong correctness for labeling functions with restricted propagation)**.** *Assume (A, B) is an unsatisfiable pair of sets of NNF formulas, $\hat{A} = \tau(A)$, $\hat{B} = \tau(B)$, ρ is a refutation tree for $(\hat{A}, \hat{B})$, σ is a partial assignment of variables from $Var(A) \cup Var(B)$. If Lab is a structure-aware locality-preserving labeling function with* restricted propagation *for (A, B, σ, ρ) then the formula I produced by SALIS for $(A, B, \sigma, \rho, Lab)$ satisfies the following conditions:*

1. $\sigma \models A \rightarrow I$
2. $\sigma \models B \rightarrow \neg I$
3. $Var(I) \subseteq Var(\hat{A}^-_\sigma) \cap Var(\hat{B}^-_\sigma)$
4. $Var(I) \cap Var(\sigma) = \emptyset$

Proof. It suffices to go through the proofs of weak and strong correctness for unrestricted labeling function and check that the restriction does not break the proofs. Note that in the proof of strong correctness it is important that in the satisfied subtree label g can be propagated from a satisfied leaf to all nodes in this satisfied subtree. Since the propagation is done using encoding variables only, a restricted labeling function still achieves this goal. □

For the rest of this chapter, we work only with the restricted labeling functions.

3.2 Strength

As we have mentioned before, in the previous systems, to compare two labeling functions strength-wise, it was enough to compare the labels of literals in the leaves of the refutation tree. In our case, additional constraints on assignments are needed if we want to have this property. First, we show what these constraints are, and then we turn to the strength of the computed interpolant. Similarly to LIS and LPAIS, we show that if a stronger labeling is used, then it produces stronger interpolant.

Lemma 3.3. *Let (A, B) and (A', B') be two divisions of the same unsatisfiable set of NNF formulas. Let ρ be the refutation tree for $\hat{A} \cup \hat{B}$ and let σ and σ' be partial variable assignments. Let Lab be a locality-preserving labeling function with restricted propagation for (A, B, σ, ρ) and let Lab' be a locality-preserving labeling function with restricted propagation for (A', B', σ', ρ). If $Lab \preceq_L Lab'$ and*

1. *if $Lab(l, v) = g$ and $Lab'(l, v) \neq g$ then $Lab'(l, v) = a$,*
2. *if $Lab'(l, v) = g$ and $Lab(l, v) \neq g$ then $Lab(l, v) = b$,*

then $Lab \preceq Lab'$.

Proof. We proceed by structural induction. We are assuming the claim holds for the leaves, so the base case is done. For the induction step, let v be an inner vertex such that v_1 is its positive predecessor with clause $\langle p, \Theta_1 \rangle$ and v_2 is its negative predecessor with clause $\langle \bar{p}, \Theta_2 \rangle$. From the induction hypothesis we have that $\forall l \in \Theta_1$ $Lab(l, v_1) \preceq Lab'(l, v_1)$ and $\forall l \in \Theta_2$ $Lab(l, v_2) \preceq Lab'(l, v_2)$. We want to show that $\forall l \in \Theta_1 \cup \Theta_2$ $Lab(l, v) \preceq Lab'(l, v)$. We use the proof by case analysis, depending on the value of $Lab(l, v)$:

$Lab(l, v) = b$**:** Since b is the strongest label, the claim holds trivially.

$Lab(l, v) = ab$**:** Suppose the claim does not hold. Then, it must be the case that $Lab'(l, v) = b$ or $Lab'(l, v) = g$. However, $Lab'(l, v) = g$ entails that l is an encoding variable, because only encoding variables can be labeled g if restricted propagation is used. However, an encoding variable is always local to one part of the problem, so it can never have label ab, contradicting the original assumption that $Lab(l, v) = ab$. Thus, suppose that $Lab'(l, v) = b$. Then for one of the predecessors, say v_1, it must hold that the label of l is also b, $Lab'(l, v_1) = b$. From the induction hypothesis, we get that $Lab(l, v_1) = b$. But then $Lab(l, v_2) = a$ which entails $Lab'(l, v_2) = a$, leading to contradiction as that would result in $Lab'(l, v) = ab$.

$Lab(l, v) = a$**:** If l has label a in v then it has to have label a also in one of the predecessors. Without loss of generality let it be v_1. So $Lab(l, v_1) = a$ and from the induction hypothesis we get that also $Lab'(l, v_1) = a$. Let us consider the possible labels of l in v_2.

If $Lab(l, v_2) = a$ or $Lab(l, v_2) = g$ or $Lab(l, v_2) = \bot$, then also $Lab'(l, v_2) = \bot$ or $Lab'(l, v_2) = a$ or $Lab'(l, v_2) = g$. If the label in v_1 for Lab' is not reconsidered

as g before the supremum rule is applied, then $Lab'(l, v) = a$ follows. However, if it is reconsidered then it may happen that $Lab'(l, v) = g$, but this violates our second explicit assumption.

The other labels, b, ab and d^+, cannot be assigned to l in v_2, because to satisfy that $Lab(l, v) = a$ the label would have to be changed to g before applying the supremum rule and this can happen only to the encoding variables. However, from the fact that $Lab(l, v_1) = a$, this encoding variable has to be local to the A-part of the problem, so it can never have label b, ab nor d^+.

$Lab(l, v) = d^+$**:** We can argue as in the case $Lab(l, v) = ab$. If the claim does not hold then $Lab'(l, v) = b$ or $Lab'(l, v) = g$. However, $Lab'(l, v) \neq g$ because that would entail it is an encoding variable while $Lab(l, v) = d^+$ entails it is not an encoding variable, because encoding variables are not assigned. If $Lab'(l, v) = b$ then $Lab'(l, v') = b$ for $v' = v_1$ or $v' = v_2$. However, this would require that $Lab(l, v') = b$ which cannot happen, because l is satisfied under σ.

$Lab(l, v) = g$**:** By the first explicit assumption, it holds that either $Lab'(l, v) = g$ or $Lab'(l, v) = a$, so the claim is satisfied. □

We have shown that under certain conditions, if two labeling functions are comparable at leaves, they are comparable everywhere. However, the conditions as stated are not easily verifiable for given labeling functions. Below we show we can guarantee these conditions given another set of conditions which are much easier to verify.

Lemma 3.4. *Let (A, B) and (A', B') be two divisions of the same unsatisfiable set of NNF formulas. Let ρ be the refutation tree for $\hat{A} \cup \hat{B}$ and let σ and σ' be partial variable assignments. Let Lab be a locality-preserving labeling function with restricted propagation for (A, B, σ, ρ) and let Lab' be a locality-preserving labeling function with restricted propagation for (A', B', σ', ρ). If $Lab \preceq_L Lab'$ and the following are true:*

1. *all occurrences of literals satisfied by σ and not satisfied by σ' lie in the derivation trees of A',*
2. *all occurrences of literals satisfied by σ' and not satisfied by σ lie in the derivation trees of B,*
3. *if Lab and Lab' both cut off a branch at some vertex, they agree on which branch is cut off,*

then $Lab \preceq Lab'$.

Proof. We show by contradiction that the conditions from Lemma 3.3 are satisfied.

Suppose that the first condition is not satisfied, i.e. there exists a vertex v and a literal l such that $Lab(l, v) = g$ and $Lab'(l, v) = b$. It follows that the variable of l is an encoding variable from a derivation tree from B'. Consider vertex v' where l obtained label g in case of Lab but did not obtain it in case of Lab'. As a result, a branch has been cut off at v' and the pivot responsible for the cutting (the one

with label d^+ or g) is from the same derivation tree as l. On the other hand, in case of Lab' literal l did not obtain label g which means that the same branch was not cut off. Since Lab' and Lab cannot disagree on which branch is cut off at given vertex by our assumption, it follows that the pivot labeled d^+ or g by Lab does not have the same label assigned by Lab'. This means that either there is an occurrence of a σ-satisfied but not σ'-satisfied literal in a derivation tree from B', or there is another literal satisfying the same conditions as l, because its variable is also an encoding variable from a derivation tree from B'. The first variant contradicts the first assumption of this lemma. In the second variant, the same reasoning can be applied. In this way we would eventually have to encounter a literal for which this first variant must hold, but this contradicts our original assumptions.

In case the second assumption is not satisfied we can use symmetric reasoning to obtain a contradiction with the second assumption of this lemma. □

Note that these assumptions are more comprehensible and easier to guard, since the first two are just slightly stronger conditions then the ones needed to ensure that $Lab \preceq_L Lab'$ in the first place. The third assumption can still be checked only by computing the labels in inner vertices, but can again be achieved simply if we surrender some generality. Moreover, we shall encounter this assumption again in the proof of path interpolation, so it is not unreasonable to require it.

Corollary 3.5. *Let (A, B) and (A', B') be two divisions of the same unsatisfiable set of NNF formulas. Let ρ be the refutation tree for $\hat{A} \cup \hat{B}$ and let σ be a partial variable assignment. Let Lab and Lab′ be locality-preserving labeling functions with restricted propagation for (A, B, σ, ρ) and (A', B', σ, ρ), respectively. If $Lab \preceq_L Lab'$ then $Lab \preceq Lab'$.*

Proof. Since the same assignment is used, the first two conditions of Lemma 3.4 are trivially satisfied. Moreover, if the same assignment is used, then Lab and Lab' agree on all d^+ and on all g labels, so they cannot disagree on which branch is cut off. □

We continue with the proofs that stronger labeling functions yield stronger interpolants in SALIS. First, we consider labeling functions under the same assignment, later we allow different assignments to be used.

Definition 3.6 (Weakened-labeling filter)**.** Let Lab and Lab' be labeling functions for the same refutation tree, let v be a vertex of the tree and Θ be a clause. We define a weakened-labeling filter $\downarrow_v^{Lab,Lab'}$ as follows:

$$\langle\Theta\rangle\downarrow_v^{Lab,Lab'} = \{l \in \langle\Theta\rangle \mid Lab(l) \neq a \wedge Lab'(l) \neq b \wedge Lab(l) \neq g \wedge Lab'(l) \neq g\}$$

or equivalently:

$$\langle\Theta\rangle \setminus \langle\Theta\rangle\downarrow_v^{Lab,Lab'} = \{l \in \langle\Theta\rangle \mid Lab(l) = a \vee Lab'(l) = b \vee Lab(l) = g \vee Lab'(l) = g\}.$$

Lemma 3.7. *Let v be a vertex in the refutation tree with parents v_1 with clause $\langle\Theta_1, p\rangle$ and v_2 with clause $\langle\Theta_2, \bar{p}\rangle$ and p be the pivot variable so the clause in v is $\langle\Theta_1, \Theta_2\rangle$. Let Lab and Lab′ be labeling functions for the refutation tree. If the*

resolution at v is such that the supremum rule is used to propagate labels in both labeling functions, then it holds that

$$\neg\langle\Theta_1\rangle{\downarrow_v^{Lab,Lab'}} \to \neg\langle\Theta_1\rangle{\downarrow_{v_1}^{Lab,Lab'}} \text{ and } \neg\langle\Theta_2\rangle{\downarrow_v^{Lab,Lab'}} \to \neg\langle\Theta_2\rangle{\downarrow_{v_2}^{Lab,Lab'}}$$

Proof. Assume l is a literal not filtered out by the filter in v_1, $l \in \langle\Theta_1\rangle{\downarrow_{v_1}^{Lab,Lab'}}$. Then $Lab(v_1, l)$ is neither a nor g. According to the assumption the supremum rule is used to label l in v, so l cannot have label a or g assigned in v by Lab (consult Fig. 2.1). The same argument shows that if $Lab'(v_1, l)$ is not b nor g then also $Lab'(v, l)$ is not b nor g. As a result $l \in \langle\Theta_1\rangle{\downarrow_v^{Lab,Lab'}}$. The case of the second vertex is symmetric. □

From now on, for the clarity of notation, we omit the labeling functions from the filter if they are clear from the context.

