# Analysis of distributed-search contraction-based strategies

**Maria Paola Bonacina** *

Dept. of Computer Science, University of Iowa, Iowa City, IA 52242-1419, USA
bonacina@cs.uiowa.edu

**Abstract.** We present a model of parallel search in theorem proving for forward-reasoning strategies, with contraction and distributed search. We extend to parallel search the bounded-search-spaces approach to the measurement of infinite search spaces, capturing both the advantages of parallelization, e.g., the subdivision of work, and its disadvantages, e.g., the cost of communication, in terms of search space. These tools are applied to compare the search space of a distributed-search contraction-based strategy with that of the corresponding sequential strategy.

## 1 Introduction

The difficulty of fully-automated theorem proving has led to investigate ways of enhancing theorem-proving strategies with parallelism. We distinguish among *parallelism at the term level* (i.e., parallelizing the inner algorithms of the strategy), *parallelism at the clause level* (i.e., parallel inferences within a single search) and *parallelism at the search level* or parallel search [8]. This paper considers *parallel search*: deductive processes search in parallel the space of the problem, and the parallel search succeeds as soon as one of the processes succeeds. A parallel-search strategy may subdivide the search space among the processes (*distributed search*), or assign to the processes different search plans (*multi-search*), or combine both principles. The processes *communicate* to merge their results, and preserve completeness (e.g., if the search space is subdivided or unfair search plans are used). This paper concentrates on distributed search.

Parallel search applies to theorem-proving strategies in general. This paper studies *forward-reasoning* and in particular *contraction-based* strategies (e.g., [13, 19, 9, 4]). There has been much interest in parallelizing these strategies (e.g., [12, 11, 5, 15, 6] and [8, 21] for earlier references), because they behave well sequentially (e.g., Otter [17], RRL [14], Reveal [2], EQP [18] are based on these strategies, and thanks to them succeeded in solving challenge problems, e.g., [2, 1, 18]). However, the parallelization of contraction-based strategies is difficult, primarily because of *backward contraction*: the clauses used as premises are subject to being deleted, the database is highly dynamic, and eager backward contraction (inter-reduce before expanding) diminishes the degree of concurrency of the inferences. The qualitative analysis in [8] showed how these factors affect adversely approaches based on parallelism at the term or clause level.

* Supported in part by NSF grants CCR-94-08667 and CCR-97-01508.

This paper begins a quantitative analysis of *distributed-search contraction-based strategies*, which poses several problems. We need to represent *subdivision* of the search space and *communication*, and measure their advantage and cost, respectively. Since the subdivision may not be effective, the search processes *overlap*: one needs to capture also the disadvantage of duplicated search. These problems are made worse by a fundamental difficulty: the search spaces in theorem proving are *infinite*. Therefore, we cannot analyze subdivision and overlap in terms of total size of the search space. Neither can we rely on the classical complexity measure of time to capture the cost of communication, because theorem-proving strategies are semidecision procedures that may not halt, so that "time" is not defined. Finally, the problems are not well-defined. Since parallel theorem proving is a young field, and an analysis of this kind was not attempted before, there are no standard formal definitions for many of the concepts involved. In recent work [10], we proposed an approach to the analysis of strategies, comprising *a model for the representation of search*, a notion of *complexity of search* in infinite spaces, and measures of this complexity, termed *bounded search spaces*. In this paper we build on this previous work to address the problems listed above.

The first part of the paper (Sections 2 and 3) develops **a framework of definitions for parallel theorem proving** by forward-reasoning distributed-search strategies. Three important properties of parallel search plans (*monotonicity of the subdivision*, *fairness* and *eager contraction*) are identified, and sufficient conditions to satisfy them are given. We point out that it is not obvious that a parallelization of a contraction-based strategy is contraction-based. On the contrary, this issue is critical in the parallelization of forward reasoning. **A model of parallel search** is presented in Section 4. A strategy with contraction not only visits, but also *modifies* the search space [10]. In distributed search also *subdivision* and *communication* modify the search space, and many processes are active in parallel. Our solution is based on distinguishing the *search space* and the *search process*, and yet representing them together in a *parallel marked search graph*. The structure of the graph represents the search space of all the possible inferences, while the *marking* represents the search process, including contraction, subdivision and communication.

Once we have a model of the search, we turn to **measuring benefits and costs of parallelization in terms of search complexity** (Section 5). The methodology of [10] is based on the observation that for infinite search spaces it is not sufficient to measure the generated search space. It is necessary to measure also the effects of the actions of the strategy on the infinite space that lies ahead. An exemplary case is that of contraction, where the deletion of an existing clause may prevent the generation of others. Our approach is to enrich the search space with a notion of *distance*, and consider the *bounded search space* made of the clauses whose distance from the input is within a given bound. The infinite search space is reduced to an infinite succession of *finite* bounded search spaces. Since the bounded search spaces are finite, they can be compared (in a multiset ordering), eliminating the obstacle of the impossibility of comparing

infinite spaces. The second fundamental property of the bounded search spaces is that they change with the steps performed by the strategy. In the sequential case, the only factor which modifies the bounded search spaces is contraction, which makes clauses unreachable (infinite distance). In the parallel case, there are also subdivision and communication: subdivision makes the bounded search spaces for the parallel processes *smaller* (advantage of parallelism), while comunication *undoes* in part the effect of the subdivision (disadvantage of parallelism). The *parallel bounded search spaces* for the parallel derivation as a whole measure also the cost of duplicated search due to overlapping processes.

