Labelled Splitting

Arnaud Fietzke and Christoph Weidenbach

MPI–I–2008–RG1-001 September 2008

Authors' Addresses

Arnaud Fietzke
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Christoph Weidenbach
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Abstract

We define a superposition calculus with explicit splitting and an explicit, new backtracking rule on the basis of labelled clauses. For the first time we show a superposition calculus with explicit backtracking rule sound and complete. The new backtracking rule advances backtracking with branch condensing known from SPASS. An experimental evaluation of an implementation of the new rule shows that it improves considerably the previous SPASS splitting implementation. Finally, we discuss the relationship between labelled first-order splitting and DPLL style splitting with intelligent backtracking and clause learning.

Keywords

Superposition, Splitting, Labels

Contents

1 Introduction

Splitting is an inference rule for case analysis. It is well-known from the Davis-Putnam-Logemann-Loveland (DPLL) [6] decision procedure for propositional logic, where a propositional clause set N is split into the clause sets $N \cup \{A\}$ and $N \cup \{\neg A\}$ for some propositional variable A occurring in N. Obviously, N is satisfiable iff one of the two split clause sets is satisfiable. Furthermore, both split clause sets are simpler than N in the sense that any clause from N containing A can be removed, and in all other clauses from N any occurrence of $\neg A$ can be removed for the split clause set $N \cup \{A\}$ (accordingly for $N \cup \{\neg A\}$). The DPLL decision procedure does not consider the two split clause sets in parallel by duplicating N, but traverses the eventual tree generated by splitting and backtracking in a depth-first way. By appropriate implementation mechanisms, backtracking then becomes quite cheap. As any split set is a subset of subclauses after reduction with the split variable, updates can be made my marking and there is no need to generate new clause objects[1]. Nieuwenhuis et al. [15] presented the DPLL procedure performing depth-first search by an abstract calculus. One contribution in this paper is to perform the same exercise for the first-order case and splitting.

In first-order logic the DPLL splitting style does typically not make sense, because for a given first-order atom A there exist infinitely many ground instances $A\sigma$ of A and it is not known which instances eventually contribute to a proof or model. Furthermore, in case of models having infinite domains, such a style of splitting won't terminate. Therefore, for many superposition based decidability results of first-order logic fragments, e.g., [3], a different style of splitting is used. Given a clause $C \in N$ that can be decomposed into two non-trivial variable disjoint subclauses C_1, C_2, we split into the clause sets $N \cup \{C_1\}$ and $N \cup \{C_2\}$. Very often the rule is further restricted to require that both C_1 and C_2 contain at least one positive literal, i.e., we split

[1] Learning clauses is a separate issue.

into clause sets that are closer to Horn. The rationale behind this restriction is that for Horn clause sets decidability results are typically "easier" to establish and more efficient algorithms exist. For example, satisfiability of propositional Horn clause sets can be decided in linear time, whereas satisfiability of arbitrary propositional clause sets is an NP-complete problem.

A further major difference between first-order splitting and propositional splitting is that in the first-order case effective theorem proving typically relies on the generation of new clauses, either via inferences or reductions. Therefore, the bookkeeping for backtracking of a depth-first approach to splitting gets more involved, because marking algorithms on existing clauses are no longer sufficient to track changes. We need to extend the labels used in the abstract DPLL calculus that are sequences of decision and propagation literals, to a sequence of split elements, called the split stack, holding in particular the potential second part of the split clause and all clauses that became redundant in the course of splitting and may have to be reactivated.

Our starting point is the splitting approach as it was implemented in SPASS [20, 19]. On the basis of this calculus we develop the labelled splitting calculus that in particular refines the previous one with an improved backtracking rule (Chapter 2). We show the labelled splitting calculus to be sound and complete where we introduce a new notion of fairness, taking into account an infinite path in the split tree. Labelled splitting is implemented in SPASS (`http://spass-prover.org/`) and improves significantly on the previous implementation (Chapter 3). We compare the calculus to other approaches to splitting, in particular, to the DPLL approach with intelligent backtracking and clause learning (Chapter 3). The report ends with a summary of the achieved results and directions for future work (Chapter 4).

2 Labelled Splitting

2.1 Preliminaries

We employ the usual notions and notations of first-order logic and superposition in a way consistent with [19]. When traversing the tree generated by successive applications of the splitting rule (the *split tree*), the conclusions of splits that were used in deriving a clause determine the clause's scope in the tree, i.e., those parts of the tree in which the clause may participate in proof search. In order to capture this information, we identify each case of a splitting application on a given path through the tree with a unique integer, its *split level*, and label each clause with a set of integers, representing all splits that contributed to the derivation of the clause.

Formally, a labelled clause $L : C$ consists of a finite set $L \subseteq \mathbb{N}$ and a clause $C = \Gamma \to \Delta$ where Γ and Δ contain the negatively and positively occurring atoms, respectively. The empty clause with label L is denoted by $L\colon \Box$. We call the greatest element in L the *split level* of the clause. We say that $L : C$ *depends* on l if $l \in L$. We extend the usual notions about clause sets to sets of labelled clauses in the natural way: for example, we will say that a set N of labelled clauses entails some formula ϕ (written $N \models \phi$) if and only if the corresponding set of unlabelled clauses entails ϕ. Similarly we extend clause orderings [19] to labelled clauses by abstracting from labels.

A *labelled clause set* is of the form $\Psi : N$ where N is a set of labelled clauses and Ψ is the *split stack*. Split stacks are sequences $\Psi = \langle \psi_n, \ldots, \psi_1 \rangle$ of length $n \geq 0$ ($\Psi = \langle\rangle$ if $n = 0$) and correspond to paths in the split tree. The ψ_i are tuples $\psi_i = (l_i, B_i, D_i, \varphi_i)$ called *splits*, where $l_i \in \mathbb{N}$ is the split level, B_i is the set of *blocked* clauses, D_i is the set of *deleted* clauses and $\varphi_i \in \{\varnothing, \{L\}\}$ with $L \subseteq \mathbb{N}$, is the *leaf marker* which records which splits were involved in refuting a branch. This information is exploited during backtracking to get rid of unnecessary splits. Splitting a clause results in a new split being put onto the stack, which can be thought of as entering the corresponding left branch of the split tree. Once the branch has been

refuted, the split is removed and possibly replaced by a split representing the right branch of the splitting step. Splits corresponding to left branches will be assigned odd levels, while splits corresponding to right branches will have even levels. Therefore we define two predicates, left and right, as follows: $\mathrm{left}(l) = \mathrm{true}$ iff $l \bmod 2 = 1$ and $\mathrm{right}(l) = \mathrm{true}$ iff $l \bmod 2 = 0$. We call $\psi = (l, B, D, \varphi)$ a *left split* if $\mathrm{left}(l)$, and a *right split* otherwise. In a left split, the set B will contain clauses to be reinserted when entering the right branch. In the present framework, B will always consist of just the second split clause. However using a set allows for additional clauses to be added to the right branch, for example the negation of the first split clause in case it is ground. Furthermore, the reason why split levels are made explicit (instead of taking the split level of ψ_k to be k, for example) is that because of branch condensation, split removal during backtracking is not limited to toplevel splits and hence "holes" may appear in the sequence of split levels. This will become clear when we discuss backtracking. For better readability, we will use the notation $\psi[x := v]$ where x is one of l, B, D, φ to denote a split identical to ψ up to component x, which has value v. We write $\psi[x_1 := v_1, x_2 := v_2]$ instead of $\psi[x_1 := v_1][x_2 := v_2]$.

For the definition of the labelled calculus we distinguish inference rules

$$\mathcal{I}\frac{\Psi : N \quad L_1 \colon \Gamma_1 \to \Delta_1 \quad \ldots \quad L_n \colon \Gamma_n \to \Delta_n}{\Psi' : N' \qquad\qquad K \colon \Pi \to \Lambda}$$

and reduction rules

$$\mathcal{R}\frac{\Psi : N \quad L_1 \colon \Gamma_1 \to \Delta_1 \quad \ldots \quad L_n \colon \Gamma_n \to \Delta_n}{\begin{array}{l}\Psi' : N' \qquad K_1 \colon \Pi_1 \to \Lambda_1 \\ \qquad\qquad\qquad\quad \vdots \\ \qquad\qquad\qquad K_k \colon \Pi_k \to \Lambda_k\end{array}}$$

The clauses $L_i \colon \Gamma_i \to \Delta_i$ are called the *premises* and the clauses $K_{(i)} \colon \Pi_{(i)} \to \Lambda_{(i)}$ the *conclusions* of the respective rule. A rule is applicable to a labelled clause set $\Psi : N$ if the premises of the rule are contained in N. In the case of an inference, the resulting labelled clause set is $\Psi' : (N' \cup \{K \colon \Pi \to \Lambda\})$. In the case of a reduction, the resulting labelled clause set is $\Psi' : (N' \setminus \{L_i \colon \Gamma_i \to \Delta_i \mid 1 \leq i \leq n\} \cup \{K_j \colon \Pi_j \to \Lambda_j \mid 1 \leq j \leq k\})$. Furthermore, we say that a ground clause C is *redundant* in N if it follows from smaller clauses in N, and a ground inference is redundant in N, if its conclusion follows logically from clauses in N that are smaller than its maximal premise. A general clause C is redundant in N if all its ground instances $C\sigma$ are redundant with respect to all ground instances of clauses in N. A general inference is redundant if all its ground instances are redundant with respect to all ground instances of clauses in N.

2.2 Labelled Calculus

We present a basic set of inference and reduction rules which together yield a sound and refutationally complete calculus for first-order logic without equality. The emphasis of this paper lies on the modelling of the splitting process, hence more advanced rules like those discussed in [19] have been omitted, since their presentation here would not add to the understanding of splitting-specific issues. Such rules can be integrated to the present setting in a straightforward way and are contained in our implementation. For the same reason we omit the usual ordering restrictions and selection strategies.

Definition 1 (Splitting). *The inference*

$$\mathcal{I}\frac{\Psi : N \quad L\colon \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2}{\Psi' : N \qquad L'\colon \Gamma_1 \rightarrow \Delta_1}$$

where $\Psi = \langle \psi_n, \ldots, \psi_1 \rangle$ ($l_n = 0$ *if* $n = 0$), $l_{n+1} = 2\left\lceil \frac{l_n}{2} \right\rceil + 1$, $L' = L \cup \{l_{n+1}\}$, $L'' = L \cup \{l_{n+1} + 1\}$, $\Psi' = \langle \psi_{n+1}, \psi_n, \ldots, \psi_1 \rangle$ *with* $\psi_{n+1} = (l_{n+1}, \{L''\colon \Gamma_2 \rightarrow \Delta_2\}, \varnothing, \varnothing)$, $vars(\Gamma_1 \rightarrow \Delta_1) \cap vars(\Gamma_2 \rightarrow \Delta_2) = \varnothing$, *and* $\Delta_1 \neq \varnothing$ *and* $\Delta_2 \neq \varnothing$ *is called* splitting.

Splitting creates a new split representing the left branch $\Gamma_1 \rightarrow \Delta_1$ on the stack. The remainder is kept in the new split's blocked clause set to be restored upon backtracking. The split level l_{n+1} is the smallest odd number larger than l_n (hence left(l_{n+1}) holds) and the blocked clause has $l_{n+1} + 1$ added to its label (right($l_{n+1} + 1$) holds). Furthermore, note that splitting is an inference and the parent clause $\Gamma_1, \Delta_1 \rightarrow \Gamma_2, \Delta_2$ is not removed from the clause set. A concrete proof strategy may require to apply subsumption deletion to the parent clause immediately after each splitting step (and after each backtracking step, when the corresponding right branch was entered), thus turning splitting into a reduction. In the current SPASS implementation, splitting is a reduction rule in this sense.

In case the first split part $L'\colon \Gamma_1 \rightarrow \Delta_1$ is ground, the clauses resulting from its negation can be added to the set of blocked clauses, i.e., the right branch. With this modification, the above splitting rule is as powerful as the DPLL splitting rule concerning proof complexity.

Definition 2 (Resolution). *The inference*

$$\mathcal{I}\frac{\Psi : N \quad L_1\colon \Gamma_1 \rightarrow \Delta_1, A \quad L_2\colon \Gamma_2, B \rightarrow \Delta_2}{\Psi : N \qquad L_1 \cup L_2\colon (\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2)\sigma}$$

where σ *is the most general unifier of* A *and* B *is called* resolution.