Theorem 3.8 (Interpolant strength – single assignment)**.** *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A}, \hat{B})$ and let σ be a partial variable assignment. Let Lab and Lab' be locality-preserving labeling functions with restricted propagation for (A, B, σ, ρ) such that $Lab \preceq Lab'$. Let I and I' be the interpolants computed by SALIS for Lab and Lab', respectively. Then*

$$\sigma \models I \to I'$$

Proof. We proceed by structural induction in the refutation tree and prove that the following invariant holds for each vertex v:

$$\sigma \models I_v \wedge \neg\langle\Theta\rangle{\downarrow_v} \to I'_v$$

where $\langle\Theta\rangle$ is the clause in vertex v and I_v and I'_v are the partial interpolants in vertex v using labeling functions Lab and Lab', respectively.

Leaves. Note that the same assignment σ is used, so either the clause in a leaf is satisfied in both computations or unsatisfied in both computations. Also note that in leaf there is no label g, so for the literal l to be preserved by weakened-labeling filter in leaves only $Lab(l) \neq a \wedge Lab'(l) \neq b$ must hold.

- If the clause $\langle\Theta\rangle$ in a leaf is a satisfied clause, than its interpolant is $\top$, so the invariant is trivially satisfied.
- If the clause is unsatisfied from A part, then $I = \langle\Theta\rangle[\sigma]\|_{b,v,Lab}$. It holds that

  $$\langle\Theta\rangle\|_{b,v,Lab} \wedge \neg\langle\Theta\rangle{\downarrow_v} \to \langle\Theta\rangle\|_{b,v,Lab'},$$

  because if $l \in \langle\Theta\rangle\|_{b,v,Lab}$ then $Lab(l) = b$ and either $Lab'(l) = b$, in which case $l \in \langle\Theta\rangle\|_{b,v,Lab'}$, or $Lab'(l) \neq b$ in which case it is preserved by the filter, so $l \in \langle\Theta\rangle{\downarrow_v}$. Since $\sigma \models \langle\Theta\rangle \leftrightarrow \langle\Theta\rangle[\sigma]$, the invariant follows.
- If the clause is unsatisfied from B part, then $I = \neg\langle\Theta\rangle[\sigma]\|_{a,v,Lab}$. It holds that

  $$\neg\langle\Theta\rangle\|_{a,v,Lab} \wedge \neg\langle\Theta\rangle{\downarrow_v} \to \neg\langle\Theta\rangle\|_{a,v,Lab'},$$

  because if $l \in \langle\Theta\rangle\|_{a,v,Lab'}$ then $Lab'(l) = a$ and either $Lab(l) = a$, in which case $l \in \langle\Theta\rangle\|_{a,v,Lab}$, or $Lab(l) \neq a$ in which case it is preserved by the filter, so $l \in \langle\Theta\rangle{\downarrow_v}$. Since $\sigma \models \langle\Theta\rangle \leftrightarrow \langle\Theta\rangle[\sigma]$, the invariant follows.

Inner vertices. Let v be an inner vertex such that v_1 is its positive predecessor with clause $\langle p, \Theta_1 \rangle$ and v_2 is its negative predecessor with clause $\langle \bar{p}, \Theta_2 \rangle$. From the induction hypothesis we have that the invariant holds in the predecessors:

$$\sigma \models I_1 \wedge \neg\langle p, \Theta_1 \rangle\downarrow_{v_1} \rightarrow I_1'$$
$$\sigma \models I_2 \wedge \neg\langle \bar{p}, \Theta_2 \rangle\downarrow_{v_2} \rightarrow I_2'$$

We distinguish several cases based on the type of the resolution at v for both labeling functions. Primed notation is used if we are referring to the case of Lab' and normal notation if referring to Lab. For example we use notation Res-a-b' if the resolution was of type Res-a for Lab and of type Res-b for Lab'. Note that the following cases are forbidden by the assumption that $Lab \preceq Lab'$: Res-a-b', Res-a-ab', Res-a-d'^{+}, Res-a-g', Res-ab-b', Res-d^{+}-b', Res-d^{+}-g', Res-ab-g', Res-g-b', Res-g-ab', Res-g-d'^{+}. For example in Res-ab-b, there is at least one pivot with label a or ab in Lab but in Lab' both pivots have label b, so at least got stronger label in Lab' contradicting the assumption that $Lab \preceq Lab'$. The argument in other cases is similar.

In addition, note that the cases Res-d^{+}-ab, Res-d^{+}-a, Res-b-d'^{+}, Res-ab-d'^{+} are forbidden due to the fact that the same assignment σ is used in both cases. If the same assignment is used then Lab and Lab' agrees completely on assigning label d^{+} and consequently on whether or not a resolution is Res-d^{+} or not. As a consequence, they also agree on all labels g and all resolutions of type Res-g. Therefore all the cases containing exactly one g-resolution are also forbidden.

We go through the remaining cases:

Res-a-a'. Both pivots in both labeling functions have label a, so they are both filtered out by the filter in their corresponding vertices: $\langle p \rangle\downarrow_{v_1} \leftrightarrow \langle \bar{p} \rangle\downarrow_{v_2} \leftrightarrow \top$. Moreover, the assumption of Lemma 3.7 is satisfied, thus $\neg\langle \Theta_1 \rangle\downarrow_v \rightarrow \neg\langle \Theta_1 \rangle\downarrow_{v_1}$ and $\neg\langle \Theta_2 \rangle\downarrow_v \rightarrow \neg\langle \Theta_2 \rangle\downarrow_{v_2}$.

Combining all this together, we obtain that:

$$\begin{aligned} \sigma \models I_1 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v \implies I_1 \wedge \neg\langle \Theta_1 \rangle\downarrow_v \implies I_1 \wedge \neg\langle \Theta_1 \rangle\downarrow_{v_1} \implies \\ \implies I_1 \wedge \neg\langle p, \Theta_1 \rangle\downarrow_{v_1} \implies I_1' \end{aligned}$$

$$\begin{aligned} \sigma \models I_2 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v \implies I_2 \wedge \neg\langle \Theta_2 \rangle\downarrow_v \implies I_2 \wedge \neg\langle \Theta_2 \rangle\downarrow_{v_2} \implies \\ \implies I_2 \wedge \neg\langle \bar{p}, \Theta_2 \rangle\downarrow_{v_2} \implies I_2' \end{aligned}$$

Using these, we can prove the invariant:

$$\begin{aligned} \sigma \models (I_1 \vee I_2) \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v \implies (I_1 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v) \vee (I_2 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v) \implies \\ \implies I_1' \vee I_2' \end{aligned}$$

Res-b-b'. This case is almost identical to the previous one. Again, as both pivots have label b in both labeling functions, they are filtered out, so $\langle p \rangle\downarrow_{v_1} \leftrightarrow \langle \bar{p} \rangle\downarrow_{v_2} \leftrightarrow \top$. As supremum rule is used in Res-b, Lemma 3.7 yields that $\neg\langle \Theta_1 \rangle\downarrow_v \rightarrow \neg\langle \Theta_1 \rangle\downarrow_{v_1}$ and $\neg\langle \Theta_2 \rangle\downarrow_v \rightarrow \neg\langle \Theta_2 \rangle\downarrow_{v_2}$ hold in this case, too. As a result, the same auxiliary implications as in the previous case hold. Together, we get the invariant

$$\begin{aligned} \sigma \models (I_1 \wedge I_2) \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v \implies (I_1 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v) \wedge (I_2 \wedge \neg\langle \Theta_1, \Theta_2 \rangle\downarrow_v) \implies \\ \implies I_1' \wedge I_2' \end{aligned}$$

Res-ab-ab'. Note the regardless of vertex or labeling, it always holds that $p \to \neg\langle\bar{p}\rangle\downarrow$, because the single literal is either filtered out, in which is the consequent evaluates to $\top$, or is not filtered out, in which case the consequent evaluates to p. By the same reasoning $\bar{p} \to \neg\langle p\rangle\downarrow$ also holds.

The following auxiliary implications are needed in the proof of the invariant:

$$\begin{aligned}\sigma \models I_1 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v &\implies p \vee (\bar{p} \wedge I_1 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v)\\ &\implies p \vee (\neg\langle p\rangle\downarrow_{v_1} \wedge I_1 \wedge \neg\langle\Theta_1\rangle\downarrow_{v_1})\\ &\implies p \vee {I_1}'\end{aligned}$$

$$\begin{aligned}\sigma \models I_2 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v &\implies \bar{p} \vee (p \wedge I_2 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v)\\ &\implies \bar{p} \vee (\neg\langle\bar{p}\rangle\downarrow_{v_2} \wedge I_2 \wedge \neg\langle\Theta_2\rangle\downarrow_{v_2})\\ &\implies \bar{p} \vee {I_2}'\end{aligned}$$

The first implication is a simple logical consequence, the second implication combines the observation from the beginning of this case and Lemma 3.7, the third implication is an application of the induction hypothesis. Note that the only assumptions needed to prove these auxiliary implications are the assumptions of Lemma 3.7. We will make use of it when analyzing the other cases.

Note that the assumption of the invariant we are trying to prove can be rewritten as three possibilities using the logical equivalence

$$(p \vee I_1) \wedge (\bar{p} \vee I_2) \iff (p \wedge I_2) \vee (\bar{p} \wedge I_1) \vee (I_1 \wedge I_2).$$

All three possibilities lead to the required result:

$$\begin{aligned}\sigma \models p \wedge I_2 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v &\implies p \wedge (\bar{p} \vee I_2') &&\implies (p \vee I_1') \wedge (\bar{p} \vee I_2')\\ \sigma \models \bar{p} \wedge I_1 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v &\implies \bar{p} \wedge (p \vee I_1') &&\implies (p \vee I_1') \wedge (\bar{p} \vee I_2')\\ \sigma \models I_1 \wedge I_2 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v & &&\implies (p \vee I_1') \wedge (\bar{p} \vee I_2')\end{aligned}$$

The required invariant $\sigma \models (p \vee I_1) \wedge (\bar{p} \vee I_2) \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v \implies (p \vee I_1) \wedge (\bar{p} \vee I_2')$ have thus been proved. Remember that the only assumptions actually needed to prove this were the assumptions of Lemma 3.7. Everything else was just logical derivations or the induction hypothesis.

Res-b-ab', Res-b-a', Res-ab-a'. All of these cases can be covered together using the implication we have derived at the end of the previous case, because the following is true:

$$\begin{aligned}\sigma \models I_1 \wedge I_2 \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v &\implies (p \vee I_1) \wedge (\bar{p} \vee I_2) \wedge \neg\langle\Theta_1,\Theta_2\rangle\downarrow_v\\ &\implies (p \vee I_1') \wedge (\bar{p} \vee I_2')\\ &\implies I_1' \vee I_2'.\end{aligned}$$

The first and third implications are just logical tautologies. The second implication holds, because in each of these three cases, the supremum rule is used to propagate labels, so Lemma 3.7 is applicable.

Res-d^+-d'^+. Assume that $Lab(p, v_1) = d^+$, the case when the second pivot is satisfied is symmetric. As Lab is locality-preserving, $\sigma \models p$. Therefore, it holds that $\sigma \models \neg\langle\bar{p}\rangle\downarrow_{v_2} \leftrightarrow \top$, because $\neg\langle\bar{p}\rangle\downarrow_{v_2}$ is either $\top$ or p, depending on whether it is filtered out or not, but in both cases it is equivalent to $\top$ under σ.