Section 6 applies these tools to **the analysis of distributed-search contraction-based strategies**. In distributed search, eager contraction depends on communication (e.g., to bring to a process a needed simplifier). We discover two related patterns of behaviour, called *late contraction* and *contraction undone*, where eager contraction fails. It follows that search paths that eager contraction would prune are not pruned. While in a sequential derivation the bounded search spaces decrease monotonically due to contraction, in a parallel derivation they may oscillate *non-monotonically*, because they reflect the conflict of subdivision and communication, and the conflict of contraction and communication. However, the incidence of late contraction and contraction undone decreases as the speed of communication increases, and at the limit, if communication takes no time, they disappear. For the overlap, we give sufficient conditions to minimize it relying on local eager contraction, independent of communication.

The last task is to compare a sequential contraction-based strategy $\mathcal{C}$ with its parallelization $\mathcal{C}'$. We prove that if $\mathcal{C}'$ has instantaneous communication and minimizes the overlap, its parallel bounded search spaces are smaller than those of $\mathcal{C}$. On one hand, this result represents a limit that concrete strategies may approximate. For instance, this theorem justifies formally the intuition about improving performance by devising subdivision criteria that reduce the overlap (e.g., [6]). On the other hand, since the hypothesis of instantaneous communication is needed, it represents a negative result on the parallelizability of contraction-based strategies, which contributes to explain the difficulty with obtaining generalized performance improvements by parallel theorem proving.

This kind of analysis is largely new, especially for parallel strategies. Most studies of complexity in deduction analyze the length of propositional proofs as part of the $NP \neq co\text{–}NP$ quest (e.g., [22]), or work with Herbrand complexity and proof length to obtain lower bounds for sets of clauses (e.g., [16]). The study in [20] analyzes measures of duplication in the search spaces of theorem-proving strategies. The full version of this paper, with the proofs and more references, can be found in [7].

## 2 Parallel theorem-proving strategies

In the *inference system* of a *theorem-proving strategy*, *expansion rules* (e.g., resolution) generate clauses, while *contraction rules* (e.g., subsumption and simplification) delete or reduce clauses according to a well-founded ordering $\succ$ (e.g., the

multiset extension of a complete simplification ordering on atoms). An inference rule can be seen as a function which takes a tuple of premises and returns a set of clauses to be added and a set of clauses to be deleted:

**Definition 21** *Let $\Theta$ be a signature, $\mathcal{L}_\Theta$ the language of clauses on $\Theta$, and $\mathcal{P}(\mathcal{L}_\Theta)$ its powerset. An inference rule $f^n$ of arity $n$ is a function $f^n\colon \mathcal{L}_\Theta^n \to \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$. (If $f^n$ does not apply to $\bar{x}$, $f^n(\bar{x}) = (\emptyset, \emptyset)$.)*

Given $\bar{x} = (\varphi_1 \dots \varphi_n)$, let $X$ be the multiset $\{\varphi_1 \dots \varphi_n\}$, and $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$ the projection functions:

**Definition 22** *Given a well-founded ordering $(\mathcal{L}_\Theta, \succ)$, $f^n$ is an* expansion rule *if $\forall \bar{x} \in \mathcal{L}_\Theta^n$, $\pi_2(f^n(\bar{x})) = \emptyset$. It is a* contraction rule *w. r. t. $\succ$ if either $\pi_1(f^n(\bar{x})) = \pi_2(f^n(\bar{x})) = \emptyset$, or $\pi_2(f^n(\bar{x})) \neq \emptyset$ and $X - \pi_2(f^n(\bar{x})) \cup \pi_1(f^n(\bar{x})) \prec_{mul} \pi_2(f^n(\bar{x}))$.*

The *closure* of a set of clauses $S$ with respect to an inference system $I$ is the set $S_I^* = \bigcup_{k \geq 0} I^k(S)$, where $I^0(S) = S$, $I^k(S) = I(I^{k-1}(S))$ for $k \geq 1$ and $I(S) = S \cup \{\overline{\varphi} |\ \varphi \in \pi_1(f(\varphi_1 \dots \varphi_n)),\ f \in I,\ \varphi_1 \dots \varphi_n \in S\}$.