Definition 3 (Factoring)**.** *The inference*

$$\mathcal{I}\frac{\Psi : N \quad L\colon \Gamma \to \Delta, A, B}{\Psi : N \quad L\colon (\Gamma \to \Delta, A)\sigma}$$

where σ is the most general unifier of A and B is called factoring*.*

Definition 4 (Subsumption Deletion)**.** *The reduction*

$$\mathcal{R}\frac{\Psi : N \quad L_1\colon \Gamma_1 \to \Delta_1 \quad L_2\colon \Gamma_2 \to \Delta_2}{\Psi' : N \qquad\qquad L_1\colon \Gamma_1 \to \Delta_1}$$

where $\Gamma_2 \to \Delta_2$ is subsumed by $\Gamma_1 \to \Delta_1$, $l_{m_1} = max(L_1)$, $l_{m_2} = max(L_2)$, $\Psi = \langle \psi_n, \ldots, \psi_{m_1}, \ldots \psi_1 \rangle$, and

$$\Psi' = \begin{cases} \Psi & \text{if } m_1 = m_2 \\ \langle \psi_n, \ldots, \psi_{m_1}[D := D_{m_1} \cup \{L_2\colon \Gamma_2 \to \Delta_2\}], \ldots, \psi_1 \rangle & \text{otherwise} \end{cases}$$

is called subsumption deletion*.*

The subsumption deletion rule is presented here as one prominent example of a reduction rule, of which many more exist [19]. They all have in common that a clause is simplified (or removed), either because of some property inherent to the clause itself (e.g., tautology deletion), or because of the presence of another clause (as with subsumption deletion). In the first case, no particular precautions are needed with respect to splitting. In the second case however, we must account for the possibility that the reducing (here: subsuming) clause will eventually be removed from the clause set, e.g., because a split that the clause depends on is removed during backtracking. This is why we store the subsumed clause at the subsuming clause's level on the split stack. As an example, consider the clauses $\{1, 6\} : P(x)$ and $\{1, 3\}\colon P(a)$. Applying subsumption deletion would cause $\{1, 3\}\colon P(a)$ to be removed from the clause set and added to the deleted set at the split with level 6. On the other hand, if both the subsumer and the subsumee have the same split level, then there always remains a subsuming clause in the current clause set as long as the corresponding split is not deleted, hence we can remove and forget the subsumed clause.

2.2.1 Backtracking

We will now define rules that formalize backtracking. In particular, we focus our attention on the deletion of splits from the split stack to ensure that all the bookkeeping is done correctly. We denote by $\text{maxr}(\Psi) := max(\{1 \leq i \leq n \mid \text{right}(l_i)\} \cup \{0\})$ the last right split in Ψ. For any given split stack

Ψ, we define the set $levels(\Psi)$ to be the set of split levels occurring in Ψ, i.e., $levels(\Psi) := \{l_1, \ldots, l_n\}$ for $\Psi = \langle \psi_n, \ldots, \psi_1 \rangle$. Finally, for any labelled clause set N and set of split levels $K \subseteq \mathbb{N}$ we define $N|_K := \{L\colon \Gamma \to \Delta \in N \mid L \subseteq K\}$. and $N|_{\overline{K}} := \{L\colon \Gamma \to \Delta \in N \mid L \cap K = \varnothing\}$.

When removing a split ψ_k from the stack, we have to take care to undo all reductions that involved a clause depending on (the level of) ψ_k. In particular, if a clause C_1 depending on ψ_k was used to reduce some other clause C_2, then C_2 must be reinserted into the current clause set. The reason is that C_1 will be removed from the current clause set, and C_2 may then no longer be redundant. If C_2 was reduced by C_1, then C_2 will be in the set of deleted clauses at the level of C_1. Note that although we know that C_1 depends on ψ_k, C_1 may also depend on other splits and thus have a split level higher than l_k. Our goal is to reinsert the deleted clauses at C_1's split level. But C_1 itself, after having reduced C_2, may have been reduced by some clause C_3, hence C_1 will not necessarily be in the current clause set, but in the deleted set at the level of C_3. This means that we need to reinsert all deleted clauses from split levels of clauses depending on ψ_k. So let $\Psi : N$ be an arbitrary labelled clause set with Ψ of length n, and $1 \leq k \leq n$. Now define

$$D(k) := \bigcup_{\substack{i=1 \\ i \neq k}}^{n} D_i \quad \text{and} \quad R(k) := \{j \mid L : C \in N \cup D(k),\ l_j = max(L),\ l_k \in L\}.$$

The set $R(k)$ describes the splits corresponding to all levels of clauses that depend on ψ_k, both in N and any deleted set D_i. It follows that the set $\bigcup_{j \in R(k)} D_j$ contains all clauses that may have been reduced by a clause in N depending on ψ_k. The reason for excluding D_k from $D(k)$ is that D_k will always be reinserted when deleting ψ_k, as the following definition shows:

Definition 5 (Delete Split)**.** *We define* $\text{delete}(\Psi : N, k) := \Psi' : N'$ *where* $\Psi' = \langle \psi'_n, \ldots, \psi'_{k+1}, \psi'_{k-1}, \ldots, \psi'_1 \rangle$, *and* $N' = (N \cup D_k \cup \bigcup_{j \in R(k)} D_j)|_{levels(\Psi')}$ *with*

$$\psi'_j = \begin{cases} \psi_j[D := \varnothing] & \text{if } j \in R(k) \\ \psi_j[D := \{L\colon \Gamma \to \Delta \in D_j \mid l_k \notin L\}] & \text{otherwise} \end{cases}$$

which removes split ψ_k*, all clauses depending on* ψ_k*, and reinserts all clauses reduced by a clause depending on split* ψ_k*.*

Note that reinserting all clauses in D_j with $l_j \in R(k)$ is an over-approximation, since not every clause in D_j was necessarily reduced by a clause depending on ψ_k. In fact, it may well be that no clause in D_j was reduced by a clause depending on ψ_k. If we wanted to reinsert only clauses reduced by clauses

depending on ψ_k, we would have to record which clause was used in each reduction step, not only the reducing clause's split level. It is not clear whether that additional effort would pay off in practice.

We now define a reduction[1] relation $\Psi : N \to \Psi' : N'$ on labelled clause sets to capture the structural transformations of the stack taking place during backtracking. The reduction relation is defined by the following four rules, where we assume that $\Psi : N$ is a labelled clause set and Ψ is of length n.

Definition 6 (Backjump). *If* $n > 0$, $L\colon \Box \in N$ *and* $\max(L) < l_n$, *then*

$$\Psi : N \to \mathrm{delete}(\Psi : N, n).$$

The Backjump rule removes the toplevel split if it did not contribute to the empty clause $L\colon \Box$. Applying Backjump exhaustively yields a split stack that is either empty (if $L = \varnothing$) or has a toplevel split with a split level $l \in L$.

Definition 7 (Branch-condense). *If* $n > 0$, $L\colon \Box \in N$, $\max(L) = l_n$, $\mathrm{left}(l_n)$ *and* $k_{max} := \max\ \{k \mid \mathrm{maxr}(\Psi) < k \leq n \text{ and } l_k \notin L\}$ *exists, then*

$$\Psi : N \to \mathrm{delete}(\Psi : N, k_{max}) \quad .$$

The rule Branch-condense removes an inner (i.e., non-toplevel) split if it did not contribute to the empty clause. However, only splits up to the last right split are considered. Dropping this restriction results in an unsound procedure, since the splits used to close a left branch (represented by the leaf marker) must be taken into consideration when analyzing the dependencies of the corresponding right branch. This analysis is performed by the following rule.

Definition 8 (Right-collapse). *If* $n > 0$, $L_2\colon \Box \in N$, $\max(L_2) = l_n$ *and* $\mathrm{right}(l_n)$, *and* $\varphi_n = \{L_1\}$, *then*

$$\Psi : N \to \Psi' : (N' \cup \{L\colon \Box\}),$$

where $\Psi' : N' = \mathrm{delete}(\Psi : N, n)$ *and* $L = L_1 \cup L_2 \setminus \{l_n - 1, l_n\}$.

The Right-collapse rule analyzes the dependencies involved in refuting left and right branches and computes a newly labelled empty clause by removing complementary split levels. The rule improves upon previous backracking mechanisms (see [21]) by allowing consecutive sequences of Backjump steps (interleaved by applications of Right-collapse) to take place within a single Backtracking step, thus possibly pruning larger parts of the split tree.

Definition 9 (Enter-right). *If* $n > 0$, $L\colon \Box \in N$, $\max(L) = l_n$, $\mathrm{left}(l_n)$ *and* $l_k \in L$ *for all* k *with* $\mathrm{maxr}(\Psi) < k \leq n$, *then* $\Psi'' : N'' := \mathrm{delete}(\Psi : N, n)$ *and*

$$\Psi : N \to \Psi' : N',$$

where $\Psi' = \langle (l_n + 1, \varnothing, \varnothing, \{L\}), \psi''_{n-1}, \ldots, \psi''_1 \rangle$ *and* $N' = N'' \cup B_n$.

[1] Not to be confused with reduction rules for clause sets. See [1] for a discussion of abstract reduction systems.

Finally, Enter-right replaces a toplevel left split by a right split representing the second case of the corresponding splitting step.

As shown in Lemma 12, any labelled clause $\Psi : N$ arising in a derivation and containing at most one empty clause has a unique normal form with respect to $\rightarrow$, written $\Psi : N{\downarrow}$.

Definition 10 (Backtracking)**.** *The reduction*

$$\mathcal{R}\frac{\Psi : N \qquad L\colon \square}{(\Psi : N')\downarrow}$$

where $N' = \{L' \colon C \in N \mid C \neq \square\} \cup \{L\colon \square\}$ *is called* backtracking.

Since we have not placed any restrictions on when to apply backtracking, there may be more than one empty clause in $\Psi : N$. Choosing one and removing all others before applying stack reductions ensures that the result of backtracking is uniquely defined (see Lemma 12). In a practical system, one would typically choose the most general empty clause for backtracking, i.e., the one whose label represents a minimal scope in the split tree. The following example shows the stack reduction rules in action.

Example 11. *Consider the clause set*

$$\begin{array}{llll} \{ & \rightarrow P(a), Q(b); & P(x) \rightarrow P(f(x)), R(c); & P(f(f(a))) \rightarrow; \\ & \rightarrow Q(x), S(y); & Q(x), P(x) \rightarrow; & \\ & \rightarrow R(a), R(b); & S(x), P(x) \rightarrow \} & \end{array}$$

where we omit empty labels. Figure 11 shows the development of the split tree over three backtracking steps. Bold line segments indicate the current path in the split tree, and numbers next to branches correspond to split levels. The split stack representing the first tree is

$$\begin{array}{ll} \langle & (7, \{\{8\} \colon R(c)\}, \varnothing, \varnothing), \quad (5, \{\{6\} \colon R(b)\}, \varnothing, \varnothing), \\ & (3, \{\{4\} \colon R(c)\}, \varnothing, \varnothing), \quad (1, \{\{2\} \colon Q(b)\}, \varnothing, \varnothing) \ \rangle \end{array}$$

which is obtained in seven steps from the initial clause set: (1) clause $\rightarrow P(a), Q(b)$ *is split, (2) resolution is applied to* $P(a)$ *and* $P(x) \rightarrow P(f(x)), R(c)$*, (3) the resulting clause* $\{1\}{:}\rightarrow P(f(a)), R(c)$ *is split, (4) clause* $\rightarrow R(a), R(b)$ *is split, (5) resolution is applied to* $\{1, 3\} \colon P(f(a))$ *and* $P(x) \rightarrow P(f(x)), R(c)$*, resulting in* $\{1, 3\}{:}\rightarrow P(f(f(a))), R(c)$*, which is again split (6), finally the empty clause* $\{1, 3, 7\} \colon \square$ *is derived by resolving* $\{1, 3, 7\} : P(f(f(a)))$ *and* $P(f(f(a))) \rightarrow$ *(7). The third split did not contribute to the contradiction, so it is removed by Branch-condense in step 1, followed by Enter-right, which*

produces $(8, \varnothing, \varnothing, \{\{1,3,7\}\})$ *as toplevel split. The clause* $\rightarrow Q(x), S(y)$ *is then split, resolution is applied to* $\{1\} : P(a)$ *and* $Q(x), P(x) \rightarrow$ *to derive* $\{1\}$: $Q(x) \rightarrow$*, which is resolved with* $\{9\} : Q(x)$ *to yield* $\{1,9\}$: $\square$*. Enter-right is applied (step 3), producing toplevel split* $(10, \varnothing, \varnothing, \{\{1,9\}\})$*, and the empty clause* $\{1,10\} : \square$ *is derived using clauses* $\{1,10\} : S(y)$ *and* $S(x), P(x) \rightarrow$ *and* $\{1\} : P(a)$*. In step 4, the clause labels* $\{1,9\}$ *and* $\{1,10\}$ *are collapsed into* $\{1\}$*. Finally, two Backjump steps followed by Enter-right yield the last tree, which corresponds to the split stack* $\langle(2, \varnothing, \varnothing, \{\{1\}\})\rangle$*. Observe how the empty clause generated in step 4 allows us to skip branch 4 and jump directly to branch 2, which would not be possible without the Right-collapse rule.*