Moreover, $\neg\langle\Theta_2\rangle\downarrow_v \rightarrow \neg\langle\Theta_2\rangle\downarrow_{v_2}$. To see this, consider $l \in \langle\Theta_2\rangle\downarrow_{v_2}$. Then $Lab(l, v_2) \neq a \wedge Lab(l, v_2) \neq g \wedge Lab'(l, v_2) \neq b \wedge Lab'(l, v_2) \neq g$. For both labeling functions the label of l in v is the supremum of l in v_2 and some other label. Since for both sets $\{a, g\}$ and $\{b, g\}$ it holds that if $c \in M$ and $c = c_1 \sqcup c_2$ then $c_1 \in M \wedge c_2 \in M$ for $M = \{a, g\}$ or $M = \{b, g\}$. As a result, $Lab(l, v) \neq a \wedge Lab(l, v) \neq g \wedge Lab'(l, v) \neq b \wedge Lab'(l, v) \neq g$, so $l \in \langle\Theta_2\rangle\downarrow_v$. Combined together with the induction hypothesis, the required invariant is obtained:

$$\sigma \models I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \implies I_2 \wedge \neg\langle\bar{p}, \Theta_2\rangle\downarrow_{v_2} \implies I'_2$$

Res-g-g'. The argument is similar to the previous case. Assume that $Lab(\bar{p}) = g$ and the branch of the positive pivot has been cut off, the other case is symmetric. From the definition of the filter we see that $\neg\langle\bar{p}\rangle\downarrow_{v_2} \leftrightarrow \top$, because literals with label g are filtered out. Moreover, $\neg\langle\Theta_2\rangle\downarrow_v \rightarrow \neg\langle\Theta_2\rangle\downarrow_{v_2}$. This can be proved using the same reasoning as in the previous case. Combined together we the induction hypothesis, the required invariant is obtained:

$$\sigma \models I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \implies I_2 \wedge \neg\langle\bar{p}, \Theta_2\rangle\downarrow_{v_2} \implies I'_2$$

We covered all the possible cases, therefore, the invariant holds for every vertex in the refutation tree. The invariant in the sink gives us the required result $\sigma \models I \rightarrow I'$. □

Theorem 3.9 (Interpolant strength – different assignments)**.** *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A}, \hat{B})$ and let σ and σ' be partial variable assignments. Let Lab be locality-preserving labeling functions with restricted propagation for (A, B, σ, ρ) and let Lab′ be locality-preserving labeling functions with restricted propagation for (A, B, σ', ρ). Let I and I' be the interpolants computed by SALIS for Lab and Lab′, respectively. Moreover, assume that if Lab and Lab′ both cut off a branch at some vertex, they agree on which branch is cut off. If $Lab \preceq Lab'$ then $\sigma, \sigma' \models I \rightarrow I'$.*

Proof. First notice that if σ and σ' disagree on some variable, i.e. there is a literal l such that $\sigma \models l$ and $\sigma' \models \bar{l}$ then $\sigma, \sigma' \models \bot$ and the claim of the theorem is trivially satisfied. Thus, for the rest of the proof we assume the assignments do not disagree. We proceed as in the proof of Theorem 3.8. We show by structural induction that for each vertex v in the refutation tree, the following invariant holds:

$$\sigma, \sigma' \models I_v \wedge \neg\langle\Theta\rangle\downarrow_v \rightarrow I'_v,$$

where I_v, I'_v are the partial interpolants computed for v by Lab and Lab', respectively, and $\langle\Theta\rangle$ is the clause of vertex v.

Leaves. Note that the division of the clauses to A and B part is the same for both labeling functions. Suppose that $\langle\Theta\rangle$ as a clause for some leaf in ρ. The cases when Lab and Lab' agree on whether $\langle\Theta\rangle$ is satisfied or not were already

covered in the proof of Theorem 3.8. The only difference is that here different assignment filters are applied, but that is not a problem because both assignments are assumptions of the invariant. We discuss the cases when they do not agree:

- Hyp-$A_{\bar{\sigma}}$-$A_{\sigma'}$, Hyp-$B_{\bar{\sigma}}$-$B_{\sigma'}$: In this case $I'_v = \top$ so the invariant is trivially satisfied.
- Hyp-A_{σ}-$A_{\bar{\sigma}'}$, Hyp-B_{σ}-$B_{\bar{\sigma}'}$: We show that in this case $\sigma, \sigma' \models \neg\langle\Theta\rangle\downarrow_v \to \bot$, which is enough to ensure that the invariant holds. Since $\langle\Theta\rangle$ is satisfied by σ, it follows that there is a literal l in Θ which is satisfied under σ, therefore $Lab(l, v) = d^+$. From the assumption that $Lab \preceq Lab'$ we have that $Lab'(l, v) \neq b$. As there are no g labels in leaves, it follows that l is preserved by the filter $\downarrow_v$. Consequently, $\neg\langle\Theta\rangle\downarrow_v \to \neg l$. As l is true under σ, we have that $\sigma \models \neg\langle\Theta\rangle\downarrow_v \to l \wedge \neg l$.

Inner vertices. Let v be an inner vertex such that v_1 is its positive predecessor with clause $\langle p, \Theta_1\rangle$ and v_2 is its negative predecessor with clause $\langle \bar{p}, \Theta_2\rangle$. From the induction hypothesis we have that the invariant holds in the predecessors:

$$\sigma, \sigma' \models I_1 \wedge \neg\langle p, \Theta_1\rangle\downarrow_{v_1} \to I'_1$$
$$\sigma, \sigma' \models I_2 \wedge \neg\langle \bar{p}, \Theta_2\rangle\downarrow_{v_2} \to I'_2$$

As in the proof of Theorem 3.8 we discuss the possible resolutions in v for both Lab and Lab'. The same combinations as before are forbidden by the strength restriction. Moreover, the proves of other cases where there is no d^+ or g, such as Res-ab-ab', are still valid in current situation, since those proofs did not depend on the partial assignment. We go through the cases where at least one resolution is of type Res-d^+ or Res-g:

Res-d^+-d'^+. We assumed at the beginning that the assignments do not disagree on any variable. Therefore, the labeling functions agree which pivot is satisfied and consequently, which branch is cut off. As a result, the same reasoning as in the proof of Theorem 3.8 applies here.

Res-d^+-ab', Res-d^+-a'. Suppose that the positive pivot is satisfied, $\sigma \models p$, the other case is symmetric. First, we prove that

$$\sigma, \sigma' \models I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \to I'_2.$$

Note that by the argument we have made before it holds that $\sigma \models \neg\langle\bar{p}\rangle\downarrow_{v_2}$ and also that $\models \neg\langle\theta_2\rangle\downarrow_v \to \neg\langle\theta_2\rangle\downarrow_{v_2}$. The first claim is due to the fact that $\models p \to \neg\langle\bar{p}\rangle\downarrow_{v_2}$. The second claim holds because if $l \in \langle\theta_2\rangle\downarrow_{v_2}$ then $Lab(l, v_2) \neq a$, $Lab(l, v_2) \neq g$, $Lab'(l, v_2) \neq b$ and $Lab'(l, v_2) \neq g$. According to the rules of propagating labels, in each case from Res-ab, Res-a and Res-d^+ the label of l in v is determined as a supremum of label l in v_2 and some other label. As we have proved before, if l does not have either a or g assigned by Lab in v_2, it cannot have a nor g in v. Similarly for the pair b, g and Lab'. As a result, $Lab(l, v) \neq a$, $Lab(l, v) \neq g$, $Lab'(l, v) \neq b$ and $Lab'(l, v) \neq g$, so $l \in \langle\theta_2\rangle\downarrow_v$.

Combining these claims with the induction hypothesis, we get that

$$\sigma, \sigma' \models I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \implies I_2 \wedge \neg\langle\bar{p}, \Theta_2\rangle\downarrow_v \implies I_2'.$$

The invariant for Res-d^+-a' follows from the logical tautology $\models I_2' \to I_1' \vee I_2'$. which is enough to prove the invariant for v. The invariant for Res-d^+-ab' follows from the fact that $\sigma \models I_2' \iff (p \vee I_1') \wedge (\bar{p} \vee I_2')$ because $\sigma \models p$.

Res-ab-d'^+, Res-b-d'^+. This case is almost identical to the previous one. Suppose that the positive pivot is satisfied, $\sigma' \models p$, the other case is symmetric. First, the same reasoning as in the previous case can be used to prove that

$$\sigma, \sigma' \models I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \to I_2'.$$

The invariant for Res-b-d'^+ follows from the logical tautology $\models I_1 \wedge I_2 \to I_2$. The invariant for Res-$ab$-$d'^+$ follows from the fact that $\sigma' \models I_2 \iff (p \vee I_1) \wedge (\bar{p} \vee I_2)$ because $\sigma' \models p$.

Res-g-g'. If Lab and Lab' agree on which branch is cut off, then the same reasoning as in the proof of Theorem 3.8 can be used. The case when the labeling functions disagree on which branch is cut off is explicitly forbidden in the assumptions of this theorem.

Res-g-ab', Res-g-d'^+, Res-ab-g', Res-d^+-g' These cases cannot occur when a restricted propagation is used for both labeling functions, because only encoding variables can then obtain label g. Since these are not assigned by the assignments and they are always local to either A part or to B part, no literal with variable that is an encoding variable can have label d^+ or ab.

Res-b-g' Suppose that the positive pivot has label g and the negative branch is cut off. The other case is symmetric. We want to show that

$$\sigma, \sigma' \models I_1 \wedge I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \to I_1'.$$

First, notice that $\vDash \neg\langle p\rangle\downarrow_{v_1} \Leftrightarrow \top$ because $Lab(v_1, p) = g$, so the literal is filtered out. Then, we can prove that $\vDash \neg\langle\theta_1\rangle\downarrow_v \to \neg\langle\theta_1\rangle\downarrow_{v_1}$ by now already familiar argument. If $l \in \langle\theta_1\rangle\downarrow_{v_1}$ then $Lab(l, v_1) \neq a$, $Lab(l, v_1) \neq g$, $Lab'(l, v_1) \neq b$ and $Lab'(l, v_1) \neq g$. According to the rules of propagating labels, the label for l in v is determined as supremum of label l in v_1 and some other label, for both labeling functions. However, if the label in v_1 is not a or g then the label in v also cannot be a nor g. Similarly for the pair b and g. As a result, $l \in \langle\theta_1\rangle\downarrow_v$.

Combining these claims with the induction hypothesis, we obtain the invariant:

$$\sigma, \sigma' \models I_1 \wedge I_2 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \implies I_1 \wedge \neg\langle p, \Theta_1\rangle\downarrow_{v_1} \implies I_1'$$

Res-g-a' This case is almost identical to the previous one. Suppose that the positive pivot has label g and the negative branch is cut off. The other case is symmetric. We want to show that

$$\sigma, \sigma' \models I_1 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \to I_1' \vee I_2'.$$

Using the same reasoning as in the previous case, we obtain that $\vDash \neg\langle p\rangle\downarrow_{v_1} \Leftrightarrow \top$ and that $\vDash \neg\langle\theta_1\rangle\downarrow_v \rightarrow \neg\langle\theta_1\rangle\downarrow_{v_1}$. Combining these claims with the induction hypothesis, the required invariant is obtained:

$$\sigma, \sigma' \models I_1 \wedge \neg\langle\Theta_1, \Theta_2\rangle\downarrow_v \implies I_1 \wedge \neg\langle p, \Theta_1\rangle\downarrow_{v_1} \implies I_1' \implies I_1' \vee I_2'.$$

We covered all possible cases, hence we have shown that the invariant holds for every vertex in the refutation tree. The invariant in the sink gives us the required result $\sigma, \sigma' \models I \rightarrow I'$. □

3.3 Proof of path interpolation property

The goal is to show that given an unsatisfiable set of NNF formulas, its refutation tree ρ and a division of the set into three parts A, S and B, the interpolants for the division $A, S\cup B$ and for the division $A\cup S, B$ can be comparable under certain conditions even for different partial assignments σ and σ'. Formally, for Lab a structure-aware locality-preserving labeling function with restricted propagation for $(A, S\cup B, \sigma, \rho)$ and Lab' a structure-aware locality-preserving labeling function with restricted propagation for $(A \cup S, B, \sigma', \rho)$ we want to prove that $\sigma, \sigma' \models I \wedge S \rightarrow I'$ where I is the interpolant computed by SALIS for $(Lab, A, S \cup B, \sigma)$ and I' is the interpolant computed by SALIS for $(Lab', A \cup S, B, \sigma)$. The proof is based entirely on the proof of the path interpolation property for LPAIS that can be found in [**?**].