Clauses deleted by contraction are *redundant*, and inferences that use redundant clauses (without deleting them) are also redundant (e.g., [9, 4]). Using the notion of *redundancy criterion* [3], $R(S)$ denotes the set of clauses redundant with respect to $S$ according to $R$. A redundancy criterion $R$ and a set of contraction rules $I_R$ *correspond* if whatever is deleted by $I_R$ is redundant according to $R$ ($\pi_2(f^n(\bar{x})) \subseteq R(X - \pi_2(f^n(\bar{x})) \cup \pi_1(f^n(\bar{x})))$), and if $\varphi \in R(S) \cap S$, $I_R$ can delete $\varphi$ without adding other clauses to make it redundant ($\pi_1(f^n(\bar{x})) = \emptyset$ and $\pi_2(f^n(\bar{x})) = \{\varphi\}$). $I_R$ and $R$ are based on the same ordering. Let $I = I_E \cup I_R$, distinguishing expansion and contraction rules in $I$.

Next, a parallel strategy has a system $M$ of *communication operators*, such as $receive\colon \mathcal{L}_\Theta^* \to \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$, and $send\colon \mathcal{L}_\Theta^* \to \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$, where $receive(\bar{x}) = (\bar{x}, \emptyset)$ (adds received clauses to the database of the receiver), and $send(\bar{x}) = (\emptyset, \emptyset)$ (sending something does not modify the database of the sender).

The other component of a strategy is the *search plan*, which chooses inference rule and premises at each stage of a derivation $S_0 \vdash \dots S_i \vdash \dots$, where $S_i$ is the state of the derivation after $i$ steps, usually the multiset of existing clauses. We use $States$ for the set of states and $States^*$ for sequences of states. In distributed search, the search plan also controls *communication* and *subdivision*. Since $S_I^*$ is infinite and unknown, the subdivision is built *dynamically*: at stage $i$ the search plan subdivides the inferences that can be done in $S_i$. For each process $p_k$, an inference is either *allowed* (assigned to $p_k$), or *forbidden* (assigned to others):

**Definition 23** *A* parallel search plan *is a 4-tuple $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$:*

1. *The* rule-selecting function $\zeta\colon States^* \times \mathbb{N} \times \mathbb{N} \to I \cup M$ *takes as arguments the partial history of the derivation, the number of processes and the identifier of the process executing the selection, and returns an inference rule or a communication operator.*
2. *The* premise-selecting function $\xi\colon States^* \times \mathbb{N} \times \mathbb{N} \times (I \cup M) \to \mathcal{L}_\Theta^*$ *also takes in input the selection of $\zeta$, and satisfies $\xi((S_0 \dots S_i), n, k, f^m) \in S_i^m$.*

3. *The* subdivision function $\alpha\colon States^* \times \mathbb{N} \times \mathbb{N} \times (I \cup M) \times \mathcal{L}^*_\Theta \to Bool$ *also takes as argument the selection of* $\xi$*, and returns true (process allowed to perform the step), or* $false$ *(forbidden), or* $\perp$ *(undefined).*
4. *The* termination-detecting function $\omega\colon States \to Bool$ *returns true if and only if the given state contains the empty clause.*

**Definition 24** *Given a theorem-proving problem* $S$*, the* parallel derivation *generated by a strategy* $\mathcal{C} = \langle I, M, \Sigma\rangle$*, with* $\Sigma = \langle \zeta, \xi, \alpha, \omega\rangle$*, for processes* $p_0 \dots p_{n-1}$ *is made of* $n$ *asynchronous* local derivations $S = S_0^k \vdash_\mathcal{C} \dots S_i^k \vdash_\mathcal{C} \dots$*, s. t.* $\forall k$*,* $0 \le k \le n-1$*,* $\forall i \ge 0$*, if* $\omega(S_i^k) = false$*,* $\zeta((S_0^k \dots S_i^k), n, k) = f$*, either* $f = receive$ *and* $\bar{x}$ *is received, or* $f \ne receive$ *and* $\xi((S_0^k \dots S_i^k), n, k, f) = \bar{x}$*, and* $\alpha((S_0^k \dots S_i^k), n, k, f, \bar{x}) = true$*, then* $S_{i+1}^k = S_i^k \cup \pi_1(f(\bar{x})) - \pi_2(f(\bar{x}))$*.*

A sequential search plan $\Sigma = \langle \zeta, \xi, \omega\rangle$ has $\zeta\colon States^* \to I$, $\xi\colon States^* \times I \to \mathcal{L}^*_\Theta$ and $\omega$. A parallel search plan $\Sigma' = \langle \zeta', \xi', \alpha, \omega\rangle$ is a *parallelization by subdivision* of $\Sigma$, if: whenever $\zeta'((S_0 \dots S_i), n, k) \in I$, $\zeta'((S_0 \dots S_i), n, k) = \zeta(S_0 \dots S_i)$; and $\xi'((S_0 \dots S_i), n, k, f) = \xi((S_0 \dots S_i), f)$. $\mathcal{C}' = \langle I, M, \Sigma'\rangle$ is a parallelization of $\mathcal{C} = \langle I, \Sigma\rangle$, if $\Sigma'$ is a parallelization of $\Sigma$.