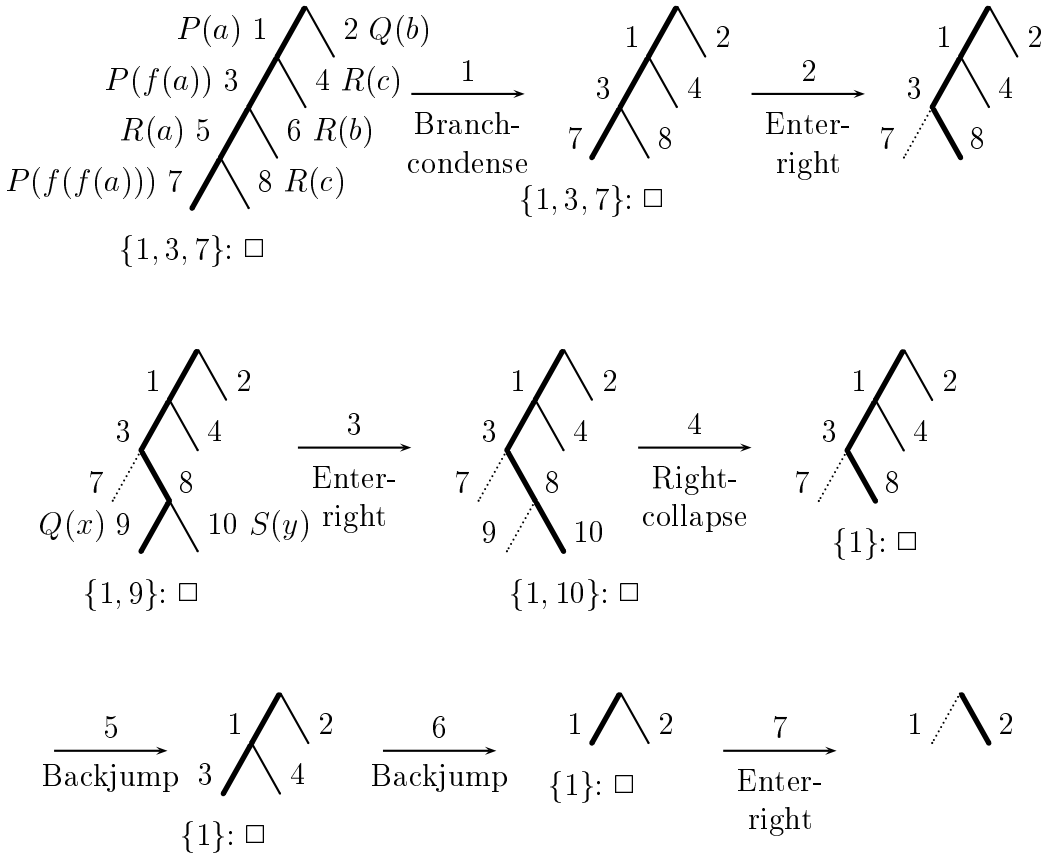

Figure 2.1: Split Tree Development over 3 Backtracking Steps

2.3 Correctness

In this section, we introduce the concept of satisfiability of labelled clause sets, we establish some important properties of derivations in the labelled calculus, and we show that all the transformations of the stack presented in Section 2.2 maintain labelled clause set satisfiability, which implies soundness of the labelled calculus. Finally, we give a completeness result.

A *derivation* in the labelled calculus is a sequence of labelled clause sets

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \Psi_2 : N_2 \rhd \ldots$$

such that $\Psi_0 = \langle\rangle$, N_0 is the initial labelled clause set and for each $i \geq 1$, the labelled clause set $\Psi_i : N_i$ is the result of applying a rule of the calculus to $\Psi_{i-1} : N_{i-1}$. We call the step $\Psi_{i-1} : N_{i-1} \rhd \Psi_i : N_i$ the ith step of $\mathcal{D}$. We write $\Psi : N \rhd^* \Psi' : N'$ to indicate that $\Psi' : N'$ is obtained from $\Psi : N$ by zero or more applications of calculus rules. We use superscripts to make explicit that a split belongs to a particular split stack in a derivation: for example, we write ψ_j^i for the jth split of Ψ_i, and D_j^i for the deleted set of ψ_j^i. Furthermore, we write D^i for $\bigcup_{j=1}^{n} D_j^i$, where n is the length of Ψ_i.

Each ψ_j is the result of a unique splitting step of a clause $\Gamma_1^j, \Gamma_2^j \to \Delta_1^j, \Delta_2^j$ into components $\Gamma_1^j \to \Delta_1^j$ and $\Gamma_2^j \to \Delta_2^j$, since the splitting rule is the only rule extending the stack. For a given stack Ψ, we denote those components by $s_\Psi^1(j)$ and $s_\Psi^2(j)$, respectively, and the "active" component by

$$s_\Psi(j) := \begin{cases} s_\Psi^1(j) & \text{if } \mathrm{left}(l_j) \\ s_\Psi^2(j) & \text{if } \mathrm{right}(l_j). \end{cases}$$

In the following, we omit the subscript whenever Ψ is clear from the context.

Lemma 12 (Existence of normal forms)**.** *A labelled clause set $\Psi : N$ has a unique normal form $\Psi : N{\downarrow}$ under stack reduction rules, if (1) N contains at most one empty clause and that empty clause is label-valid, and (2) Ψ is of finite size.*

Proof. For any given $L : \Box \in \Psi : N$, at most one stack reduction rule is applicable in $\Psi : N$, since the preconditions are mutually exclusive. Thus by condition (1), there is a unique sequence of reductions from $\Psi : N$. Furthermore, for all rules except ENTER-RIGHT, applying the rule to a label-valid $L : \Box$ reduces the stack size by one, whereas ENTER-RIGHT removes the empty clause. Thus by condition (2), the sequence of reductions terminates. □

2.3.1 Satisfiability of Labelled Clause Sets

In order to prove soundness and completeness results for our labelled calculus, we need to extend the notion of satisfiability from clause sets to labelled clause sets. Since we are exploring a tree whose branches represent alternatives of successive case distinctions, we associate a clause set with each unexplored branch on the stack. Formally, for any derivation $\Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \Psi_2 : N_2 \rhd \ldots$ and any labelled clause set $\Psi_i : N_i$ ocurring in it, we define the following set of clause sets:

$$\mathcal{N}_i := \{(N_i \cup D^i)|_{L_k} \cup B_k \mid k \in \{1, \ldots, n\}, L_k = \{l_1, \ldots, l_{k-1}\} \text{ and } \text{left}(l_k)\}.$$

We use the notation N_i^k to denote the set $(N_i \cup D^i)|_L \cup B_k \in \mathcal{N}_i$. We call N_i the *active clause set* of $\Psi_i : N_i$, and $\mathcal{N}_i$ the set of *inactive clause sets* of $\Psi_i : N_i$.

Definition 13 (Satisfiability of labelled clause sets)**.** *We say that $\Psi_i : N_i$ is* satisfiable*, if and only if*

- *N_i is satisfiable, or*
- *some $N_i^k \in \mathcal{N}_i$ is satisfiable.*

Our goal is to show that the rules of the labelled calculus preserve satisfiability of labelled clause sets, i.e., for each step $\Psi_{i-1} : N_{i-1} \rhd \Psi_i : N_i$ in a derivation, $\Psi_{i-1} : N_{i-1}$ is satisfiable if and only if $\Psi_i : N_i$ is satisfiable.

2.3.2 Some Derivation Invariants

Lemma 14 (Existence of leaf markers)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and let $j \in \{1, \ldots, n\}$ be arbitrary. Then $\varphi_j = \varnothing$ whenever* $\text{left}(l_j)$*, and $\varphi_j = \{L\}$ for some L, whenever* $\text{right}(l_j)$*.*

Proof. It is easy to see that the stack reduction rules maintain this property, since leaf markers are set to $\{L\}$ by ENTER-RIGHT and not modified by any other rules. For the calculus rules, the only relevant case is the splitting rule, which only produces left splits with $\varphi = \varnothing$. □

Next we show that clause labels are correctly inherited throughout derivations:

Lemma 15 (Label monotonicity)**.** *Let $\mathcal{D}$ be a derivation, $\Psi_i : N_i$ an arbitrary labelled clause set in $\mathcal{D}$, and $l \in \text{levels}(\Psi_i)$, and let k be maximal such that $k \leq i$ and the splitting rule was applied at step k of $\mathcal{D}$ with l as the split level of the conclusion. Furthermore, let L be the label of the premise at step k. Then for any $L' : C \in N_i \cup D^i$ with $l \in L'$, we have $L \subseteq L'$.*

Proof. It is easy to see that the property is maintained by all stack reduction rules, since no new clauses are added, except for rule RIGHT-COLLAPSE, in which case the statement easily follows from the definition of the new empty clause's label. Now consider the calculus rules. The only interesting case is resolution, but again, it is easy to see that the property is maintained by the definition of the label of the conclusion. □

2.3.3 Label-validity

A basic property we require of all labelled clause sets is *label validity*, which expresses that clause labels only refer to existing splits. The fact that label-validity is invariant in all derivations justifies our use of the notation l_j when referring to an element of a clause label or leaf marker (in the sense that it guarantees the existence of split ψ_j with split level l_j).

Definition 16 (Label-validity)**.** *Let $\Psi_i : N_i$ be a labelled clause set.*

1. *We say that the active set N_i is* label-valid*, if $L \subseteq \text{levels}(\Psi)$ for every $L\colon \Gamma \to \Delta \in N_i$;*
2. *we say that some inactive set $N_i^k \in \mathcal{N}_i$ is* label-valid*, if $L \subseteq \{l_1, \ldots, l_k\}$ for every $L\colon \Gamma \to \Delta \in N_i^k$;*
3. *finally, we say that a leaf-marker $\varphi_j = \{L\}$ is* label-valid*, if $L \subseteq \{l_1, \ldots, l_{j-1}, l_j - 1\}$.*

We call the labelled clause set $\Psi_i : N_i$ label-valid*, if its active set, all its inactive sets and all its leaf markers are label-valid.*

Lemma 17 (Split deletion and label-validity)**.** *Let $\Psi : N$ be an arbitrary labelled clause set, let $m \in \{1, \ldots, n\}$ be arbitrary and assume that for all $j > m$, the parent clause of split j does not depend on l_m, and that $\varphi_j = \varnothing$. Then $\Psi' : N' := \text{delete}(\Psi : N, m)$ is label-valid if $\Psi : N$ is label-valid.*

Proof. 1. Label-validity of $N' = (N \cup D_k \cup \bigcup_{l_j \in R} D_j)|_{\text{levels}(\Psi')}$ is immediate.

2. Let $k \in \{1, \ldots, n-1\} \setminus \{m\}$ be arbitrary. Splitting is the only rule extending the blocked set B'_k, and by definition of the splitting rule, all clauses in B'_k have the label $L \cup \{l_k\}$, where L is the label of the parent clause of split k. Hence by assumption, no clause in B'_k depends on l_m. Therefore $N'^k = (N' \cup D'^i)|_{\text{levels}(\Psi')} \cup B'_k$ is label-valid.

3. Let $k \in \{1, \ldots, n-1\} \setminus \{m\}$ again be arbitrary. If $k < m$, then φ'_k is label-valid by assumption. If $k \geq m$, then $\varphi'_k = \varnothing$ by assumption, and hence φ' is trivially label-valid.

□

Lemma 18 (Stack reductions maintain label-validity)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume that $\Psi_i : N_i \to \Psi'_i : N'_i$. Then $\Psi'_i : N'_i$ is label-valid if $\Psi_i : N_i$ is label-valid.*

Proof. Assume $\Psi_i : N_i$ is label-valid. We distinguish which rule was applied to obtain $\Psi'_i : N'_i$.

Backjump: Follows directly with Lemma 17.

Branch-condense: Follows again with Lemma 17. Note that the assumptions of Lemma 17 are fulfilled: Since $l_{k_{max}}$ is the greatest split level (below the last backtracking level) that the empty clause doesn't depend on, it follows by Lemma 15 that the parent clauses of all splits below k_{max} don't depend on $l_{k_{max}}$ either. Furthermore, since for all $j > k_{max}$, it holds that left(l_j), we know by Lemma 14 that $\varphi_j = \varnothing$.

Right-collapse: Like in the Backjump-case, Lemma 17 guarantees label-validity of all clauses, except the new empty clause L: □. Hence we need to show that $L \subseteq \text{levels}(\Psi')$. But this is obvious, since by assumption, both L_1 and L_2 are subsets of levels(Ψ), and by the definition of L, both $l_n - 1 \notin L$ and $l_n \notin L$

Enter-right: We again use Lemma 17, and note that the new leaf marker is label-valid, since the empty clause L: □ was label-valid by assumption. □

We now show that label-validity is an invariant of any derivation.