The proof is divided into three steps:

1. Extend assignment σ to $\sigma \cup \sigma'$
2. Move S from B-part to A-part
3. Restrict assignment $\sigma \cup \sigma'$ to σ'

3.3.1 Step 1 – extending assignment

Definition 3.10 (Extended-assignment labeling function)**.** Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that σ is an extension of π. Let Lab be a labeling function for (A, B, π, ρ). The extended-assignment labeling function $Lab^+_{\pi\rightarrow\sigma}$ for Lab is defined as follows: $\forall v \in Leaves(\rho)\ \forall l \in cl(v)$: $Lab^+_{\pi\rightarrow\sigma}(l, v) =$

1. d^+; if $\sigma \models l$
2. a; if $var(l)$ is unassigned by σ and either $\hat{A}^-_\sigma$-local or $\hat{A}_{\bar{\sigma}}$-local
3. b; if $var(l)$ is unassigned by σ and either $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local
4. a; if $var(l)$ is unassigned, $\hat{A}^-_\sigma\hat{B}^-_\sigma$-clean, not $\hat{A}_{\bar{\sigma}}$-local, not $\hat{B}_{\bar{\sigma}}$-local and there exists $v' \in Leaves$ and a literal l' such that $var(l') = var(l)$ and $Lab(l', v') \in \{a, ab\}$
5. $Lab(l, v)$ otherwise

Lemma 3.11. *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that σ is an extension of π. Let Lab be a structure-aware locality-preserving labeling function for (A, B, π, ρ). The extended-assignment labeling function $Lab^{+}_{\pi \to \sigma}$ for Lab is a structure-aware locality-preserving labeling function for (A, B, σ, ρ).*

Proof. It is easy to see that the extended-assignment labeling function is defined to be structure-aware locality-preserving. The only condition not immediately satisfied is condition 3 of Definition 2.15, which is strictly stronger than condition 4 of Definition 2.4. If a literal is satisfied under σ, then it is labeled d^{+} due to the alternative 1 of Definition 3.10. Suppose that variable x is not satisfied by σ and $\hat{A}^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-clean. The following situations are possible:

- Variable x is also $\hat{A}_{\bar{\sigma}}$-local. Then it is consistently labeled a due to the alternative 2 of Definition 3.10.
- Variable x is also $\hat{B}_{\bar{\sigma}}$-local. Then it is consistently labeled b due to the alternative 3 of Definition 3.10.
- Variable x is not $\hat{A}_{\bar{\sigma}}$-local, nor $\hat{B}_{\bar{\sigma}}$-local. If it is consistently labeled b in all leaves by Lab, then the alternative 5 of Definition 3.10 applies to x and x is consistently labeled b by $Lab^{+}_{\pi \to \sigma}$. Otherwise there is some vertex v and literal l with $var(l) = x$ such that $Lab^{+}_{\pi \to \sigma}(l, v) \in \{a, ab\}$. In this case alternative 4 of Definition 3.10 applies to x and x is consistently labeled a.

Thus, even condition 3 of Definition 2.15 is satisfied. □

Note that whether or not a labeling function is structure-aware locality-preserving depends only on labeling the leaves, not on the propagation of labels in inner vertices. Thus, extended-assignment labeling function is structure-aware locality-preserving even if full propagation is used. However, to prove further properties, the restricted propagation is necessary.

Lemma 3.12. *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that σ is an extension of π. Let Lab and Lab' be structure-aware locality-preserving labeling functions with restricted propagation for (A, B, π, ρ) and for (A, B, σ, ρ), respectively. Let $Lab^{+}_{\pi \to \sigma}$ be the extended-assignment labeling function for Lab with restricted propagation.*

If $Lab \preceq_L Lab'$ then $Lab^{+}_{\pi \to \sigma} \preceq Lab'$.

Proof. By Corollary 3.5 it is enough to show that $Lab^{+}_{\pi \to \sigma} \preceq_L Lab'$. For a contradiction assume that there exist a vertex v and a literal l such that $Lab^{+}_{\pi \to \sigma}(l, v) \not\preceq Lab'(l, v)$. We analyze which alternative of Definition 3.10 could have been used to set the value of $Lab^{+}_{\pi \to \sigma}(l, v)$:

1. If the first alternative is used then $\sigma \models l$, so $Lab'(l, v) = d^{+} = Lab^{+}_{\pi \to \sigma}(l, v)$ because Lab' is structure-aware locality-preserving.
2. If the second alternative is used then $var(l)$ is unassigned and $\hat{A}^{-}_{\sigma}$-local or $\hat{A}_{\bar{\sigma}}$-local, so $Lab'(l, v) = a = Lab^{+}_{\pi \to \sigma}(l, v)$ because Lab' is structure-aware locality-preserving.

3. If the third alternative is used then $var(l)$ is unassigned and $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local, so $Lab'(l,v) = b = Lab^+_{\pi\to\sigma}(l,v)$ because Lab' is structure-aware locality-preserving.

4. If the fourth alternative is used then $var(l)$ is unassigned and $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean and not $\hat{A}_{\bar{\sigma}}$-local or $\hat{B}_{\bar{\sigma}}$-local, so Lab' labels it consistently a or b. Moreover there exists v', l' such that $var(l) = var(l')$ and $Lab(l', v') \in \{a, ab\}$. Since $Lab \preceq_L Lab'$, $Lab'(l', v') = a$ and also $Lab'(l, v) = a = Lab^+_{\pi\to\sigma}(l, v)$.

5. If the fifth alternative is used then $Lab^+_{\pi\to\sigma}(l,v) = Lab(l,v) \preceq Lab'(l,v)$.

As a result, no alternative could be used to get that $Lab^+_{\pi\to\sigma}(l,v) \not\preceq Lab'(l,v)$. Therefore, $Lab^+_{\pi\to\sigma} \preceq_L Lab'$. □

Lemma 3.13. *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that σ is an extension of π. Let Lab be a structure-aware locality-preserving labeling function with restricted propagation for (A, B, π, ρ) Let $Lab^+_{\pi\to\sigma}$ be the extended-labeling function for Lab with restricted propagation. If the following holds:*

- *all newly σ-assigned variables (not assigned by π) have the strongest label b assigned consistently in Lab and all occurrences of literals satisfied by σ and not satisfied by π lie in the derivation trees of B,*
- *if Lab and $Lab^+_{\pi\to\sigma}$ both cut off a branch at some vertex, they agree on which branch is cut off,*

then $Lab \preceq Lab^+_{\pi\to\sigma}$.

Proof. First, we show that $Lab \preceq_L Lab^+_{\pi\to\sigma}$. For a contradiction assume that there exist a vertex v and a literal l such that $Lab(l,v) \not\preceq Lab^+_{\pi\to\sigma}(l,v)$. We analyze which alternative of Definition 3.10 could have been used to set the value of $Lab^+_{\pi\to\sigma}(l,v)$:

- If the first alternative is used then $\sigma \models l$, so $Lab^+_{\pi\to\sigma}(l,v) = d^+$. Either $\pi \models l$, in which case $Lab(l,v) = d^+ = Lab^+_{\pi\to\sigma}(l,v)$, or $var(l)$ is newly σ-assigned variable. According to the assumptions, $var(l)$ is consistently assigned the strongest label b by Lab.
- If the second alternative is used then the weakest label a is assigned to l consistently by $Lab^+_{\pi\to\sigma}$.
- If the third alternative is used then $Lab^+_{\pi\to\sigma}(l,v) = b$ and $var(l)$ is either $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local. According to the assumptions, all literals satisfied by σ and not satisfied by π lie in the derivation trees of B, so $\hat{A}^-_\sigma = \hat{A}^-_\pi$ and $\hat{A}_{\bar{\sigma}} = \hat{A}_{\bar{\pi}}$. As a result, all $\hat{B}^-_\sigma$-local variables are already $\hat{B}^-_\pi$-local and $\hat{B}_{\bar{\sigma}}$-local variables are already $\hat{B}_{\bar{\pi}}$-local. Since Lab is structure-aware locality-preserving labeling function and $var(l)$ is unassigned by π and either $\hat{B}^-_\pi$-local or $\hat{B}_{\bar{\pi}}$-local, we obtain that $Lab(l,v) = b$.
- If the fourth alternative is used then the weakest label a is assigned to l consistently by $Lab^+_{\pi\to\sigma}$.

- If the fifth alternative is used then $Lab(l,v) = Lab^{+}_{\pi\to\sigma}(l,v)$ by definition.

As a result, no alternative can violate the claim that $Lab(l,v) \preceq Lab^{+}_{\pi\to\sigma}(l,v)$, so we have proved that $Lab \preceq_L Lab^{+}_{\pi\to\sigma}$. To conclude that $Lab \preceq Lab^{+}_{\pi\to\sigma}$, notice that Lemma 3.4 is applicable in this situation, because the first condition is trivially satisfied (there are no variables assigned by π not assigned by σ), while the other two conditions are among the assumptions of this lemma. □

3.3.2 Step 2 – moving formulas

Definition 3.14 (Strongest successor labeling)**.** Let (A,S,B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A}\cup\hat{S}\cup\hat{B})$ and let σ be a partial variable assignment. Let Lab be a labeling function for $(A, S\cup B, \sigma, \rho)$. The strongest successor labeling Lab^S for Lab induced by S is defined as follows: $\forall v \in Leaves(\rho)\ \forall l \in cl(v)$

$$Lab^S(l,v) = \begin{cases} a & \text{if } var(l) \text{ is unassigned and either } (\hat{A}\cup\hat{S})^{-}_{\sigma}\text{-local} \\ & \text{or } (\hat{A}\cup\hat{S})_{\bar{\sigma}}\text{-local} \\ Lab(l,v) & \text{otherwise} \end{cases}$$

Lemma 3.15. *Let (A,S,B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A}\cup\hat{S}\cup\hat{B})$ and let σ be a partial variable assignment. If Lab is a structure-aware locality-preserving labeling function for $(A, S\cup B, \sigma, \rho)$, then Lab^S is a structure-aware locality-preserving labeling function for $(A\cup S, B, \sigma, \rho)$.*

Proof. We go through the conditions on structure-aware locality-preserving:

- $Lab^S(l,v)$ iff $\sigma \models l$. If $Lab^S(l,v) = d^+$ then also $Lab^S(l,v) = d^+$ (the second alternative had to be used) and $\sigma \models l$ follows because Lab is structure-aware locality-preserving. On the other hand, if $\sigma \models l$ then $Lab(l,v) = d^+$ and also $Lab^S(l,v) = d^+$ because the second alternative is used for assigned variables.

- If $var(l)$ is unassigned and either $(\hat{A}\cup\hat{S})^{-}_{\sigma}$-local or $(\hat{A}\cup\hat{S})_{\bar{\sigma}}$-local then the first alternative kicks in an $Lab^S(l,v) = a$.

- If $var(l)$ is unassigned and either $\hat{B}^{-}_{\sigma}$-local or $\hat{B}_{\bar{\sigma}}$-local then the first alternative cannot be used, so $Lab^S(l,v) = Lab(l,v)$. Since $\hat{B}^{-}_{\sigma}$-locality implies $(\hat{S}\cup\hat{B})^{-}_{\sigma}$-locality and $\hat{B}_{\bar{\sigma}}$-locality implies $(\hat{S}\cup\hat{B})_{\bar{\sigma}}$-locality, $Lab(l,v) = b$ because Lab is structure-aware locality-preserving.

- If $var(l)$ is unassigned and $(\hat{A}\cup\hat{S})^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-clean then it is also $\hat{A}^{-}_{\sigma}(\hat{S}\cup\hat{B})^{-}_{\sigma}$-clean. If $var(l)$ is also $(\hat{A}\cup\hat{S})^{-}_{\sigma}$ local then it is consistently labeled a by the first alternative. Otherwise the second alternative is used for all its occurrences and the claim follows from the fact that Lab is structure-aware locality-preserving.