## 3 Monotonicity, fairness and eager contraction

If $\zeta$ and $\xi$ can select a certain $f$ and $\bar{x}$, $\alpha$ needs to be defined on their selection, and it should not "forget" its decisions when clauses are deleted:

**Definition 31** *A subdivision function* $\alpha$ *is* total on generated clauses *if for all* $S_0 \vdash \dots S_i \vdash \dots$*,* $k$*,* $n$*,* $f^m$ *and* $\bar{x} \in (\bigcup_{j=0}^{i} S_j)^m$*,* $\alpha((S_0 \dots S_i), n, k, f^m, \bar{x}) \ne \perp$*.*

Since it is undesirable that permission changes after it has been decided, $\alpha$ is required to be *monotonic* w. r. t. $\perp < false$ and $\perp < true$:

**Definition 32** *A subdivision function* $\alpha$ *is* monotonic *if for all* $S_0 \vdash \dots S_i \vdash \dots$*,* $n$*,* $k$*,* $f$*,* $\bar{x}$*, and* $i \ge 0$*,* $\alpha((S_0 \dots S_i), n, k, f, \bar{x}) \le \alpha((S_0 \dots S_{i+1}), n, k, f, \bar{x})$*.*

A strategy is *complete* if it succeeds whenever $S_0$ is inconsistent. Completeness is made of *refutational completeness* of the inference system, and *fairness* of the search plan [9]. A sufficient condition for fairness is (e.g., [3]):

**Definition 33** *A derivation* $S_0 \vdash \dots S_i \vdash \dots$ *is* uniformly fair *w. r. t.* $I$ *and* $R$ *if* $I(S_\infty - R(S_\infty)) \subseteq \bigcup_{j \ge 0} S_j$*, where* $S_\infty = \bigcup_{j \ge 0} \bigcap_{i \ge j} S_i$ *(persistent clauses).*

In distributed search a process only needs to be fair with respect to what is allowed. Since $\alpha$ is monotonic, allowed $(\exists j\ \alpha((S_0^k \dots S_j^k), n, k, f, \bar{x}) = true)$ is equivalent to persistently allowed $(\exists i\ \forall j \ge i\ \alpha((S_0^k \dots S_j^k), n, k, f, \bar{x}) = true)$.

**Definition 34** *A local derivation* $S_0^k \vdash \dots S_i^k \vdash \dots$ *is* uniformly fair *w.r.t.* $I$*,* $R$ *and* $\alpha$*, if* $\varphi \in \pi_1(f^m(\bar{x}))$*,* $\bar{x} \in (S_\infty^k - R(S_\infty^k))^m$ *and* $\exists i \ge 0\ \forall j \ge i$ $\alpha((S_0^k \dots S_j^k), n, k, f^m, \bar{x}) = true$ *imply* $\varphi \in \bigcup_{j \ge 0} S_j^k$*.*

The search plan needs to ensure that for every inference from a tuple of persistent (in $S_\infty = \bigcup_{k=0}^{n-1} S_\infty^k$) non-redundant premises, there is a $p_k$ which is allowed (*fairness of subdivision*) and has all the premises (*fairness of communication*):

**Definition 35** *A parallel derivation $S_0^k \vdash_{\mathcal{C}} \ldots S_i^k \vdash_{\mathcal{C}} \ldots$, for $k \in [0, n-1]$, by $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, is* uniformly fair *w. r. t. $I$ and $R$, if:*

1. *$\forall p_k$, $S_0^k \vdash_{\mathcal{C}} \ldots S_i^k \vdash_{\mathcal{C}} \ldots$ is uniformly fair w. r. t. $I$, $R$ and $\alpha$* (local fairness).
2. *$\forall f^m \in I$, $\forall \bar{x} \in (S_\infty - R(S_\infty))^m$, s. t. $\pi_1(f^m(\bar{x})) \neq \emptyset$, $\exists$ $p_k$ s. t. $\bar{x} \in (S_\infty^k - R(S_\infty))^m$ (*fairness of communication*), and $\exists i \geq 0$, s. t. $\forall j \geq i$, $\alpha((S_0^k \ldots S_j^k), n, k, f^m, \bar{x}) = true$* (fairness of subdivision).

If $\Sigma$ is uniformly fair, its parallelization $\Sigma'$ inherits local fairness from $\Sigma$.