Proposition 19 (Label-validity in derivations)**.** *Any labelled clause set in any derivation is label-valid.*

Proof. Let $\Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$ be an arbitrary derivation. We show that any $\Psi_i : N_i$ is label-valid by induction over the derivation. Clearly, $\Psi_0 : N_0$ is label-valid, since Ψ_0 is empty. For the inductive case, we assume $\Psi_{i-1} : N_{i-1}$ is label-valid and distinguish which calculus rule was applied to obtain $\Psi_i : N_i$. For backtracking, the result follows with Lemma 18. In the

case of splitting, we know by induction hypothesis that $L \subseteq \text{levels}(\Psi_{i-1})$, hence also $L' \subseteq \text{levels}(\Psi_i)$ and therefore N_i is label-valid since N_{i-1} is label-valid by induction hypothesis. Label-validity of the N_i^k and the leaf markers is obvious. For resolution, note that the union of two valid labels is again valid. Finally, the subsumption deletion and factoring cases are trivial. □

2.3.4 Path-validity

We now define a property of labelled clause sets, *path-validity*, which states that the clauses in the active and deleted sets follow logically from the initial clause set and all active split clauses, and that the initial clause set together with the active split clauses described by a leaf marker is unsatisfiable.

Definition 20 (Path-validity)**.** *Let $\Psi_i : N_i$ be a labelled clause set in a derivation. We call $\Psi_i : N_i$* path-valid*, if*

1. $N_0 \cup \bigcup_{l_j \in L} s(j) \models C$ *for every* $L : C \in N_i \cup D^i$*, and*
2. $N_0 \cup \bigcup_{l_j \in L \setminus \{l_k - 1\}} s(j) \cup \bigcup_{l_j \in L \cap \{l_k - 1\}} s^1(j) \models \bot$ *for every* $\varphi_k = \{L\}$.

Lemma 21 (Stack reductions maintain path-validity)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume that $\Psi_i : N_i \to \Psi_i' : N_i'$. Then $\Psi_i' : N_i'$ is path-valid if all $\Psi_j : N_j$ are path-valid, for $j \in \{0, \ldots, i\}$.*

Proof. Assume all $\Psi_j : N_j$ are path-valid, for $j \in \{0, \ldots, i\}$. We distinguish which rule was applied to obtain $\Psi_i' : N_i'$. The cases BACKJUMP and BRANCH-CONDENSE follow immediately by assumption, since $(N_i' \cup \bigcup_{j=1}^{n-1} D_j') \subseteq (N_i \cup \bigcup_{j=1}^{n} D_j)$, and leaf markers are not extended.

RIGHT-COLLAPSE: We need to show that path-validity is maintained by the addition of the new empty clause $L : \square$, that is, we have to show $N_0 \cup \bigcup_{l_j \in L} \{s(j)\} \models \bot$ for $L = L_1 \cup L_2 \setminus \{l_n - 1, l_n\}$. Assume the kth step of the derivation was the splitting step that produced split n and let L' be the label of the parent clause. By path-validity of $\Psi_{k-1} : N_{k-1}$, we know that

$$N_0 \cup \bigcup_{l_j \in L'} \{s(j)\} \models s^1(n) \vee s^2(n)$$

By path-validity of $\Psi_i : N_i$, we know that

$$N_0 \cup \bigcup_{l_j \in L_1 \setminus \{l_n\}} \{s(j)\} \cup \{s^1(n)\} \models \bot$$

and

$$N_0 \cup \bigcup_{l_j \in L_2 \setminus \{l_n\}} \{s(j)\} \cup \{s^2(n)\} \models \bot.$$

Or, equivalently:

$$N_0 \cup \bigcup_{l_j \in L_1 \setminus \{l_n\}} \{s(j)\} \models \neg s^1(n)$$

and

$$N_0 \cup \bigcup_{l_j \in L_2 \setminus \{l_n\}} \{s(j)\} \models \neg s^2(n).$$

By label monotonicity, $L' \subseteq L_1$ and $L' \subseteq L_2$, therefore

$$N_0 \cup \bigcup_{l_j \in L} \{s(j)\} \models (s^1(n) \vee s^2(n)) \wedge \neg s^1(n) \wedge \neg s^2(n)$$

or, equivalently:

$$N_0 \cup \bigcup_{l_j \in L} \{s(j)\} \models \bot.$$

ENTER-RIGHT: By assumption that $N_0 \cup \bigcup_{l_j \in L} D_j \models \bot$, for the empty clause L: $\square$. This immediately implies statement 2, since the only new leaf marker is $\varphi'_n = \{L\}$. Let us show that the blocked clauses $B'_n = \{s^2(n)\}$ are also path-valid: Again let L': $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$ be the parent clause of split n. By assumption, $N_0 \cup \bigcup_{l_j \in L'} s(j) \models s^1(n) \vee s^2(n)$. Since $l_n \in L$, we know that $N_0 \cup \bigcup_{l_j \in L'} s(j) \cup \{s^1(n)\} \models \bot$ and hence $N_0 \cup \bigcup_{l_j \in L'} s(j) \models s^2(n)$. $\square$

Proposition 22 (Path-validity in derivations)**.** *Any labelled clause set in any derivation is path-valid.*

Proof. Let $\Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$ be an arbitrary derivation. We show that any $\Psi_i : N_i$ is path-valid by induction over the derivation. Clearly, $\Psi_0 : N_0$ is path-valid. For the inductive case, we assume all $\Psi_j : N_j$ are path-valid, for $j \in \{0, \ldots, i-1\}$, and distinguish which calculus rule was applied to obtain $\Psi_i : N_i$.

- For splitting, we are adding the left split clause to the active set, so $s(n) = s^1(n) \in N_i$ and path-validity is trivially maintained.
- For backtracking, the statement follows by induction hypothesis and Lemma 21.
- Since resolution is sound, we know that the conclusion follows logically from the premises. The statement then follows from the fact that the conclusion's label is the union of both premises' labels.

- For subsumption deletion, the statement follows from the fact that

$$N_i \cup D^i \subseteq N_{i-1} \cup D^{i-1}$$

  and induction hypothesis.

- The factoring case is trivial, since the conclusion follows logically from the premise.

□

Corollary 23 (Restriction expansion)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and let $M \subseteq \text{levels}(\Psi_i)$ and $k \in \{1, \ldots, n\}$. If $(N_i \cup D^i)|_M \cup \{s(k)\}$ is satisfiable, then $(N_i \cup D^i)|_{M \cup \{l_k\}}$ is satisfiable.*

Proof. Let $L : C \in (N_i \cup D^i)|_{M \cup \{l_k\}} \setminus (N_i \cup D^i)|_M \cup \{s(k)\}$. By Proposition 22, $L : C$ follows logically from $(N_i \cup D^i)|_M \cup \{s(k)\}$. □

The following proposition follows trivially from the definition of subsumption deletion:

Proposition 24 (Existence of a subsumption deletion step)**.** *Let $\mathcal{D}$ be a derivation and $\Psi_k : N_k$ an arbitrary labelled clause set in $\mathcal{D}$. For each $L : C$ in any D^k_j, there exists $k' \leq k$ such that subsumption deletion was applied at step k' of $\mathcal{D}$ with $L : C$ the subsumed clause, and a subsuming clause with split level l_j.*

Proof. By induction over the derivation, and induction over sequences of stack reductions for the backtracking case: For Ψ_0, all D_j are empty, and subsumption deletion is the only rule extending sets of deleted clauses, no stack reduction rule extends sets of deleted clauses. □

Proposition 25 (Existence of subsuming clauses)**.** *Let $\mathcal{D}$ be a derivation, $\Psi_k : N_k$ an arbitrary labelled clause set in $\mathcal{D}$, and $L : C \in D^k_j$ for some j. Furthermore, let k' be maximal such that subsumption deletion was applied at step k' of $\mathcal{D}$ with $L : C$ as subsumed clause, and a subsuming clause with split level l_j. Then for all $\Psi_i : N_i$ with $i \in \{k', k'+1, \ldots, k\}$:*

1. *$L : C \in D^i_{j^i}$, where $l^i_{j^i} = l^k_j$, and*
2. *there exists a clause $L' : C'$ with split level l_j, such that $L' : C'$ subsumes $L : C$ and either $L' : C' \in N_i$ or $L' : C' \in D^i_l$ for some $l \in \text{levels}(\Psi_i)$. Furthermore, if $L' : C' \in D^i_l$, then it holds that $h > k'$, where h is maximal such that subsumption deletion was applied at step h of $\mathcal{D}$ with $L' : C'$ as subsumed clause, and there exists a clause $L'' : C'' \in N_i$ that subsumes $L : C$.*

Proof. First of all, note that k' exists, by Proposition 24. We first show 1. By definition of subsumption deletion, $L : C \in D^{k'}_{j'}$ where $l^{k'}_{j'} = l^k_j$. Assume for contradiction that there exists some $i \in \{k'+1, \ldots, k-1\}$ such that $L : C \notin D^i_{j^i}$ for $l^i_{j^i} = l^k_j$. Since $L : C \in D^k_j$, and subsumption deletion is the only rule extending the deleted sets, one of the steps $k'+1, \ldots, k$ must have been a subsumption step with $L : C$ as subsumed clause and a subsuming clause with split level l_j. This is a contradiction to the maximality of k'.

Let us now show 2. We first show the property is maintained by all stack reduction rules. So let $i \in \{k', \ldots, k-1\}$ and assume there exists a clause $L' : C'$ with split level l_j, such that $L' : C'$ subsumes $L : C$ and either $L' : C' \in N_i$ or $L' : C' \in D^i_l$ for some $l \in \text{levels}(\Psi_i)$, and assume $\Psi_i : N_i \to \Psi'_i : N'_i$. Note that any stack reduction rule invokes delete$(\Psi_i : N_i, m)$ for some $m \in \text{levels}(\Psi_i)$ (e.g., $m = k_{max}$ for BRANCH-CONDENSE). We know that $l_m \notin L'$, since otherwise we would have $D'_{j'} = \varnothing$ (where $l'_{j'} = l_j$) by definition of delete$(\Psi_i : N_i, m)$, a contradiction to 1. So $L' : C'$ does not depend on l_m, and hence if $L' : C' \in N_i$, then also $L' : C' \in N'_i$. On the other hand, if $L' : C' \in D^i_l$, then either $L' : C' \in D'_l$, or $L' : C' \in N'_i$ if $l \in R \cup \{l_m\}$.

We now show that the property indeed holds for all $\Psi_i : N_i$ with $i \in \{k', k'+1, \ldots, k\}$, by induction on i. We have already shown that it holds for $i = k'$. For the inductive step, assume it holds for $i-1$. The cases splitting, resolution and factoring are trivial, and the backtracking case follows from the above discussion. In the case of subsumption deletion, it is easy to see that the property is maintained if the subsumed clause does not itself subsume $L : C$. So let $h = i$ be maximal such that step h of $\mathcal{D}$ is subsumption deletion with $L' : C'$ as subsumed clause, $L'' : C''$ as subsuming clause, let l_m be the split level of $L'' : C''$ and assume that $L' : C'$ subsumes $L : C$. Observe that If $l_m \neq \max(L')$, then $L' : C' \in D^h_m$, by definition of subsumption deletion, and so property 2. holds. On the other hand, if $l_m = \max(L')$, then $L' : C'$ is removed. But since the subsuming clause also subsumes $L : C$ and is in N_h, property 2 still holds. □

Corollary 26 (Active clause sets and deleted clauses)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, where Ψ_i has length n. Then N_i is satisfiable if and only if $N_i \cup D^i$ is satisfiable.*

Proof. For the forward direction, we know by Proposition 25 that for every clause $L : C$ in any D_j, there exists a subsuming clause in N_i. Hence $N_i \cup D^i$ is satisfiable if N_i is satisfiable. The backward direction is trivial. □

Lemma 27 (Split deletion and inactive clause sets)**.** *Let $\Psi : N$ be an arbitrary labelled clause set with Ψ of length n, and let $m \in \{1, \ldots, n\}$ be such that for all $j > m$, the parent clause of split j does not depend on l_m, and let*

$\Psi' : N' = \text{delete}(\Psi : N, m)$. *Then for all* $k \in \{1, \ldots, n-1\}$,

$$N'^k = \begin{cases} N^k & \text{if } k < m \\ N^{k+1}|_{\overline{\{l_m\}}} & \text{if } k \geq m. \end{cases}$$

Proof. Assume $k < m$, and let $L := \{l'_1, \ldots, l'_{k-1}\} = \{l_1, \ldots, l_{k-1}\}$. Then

$$\begin{aligned}
N'^k &= (N' \cup \bigcup_{j=1}^{n-1} D'_j)|_L \cup B'_k \\
&= \left((N \cup D_m \cup \bigcup_{l_j \in R} D_j)|_{\text{levels}(\Psi')} \cup \bigcup_{j=1}^{n-1} D'_j \right) \Bigg|_L \cup B_k && (1) \\
&= \left((N \cup D_m \cup \bigcup_{l_j \in R} D_j \cup \bigcup_{j=1}^{m-1} D_j \cup \bigcup_{j=m+1}^{n} D_j)|_{\text{levels}(\Psi')} \right) \Bigg|_L \cup B_k && (2) \\
&= (N \cup \bigcup_{j=1}^{n} D_j)|_L \cup B_k && (3) \\
&= N^k.
\end{aligned}$$

Equation (1) is obtained by plugging in the definition of N', and observing that $B'_k = B_k$. For equation (2), we use the fact that $\bigcup_{j=1}^{n-1} D'_j = \bigcup_{j=1}^{m-1} D_j|_{\text{levels}(\Psi')} \cup \bigcup_{j=m+1}^{n} D_j|_{\text{levels}(\Psi')}$ because of level shifting. Finally, equation (3) follows from the fact that $L \subseteq \text{levels}(\Psi')$.