□

Note that the definition of Lab^S and the proof that it is structure-aware locality-preserving does not depend on whether a full or restricted propagation is used for inner vertices. However, to prove further properties, the restricted propagation is necessary.

Lemma 3.16. *Let (A, S, B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{S} \cup \hat{B})$ and let σ be a partial variable assignment. If Lab is a structure-aware locality-preserving labeling function with restricted propagation for $(A, S \cup B, \sigma)$ and Lab^S is the strongest successor labeling function for Lab induced by S with restricted propagation, then $Lab \preceq Lab^S$.*

Proof. From the definition, it is obvious that $Lab \preceq_L Lab^S$ because Lab^S assigns either the same label as Lab or the weakest label a. From Lemma 3.15 we have that Lab^S is also structure-aware locality-preserving. Since the same assignment is used by Lab and Lab^S Lemma 3.4 yields the required result $Lab \preceq Lab^S$. □

Lemma 3.17. *Let (A, S, B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{S} \cup \hat{B})$ and let σ be a partial variable assignment. Let Lab be a locality-preserving labeling function with restricted propagation for $(A, S \cup B, \sigma)$, let Lab^S be the strongest successor labeling for Lab induced by S with restricted propagation. Let I and I' be the interpolants computed by SALIS for $(A, S \cup B, \sigma, \rho, Lab)$ and $(A \cup S, B, \sigma, \rho, Lab^S)$, respectively. Then*

$$\sigma \models I \wedge S \rightarrow I'.$$

Proof. We show by structural induction over the resolution proof that the following invariant holds for every vertex v:

$$\sigma \models I_v \wedge \hat{S} \wedge \neg\langle\Theta\rangle{\downarrow_v} \rightarrow I'_v,$$

where $\langle\Theta\rangle$ is the clause in the vertex v, $\downarrow_v$ is the weakened-labeling filter applied in v, I_v and I'_v are partial interpolants computed by SALIS for vertex v using labeling functions Lab and Lab^S, respectively.

Leaves. We distinguish cases depending on categorizing the clause $\langle\Theta\rangle$. Note that the set of satisfied clauses is the same for both labeling functions as the same assignment is used.

- $\langle\Theta\rangle$ is a satisfied clause. In this case $I'_v = \top$ and the invariant holds trivially.
- $\langle\Theta\rangle \in \hat{A}_{\bar{\sigma}}$. In this case $I_v = \langle\Theta\rangle[\sigma]\|_{b,v,Lab}$ and $I'_v = \langle\Theta\rangle[\sigma]\|_{b,v,Lab^S}$ Using the same reasoning as in the proof of Theorem 3.8, we derive that

  $$\langle\Theta\rangle|_{b,v,Lab} \wedge \neg\langle\Theta\rangle{\downarrow_v} \rightarrow \langle\Theta\rangle|_{b,v,Lab^S},$$

  because if $l \in \langle\Theta\rangle|_{b,v,Lab}$ then $Lab(l) = b$ and either $Lab^S(l) = Lab(l) = b$, in which case $l \in \langle\Theta\rangle|_{b,v,Lab^S}$, or $Lab^S(l) = a$ in which case it is preserved by the filter, so $l \in \langle\Theta\rangle{\downarrow_v}$. As a result we get

  $$\langle\Theta\rangle[\sigma]\|_{b,v,Lab} \wedge \neg\langle\Theta\rangle{\downarrow_v} \rightarrow \langle\Theta\rangle[\sigma]\|_{b,v,Lab^S},$$

  because the same literals are filtered out by $[\sigma]$ in the antecedent and in the consequent. This claim is even stronger then the required invariant.

- $\langle\Theta\rangle \in \hat{B}_{\bar{\sigma}}$. In this case $I_v = \neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab}$ and $I'_v = \neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab^S}$ Again as in proof of Theorem 3.8,

  $$\neg\langle\Theta\rangle|_{a,v,Lab} \wedge \neg\langle\Theta\rangle\downarrow_v \rightarrow \neg\langle\Theta\rangle|_{a,v,Lab^S}$$

  can be derived. If $l \in \langle\Theta\rangle|_{a,v,Lab^S}$ then $Lab^S(l) = a$ and either $Lab(l) = a$, in which case $l \in \langle\Theta\rangle|_{a,v,Lab}$, or $Lab(l) \neq a$ in which case it is preserved by the filter, so $l \in \langle\Theta\rangle\downarrow_v$. It follows that every literal from the conjunction in the consequent is already present in the conjunction in the antecedent. As a result we get

  $$\neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab} \wedge \neg\langle\Theta\rangle\downarrow_v \rightarrow \neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab^S},$$

  because the same literals are filtered out by $[\sigma]$ in the antecedent and in the consequent. This claim is even stronger then the required invariant.

- $\langle\Theta\rangle \in \hat{S}_{\bar{\sigma}}$. This is the new case not seen in the proof of Theorem 3.8 where the additional assumption $\hat{S}$ is used. In this case $I_v = \neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab}$ and $I'_v = \langle\Theta\rangle[\sigma]\rangle_{b,v,Lab^S}$. First, notice that the following implication is a logical truth: $\langle\Theta\rangle \rightarrow \langle\Theta\rangle|_{a,v,Lab} \vee \langle\Theta\rangle|_{b,v,Lab^S} \vee \langle\Theta\rangle\downarrow_v$. For each literal l from Θ one of the three possibilities must hold

  - If $Lab(l, v) = a$ then $l \in \langle\Theta\rangle|_{a,v,Lab}$.
  - If $Lab^S(l, v) = b$ then $l \in \langle\Theta\rangle|_{b,v,Lab^S}$.
  - If $Lab(l, v) \neq a$ and $Lab^S(l, v) \neq b$ then $l \in \langle\Theta\rangle\downarrow_v$ follows from the definition of $\downarrow_v$ (Definition 3.6) as there are no g labels in the leaves.

  From this implication we can further derive that $\pi \models \langle\Theta\rangle \rightarrow \langle\Theta\rangle[\sigma]\rangle_{a,v,Lab} \vee \langle\Theta\rangle[\sigma]\rangle_{b,v,Lab^S} \vee \langle\Theta\rangle\downarrow_v$, because $\langle\Theta\rangle \in \hat{S}_{\bar{\sigma}}$ implies there are no satisfied literals in $\langle\Theta\rangle$, so under σ the literals there are filtered out by $[\sigma]$ evaluates to $\bot$. Since $\theta \in S$ we have everything we need to derive the invariant:

  $$\begin{aligned}
  &\sigma \models I_v \wedge \hat{S} \wedge \neg\langle\Theta\rangle\downarrow_v \implies I_v \wedge \langle\Theta\rangle \wedge \neg\langle\Theta\rangle\downarrow_v \implies \\
  &\neg\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab} \wedge (\langle\Theta\rangle[\sigma]\rangle_{a,v,Lab} \vee \langle\Theta\rangle[\sigma]\rangle_{b,v,Lab^S} \vee \langle\Theta\rangle\downarrow_v) \wedge \neg\langle\Theta\rangle\downarrow_v \implies \\
  &\langle\Theta\rangle[\sigma]\rangle_{b,v,Lab^S} \equiv I'_v
  \end{aligned}$$

This concludes the base case of the induction.

Inner vertices. Here, we should check all the possible combinations of resolution types that could occur in an inner vertex. However, one can notice the situation is very similar to the situation in the proof of Theorem 3.8. Indeed, when checking that proof we see that in the inner vertices the distribution of the clauses between the two parts does not play any role, only current labels of literals are important. There is only one thing where the situations differ and that is the induction hypothesis (now there is an additional assumption $\hat{S}$). In the proofs, an additional assumption $\hat{S}$ is needed to apply the induction hypothesis to predecessors. Fortunately, $\hat{S}$ is provided as an assumption of the invariant in the child successor, so all the proofs are directly transferable to our current situation.

As a consequence, the invariant holds in all vertices of the resolution proof. The invariant of the sink gives us that $\sigma \models I \wedge \hat{S} \rightarrow I'$. To conclude that $\sigma \models I \wedge S \rightarrow I'$ reason as follows: Fix any assignment σ' which extends σ and satisfies $I \wedge S$. From the properties of the Tseitin's encoding we know that there is an assignment σ'' such that it also extends σ, it agrees with σ' on $Var(I) \cup Var(S)$ and it satisfies $I \wedge \hat{S}$ (σ'' just sets the right values to encoding variables of $\hat{S}$). It follows that $\sigma'' \models I'$. However, I' does not contain any encoding variables from $\hat{S}$, so $\sigma' \models I'$. □

Lemma 3.18. *Let (A, S, B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{S} \cup \hat{B})$ and let σ be a partial variable assignment. Let Lab be a structure-aware locality-preserving labeling function with restricted propagation for $(A, S \cup B, \sigma)$, let Lab^S be the strongest successor labeling for Lab induced by S with restricted propagation and let Lab' be a structure-aware locality-preserving labeling function for $(A \cup S, B, \sigma)$. If $Lab \preceq_L Lab'$ then $Lab^S \preceq Lab'$.*

Proof. First, we show that $Lab^S \preceq_L Lab'$. Consider a leaf v and a literal $l \in cl(v)$. We show that $Lab^S(l, v) \preceq Lab'(l, v)$. If the first alternative of Definition 3.14 was used to label l in v, then $var(l)$ is unassigned and either $(\hat{A} \cup \hat{S})^-_\sigma$-local or $(\hat{A} \cup \hat{S})_{\bar{\sigma}}$-local. Since Lab' is also structure-aware locality-preserving, it also assigns a to l in v. If the second alternative is used then $Lab^S(l, v) = Lab(l, v) \preceq Lab'(l, v)$ by the assumption of this lemma. Thus $Lab^S \preceq_L Lab'$.

Now, assumptions of Corollary 3.5 are satisfied, so we get that $Lab^S \preceq Lab'$. □

Theorem 3.19. *Let (A, S, B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{S} \cup \hat{B})$ and let σ be a partial variable assignment. Let Lab be a structure-aware locality-preserving labeling function with restricted propagation for $(A, S \cup B, \sigma, \rho)$, let Lab^S be the strongest successor labeling for Lab induced by S with restricted propagation and Lab' be a structure-aware locality-preserving labeling function with restricted propagation for $(A \cup S, B, \sigma, \rho)$. Let I and I' be the interpolants computed by SALIS for $(A, S \cup B, \sigma, \rho, Lab)$ and $(A \cup S, B, \sigma, \rho, Lab')$, respectively. If $Lab \preceq_L Lab'$ then $\sigma \models I \wedge S \rightarrow I'$.*

Proof. By Lemma 3.16 we have that $Lab \preceq Lab^S$ and by Lemma 3.18 we have that $Lab^S \preceq Lab'$. By Theorem 3.17 we get that $\sigma \models I \wedge S \rightarrow I^S$ where I^S is the interpolant produced by SALIS for Lab^S. Finally, by Theorem 3.8 we get that $\sigma \models I^S \rightarrow I'$. By chaining these implication we get that $\sigma \models I \wedge S \rightarrow I'$ □

3.3.3 Step 3 – restricting assignment

Definition 3.20 (Restricted-assignment labeling function)**.** Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that π is an extension of σ. Let Lab be a labeling function for (A, B, π, ρ). The restricted-assignment labeling function $Lab^-_{\pi \rightarrow \sigma}$ for Lab is defined as follows: $\forall v \in Leaves(\rho)\ \forall l \in cl(v)$: $Lab^-_{\pi \rightarrow \sigma}(l, v) =$

1. ab; if $\pi \models l$, $var(l)$ is unassigned by σ and $\hat{A}^-_\sigma \hat{B}^-_\sigma$-shared
2. a; if $var(l)$ is unassigned by σ and either $\hat{A}^-_\sigma$-local or $\hat{A}_{\bar{\sigma}}$-local

3. b; if $var(l)$ is unassigned by σ and either $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local