**Theorem 31** *If a parallel derivation $S_0^k \vdash_{\mathcal{C}} \ldots \vdash_{\mathcal{C}} S_i^k \vdash_{\mathcal{C}} \ldots$, for $k \in [0, n-1]$, is uniformly fair with respect to $I$ and $R$, then $I(S_\infty - R(S_\infty)) \subseteq \bigcup_{k=0}^{n-1} \bigcup_{j \geq 0} S_j^k$.*

A sequential strategy is *contraction-based* if it features contraction rules and an *eager-contraction* search plan:

**Definition 36** *A derivation $S_0 \vdash_I \ldots S_i \vdash_I \ldots$ has* eager contraction*, if for all $i \geq 0$ and $\varphi \in S_i$, if there are $f^m \in I_R$ and $\bar{x} \in S_i^m$, such that $\pi_2(f^m(\bar{x})) = \{\varphi\}$, then $\exists l \geq i$ such that $S_l \vdash S_{l+1}$ deletes $\varphi$, and $\forall j, i \leq j \leq l$, $S_j \vdash S_{j+1}$ is not an expansion inference, unless the derivation succeeds sooner.*

The parallelization of eager contraction is difficult. First, each local derivation needs to have *local eager contraction*, which is defined as in Def. 36 with expansion replaced by expansion or communication. Second, the strategy should ensure that $p_k$ delete eagerly a clause reducible by a premise generated by $p_h$. This depends on communication:

**Definition 37** *Let $\varphi \in S_\infty - R(S_\infty)$ and $i$ be the first stage s. t. $\varphi \in \bigcup_{h=0}^{n-1} S_i^h$. A parallel derivation has* propagation of clauses up to redundancy *if $\forall p_h$ $\exists j$ $\varphi \in S_j^h$,* instantaneous propagation of clauses up to redundancy *if $\forall p_h$ $\varphi \in S_i^h$.*

**Definition 38** *A parallel derivation has* distributed global contraction*, if for all $p_k$, $i \geq 0$, and $\varphi \in S_i^k$, if there are $f^m \in I_R$ and $\bar{x} \in (\bigcup_{h=0}^{n-1} S_i^h)^m$ such that $\pi_2(f^m(\bar{x})) = \{\varphi\}$, then $\exists l \geq i$ such that $S_l^k \vdash S_{l+1}^k$ deletes $\varphi$, unless $p_k$ halts sooner. It has* global eager contraction *if, in addition, $\forall j, i \leq j \leq l$, $S_j^k \vdash S_{j+1}^k$ is neither an expansion nor a communication step.*

Global eager contraction generalizes eager contraction to parallel derivations, while distributed global contraction is a weaker requirement which guarantees contraction, but not priority over expansion. In the presence of communication delays, $p_k$ may use as expansion premise a clause which is reducible with respect to $\bigcup_{h=0}^{n-1} S^h$ before it becomes reducible with respect to $S^k$. Thus, we have:

**Lemma 31** *Local eager contraction and propagation of clauses up to redundancy (instantaneous propagation of clauses) imply distributed global contraction (global eager contraction, respectively).*

A parallel strategy is *contraction-based* if it has contraction rules and its search plan has local eager contraction and distributed global contraction.

**Theorem 32** *Let $\mathcal{C} = \langle I, \Sigma \rangle$ be a contraction-based strategy and $\mathcal{C}'$ with $\Sigma' = \langle \zeta', \xi', \alpha, \omega' \rangle$ a parallelization by subdivision of $\mathcal{C}$. If $\Sigma'$ propagates clauses up to redundancy, and for all $f \in I_R$, $i$, $n$, $k$ and $\bar{x}$, $\alpha((S_0^k \dots S_i^k), n, k, f, \bar{x}) = true$, $\mathcal{C}'$ is also contraction-based.*

The requirement that all contractions are allowed to all is strong, especially if contraction is not only deletion but also deduction (e.g., simplification). Assume that $\alpha$ forbids contractions, e.g., $f(\bar{x}) = (\{\psi_1, \dots, \psi_m\}, \{\varphi_1, \dots, \varphi_p\})$. Consider a strategy that lets a process delete the $\varphi_j$, but not generate the $\psi_i$, when $\zeta$ selects $f$, $\xi$ selects $\bar{x}$, and $\alpha$ forbids the step. It is sufficient that at least one process generates and propagates the $\psi_i$ to preserve completeness:

**Theorem 33** *Let $\mathcal{C}$ and $\mathcal{C}'$ be as in Theorem 32. If $\Sigma'$ propagates clauses up to redundancy, and whenever $\zeta((S_0^k \dots S_i^k), n, k) = f \in I_R$, $\xi((S_0^k \dots S_i^k), n, k, f) = \bar{x}$, $\alpha((S_0^k \dots S_i^k), n, k, f, \bar{x}) = false$, it is $S_{i+1}^k = S_i^k - \pi_2(f(\bar{x}))$, $\mathcal{C}'$ is also contraction-based.*