Now assume $k \geq m$, and let $L := \{l'_1, \ldots, l'_{k-1}\}$. Note that

$$L = \begin{cases} \{l_1, \ldots, l_{m-1}\} & \text{if } k = m \\ \{l_1, \ldots, l_{m-1}, l_{m+1}, \ldots, l_k\} & \text{if } k > m \end{cases}$$

or, in other words, $L = \{l_1, \ldots, l_k\} \setminus \{l_m\}$. Hence

$$\begin{aligned}
N'^k &= (N' \cup \bigcup_{j=1}^{n-1} D'_j)|_L \cup B'_k \\
&= \left((N \cup D_m \cup \bigcup_{l_j \in R} D_j \cup \bigcup_{j=1}^{m-1} D_j \cup \bigcup_{j=m+1}^{n} D_j)|_{\text{levels}(\Psi')} \right) \Bigg|_L \cup B_{k+1} && (1) \\
&= (N \cup \bigcup_{j=1}^{n} D_j)|_{\{l_1, \ldots, l_k\} \setminus \{l_m\}} \cup B_{k+1} && (2) \\
&= N^{k+1}|_{\overline{\{l_m\}}}.
\end{aligned}$$

We again use $L \subseteq \text{levels}(\Psi')$ for equation (2), and the assumption that the splits below m do not depend on l_m. □

2.3.5 Soundness

We can now tackle the preservation of labelled clause set satisfiability, as discussed in Section 2.3.1. We again begin by showing that satisfiability is preserved by each of the stack reduction rules.

Lemma 28 (BACKJUMP maintains satisfiability)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume $\Psi_i : N_i \to \Psi_i' : N_i'$ with rule* BACKJUMP*. Then $\Psi_i : N_i$ is satisfiable if and only if $\Psi_i' : N_i'$ is satisfiable.*

Proof. Clearly, N_i is unsatisfiable, since $L\colon \square \in N_i$. Furthermore, if $N_i^n \in \mathcal{N}_i$ exists – this is the case if $l_n = left$ – then $L\colon \square \in N_i^n$, since $l_n \notin L$, and thus N_i^n is unsatisfiable. Hence $\Psi_i : N_i$ is satisfiable if and only if some $N_i^k \in \mathcal{N}_i$ is satisfiable, for $k < n$. On the other hand, $L\colon \square \in N_i'$ since $l_n \notin L$, thus N_i' is also unsatisfiable. Therefore, $\Psi_i' : N_i'$ is satisfiable if and only if some $N_i'^k \in \mathcal{N}_i'$ is satisfiable. The statement then follows directly with Lemma 27. □

Lemma 29 (BRANCH-CONDENSE maintains satisfiability)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume $\Psi_i : N_i \to \Psi_i' : N_i'$ with rule* BRANCH-CONDENSE*. Then $\Psi_i : N_i$ is satisfiable if and only if $\Psi_i' : N_i'$ is satisfiable.*

Proof. Again, N_i is unsatisfiable, since $L\colon \square \in N_i$. Because $l_{k_{max}} \notin L$, we also know $L\colon \square \in N_i'$, and thus N_i' is unsatisfiable. By Lemma 27, we know that $N_i^k = N_i'^k$ for $k < k_{max}$. Hence it suffices to show that there exists a satisfiable $N_i^k \in \mathcal{N}_i$ with $k \in \{k_{max}, \ldots, n\}$ if and only if there exists a satisfiable $N_i'^{k'} \in \mathcal{N}_i'$ with $k' \in \{k_{max}, \ldots, n-1\}$. Also note that since $b_k = left$ for all $k \in \{k_{max}, \ldots, n\}$, we have $\mathcal{N}_i = \{N_i^{k_{max}}, N_i^{k_{max}+1}, \ldots, N_i^n\}$.

For the forward direction, let us first assume that $N_i^{k_{max}}$ is satisfiable. We show that there exists a satisfiable $N_i'^k \in \mathcal{N}_i'$ with $k \in \{k_{max}, \ldots, n-1\}$, by induction on k. For each k, we show that either

1. N'^k is satisfiable, or
2. $(N_i \cup D^i)|_{\{l_1,\ldots,l_{k+1}\}\setminus\{l_{k_{max}}\}}$ is satisfiable.

Since we know $(N_i \cup D^i)|_{\{l_1,\ldots,l_n\}\setminus\{l_{k_{max}}\}}$ is unsatisfiable, it follows that there must be a satisfiable $N_i'^k \in \mathcal{N}_i'$.
We have assumed $N_i^{k_{max}} = (N_i \cup D^i)|_{\{l_1,\ldots,l_{k_{max}-1}\}} \cup B_{k_{max}}$ to be satisfiable,

hence also $N_i \cup D^i|_{\{l_1,\ldots,l_{k_{max}-1}\}}$ is satisfiable – this provides the induction base.
For the inductive step, let $k \geq k_{max}$ and assume that

$$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}}$$

is satisfiable. Assume parent clause of the $k+1$st split of Ψ'_i (or the kth split of Ψ_i) was $L' : s^1(k+1) \vee s^2(k+1)$. Since $l_{k+1} \in L$, we know by Lemma 15 that $L' \subseteq L$. Since $l_{k_{max}} \notin L$, we also have $l_{k_{max}} \notin L'$. But then, by path-validity (Proposition 22), we know that

$$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \models s^1_{\Psi_i}(k+1) \vee s^2_{\Psi_i}(k+1).$$

Hence either

1. $$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \cup \{s^1_{\Psi_i}(k+1)\}$$
   is satisfiable, but then by Corollary 23, also
   $$(N_i \cup D^i)|_{\{l_1,\ldots,l_k,l_{k+1}\}\setminus\{l_{k_{max}}\}}$$
   is satisfiable.

2. Or
   $$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \cup \{s^2_{\Psi_i}(k+1)\} \;=\; N_i^{k+1}|_{\overline{\{l_{k_{max}}\}}}$$
   is satisfiable. By Lemma 27, we have $N_i^{k+1}|_{\overline{\{l_{k_{max}}\}}} = N_i'^k$, hence $N_i'^k$ is satisfiable.

Still for the forward direction, let us now assume that some N_i^k is satisfiable, for $k \in \{k_{max}+1,\ldots,n\}$. Then it immediately follows with Lemma 27 that $N_i'^{k-1} = N_i^k|_{\overline{\{l_{k_{max}}\}}}$ is satisfiable.

Let us now prove the backward direction. So assume $N_i'^k$ is satisfiable, for some $k \in \{k_{max},\ldots,n-1\}$. Again by Lemma 27,

$$N_i'^k \;=\; N_i^{k+1}|_{\overline{\{l_{k_{max}}\}}} \;=\; (N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \cup B_{k+1}$$

is satisfiable. By path-validity (Proposition 22), we know that

$$(N_i \cup D^i)|_{\{l_1,\ldots,l_{k_{max}-1}\}} \models s^1_{\Psi_i}(k_{max}) \vee s^2_{\Psi_i}(k_{max}).$$

Hence we again distinguish two cases:

1. Either
$$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \cup \{s^1_{\Psi_i}(k_{max})\} \cup B_{k+1}$$
is satisfiable. But then also
$$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}} \cup B_{k+1} \;=\; N_i^{k+1}$$
is satisfiable.

2. Or
$$(N_i \cup D^i)|_{\{l_1,\ldots,l_k\}\setminus\{l_{k_{max}}\}} \cup \{s^2_{\Psi_i}(k_{max})\} \cup B_{k+1}$$
hence also
$$(N_i \cup D^i)|_{\{l_1,\ldots,l_{k_{max}-1}\}} \cup B_{k_{max}}$$
is satisfiable, since $B_{k_{max}} = \{s^2_{\Psi_i}(k_{max})\}$.

□

Lemma 30 (RIGHT-COLLAPSE maintains satisfiability)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume $\Psi_i : N_i \to \Psi_i' : N_i'$ with rule* RIGHT-COLLAPSE*. Then $\Psi_i : N_i$ is satisfiable if and only if $\Psi_i' : N_i'$ is satisfiable.*

Proof. First note that L_2: $\square \in N_i$ and L: $\square \in N_i'$, hence both N_i and N_i' are unsatisfiable. Thus it suffices to show that there exists a satisfiable $N_i'^{k'} \in \mathcal{N}_i'$ if and only if there is a satisfiable $N_i^k \in \mathcal{N}_i$. By Lemma 27 and definition of RIGHT-COLLAPSE, we obtain that for all $N_i'^k \in \mathcal{N}_i'$,
$$N_i'^k = \begin{cases} N_i^k \cup \{L\text{: }\square\} & \text{if } k > \max(L) \\ N_i^k & \text{otherwise.} \end{cases}$$
For the first case, use the fact that for $k > \max(L)$, we have $L \subseteq \{1, \ldots, k\}$, and path-validity (Proposition 22). The second case is immediate. □

Lemma 31 (ENTER-RIGHT maintains satisfiability)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume $\Psi_i : N_i \to \Psi_i' : N_i'$ with rule* ENTER-RIGHT*. Then $\Psi_i : N_i$ is satisfiable if and only if $\Psi_i' : N_i'$ is satisfiable.*

Proof. Again because of the empty clause, N_i is unsatisfiable. Furthermore, by Lemma 27, $N_i'^k = N_i^k$ for $k < n$. Thus it suffices to show that N_i^n is satisfiable if and only if N_i' is satisfiable. The forward direction is immediate since $N_i' \subseteq N_i^n$. For the backward direction, assume that N_i' is satisfiable. We show that then N_i^n must also be satisfiable. So let $L : C \in N_i^n \setminus N_i'$ be

arbitrary but fixed. Let us show that $L : C$ is redundant with respect to N_i'. We inductively define integers j_l and k_l, and clauses $L_l : C_l$ for $l \geq 1$ as follows: By definition of N_i^n and N_i', we know that $L : C \in D_{j_1}$ such that $l_{j_1} \notin R \cup \{l_n\}$, and $l_n \notin L$. Let k_1 be the step of the derivation where $L : C$ was subsumed (k_1 exists by Proposition 24). By Proposition 25, there exists a clause $L_1 : C_1 \in N_i \cup D^i$ such that $L_1 : C_1$ subsumes $L : C$ and $\max(L_1) = l_{j_1}$.

- If $l_n \in L_1$, then we know $L : C \in N_i'$, by definition of *delete_split*.
- Otherwise, if $l_n \notin L_1$
  - If $L_1 : C_1 \in N_i$, then also $L : C \in N_i'$, by definition of *delete_split*.
  - Otherwise, $L_1 : C_1 \in D_{j_2}$. Let k_2 be the step of the derivation where $L_1 : C_1$ was subsumed. By Proposition 25, we know that $k_2 > k_1$, and that there exists $L_2 : C_2$ with $\max(L_1) = l_{j_1}$, such that $L_2 : C_2$ subsumes $L_1 : C_1$ and hence also subsumes $L : C$

Clearly, some k_m will be the last subsumption step in the derivation where $L_{m-1} : C_{m-1} \in D_{j_m}$. The clause $L_m : C_m$ subsumes $L_{m-1} : C_{m-1}$ and hence also subsumes $L : C$, but $L_m : C_m$ was not itself subsumed. Therefore, $L_m : C_m \in N_i$.

- If $l_n \in L_m$, then $L_{m-1} : C_{m-1} \in N_i'$, by definition of *delete_split*.
- Otherwise $l_n \notin L_m$, but then $L_m : C_m \in N_i'$, by definition of *delete_split*.