4. a; if $var(l)$ is assigned by π unassigned by σ, $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean, not $\hat{A}_{\bar{\sigma}}$-local, not $\hat{B}_{\bar{\sigma}}$-local and there exists $v' \in Leaves$ and a literal l' such that $var(l') = var(l)$ and $Lab(l', v') \in \{a, ab, d^+\}$

5. $Lab(l, v)$; otherwise

Lemma 3.21. *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that π is an extension of σ. Let Lab be a structure-aware locality-preserving labeling function for (A, B, π, ρ). The restricted-assignment labeling function $Lab^-_{\pi\to\sigma}$ for Lab is a structure-aware locality-preserving labeling function for (A, B, σ, ρ).*

Proof. If a literal is satisfied under σ, then it is labeled d^+ because it has the same label in $Lab^-_{\pi\to\sigma}$ as in Lab and that is d^+ because the literal is also satisfied by π and Lab is locality-preserving. If a literal is falsified under σ, then it is not labeled d^+ because it has the same label in $Lab^-_{\pi\to\sigma}$ as in Lab and that is not d^+ because the literal is also falsified by π and Lab is locality-preserving. For the rest of the proof suppose that $var(l)$ is not assigned by σ. We want to show that $Lab^-_{\pi\to\sigma}(l, v) \neq d^+$. Suppose it is not true, so $Lab^-_{\pi\to\sigma}(l, v) = d^+$. Then the last alternative was applied, so $Lab(l, v) = d^+$ and this means that $\pi \models l$. Since the first alternative was not applied, $var(l)$ is not $\hat{A}^-_\sigma \hat{B}^-_\sigma$-shared. Since the second and third alternative was not applied, $var(l)$ is not $\hat{A}^-_\sigma$-local nor $\hat{B}^-_\sigma$-local. Since the fourth alternative was not applied, $var(l)$ is not $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean, because there is a vertex, namely v, where l is assigned d^+ by Lab. We have run out of categories for $var(l)$ which means the original assumption that $Lab^-_{\pi\to\sigma}(l, v) = d^+$ was false. Thus we have covered the first condition of locality-preserving labeling function.

If $var(l)$ is either $\hat{A}^-_\sigma$-local or $\hat{A}_{\bar{\sigma}}$-local then alternative 2 is applied and label a is assigned as required by definition of structure-aware locality-preserving.

If $var(l)$ is either $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local then alternative 3 is applied and label b is assigned as required by definition of structure-aware locality-preserving.

If $var(l)$ is $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean, we need to show that it is consistently labeled a or b. Note that the first alternative does not apply in this case. If the second, third, or fourth alternative is used, then the requirement is met. Suppose the last alternative was used. If $var(l)$ is not assigned by π then it is also $\hat{A}^-_\pi \hat{B}^-_\pi$-clean because π is an extension of σ. As Lab is structure-aware locality-preserving, it follows that it is consistently labeled a or b by Lab, and thus also by $Lab^-_{\pi\to\sigma}$. If it is assigned by π then Lab labels it consistently b because the fourth alternative was not used. So it is also labeled consistently b by $Lab^-_{\pi\to\sigma}$. □

Note again that whether or not a labeling function is structure-aware locality-preserving depends only on labeling the leaves, not on the propagation of labels in inner vertices. Thus, restricted-assignment labeling function is structure-aware locality-preserving even if full propagation is used. However, to prove further properties, the restricted propagation is necessary.

Lemma 3.22. *Let (A, B) be an unsatisfiable pair of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{B})$ and let π and σ be partial variable assignments such that π is an extension of σ. Let Lab and Lab$'$ be structure-aware locality-preserving labeling functions with restricted propagation for (A, B, π, ρ) and for*

(A, B, σ, ρ), *respectively. Let* $Lab^{-}_{\pi\to\sigma}$ *be the restricted-assignment labeling function for* Lab *with restricted propagation. If* $Lab \preceq_L Lab'$ *then* $Lab^{-}_{\pi\to\sigma} \preceq Lab'$.

Proof. By Corollary 3.5 it is enough to show that $Lab^{-}_{\pi\to\sigma} \preceq_L Lab'$. For contradiction assume that there exist a vertex v and a literal l such that $Lab^{-}_{\pi\to\sigma}(l, v) \not\preceq Lab'(l, v)$. We analyze which alternative of Definition 3.20 could have been used to set the value of $Lab^{-}_{\pi\to\sigma}(l, v)$:

1. If the first alternative is used then $Lab^{-}_{\pi\to\sigma}(l, v) = ab$ and $\pi \models l$, so $Lab(l, v) = d^+$. As $Lab(v, l) \preceq Lab'(v, l)$, we have that $Lab^{-}_{\pi\to\sigma}(l, v) = ab \approx d^+ \preceq Lab'(l, v)$.

2. If the second alternative is used then $var(l)$ is unassigned by σ and $\hat{A}^{-}_{\sigma}$-local or $\hat{A}_{\bar{\sigma}}$-local, so $Lab'(l, v) = a = Lab^{-}_{\pi\to\sigma}(l, v)$ because Lab' is structure-aware locality-preserving.

3. If the third alternative is used then $var(l)$ is unassigned by σ and $\hat{B}^{-}_{\sigma}$-local or $\hat{B}_{\bar{\sigma}}$-local, so $Lab'(l, v) = b = Lab^{-}_{\pi\to\sigma}(l, v)$ because Lab' is structure-aware locality-preserving.

4. If the fourth alternative is used then $var(l)$ is unassigned by σ and $\hat{A}^{-}_{\sigma}\hat{B}^{-}_{\sigma}$-clean so Lab' labels it consistently a or b. Moreover there exist v' and l' such that $var(l) = var(l')$ and $Lab(l', v') \in \{a, ab, d^+\}$. Since $Lab \preceq_L Lab'$, $Lab'(l', v') = a$. As $var(l)$ is consistently labeled by Lab', $Lab'(l, v) = a = Lab^{+}_{\pi\to\sigma}(l, v)$.

5. If the fifth alternative is used then $Lab^{-}_{\pi\to\sigma}(l, v) = Lab(l, v) \preceq Lab'(l, v)$.

As a result, no alternative could be used to get that $Lab^{-}_{\pi\to\sigma}(l, v) \not\preceq Lab'(l, v)$. Therefore, $Lab^{-}_{\pi\to\sigma} \preceq_L Lab'$. □

Lemma 3.23. *Let* (A, B) *be an unsatisfiable pair of sets of NNF formulas. Let* ρ *be a refutation tree for* $(\hat{A} \cup \hat{B})$ *and let* π *and* σ *be partial variable assignments such that* π *is an extension of* σ. *Let* Lab *be a structure-aware locality-preserving labeling function with restricted propagation for* (A, B, π, ρ) *Let* $Lab^{-}_{\pi\to\sigma}$ *be the restricted-labeling function for* Lab *with restricted propagation.*

If the following holds:

- *no variable assigned by* π *and not assigned by* σ *is* $\hat{B}^{-}_{\sigma}$*-local or* $\hat{B}_{\bar{\sigma}}$*-local,*
- *all occurrences of literals satisfied by* π *and not satisfied by* σ *lie in the derivation trees from* A*,*
- *if* Lab *and* $Lab^{-}_{\pi\to\sigma}$ *both cut off a branch at some vertex, they agree on which branch is cut off,*

then $Lab \preceq Lab^{-}_{\pi\to\sigma}$.

Proof. First, we show that $Lab \preceq_L Lab^{-}_{\pi\to\sigma}$. For contradiction assume that there exist a vertex v and a literal l such that $Lab(l, v) \not\preceq Lab^{-}_{\pi\to\sigma}(l, v)$. We analyze which alternative of Definition 3.20 could have been used to set the value of $Lab^{-}_{\pi\to\sigma}(l, v)$:

- If the first alternative is used then from $\pi \models l$ and because Lab is locality-preserving, we get that $Lab(l, v) = d^+ \preceq ab = Lab^-_{\pi\to\sigma}(l, v)$.
- If the second alternative is used then the weakest label a is assigned to l consistently by $Lab^-_{\pi\to\sigma}$.
- If the third alternative is used then $Lab^-_{\pi\to\sigma}(l, v) = b$ and $var(l)$ is either $\hat{B}^-_\sigma$-local or $\hat{B}_{\bar{\sigma}}$-local. According to the assumptions, $var(l)$ is not assigned by π. Since all literals satisfied by π and not satisfied by σ lie in the derivation trees of A, it must hold that $\hat{B}^-_\sigma = \hat{B}^-_\pi$, $\hat{B}_{\bar{\sigma}} = \hat{B}_{\bar{\pi}}$, $\hat{A}^-_\sigma \supseteq \hat{A}^-_\pi$ and $\hat{A}_{\bar{\sigma}} \supseteq \hat{A}_{\bar{\pi}}$. As a result, $var(l)$ is either $\hat{B}^-_\pi$-local or $\hat{B}_{\bar{\pi}}$-local, so $Lab(l, v) = b$ because Lab is structure-aware locality-preserving.
- If the fourth alternative is used then the weakest label a is assigned to l consistently by $Lab^-_{\pi\to\sigma}$.
- If the fifth alternative is used then $Lab(l, v) = Lab^-_{\pi\to\sigma}(l, v)$ by definition.

As a result, no alternative can violate the claim that $Lab(l, v) \preceq Lab^-_{\pi\to\sigma}(l, v)$, so we have proved that $Lab \preceq_L Lab^-_{\pi\to\sigma}$. To conclude that $Lab \preceq Lab^-_{\pi\to\sigma}$, notice that Lemma 3.4 is applicable in this situation, because the second condition of the lemma is trivially satisfied (there are no variables assigned by σ not assigned by π), while the other two conditions are among the assumptions of this lemma. □

3.3.4 Combining partial results

Lemma 3.24. *Let (A, S, B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A} \cup \hat{S} \cup \hat{B})$ and let π and π' be partial variable assignments. Let Lab be a structure-aware locality-preserving labeling function with restricted propagation for $(A, S \cup B, \pi)$ and let Lab' be a structure-aware locality-preserving labeling function with restricted propagation for $(A \cup S, B, \pi')$. Assume $\sigma = \pi \cup \pi'$ is a valid assignment (i.e. π and π' are not contradictory together), and that the following conditions hold:*

- *All occurrences of literals satisfied by π' and not satisfied by π are in the trees from $S \cup B$.*
- *All occurrences of literals satisfied by π and not satisfied by π' are in the trees from $A \cup S$.*
- *Variables assigned by π' and not assigned by π have the strongest label b assigned by Lab.*
- *Variables assigned by π and not assigned by π' are not $\hat{B}^-_\sigma$-local nor $\hat{B}_{\bar{\sigma}}$-local.*
- *If Lab and Lab' both cut off a branch at some vertex of ρ then they agree on which branch is cut off.*

If $Lab' \preceq_L Lab$ then

$$Lab \preceq Lab^+_{\pi\to\sigma} \preceq Lab^S_\sigma \preceq Lab^-_{\sigma\to\pi'} \preceq Lab',$$

where $Lab^+_{\pi\to\sigma}$ is the extended-assignment labeling function for Lab, Lab^S_σ is the strongest successor labeling function for $Lab^+_{\pi\to\sigma}$ induced by S and $Lab^-_{\sigma\to\pi'}$ is the restricted-assignment labeling function for Lab^S_σ.

Proof. We drop the assignments from notations of the labeling functions to simplify them, because it should not cause any confusion.

Note that the assumptions of Lemma 3.23 and Lemma 3.13 are almost covered by the assumptions of this lemma. We can cover the missing assumptions from the assumption that Lab and Lab' do not disagree on cutting a branch at any vertex. From it we show that Lab and $Lab^+_{\pi\to\sigma}$ do not disagree and that Lab^S and $Lab^-_{\pi\to\sigma}$ do not disagree.