## 4 Search graphs for parallel search

Given a theorem-proving problem $S$ and an inference system $I$, the *search space* induced by $S$ and $I$ is represented by the hypergraph $G(S_I^*) = (V, E, l, h)$, where $V$ is the set of vertices, $l$ is a vertex-labelling function $l\colon V \to \mathcal{L}_\Theta/\stackrel{\bullet}{=}$ (from vertices to equivalence classes of variants, so that all variants are associated to a unique vertex), $h$ is an arc-labelling function $h\colon E \to I$, and if $f^m(\varphi_1, \dots, \varphi_m) = (\{\psi_1, \dots, \psi_s\}, \{\gamma_1, \dots, \gamma_p\})$ for $f^m \in I$, $E$ contains a hyperarc $e = (v_1 \dots v_n; w_1 \dots w_p; u_1 \dots u_s)$ where $h(e) = f^m$, $n + p = m$, and

- $\forall j\ 1 \le j \le n\ l(v_j) = \varphi_j$ and $\varphi_j \notin \{\gamma_1, \dots, \gamma_p\}$ (premises not deleted),
- $\forall j\ 1 \le j \le p\ l(w_j) = \gamma_j$ (deleted premises), and $\forall j\ 1 \le j \le s\ l(u_j) = \psi_j$ (generated clauses).

W.l.o.g. we consider hyperarcs in the form $(v_1 \dots v_n; w; u)$. Contraction inferences that purely delete clauses are represented as replacement by *true* (a dummy clause such that $\forall \varphi\ \varphi \succ true$), and a special vertex $\mathsf{T}$ is labelled by *true*.

The search graph $G(S_I^*) = (V, E, l, h)$ represents the static structure of the search space. The dynamics of the search during a derivation is described by *marking functions* for vertices and arcs:

**Definition 41** *A* parallel marked search-graph $(V, E, l, h, \bar{s}, \bar{c})$ *for* $p_0, \ldots p_{n-1}$ *has an n-tuple* $\bar{s}$ *of* vertex-marking functions $s^k: V \to Z$ *such that*

$$s^k(v) = \begin{cases} m & \text{if } m \text{ variants } (m > 0) \text{ of } l(v) \text{ are present for process } p_k, \\ -1 & \text{if all variants of } l(v) \text{ have been deleted by process } p_k, \\ 0 & \text{otherwise;} \end{cases}$$

*and an n-tuple* $\bar{c}$ *of* arc-marking functions $c^k: E \to \mathbb{N} \times Bool$ *such that* $\pi_1(c^k(e))$ *is the number of times* $p_k$ *executed* $e$ *or received a clause generated by* $e$*, and* $\pi_2(c^k(e)) = true/false$ *if* $p_k$ *is allowed/forbidden to execute* $e$*.*

Hyperarc $e = (v_1 \ldots v_n; w; u)$ is *enabled* for $p_k$ if $s^k(v_j) > 0$ for $1 \leq j \leq n$, $s^k(w) > 0$, and $\pi_2(c^k(e)) = true$.

**Definition 42** *A parallel derivation induces* $n$ successions of vertex-marking functions $\{s_i^k\}_{i \geq 0}$*, one per process. For all* $v \in V$*,* $s_0^k(v) = 0$*, and* $\forall i \geq 0$*:*

- *If at stage* $i$ $p_k$ *executes an enabled hyperarc* $e = (v_1, \ldots, v_n; w; u)$*:*

$$s_{i+1}^k(v) = \begin{cases} s_i^k(v) - 1 & \text{if } v = w \ (-1 \text{ if } s_i^k(v) = 1), \\ s_i^k(v) + 1 & \text{if } v = u \ (+1 \text{ if } s_i^k(v) = -1), \\ s_i^k(v) & \text{otherwise.} \end{cases}$$

- *If at stage* $i$ $p_k$ *receives* $\bar{x} = (\varphi_1, \ldots \varphi_n)$*, where* $\varphi_j = l(v_j)$ *for* $1 \leq j \leq n$*:*

$$s_{i+1}^k(v) = \begin{cases} s_i^k(v) + 1 & \text{if } v \in \{v_1, \ldots v_n\} \ (+1 \text{ if } s_i^k(v) = -1), \\ s_i^k(v) & \text{otherwise.} \end{cases}$$

- *If at stage* $i$ $p_k$ *sends* $\bar{x}$*,* $s_{i+1}^k(v) = s_i^k(v)$*.*

Note that $s_0^k(v) = 0$ also for input clauses: the steps of reading or receiving input clauses are included in the derivation (read steps can be modelled as expansion steps), because the subdivision function applies to input clauses.