Both $L_{m-1} : C_{m-1}$ and $L_m : C_m$ subsume $L : C$, hence in both cases, the clause $L : C$ is redundant with respect to N_i'. □

Proposition 32 (Calculus rules maintain satisfiability)**.** *Let $\Psi_i : N_i$ be an arbitrary labelled clause set in a derivation, and assume $\Psi_i : N_i \rhd \Psi_{i+1} : N_{i+1}$. Then $\Psi_i : N_i$ is satisfiable if and only if $\Psi_{i+1} : N_{i+1}$ is satisfiable.*

Proof. Let n be the length of Ψ_i. We distinguish which rule was applied to obtain $\Psi_{i+1} : N_{i+1}$.

- The cases resolution and factoring are trivial, since the conclusion follows logically from the premises.

- For the splitting case, let us first observe that Ψ_{i+1} has length $n+1$, and that for all $1 \leq k \leq n$,

$$\begin{aligned} N_{i+1}^k &= (N_{i+1} \cup \bigcup_{j=1}^{n+1} D_j^{i+1})|_{\{l_1,\ldots,l_{k-1}\}} \cup B_k^{i+1} \\ &= (N_i \cup \bigcup_{j=1}^{n} D_j^i)|_{\{l_1,\ldots,l_{k-1}\}} \cup B_k^i \\ &= N_i^k. \end{aligned}$$

  Thus it suffices to show that N_i is satisfiable if and only if N_{i+1} or N_{i+1}^{n+1} is satisfiable. By definition of the splitting rule, we have $N_{i+1} = N_i \cup \{L' \colon \Gamma_1 \to \Delta_1\}$ and

$$\begin{aligned} N_{i+1}^{n+1} &= (N_{i+1} \cup \bigcup_{j=1}^{n+1} D_j^{i+1})|_{\{l_1,\ldots,l_n\}} \cup B_{n+1}^{i+1} \\ &= (N_i \cup \bigcup_{j=1}^{n} D_j^i)|_{\{l_1,\ldots,l_n\}} \cup \{L' \colon \Gamma_2 \to \Delta_2\} \end{aligned}$$

  since $l_n + 1 \in L'$ and thus $N_{i+1}|_{\{l_1,\ldots,l_n\}} = N_i$, and the sets of deleted clauses are not changed by splitting. By Corollary 26, satisfiability of $N_i \cup \bigcup_{j=1}^{n} D_j^i$ is equivalent to satisfiability of N_i, hence N_{i+1}^{n+1} is satisfiable if and only if $N_i \cup \{L' \colon \Gamma_2 \to \Delta_2\}$. Therefore, it suffices to show that N_i is satisfiable if and only if $N_i \cup \{L' \colon \Gamma_1 \to \Delta_1\}$ or $N_i \cup \{L' \colon \Gamma_2 \to \Delta_2\}$ is satisfiable, and this is immediate since $\Gamma_1 \to \Delta_1$ and $\Gamma_2 \to \Delta_2$ are variable-disjoint.

- For backtracking, we use induction over the stack reductions together with Lemmata 28 to 31.

- For subsumption deletion, observe that N_i and N_{i+1} are equisatisfiable, since the subsuming clause $L_1 \colon \Gamma_1 \to \Delta_1$ is still in N_{i+1}.

□

Theorem 33 (Soundness)**.** *Let N be an arbitrary clause set. Let $N_0 := \{\varnothing \colon C \mid C \in N\}$ be the associated set of labelled clauses, let $\Psi_0 := \langle\rangle$, and let $\Psi_0 : N_0 \rhd^* \Psi_m : N_m$ be an arbitrary derivation starting with $\Psi_0 : N_0$. Then $\Psi_m : N_m$ is satisfiable if $\Psi_0 : N_0$ is satisfiable.*

Proof. By induction over the derivation, using Proposition 32. □

2.3.6 Completeness

When backtracking is modelled explicitly, as we have done here, classical model construction techniques (as in [2, 16]) cannot directly be used to show completeness of the calculus. In particular, defining a fair derivation to be a derivation in which any non-redundant inference from persistent clauses is eventually computed, no longer guarantees that any fair derivation from an inconsistent clause set eventually yields a contradiction. The reason is that in our calculus, the changes to the clause set are not monotonic (in the sense that in each step, only derived clauses are added or redundant clauses are removed).

For example, consider the unsatisfiable clause set

$$\begin{array}{lll} N_0 = \{ & S(a); & \neg S(a); \quad P(a); \\ & P(x) \to Q(y), P(f(x)); & \\ & Q(y) \to S(x) & \} \end{array}$$

where all clauses have the empty label. We construct an infinite derivation $\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$ as follows: we apply resolution to derive a clause $\to Q(y), P(f^{n+1}(a))$ (initially, $n = 0$) which we then split. In the left branch, we use $Q(y)$ to infer $S(x)$. We then apply subsumption deletion to $S(x)$ and $S(a)$, removing $S(a)$ from the clause set and storing it in the deleted set of the current split. We apply resolution to infer the empty clause from $S(x)$ and $\neg S(a)$ and backtrack, entering the right branch, reinserting clause $S(a)$. We then repeat the procedure with n increased by one (see Figure 2.2). The classical definition of persistent clauses as $N_\infty = \bigcup_i \bigcap_{j \geq i} N_j$ yields a set that does not contain $S(a)$ (thus N_∞ is satisfiable), because for each subsumption deletion step k, we have $S(a) \notin N_k$, hence $S(a)$ is not "persistent" when viewed over the whole derivation.

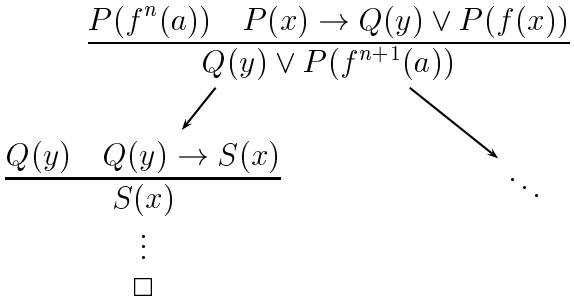

Figure 2.2: Split Tree Development

In the above example, the non-persistent clause $S(a)$ is redundant in all left branches. However, every left branch is refuted at some point, causing

$S(a)$ to be reinserted upon backtracking. Thus, the fact that the clause is redundant only in branches that are eventually closed is irrelevant in an infinite derivation. In the example, the split tree has an infinite path consisting of right branches, and along this path, the clause $S(a)$ is not redundant and should therefore eventually be considered for inferences. Our goal is therefore to define a notion of fairness that ignores closed branches and only talks about clauses on the infinite path of the split tree. In the following, we use $\mathrm{suf}_{\mathcal{D}}(i)$ to denote the suffix $\Psi_i : N_i \rhd \Psi_{i+1} : N_{i+1} \rhd \ldots$ of an infinite derivation $\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$. Note that $\mathrm{suf}_{\mathcal{D}}(0) = \mathcal{D}$.

Definition 34 (Persistent Split)**.** *Given an infinite derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots,$$

and $i \geq 1$, where $\Psi_i = \langle \psi_{n_i}, \ldots, \psi_1 \rangle$, we call ψ_k $(1 \leq k \leq n_i)$ persistent in $\mathrm{suf}_{\mathcal{D}}(i)$*, if $l_k \in \mathrm{levels}(\Psi_j)$ for all $j \geq i$.*

Definition 35 (Weakly Persistent Clause)**.** *Given an infinite derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots,$$

and $i \geq 0$, where $\Psi_i = \langle \psi_{n_i}, \ldots, \psi_1 \rangle$, we call a clause $L : C \in (N_i \cup D^i)$ weakly persistent in $\mathrm{suf}_{\mathcal{D}}(i)$*, if for all $l_j \in L$, ψ_j is persistent in* $\mathrm{suf}_{\mathcal{D}}(i)$*.*

Weakly persistent clauses are not necessarily contained in every clause set of the derivation from some point on. However, if a clause is weakly persistent, then from some point on, every clause set contains either the clause itself, or some clause that subsumes it. Also note that any clause that is weakly persistent in $\mathcal{D}$ is contained in N_0, since no split is persistent in $\mathcal{D}$.

Lemma 36. *For any infinite derivation $\mathcal{D}$ and for $i \geq 1$, if a clause $L : C \in (N_i \cup D^i_j) \setminus N_0$ is weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(i)$*, then it is also weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(k)$ *where $k \leq i$ is maximal such that step k of $\mathcal{D}$ has $L : C$ as its conclusion.*

Proof. Let k be defined as above and assume $L : C$ is not weakly persistent in $\mathrm{suf}_{\mathcal{D}}(k)$. Then there exists $k \leq k' < i$ such that some split with level in L was deleted in step k' of $\mathcal{D}$. Thus by Definition 5, $L : C \notin (N_{k'} \cup D^{k'})$. Hence there must exist $k'' > k$ such that step k'' has $L : C$ as its conclusion, a contradiction. □

Also note that Definition 35 implies that any clause that is weakly persistent in $\mathcal{D}$ is contained in N_0, since no split is persistent in $\mathcal{D}$.

Definition 37 (Persistent Step)**.** *Given an infinite derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots,$$

we say that step $i \geq 1$ of $\mathcal{D}$ is persistent*, if*

1. *step i is a splitting or backtracking step, $\Psi_i = \langle \psi_{n_i}, \ldots, \psi_1 \rangle$, and ψ_{n_i} is persistent in* $\mathrm{suf}_{\mathcal{D}}(i)$*, or*

2. *step i is an inference or reduction step and all premises of the applied inference or reduction rule are weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(i-1)$*.*

Definition 38 ($\mathrm{spt}_{\mathcal{D}}$)**.** *Given an infinite derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots,$$

let $i \geq 0$ be a backtracking step, let l be the split level of the toplevel split of Ψ_i, and let $k < i$ be maximal such that step k of $\mathcal{D}$ is a splitting step producing a split of level $l-1$. Then we define $\mathrm{spt}_{\mathcal{D}}(i) := k$*, the* associated splitting step *of backtracking step i.*

Lemma 39. *For any infinite derivation $\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$, if step $i \geq 1$ is a persistent splitting step with parent clause $L : C$ (a persistent backtracking step with $L : C$ the parent clause of the associated splitting step), then $L : C$ is weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(i-1)$*.*

Proof. Let l denote the level of the split produced at step i. Assume $L : C$ is not weakly persistent. Let k be minimal such that $k > i$ and $l' \notin \mathrm{levels}(\Psi_k)$ for some $l' \in L$. Then step k of $\mathcal{D}$ is a backtracking step. Since the split produced at step i is persistent, the split with level l' must have been removed by application of Branch-condense. Hence there is an empty clause L': $\square \in N_{k-1}$ with $l \in L'$ but $l' \notin L'$, a contradiction to Lemma 15. □

Lemma 40. *For any infinite derivation $\mathcal{D}$ and for $i \geq 1$, if the conclusion of step i is weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(i)$*, then all premises of step i are weakly persistent in* $\mathrm{suf}_{\mathcal{D}}(i-1)$*.*

Proof. Follows from Definition 35 and Lemma 39. □

We will define the limit of an infinite derivation $\mathcal{D}$ to be the derivation obtained by starting from the original labelled clause set and applying exactly the persistent steps of $\mathcal{D}$. Note that all left splits are created by splitting, whereas all right splits are created by backtracking. The limit will contain no more applications of backtracking. Instead, whenever a persistent right branch is created in the original derivation, the limit will enter that branch directly, using the rule *splitting right*:

Definition 41 (Splitting Right)**.** *The inference*

$$\mathcal{I}\frac{\Psi : N \quad L : \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Psi' : N \qquad L' : \Gamma_2 \to \Delta_2}$$

where $\Psi = \langle \psi_n, \ldots, \psi_1 \rangle$, $L' = L \cup \{l_n + 1\}$, $\Psi' = \langle \psi_{n+1}, \psi_n, \ldots, \psi_1 \rangle$ *with* $\psi_{n+1} = (l_n + 1, \varnothing, \varnothing, \varnothing)$, $vars(\Gamma_1 \to \Delta_1) \cap vars(\Gamma_2 \to \Delta_2) = \varnothing$, *and* $\Delta_1 \neq \varnothing$ *and* $\Delta_2 \neq \varnothing$ *is called* splitting right.