Suppose that Lab and $Lab^+_{\pi\to\sigma}$ disagrees on which branch is cut off at some vertex. Since σ is a valid extension of π and only encoding variables can have label g, it follows that in case of Lab one pivot has label g while in case of $Lab^+_{\pi\to\sigma}$ both pivot have label g. It follows that the second pivot has label g due to π', so it has label g also when using Lab'. As a result, Lab and Lab' would also disagree on which branch to cut off at this vertex, which is forbidden. Similar reasoning is used to show that also Lab and Lab' do not disagree on cutting a branch at any vertex.

As a result, by applying Lemmas 3.13, 3.16 and 3.23 every comparison except for the last one is obtained:

$$Lab \preceq Lab^+ \preceq Lab^S \preceq Lab^-$$

To prove the last comparison, we assume there is a vertex v and a literal l such that $Lab^-(l,v) \not\preceq Lab'(l,v)$. We go through the alternatives of Definition 3.20

- If the first alternative is used then $Lab^-(l,v) = ab$. Moreover, $\sigma \models l$ but $var(l)$ is unassigned by π'. As a result $\pi \models l$ and $Lab(v,l) = d^+$. Hence $Lab^-(l,v) = ab \approx d^+ = Lab(l,v) \preceq Lab'(l,v)$.

- If the second alternative is used then $Lab^-(l,v) = a$ and $var(l)$ is either $\hat{A}^-_{\pi'}$-local or $\hat{A}_{\bar{\pi'}}$-local. Since Lab' is structure-aware locality-preserving, we have $Lab'(l,v) = a$

- If the third alternative is used then $Lab^-(l,v) = b$ which is the strongest label.

- If the fourth alternative is used then $Lab^-(l,v) = a$. This means that $var(l)$ is assigned by π and not assigned by π' and $\hat{A}^-_{\pi'}\hat{B}^-_{\pi'}$-clean. Since Lab' is structure-aware locality-preserving, $var(l)$ is consistently labeled a or b. Since $Lab^-(l,v) \not\preceq Lab'(l,v)$, it must be true that $Lab'(l,v) = b$. Moreover, from the fact that the fourth alternative is used, we have that there exist a leaf v' and a literal l' such that $var(l) = var(l')$ and $Lab^S(l',v') \in \{a, ab, d^+\}$. Since $var(l) = var(l')$ and Lab' labels $var(l)$ consistently we get that $Lab'(l',v') = b$. Now consider the label of l' at v' for Lab. We know that $var(l')$ is assigned by π. If $\pi \models l'$ then $Lab(l',v') = d^+$ contradicting the assumption that $Lab(l',v') \preceq Lab'(l',v')$. Finally, if $\pi \models \neg l'$ then $Lab(l',v') = Lab^+(l',v') = Lab^S(l',v')$, because the last alternatives of Definitions 3.10 and 3.14 are used. Thus $Lab(l',v') \in \{a, ab, d^+\}$ contradicting the assumption that $Lab(l',v') \preceq Lab'(l',v')$.

- If the last alternative is used then $Lab^-(l,v) = Lab^S(l,v)$.

Thus we have derived that if $Lab^-(l,v) \not\preceq Lab'(l,v)$ then also $Lab^S(l,v) \not\preceq Lab'(l,v)$. We continue with the analysis which alternative of Definition 3.14 is used to label l in v:

- If the first alternative is used then $Lab^S(l,v) = a$ and $var(l)$ is either $\hat{A}^-_\sigma$-local or $\hat{A}_{\bar{\sigma}}$-local. Since the literals satisfied by π and not satisfied by π' occur only in the trees from $A \cup S$, it follows that $(\hat{A} \cup \hat{S})^-_\sigma \subseteq (\hat{A} \cup \hat{S})^-_{\pi'}$, $(\hat{A} \cup \hat{S})_{\bar{\sigma}} \subseteq (\hat{A} \cup \hat{S})_{\bar{\pi'}}$, $\hat{B}^-_\sigma = \hat{B}^-_{\pi'}$ and $\hat{B}_{\bar{\sigma}} = \hat{B}_{\bar{\pi'}}$. As a consequence, every $(\hat{A} \cup \hat{S})^-_\sigma$-local variable is also $(\hat{A} \cup \hat{S})^-_{\pi'}$-local and every $(\hat{A} \cup \hat{S})_{\bar{\sigma}}$-local variable is also $(\hat{A} \cup \hat{S})_{\bar{\pi'}}$-local. However, this means that $var(l)$ is either $(\hat{A} \cup \hat{S})^-_{\pi'}$-local or $(\hat{A} \cup \hat{S})_{\bar{\pi'}}$-local variable, but this contradicts our assumption that the last alternative of Definition 3.20 and not the second one is used for labeling $Lab^-(l,v)$.

- If the second alternative is used then $Lab^S(l,v) = Lab^+(l,v)$.

Thus we have derived that if $Lab^S(l,v) \not\preceq Lab'(l,v)$ then $Lab^+(l,v) \not\preceq Lab'(l,v)$. We continue with the analysis which alternative of Definition 3.10 is used to label l in v:

- If the first alternative is used then $Lab^+(l,v) = d^+$ and $\sigma \models l$. Since $Lab^+(l,v) \not\preceq Lab'(l,v)$, it follows that $Lab'(l,v) = b$. If $\pi \models l$ then $Lab(l,v) = d^+$ contradicting the assumption that $Lab(l,v) \preceq Lab'(l,v)$. However, if $\pi \not\models l$ then $\pi' \models l$, but this contradicts the fact that Lab' is structure-aware locality-preserving.

- If the second alternative is used then $Lab^+(l,v) = a$ and $var(l)$ is either $\hat{A}^-_\sigma$-local or $\hat{A}_{\bar{\sigma}}$-local. It is easy to see that then it is also either $(\hat{A} \cup \hat{S})^-_\sigma$-local or $(\hat{A} \cup \hat{S})_{\bar{\sigma}}$-local. By the argument shown above in the discussion of the alternative used to determine $Lab^S(l,v)$, we get that $var(l)$ is either $(\hat{A} \cup \hat{S})^-_{\pi'}$-local or $(\hat{A} \cup \hat{S})_{\bar{\pi'}}$-local. Since Lab' is structure-aware locality-preserving, $Lab'(l,v) = a$, contradicting the assumption $Lab^+(l,v) \not\preceq Lab'(l,v)$.

- If the third alternative is used then the strongest label b is assigned, contradicting the assumption that $Lab^+(l,v) \not\preceq Lab'(l,v)$.

- If the fourth alternative is used then $Lab^+(l,v) = a$, $x = var(l)$ is unassigned by σ, $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean and there exist a leaf v' and a literal l' such that $var(l') = x$ and $Lab(l',v') \in a, ab$. Let us consider the possible category of x in $(A \cup S, B, \pi')$.

  It cannot be $(\hat{A} \cup \hat{S})^-_{\pi'}$-local nor $\hat{B}^-_{\pi'}$-local, because the last alternative was used when considering the value of $Lab^-(l,v)$.

  If it is $(\hat{A} \cup \hat{S})^-_{\pi'} \hat{B}^-_{\pi'}$-clean then it is consistently labeled a or b. Since $Lab^+(l,v) \not\preceq Lab'(l,v)$, it must, in fact, be consistently labeled b. However, in this case we have that $Lab'(l',v') = b$ and $Lab(l',v') \in a, ab$ contradicting the assumption that $Lab \preceq Lab'$.

  If it is $(\hat{A} \cup \hat{S})^-_{\pi'} \hat{B}^-_{\pi'}$-shared then especially $x \in Var(\hat{B}^-_{\pi'})$. However, we know that $\hat{B}^-_{\pi'} = \hat{B}^-_\sigma$ because all literals satisfied by π and not satisfied by π' occur only in the trees from $A \cup S$. Thus $x \in Var(\hat{B}^-_\sigma)$, contradicting the assumption that it is $\hat{A}^-_\sigma \hat{B}^-_\sigma$-clean.

- If the last alternative is used then $Lab^+(l,v) = Lab(l,v)$, but this together with the assumption that $Lab^+(l,v) \not\preceq Lab'(l,v)$ contradicts the assumption that $Lab \preceq Lab'$.

We have exhausted all the possibilities, always reaching a contradiction, so the original assumption $Lab^-(l,v) \not\preceq Lab'(l,v)$ must be false. This means that $Lab^- \preceq_L Lab'$. By Corollary 3.5, $Lab^- \preceq Lab'$.

□

Theorem 3.25 (Path interpolation property – inductive step)**.** *Let (A,S,B) be an unsatisfiable triple of sets of NNF formulas. Let ρ be a refutation tree for $(\hat{A}\cup\hat{S}\cup\hat{B})$ and let π and π' be partial variable assignments. Let Lab be a structure-aware locality-preserving labeling function with restricted propagation for $(A, S\cup B, \pi)$ and let Lab' be a structure-aware locality-preserving labeling function with restricted propagation for $(A\cup S, B, \pi')$. Let the interpolant computed by SALIS for $(A, S\cup B, \pi, \rho, Lab)$ be denoted by I and let the interpolant computed by SALIS for $(A\cup S, B, \pi', \rho, Lab')$ be denoted by I'. Assume the following conditions hold:*

- *All occurrences of literals satisfied by π' and not satisfied by π are in the trees from $S\cup B$.*
- *All occurrences of literals satisfied by π and not satisfied by π' are in the trees from $A\cup S$.*
- *Variables assigned by π' and not assigned by π have the strongest label b assigned by Lab.*
- *Variables assigned by π and not assigned by π' are not $\hat{B}^-_\sigma$-local not $\hat{B}_{\bar{\sigma}}$-local.*
- *If Lab and Lab' both cut off a branch at some vertex of ρ then they agree on which branch is cut off.*

If $Lab \preceq_L Lab'$ then $\pi, \pi' \models I \wedge S \rightarrow I'$.

Proof. If π and π' are contradicting each other, then the claim holds trivially. So suppose that $\sigma = \pi \cup \pi'$ is a valid assignment. By Lemma 3.24 we have that

$$Lab \preceq Lab^+_{\pi\rightarrow\sigma} \preceq Lab^S \preceq Lab^-_{\sigma\rightarrow\pi'} \preceq Lab',$$

where $Lab^+_{\pi\rightarrow\sigma}$ is the extended-assignment labeling function for Lab, Lab^S is the strongest successor for $Lab^+_{\pi\rightarrow\sigma}$ induced by S and $Lab^-_{\sigma\rightarrow\pi'}$ is the restricted-assignment labeling function for Lab^S. Let the interpolants computed by $Lab^+_{\pi\rightarrow\sigma}$, Lab^S, and $Lab^-_{\sigma\rightarrow\pi'}$ be denoted by I^+, I^S, and I^-, respectively. We get $\pi, \pi' \models I \rightarrow I^+$ by applying Theorem 3.9, $\pi, \pi' \models I^+ \wedge S \rightarrow I^S$ by applying Lemma 3.17, $\pi, \pi' \models I^S \rightarrow I^-$ by applying Theorem 3.9, and finally $\pi' \models I^- \rightarrow I'$ by applying Theorem 3.8.

Together, the desired result is obtained:

$$\pi, \pi' \models I \wedge S \implies I^+ \wedge S \implies I^S \implies I^- \implies I'$$

□

When we compare the assumptions of Theorem 3.25 with the assumptions for the path interpolation property of LPAIS in [**?**], we see that ours imply theirs. Moreover, with the exception of the last one, our assumptions are not that much stronger. The last assumption deserves a little discussion. In general, we do not know how to decide this condition solely from the given labeling functions, without actually propagating all the labels. However, it is easy to see that if this condition is violated then in at least one tree both assignments need to satisfy at least one literal. So, if for each tree from $A \cup S \cup B$, at most one assignment satisfies a literal from this tree, then the last assumption is satisfied. This assumption is not in conflict with the intended usage as presented in [**?**], so SALIS can also be used in the context of abstract reachability graphs.

4. Evaluation

Though a proper implementation of the proposed system is not a part of this thesis, an experimental implementation of the unrestricted version of the system has been written as a part of the PeRIPLO project [**?**]. However, this implementation has not been tested thoroughly yet, because it takes a long time to prepare proper large inputs, often consisting of thousands and more variables.