**Definition 43** *A parallel derivation induces* $n$ successions of arc-marking functions $\{c_i^k\}_{i \geq 0}$*:* $\forall a \in E$*,* $\pi_1(c_0^k(a)) = 0$ *and* $\pi_2(c_0^k(a)) = true$*, and* $\forall i \geq 0$*:*

$$\pi_1(c_{i+1}^k(a)) = \begin{cases} \pi_1(c_i^k(a)) + 1 & \text{if } p_k \text{ executes } a \text{ or receives a clause generated by } a, \\ \pi_1(c_i^k(a)) & \text{otherwise;} \end{cases}$$

$$\pi_2(c_{i+1}^k(a)) = \begin{cases} \alpha((S_0 \ldots S_{i+1}), n, k, f, \bar{x}) & \text{if } \alpha((S_0 \ldots S_{i+1}), n, k, f, \bar{x}) \neq \perp, \\ true & \text{otherwise (arcs allowed by default),} \end{cases}$$

*where* $h(a) = f$ *and* $\bar{x}$ *is the tuple of premises of hyperarc* $a$*.*

## 5 Measures of search complexity

Let $G = (V, E, l, h)$ be a search graph. For all $v \in V$, if $v$ has no incoming hyperarcs, the *ancestor-graph* of $v$ is the graph made of $v$ itself; if $e = (v_1 \dots v_n; v)$ is a hyperarc and $t_1 \dots t_n$ are ancestor-graphs of $v_1 \dots v_n$, the graph with root $v$ connected by $e$ to $t_1 \dots t_n$ is an *ancestor-graph* of $v$, denoted by $(v; e; (t_1, \dots, t_n))$. (To simplify the notation, we include the deleted premise in $v_1 \dots v_n$.) The set of the ancestor-graphs of $v$ in $G$ is denoted by $at_G(v)$ (or $at_G(\varphi)$). From now on, $G = (V, E, l, h, \bar{s}, \bar{c})$ is the parallel marked search graph searched by $p_0, \dots p_{n-1}$, $G^k = (V, E, l, h, s^k, c^k)$ is the marked search graph for $p_k$ and $G^k_i = (V, E, l, h, s^k_i, c^k_i)$ is the marked search graph for $p_k$ at stage $i$ of a derivation.

If a *relevant* ancestor of $\varphi$ in $t \in at_G(\varphi)$ is deleted by contraction, it becomes impossible to reach $\varphi$ by traversing $t$:

**Definition 51** *Let $t = (v; e; (t_1 \dots t_n)) \in at_G(v)$ with $e = (v_1 \dots v_n; v)$. A vertex $w \in t$, $w \neq v$, is* relevant *to $v$ in $t$ for $p_k$ ($w \in Rev_{G^k}(t)$) if either $w \in \{v_1 \dots v_n\}$ and $\pi_1(c^k(e)) = 0$, or $\exists i\ 1 \leq i \leq n$ s. t. $w$ is relevant to $v_i$ in $t_i$ for $p_k$.*

If $\pi_1(c^k(e)) \neq 0$ because $p_k$ executed $e$, deleting $\psi$ is irrelevant for $\varphi$, since $\psi$ has been already used to generate $\varphi$. If $\pi_1(c^k(e)) \neq 0$ because $p_k$ received a variant of $\varphi$ generated by $e$, deleting $\psi$ is irrelevant for $\varphi$, since $\varphi$ came from the outside.

Given $t \in at_G(\varphi)$, the *p(ast)-distance* measures the portion of $t$ that $p_k$ has visited; the *f(uture)-distance* measures the portion of $t$ that $p_k$ needs to visit to reach $\varphi$ by traversing $t$; the *g(lobal)-distance* is their sum:

**Definition 52** *For all clauses $\varphi$, ancestor-graphs $t \in at_G(\varphi)$ and processes $p_k$:*

- *The* p-distance of $\varphi$ on $t$ for $p_k$ *is* $pdist_{G^k}(t) = \mid \{w \mid w \in t, s^k(w) \neq 0\} \mid$.
- *The* f-distance of $\varphi$ on $t$ for $p_k$ *is*

$$fdist_{G^k}(t) = \begin{cases} \infty & \text{if } s^k(\varphi) < 0, \text{ or} \\ & \exists w \in Rev_{G^k}(t),\ s^k(w) < 0, \\ \mid \{w \mid w \in t, s^k(w) = 0\} \mid & \text{otherwise.} \end{cases}$$

- *The* g-distance of $\varphi$ on $t$ for $p_k$ *is* $gdist_{G^k}(t) = pdist_{G^k}(t) + fdist_{G^k}(t)$.

*The* f-distance of $\varphi$ in $G$ for $p_k$ *is* $fdist_{G^k}(\varphi) = min\{fdist_{G^k}(t) \mid t \in at_G(\varphi)\}$.
*The* g-distance of $\varphi$ in $G$ for $p_k$ *is* $gdist_{G^k}(\varphi) = min\{gdist_{G^k}(t) \mid t \in at_G(\varphi)\}$.