Definition 42 (Limit of a Derivation)**.** *Given an infinite derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots,$$

let $i \geq 0$ *be a backtracking step, let* l *be the split level of the toplevel split of* Ψ_i*, and let* $k < i$ *be maximal such that step* k *of* $\mathcal{D}$ *is a splitting step producing a split of level* $l - 1$*. We define the monotonic function* $f_{\mathcal{D}} : \mathbb{N} \to \mathbb{N}$ *as follows:*

$$\begin{aligned} f_{\mathcal{D}}(0) &:= 0 \\ f_{\mathcal{D}}(i+1) &:= \min\{j > f_{\mathcal{D}}(i) \mid \textit{step } j \textit{ of } \mathcal{D} \textit{ is persistent}\}. \end{aligned}$$

The limit *of* $\mathcal{D}$ *is defined as*

$$\lim(\mathcal{D}) := \Psi'_0 : N'_0 \rhd \Psi'_1 : N'_1 \rhd \Psi'_2 : N'_2 \rhd \ldots$$

where $\Psi'_0 : N'_0 = \Psi_0 : N_0$ *and* $\Psi'_{i+1} : N'_{i+1}$ *is obtained from* $\Psi'_i : N'_i$ *as follows:*

1. *if step* $f_{\mathcal{D}}(i+1)$ *of* $\mathcal{D}$ *is a backtracking step: let* $(l, \varnothing, \varnothing, M)$ *be the toplevel split of* Ψ_{i+1}*, and consider the associated splitting step* $k = \mathrm{spt}_{\mathcal{D}}(i+1)$

$$\mathcal{I}\frac{\Psi_{k-1} : N_{k-1} \quad L\colon \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Psi_k : N_k \qquad L'\colon \Gamma_1 \to \Delta_1}$$

   Then step $i+1$ *of* $\lim(\mathcal{D})$ *is the splitting right step*

$$\mathcal{I}\frac{\Psi'_i : N'_i \quad L\colon \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Psi'_{i+1} : N'_{i+1} \qquad M\colon \Gamma_2 \to \Delta_2}$$

2. *otherwise:* $\Psi'_{i+1} : N'_{i+1}$ *is obtained by applying the same rule as in step* $f_{\mathcal{D}}(i+1)$ *of* $\mathcal{D}$ *to* $\Psi'_i : N'_i$*.*

Lemma 43. *For any infinite derivation* $\mathcal{D}$*,* $\lim(\mathcal{D})$ *is well-defined.*

Proof. We show by induction on $i \geq 1$ that every premise $L : C$ of step i of $\lim(\mathcal{D}) = \Psi'_0 : N'_0 \rhd \Psi'_1 : N'_1 \rhd \ldots$ is contained in N'_{i-1}. By definition of $\lim(\mathcal{D})$, step $f_{\mathcal{D}}(i)$ of $\mathcal{D}$ is persistent.

1. If step $f_{\mathcal{D}}(i)$ of $\mathcal{D}$ is an inference or reduction step, then by Definition 37, $L\colon C$ is weakly persistent in $\mathrm{suf}_{\mathcal{D}}(f_{\mathcal{D}}(i) - 1)$.

2. If step $f_{\mathcal{D}}(i)$ of $\mathcal{D}$ is a splitting or backtracking step, then by Lemma 39, $L\colon C$ is weakly persistent in $\mathrm{suf}_{\mathcal{D}}(f_{\mathcal{D}}(i) - 1)$

Assume $L : C \notin N_0$. Lemma 36 tells us that $L : C$ is weakly persistent in $\mathrm{suf}_{\mathcal{D}}(k)$, where $k < f_{\mathcal{D}}(i)$ is maximal with $L : C$ the conclusion of step k of $\mathcal{D}$. Hence by Lemma 40, all premises of step k of $\mathcal{D}$ are weakly persistent, therefore step k is a persistent step. For the base case $i = 1$, this is a contradiction, since $f_{\mathcal{D}}(i)$ is the first persistent step of $\mathcal{D}$, and we can thus conclude that $L : C \in N_0 = N'_0$. For $i > 1$, there is $j < i$ such that $f_{\mathcal{D}}(j) = k$ and $L : C$ is the conclusion of step j of $\lim(\mathcal{D})$. Since $L : C \in N_{f_{\mathcal{D}}(j)-1}$, we can conclude that $L : C$ has not been persistently subsumed and hence $L : C \in N'_{i-1}$. □

Note that in general, neither $N'_i \subseteq N_{f_{\mathcal{D}}(i)}$ nor $N'_i \supseteq N_{f_{\mathcal{D}}(i)}$ hold for arbitrary i, because $N_{f_{\mathcal{D}}(i)}$ may contain clauses that are not (weakly) persistent, and persistent clauses may have been subsumed by non-persistent ones. Therefore we can not simply take the limit to be a subsequence of the initial derivation.

Lemma 44. *For every infinite derivation $\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$, and every $i \geq 1$, if step i of $\lim(\mathcal{D})$ is an inference step, then its conclusion is contained in $N_{f_{\mathcal{D}}(i)}$.*

Proof. Follows directly from Definition 42. □

For $\lim(\mathcal{D}) = \Psi'_0 : N'_0 \rhd \Psi'_1 : N'_1 \rhd \ldots$ we define $N_\infty^{\lim(\mathcal{D})} := \bigcup_i \bigcap_{j \geq i} N'_j$.

Definition 45 (Fairness)**.** *A derivation*

$$\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$$

is called fair *if*

1. *either $\mathcal{D}$ is finite and in the final clause set N_k all resolution and factoring inferences are redundant in N_k, or $\mathcal{D}$ is infinite and every resolution or factoring inference from $N_\infty^{\lim(\mathcal{D})}$ is redundant in some $N'_i \in \lim(\mathcal{D})$; and*

2. *for every $i \geq 0$ with L: $\square \in N_i$ with $L \neq \varnothing$, there exists $j > i$ such that step j of $\mathcal{D}$ is a backtracking step with premise L: $\square$.*

Lemma 46. *For any fair infinite derivation $\mathcal{D}$ and any $i \geq 0$, no L: $\square$ with $L \neq \varnothing$ is weakly persistent in $\mathrm{suf}_{\mathcal{D}}(i)$.*

Proof. Assume $L \neq \varnothing$ and consider the backtracking step with $L : \square$ as premise (which must exist by condition 2 of Definition 45). It follows from the definitions of the stack reduction rules that the final rule applied is Enter-right, which deletes the split with the greatest split level in L. □

Our completeness proof will closely follow the one given in [2] (Theorem 4.9), with the exception that we limit ourselves to showing that unsatisfiability of N_0 implies $\varnothing\colon \Box \in N_\infty^{\lim(\mathcal{D})}$. Note that by Lemma 44, this also implies $\varnothing\colon \Box \in \bigcup_j N_j$. We use the *standard redundancy criterion* that is based on a clause ordering $\succ$ [2] and we write $\mathcal{R}_\mathcal{F}^\succ(N)$ for the set of redundant clauses with respect to N and $\mathcal{R}_\mathcal{I}^\succ(N)$ for the set of redundant inferences with respect to N. The standard redundancy criterion satisfies the following four conditions:

(R1) if $N \subseteq N'$ then $\mathcal{R}_\mathcal{F}^\succ(N) \subseteq \mathcal{R}_\mathcal{F}^\succ(N')$ and $\mathcal{R}_\mathcal{I}^\succ(N) \subseteq \mathcal{R}_\mathcal{I}^\succ(N')$;

(R2) if $N' \subseteq \mathcal{R}_\mathcal{F}^\succ(N)$ then $\mathcal{R}_\mathcal{F}^\succ(N) \subseteq \mathcal{R}_\mathcal{F}^\succ(N \setminus N')$ and $\mathcal{R}_\mathcal{I}^\succ(N) \subseteq \mathcal{R}_\mathcal{I}^\succ(N \setminus N')$;

(R3) if N is unsatisfiable, then $N \setminus \mathcal{R}_\mathcal{F}^\succ(N)$ is also unsatisfiable; and

(R4) a resolution or factoring inference γ is in $\mathcal{R}_\mathcal{I}^\succ(N)$ whenever its conclusion is in $N \cup \mathcal{R}_\mathcal{F}^\succ(N)$.

Lemma 47. *Let* $\lim(\mathcal{D}) = \Psi'_0 : N'_0 \rhd \Psi'_1 : N'_1 \rhd \ldots$ *be the limit of a derivation* $\mathcal{D}$. *Then* $\mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j) \subseteq \mathcal{R}_\mathcal{F}^\succ(N_\infty^{\lim(\mathcal{D})})$ *and* $\mathcal{R}_\mathcal{I}^\succ(\bigcup_j N'_j) \subseteq \mathcal{R}_\mathcal{I}^\succ(N_\infty^{\lim(\mathcal{D})})$. *Moreover, the set* $N_\infty^{\lim(\mathcal{D})}$ *is unsatisfiable if* N'_0 *is unsatisfiable.*

Proof. Observe that for any $i \geq 0$, one of the following holds:

(i) $N'_{i+1} = N'_i \cup \{C\}$, where C is the conclusion of a resolution or factoring step from clauses in N'_i; or

(ii) $N'_{i+1} = N'_i \cup \{C\}$, where C is a non-trivial subclause of a clause in N'_i; or

(iii) $N'_{i+1} = N'_i \setminus \{D\}$, where $D \in \mathcal{R}_\mathcal{F}^\succ(N'_i)$.

Hence by definition of $N_\infty^{\lim(\mathcal{D})}$, any clause in $(\bigcup_j N'_j) \setminus N_\infty^{\lim(\mathcal{D})}$ is in some $\mathcal{R}_\mathcal{F}^\succ(N'_i)$. Therefore $(\bigcup_j N'_j) \setminus N_\infty^{\lim(\mathcal{D})} \subseteq \bigcup_j \mathcal{R}_\mathcal{F}^\succ(N'_j)$. Moreover, by condition (R1), $\bigcup_j \mathcal{R}_\mathcal{F}^\succ(N'_j) \subseteq \mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j)$. As a consequence, we have $(\bigcup_j N'_j) \setminus \mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j) \subseteq N_\infty^{\lim(\mathcal{D})}$. Applying condition (R1) again, we may infer that $\mathcal{R}((\bigcup_j N'_j) \setminus \mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j)) \subseteq \mathcal{R}(N_\infty^{\lim(\mathcal{D})})$ (where $\mathcal{R}$ may be either $\mathcal{R}_\mathcal{F}^\succ$ or $\mathcal{R}_\mathcal{I}^\succ$). Using condition (R2), we obtain $\mathcal{R}(\bigcup_j N'_j) \subseteq \mathcal{R}(N_\infty^{\lim(\mathcal{D})})$.

For the second part, assume N'_0 is unsatisfiable. Hence also $\bigcup_j N'_j$ is unsatisfiable, and by condition (R3), $(\bigcup_j N'_j) \setminus \mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j)$ is unsatisfiable. Since $(\bigcup_j N'_j) \setminus \mathcal{R}_\mathcal{F}^\succ(\bigcup_j N'_j) \subseteq N_\infty^{\lim(\mathcal{D})}$, we can infer that $N_\infty^{\lim(\mathcal{D})}$ is unsatisfiable. $\Box$

We say that a set of labelled clauses N is *saturated up to redundancy*, if every resolution or factoring inference γ from $N \setminus \mathcal{R}_{\mathcal{F}}^{\succ}(N)$ is contained in $\mathcal{R}_{\mathcal{I}}^{\succ}(N)$.

Theorem 48 (Completeness)**.** *Let $\mathcal{D} = \Psi_0 : N_0 \rhd \Psi_1 : N_1 \rhd \ldots$ be a fair derivation. If $\varnothing\colon \square \notin \bigcup_j N_j$, then N_0 is satisfiable.*

Proof. By Lemma 44, $\varnothing\colon \square \notin \bigcup_j N_j$ implies $\varnothing\colon \square \notin N_\infty^{\lim(\mathcal{D})}$. Since $\mathcal{D}$ is fair, every inference from $N_\infty^{\lim(\mathcal{D})}$ is redundant in some N_i' and therefore, by Lemma 47, also redundant in $N_\infty^{\lim(\mathcal{D})}$. Hence $N_\infty^{\lim(\mathcal{D})}$ is saturated up to redundancy. It follows ([2], Theorem 4.9) that $N_\infty^{\lim(\mathcal{D})}$ is satisfiable if and only if it does not contain a contradiction. By Lemma 46, the only possible contradiction in $N_\infty^{\lim(\mathcal{D})}$ is $\varnothing\colon \square$. Thus by Lemma 47, if $\varnothing\colon \square \notin N_\infty^{\lim(\mathcal{D})}$, then $N_0 = N_0'$ is satisfiable. $\square$

3 Experiments and Related Work

3.1 Experiments

Our enhanced backtracking process has been integrated into the SPASS [22] theorem prover. The basis for the implementation was SPASS version 3.0. As we mentioned before, the calculus presented here represents a minimal set of inference and reduction rules, whereas the overall splitting calculus is implemented in SPASS [19], and our extension covers that entire calculus. The data structures used to represent the split stack were modified to allow storage of the dependencies of closed left branches. Minor modifications to the implementation of reduction rules were made to ensure that reduced clauses are always recorded for later reinsertion. These modifications were necessary since the original branch condensation in SPASS is performed only up to the last backtracking level, hence a redundant clause is recorded only if it has been subsumed by a clause with greater split level. Finally, a new backtracking procedure was written, implementing the stack reduction rules.