Since LPAIS is also implemented in PeRIPLO, we were able to find a small example to demonstrate our claim that if the input is given as a NNF formula and Tseitin's encoding is used to transform the input, SALIS can produce smaller interpolants than LPAIS. More precisely, variables that should not be a part of the subproblem given by a partial assignment applied to the input in its original form, can appear in interpolants computed by LPAIS but not by SALIS. We fed the following input to the solver: $A = \{a, \neg a \vee b \vee c, \neg b \vee (d \wedge (e \vee f))\}$, $B = \{\neg e, \neg d \vee \neg f, \neg c \vee (g \wedge f \wedge (d \vee e))\}$, with partial variable assignment $\sigma \models \neg b$. Note that the input is not in CNF, so PeRIPLO encodes the input to CNF using a very concise Tseitin's encoding. The refutation proof together with the partial interpolants as computed by these systems can be seen in Fig. 4.1 and 4.2. Encoding variables are visible, they are denoted by CNF_xx_xx. Leaves with clauses from A-part of the problem are colored green while leaves with clauses from B-part are colored red. Inner vertices are colored orange or gray depending on whether or not their clause is a unit clause. The color of the edge represents the label of the corresponding pivot: green for a, red for b, black for d^+, blue for g. Notice that the interpolant computed by SALIS has only one variable, while the interpolant computed by LPAIS has four variables with five occurrences. It can be easily verified that the interpolant computed by SALIS is indeed an interpolant for the narrower subproblem $(A[\sigma], B[\sigma])$ while the other interpolant is not.

After this trivial example, we were able to do at least a quick test on a non-trivial one. An unsatisfiable formula with 29 variables was chosen at random from examples from a SAT competition. Then, 10 different partitions into the A and B parts were generated and for each such a partition, 24 random variable assignments (8 for assigning one, five, and twenty variables each) were generated. The results are promising. The average size of the interpolants computed by LPAIS was 370 occurrences of variables, while the average size of the interpolants computed by unrestricted SALIS was 11 occurrences.

Even though this is in no way a proper evaluation of our approach, it shows that SALIS has a potential to significantly reduce the size of computed interpolants.

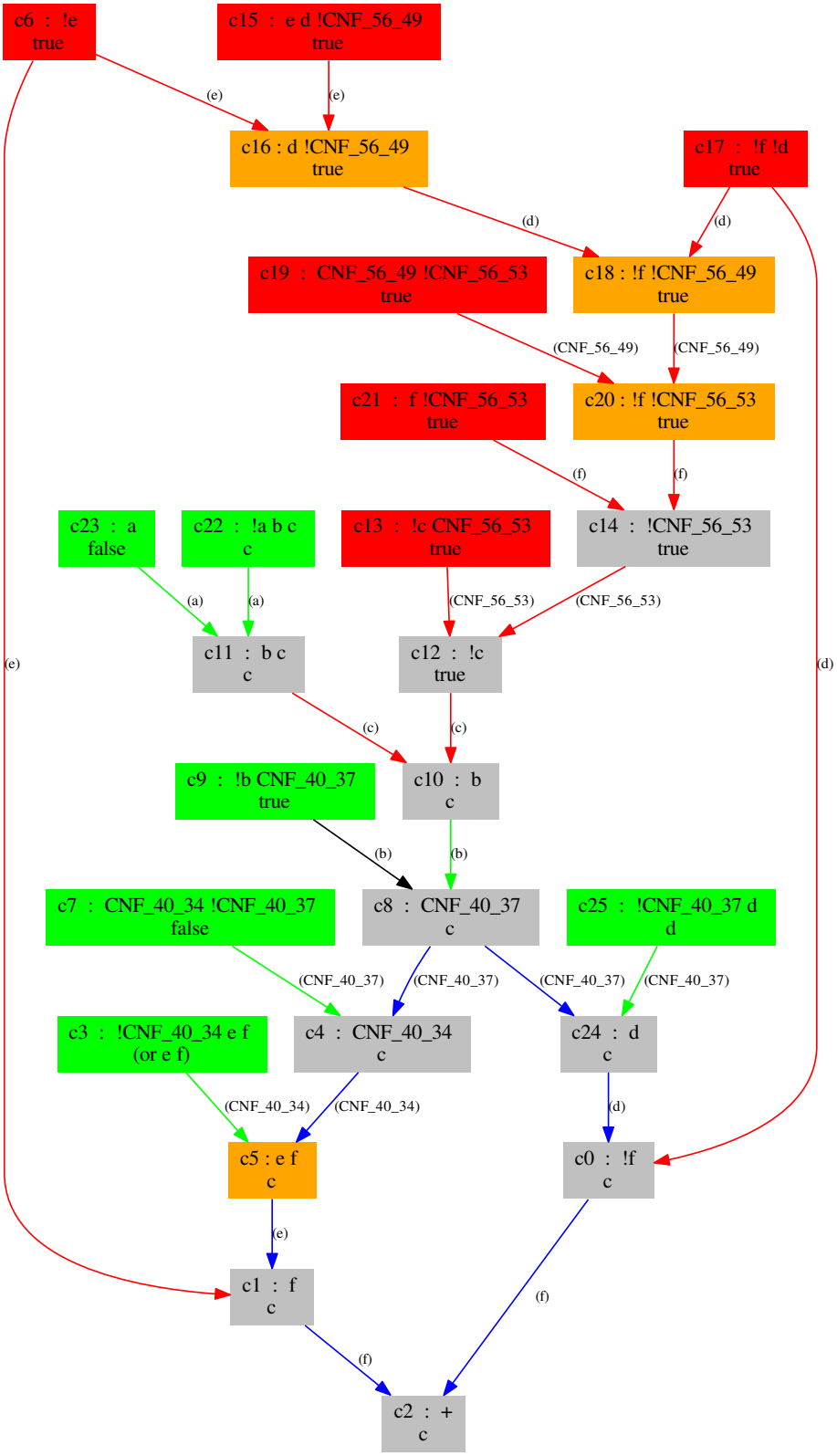

Figure 4.1: SALIS example

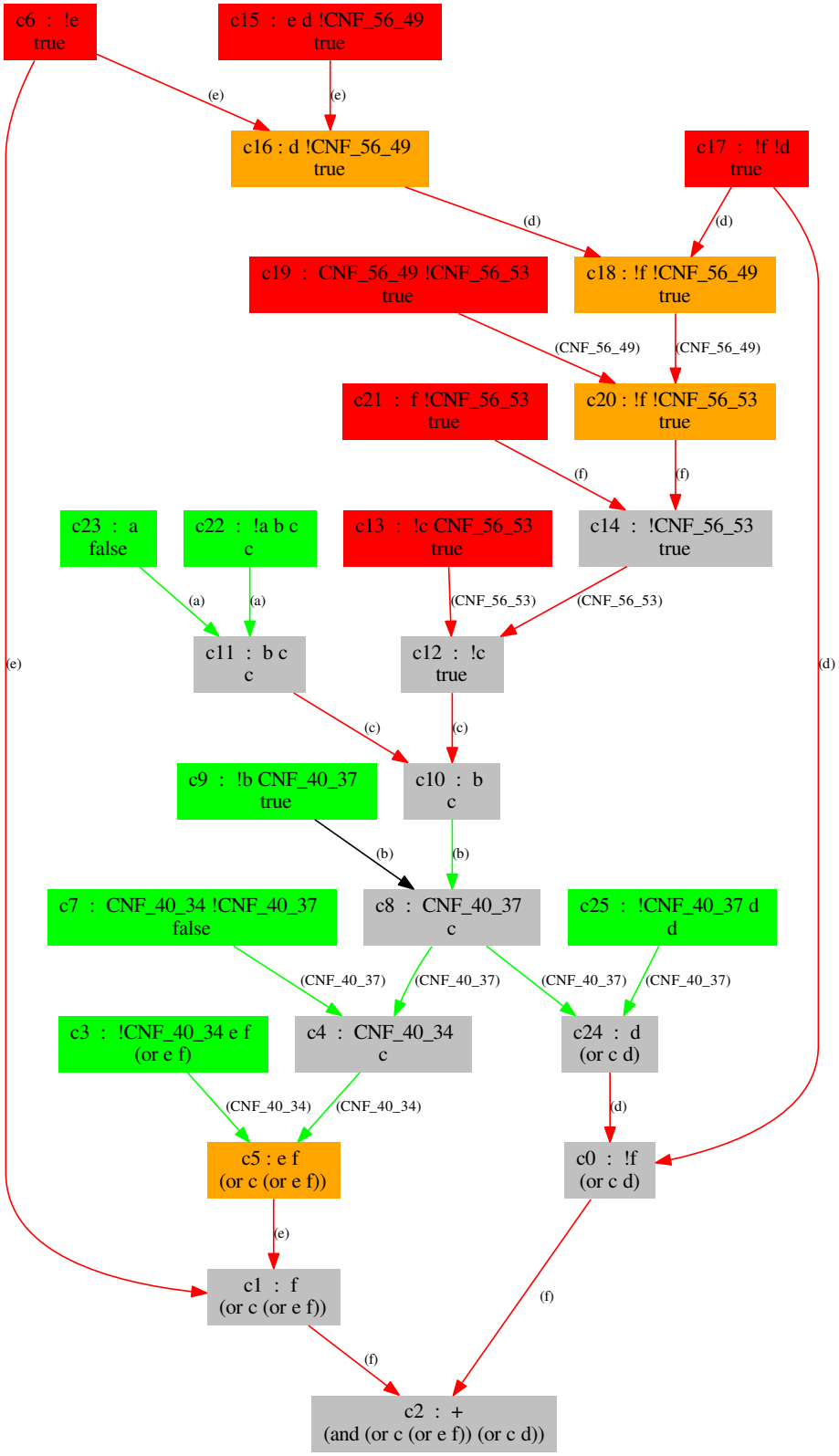

Figure 4.2: LPAIS example

5. Conclusion

In this thesis we have examined the computation of interpolants in the presence of partial variable assignments. Partial variable assignments were introduced to the computation of interpolants in [**?**], where it was shown that, in the context of abstract reachability graph, partial variable assignments can be employed in computing node interpolants, where they can not only reduce the size of computed interpolants, but also effectively solve the problem of out-of-scope variables.

We showed that if Tseitin's encoding is used to encode input to a set of clauses, then LPAIS does not exploit given partial assignment as much as it could. Since Tseitin's encoding is the standard way of encoding input to a set of clauses, this represented a possibility of significant improvement of the system. We presented a modification of the framework of LPAIS to address this issue and formalized it in the form of a new framework of structure-aware labeled interpolation systems. We proved it computes correct interpolants and also that the computed interpolants provably contain only variables common to both parts of the subproblem defined by a partial assignment even if Tseitin's encoding is applied to the input. Then we focused on the path interpolation property where we showed that additional restrictions are needed to prove the property. We successfully showed restrictions such that they do not break the correctness of the system, yet they are strong enough to prove the path interpolation property under conditions that do not prevent its intended use in the context of abstract reachability graph. Although the system has not been tested thoroughly yet, first tests indicates that the reduction of size of computed interpolants could be substantial.

Future work

The main task for the near future is to implement the proposed system, both the restricted and the unrestricted version, properly and to test the implementation thoroughly.

From the theoretical point of view, we laid some constraints on Tseitin's encoding to be used, which we have used to simplify the proofs. We would like to lift these constraints and to prove the correctness of the system even if Tseitin's encoding is optimized, for example by allowing a single encoding variable for multiple conjunctions or multiple disjunctions, or by using single encoding variable for multiple occurrences of the same subformula. We think the first one should be straightforward, but that the second one would demand that the labeling function always assigns the label of its clause (a or b depending on to which part the clause belongs to) to the literal with an encoding variable.

One possible direction of future work is to try to join interpolation with partial assignment and interpolation for theories for first order logic such as linear integer arithmetic.

List of Definitions