The implementation was tested on the TPTP problem library, version 3.2.0 [18], which consists of 8984 first-order problems. On 2513 of those problems, SPASS (in automatic mode) uses the splitting rule during proof search. For the experiments, an Opteron Linux cluster was used, and SPASS was given a time limit of 5 minutes per problem.

Overall, the number of splits performed per problem decreased by about 10% on average when using improved backtracking. In addition, SPASS with improved backtracking terminated with a solution for 24 problems (22 proofs, 2 completions) that could not be solved by the version without improved backtracking within the given time limit. The new version looses 4 problems because of the potentially different search space exploration caused by the new backtracking rule. These problems are recovered by an increased time

limit.

3.2 Related Work

The paper is an extension of the abstract DPLL calculus of Nieuwenhuis et al. [15] for first-order logic. Due to the need to create new clauses via inferences and reductions, the calculus is more involved.

An alternative approach to case analysis in saturation-based theorem proving, relying on the introduction of special propositional symbols instead of a split stack, was presented in [17]. The main difference to our framework is that when simulating splitting with new propositional symbols, the generated clauses can not be directly used for reductions: for example, splitting the clause $P(x) \vee Q(y)$ in this way produces the clauses $P(x) \vee p$ and $p \rightarrow Q(x)$, where p is a new propositional symbol. Therefore, this type of splitting does not provide the necessary support for the superposition based decidability results on non Horn first-order fragments. On the other hand, in particular if applied to unsatisfiable problems, this alternative approach to splitting can be very useful.

For the general methodology of labelled clauses there is a huge literature, see [4] for an overview. In particular, the use of clause labels to model explicit case analysis was first suggested in [13], which provided a starting point for the work presented here. We refined the abstract labelled splitting rule presented there with explicit backtracking and redundancy handling.

The DPLL procedure [6], which lies at the core of most of today's state-of-the-art boolean satisfiability solvers, has received a great deal of attention in the past years. In particular, a lot of research has gone into improving the backtracking process. Thus it is natural to ask how, in the propositional domain, our backtracking scheme compares to that of modern SAT solvers. Let us first clarify some key differences between our framework and DPLL.

In basic DPLL, when exhaustive unit propagation has neither led to a conflict nor yielded a complete truth assignment, a choice is made by assuming some yet undefined literal to be true (typically a literal occurring in some clause). If this choice leads to a conflict, the truth value of that literal is flipped, hence the two branches of the case analysis are A and $\neg A$ for some propositional variable A. This can be simulated by the refined version of the splitting rule, discussed after Definition 1. The reason why this refinement is restricted to ground split parts is that the negation of universally quantified clauses leads to the introduction of new Skolem constants, and in practice this tends to extend the search space for the second branch. This could in principle be avoided by remembering all ground terms that a variable has

been instantiated with. For example, if the left split part was $P(x)$ and a refutation of the branch involved substituting x by both a and $f(b)$ in different subtrees, then $\neg P(a) \vee \neg P(f(b))$ could be used instead of $\neg P(c)$ for some new constant c. Although this approach avoids the introduction of new symbols, tracking all instantiations adds overhead and the fact that the resulting lemmata are again in general non-units diminishes their usefulness for reductions. When dealing with purely propositional problems however, this lemma-generation can be used to simulate DPLL-style case analysis: for any clause $\Gamma \rightarrow \Delta, A$ chose A as the first split part – the lemma $\neg A$ is then available in the second branch.

Backtracking in modern SAT solvers is based on a conflict analysis during which a conflict clause is learned, i.e., added to the clause set. We compared SPASS implementing our new backtracking rule (Definition 10) without learning with MiniSat v1.14 [8] on SAT problems, by manipulating both systems such that they essentially use the same propositional variable order for branching. Out of the current SATLIB (`http://www.satlib.org`) library we selected about 200 problems that SPASS could solve in a 5 min time limit. On more than 95% of all problems Minisat needs less splits than SPASS. Out of these problems, SPASS performs three times more splits than MiniSat, on the average. This result suggests that conflict-driven clause learning is in favor of the SPASS split backtracking mechanism. So there is potential for exploring this mechanism also for the first-order case. However, again, this requires at least a detailed analysis of the closed branches as split clauses may have been used in several instantiations.

Although our backtracking rule does not include learning of conflict clauses, there are cases where it is superior to the conflict driven clause learning of modern SAT solvers, i.e., a proof requires less splits. This is documented by the 5% of examples where SPASS needed less splits than MiniSat and such examples can also be explicitly constructed.

The following example illustrates a clause set, where the labelled splitting style of backtracking is in favor of the DPLL style of backtracking.

$$\begin{array}{lll} \{ \quad \bar{A}_1 \vee \bar{A}_2 \vee \bar{A}_3 \vee A_5, & \bar{A}_1 \vee \bar{A}_2 \vee \bar{A}_3 \vee \bar{A}_5, & \bar{A}_1 \vee \bar{A}_4 \vee A_6, \\ \quad \bar{A}_1 \vee \bar{A}_4 \vee \bar{A}_6, & \bar{A}_1 \vee \bar{A}_2 \vee A_4 \vee A_7, & \bar{A}_1 \vee \bar{A}_2 \vee A_4 \vee \bar{A}_7, \\ \quad A_1 \vee C_1, & A_2 \vee C_2, & A_3 \vee C_3, \\ \quad A_4 \vee C_4, & \ldots\} & \end{array}$$

Negation is denoted by overlining and the C_i stand for further disjuncts containing positive literals. In favor of a small example, it contains potential for simplification, e.g., the first two clauses. Marking decision literals with a d, the partial truth assignment $A_1^d\, A_2^d\, A_3^d$ (corresponding to splittings of the finally given clauses) is extended to $A_1^d\, A_2^d\, A_3^d\, A_5$ by unit propagation, leading

to a conflict. In terms of the labelled resolution calculus, this corresponds to an empty clause with label $\{1, 3, 5\}$ where the first split clause is $A_1 \vee C_1$, the second $A_2 \vee C_2$, and the third $A_3 \vee C_3$. Both DPLL and labelled splitting then enter the right branch of the last split. In DPLL terms, we get the assignment $A_1^d\, A_2^d\, \bar{A}_3$, which we extend to $A_1^d\, A_2^d\, \bar{A}_3\, A_4^d$ via a further decision. Clause $\bar{A}_1 \vee \bar{A}_4 \vee A_6$ propagates A_6, after which clause $\bar{A}_1 \vee \bar{A}_4 \vee \bar{A}_6$ becomes false. Resolving these two yields the conflict clause $\bar{A}_1 \vee \bar{A}_4$, and DPLL backjumps to decision level 1, i.e. to assignment $A_1^d\, \bar{A}_4$. If this branch cannot be closed without repeating the decision on A_2 that was just undone, this overall search space needs to be explored again. On the other hand, in the labelled calculus, the conflict clause corresponds to an empty clause with label $\{1, 7\}$, where we have split $A_4 \vee C_4$. So Enter-right is the only applicable stack reduction rule. The right branch of the corresponding split can be closed by the clauses $\bar{A}_1 \vee \bar{A}_2 \vee A_4 \vee A_7$ and $\bar{A}_1 \vee \bar{A}_2 \vee A_4 \vee \bar{A}_7$ without further splitting, since the split of clause $A_2 \vee C_2$ is still available.

4 Conclusion

We have extended the abstract DPLL calculus by Nieuwenhuis et al. [15] to the full first-order case, called labelled splitting. The calculus is sound and complete. For the completeness result we introduced a new notion of an infinite path to establish fairness.

The calculus is implemented for the full superposition calculus [19] in SPASS. It shows a 10% average gain in the number of splits on all TPTP problems where splitting is involved and SPASS could decide 24 more problems compared to the previous version.

The fairness notion of Definition 45 does not directly provide an effective strategy for a fair inference selection. In SPASS we mainly use two different strategies. For the propositional case we employ exhaustive splitting, because splitting combined with the reduction matching replacement resolution [19] constitutes already a complete calculus. For the first-order case, clauses are selected for inferences with respect to their "weight" composed out of the number of contained symbols and their depth in the derivation. If such a clause can be split and the first split part has a "sufficient" reduction potential for other clauses, splitting is preferred over other inferences.

A comparison of the labelled splitting backtracking mechanism with DPLL style backtracking based on conflict-driven clause learning reveals room for further improvement. However, this is not a straightforward effort, because the negation of a first-order clause is an existentially quantified conjunction of literals that via Skolemization introduces new constants to the proof search. It is well known that the addition of new constants causes an increased complexity of the unsatisfiability problem and if potentially done infinitely often, can even cause completeness issues. So it seems to us that in the case of a conflict, an analysis of the proof and the used first-order terms in the proof is the most promising approach to enhance the presented labelled splitting backtracking mechanism with conflict-driven clause learning. This will be subject of future research.

Bibliography

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] L. Bachmair and H. Ganzinger. Resolution theorem proving. In *Handbook of Automated Reasoning*, pages 19–99. 2001.

[3] L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational Logic and Proof Theory, Third Kurt Gödel Colloquium*, volume 713 of *LNCS*, pages 83–96. Springer, August 1993.

[4] D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, L. Vigan, and o eds. Labelled deduction, 2000.

[5] P. Beame, H. Kautz, and A. Sabharwal. Understanding the power of clause learning, 2003.

[6] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[7] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[8] N. Eén and N. Sörensson. An extensible SAT-solver. *Theory and Applications of Satisfiability Testing*, pages 502–518, 2004.

[9] E. Goldberg and Y. Novikov. Berkmin: A fast and robust SAT-solver. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, page 142, Washington, DC, USA, 2002. IEEE Computer Society.

[10] R. Hähnle. Tableaux and related methods. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 3, pages 100–178. Elsevier, 2001.

[11] T. Hillenbrand and C. Weidenbach. Superposition for finite domains. Research Report MPI-I-2007-RG1-002, Max-Planck Institute for Informatics, Saarbrücken, Germany, April 2007.

[12] R. Letz, K. Mayr, and C. Goller. Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning,*, 13(3):297–338, 1994.

[13] T. Lev-Ami, C. Weidenbach, T. Reps, and M. Sagiv. Labelled clauses. In *21st International Conference on Automated Deduction (CADE-21)*, volume 4603 of *Lecture Notes in Computer Science*, pages 311–327, Bremen, Germany, 2007. Springer.

[14] J. P. Marques-Silva and K. A. Sakallah. GRASP - A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.

[15] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[16] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier, 2001.

[17] A. Riazanov and A. Voronkov. Splitting without backtracking. In *IJCAI*, pages 611–617, 2001.

[18] G. Sutcliffe and C. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[19] C. Weidenbach. Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 27, pages 1965–2012. Elsevier, 2001.

[20] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER, version 0.42. In M. McRobbie and J. Slaney, editors, *13th International Conference on Automated Deduction, CADE-13*, volume 1104 of *LNAI*, pages 141–145. Springer, 1996.

[21] C. Weidenbach, R. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. *The SPASS Handbook*. Included in the SPASS 3.0 distribution.

[22] C. Weidenbach, R. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. System description: SPASS version 3.0. In F. Pfenning, editor, *CADE-21 : 21st International Conference on Automated Deduction*, volume 4603 of *LNAI*, pages 514–520. Springer, 2007.

[23] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in boolean satisfiability solver. In *ICCAD*, pages 279–285, 2001.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available via WWW using the URL `http://www.mpi-inf.mpg.de`. If you have any questions concerning WWW access, please contact `reports@mpi-inf.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Campus E2 3
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-inf.mpg.de`

MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	F.M. Suchanek, G. de Melo, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	F. Suchanek, G. Kasneci, M. Ramanath, M. Sozio, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	W. Saleem, D. Wang, A. Belyaev, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation

MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - a core of semantic knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A neighborhood-based approach for clustering of linked document collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining linguistic and statistical analysis to extract relations from web documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment texture editing in monocular video sequences based on color-coded printing patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: index-access optimized top-k query processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-aware global df estimation in distributed information retrieval systems
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean value coordinates for arbitrary spherical polygons and polyhedra in $\mathbb{R}^3$
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and annealing particle filters: mathematics and a recipe for applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture modeling and animation by imitation
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving non-local denoising of static and time-varying range data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced dynamic reflectometry for relightable free-viewpoint video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven laplacian mesh deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On fast construction of spatial hierarchies for ray tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A framework for natural animation of digitized models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU point list generation through histogram pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and evaluation of backward compatible high dynamic range video compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On verifying complex properties using symbolic shape analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-sensitive autocompletion search
MPI-I-2006-1-006	M. Kerber	Division-free computation of subresultants using bezout matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap rounding of Bézier curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power assignment problems in wireless communication
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated retraining methods for document classification and their parameter tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An emperical model for heterogeneous translucent objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric calibration of high dynamic range cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel	Joint motion and reflectance capture for creating relightable 3D videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and design of discrete normals and curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse meshing of uncertain and noisy surface scattered data