While infinite distance captures the effect of contraction, we consider next subdivision:

**Definition 53** *An ancestor-graph $t$ is* forbidden *for process $p_k$ if there exists an arc $e$ in $t$ such that $\pi_1(c^k(e)) = 0$ and $\pi_2(c^k(e)) = false$. It is* allowed *otherwise.*

If $p_k$ receives the clause that $e$ generates, it is $\pi_1(c^k(e)) \neq 0$, and $t$ is no longer forbidden, because it is no longer true that forbidding $e$ prevents $p_k$ from exploring $t$. This may happen only as a consequence of communication, because

$e$ cannot be executed. Thus, subdivision forbids ancestor-graphs, reducing the search effort of each process, but communication may undo it. Indeed, a strategy employs communication to preserve fairness in presence of a subdivision. For fairness, it is sufficient that every non-redundant path is allowed to one process. If more than one process is allowed, their searches may overlap:

**Definition 54** *$p_k$ and $p_h$* overlap *on ancestor-graph $t$ if $t$ is allowed for both.*

An overlap represents a potential duplication of search effort, hence a waste. A goal of a parallel search plan is to preserve fairness while minimizing the overlap.

**Definition 55** *The* bounded search space *within distance $j$ for $p_k$ is the multiset of clauses $space(G^k, j) = \sum_{v \in V, v \neq \mathsf{T}} mul_{G^k}(v, j) \cdot l(v)$, where each $l(v)$ has multiplicity $mul_{G^k}(v, j) = |\{t \mid t \in at_G(v), t \text{ allowed } for\ p_k, 0 < gdist_{G^k}(t) \leq j\}|$.*

The sequential bounded search spaces $space(G, j)$ are defined in the same way, with multiplicity $mul_G(v, j) = |\{t \mid t \in at_G(v), 0 < gdist_G(t) \leq j\}|$.

**Definition 56** *The* parallel bounded search space *within distance $j$ is the multiset of clauses $pspace(G, j) = \sum_{v \in V, v \neq \mathsf{T}} pmul_G(v, j) \cdot l(v)$, where $l(v)$ has multiplicity $pmul_G(v, j) = \lfloor gmul_G(v, j)/n \rfloor$ with $gmul_G(v, j) = \sum_{k=0}^{n-1} mul_{G^k}(v, j)$.*

If $p_h$ and $p_k$ overlap on an ancestor-graph $t$, $t$ is counted in both $mul_{G^h}(v, j)$ and $mul_{G^k}(v, j)$ and twice in $gmul_G(v, j)$, reflecting the overlap.

**Theorem 51** *If $\alpha$ is the constant function true (no subdivision), and no communication occurs, then for all $p_k$, $i \geq 0$ and $j > 0$, $space(G_i^k, j) = space(G_i, j)$, and $pspace(G_i, j) = space(G_i, j)$.*

## 6 The analysis

In a sequential derivation, if a clause $\varphi$ is deleted at stage $i$ and regenerated via another ancestor-graph at some stage $j > i$, a contraction-based strategy will delete it again, and will do so before using $\varphi$ to generate other clauses [10]. It follows that when $\varphi$ is re-deleted, it is still relevant on all ancestor-graphs where it was relevant at stage $i$. Therefore, it is possible to make the following approximation: if $fdist_{G_i^k}(t)$ is infinite, $fdist_{G_j^k}(t)$ can be regarded as infinite for all $j > i$. In a parallel derivation, the situation is more complex. First, we need to consider not only the possibility that a process $p_k$ regenerates $\varphi$ via another ancestor-graph, but also the possibility that $p_k$ receives another variant of $\varphi$ from another process, which may not be aware that $\varphi$ is redundant. Second, we need to consider not only deleted clauses, but also clauses such that $fdist_{G_i^k}(\varphi) = \infty$ because a relevant ancestor has been deleted on every $t \in at_G(\varphi)$. In a sequential derivation, it is impossible for $\varphi$ to appear at some $j > i$, because it cannot be generated, but in a parallel derivation, $\varphi$ may still be received from another process at some $j > i$. Thus, we prove a more general result:

**Lemma 61** *In a derivation with local eager contraction, for all $p_k$, $i$, and $\varphi$, if $fdist_{G_i^k}(\varphi) = \infty$ (regardless of whether $\varphi$ is deleted or made unreachable) and $s_j^k(\varphi) > 0$ for some $j > i$ (regardless of whether $\varphi$ is received or regenerated), there exists a $q > j$, such that $s_q^k(\varphi) < 0$ (hence $fdist_{G_q^k}(\varphi) = \infty$), and $p_k$ does not use $\varphi$ to generate other clauses at any stage $l$, $j \leq l < q$.*

Therefore, we can make the approximation that $fdist_{G_i^k}(\varphi) = \infty$ implies $\forall j > i \; fdist_{G_j^k}(\varphi) = \infty$. This takes care of clauses that the strategy finds redundant. Consider a non-redundant clause $\varphi$ and an ancestor-graph $t$ of $\varphi$ such that $fdist_{G_i^k}(t) = \infty$ because $s_i^k(\psi) = -1$ for a relevant ancestor $\psi$ in $t$. For simplicity, let $\psi$ be a parent of $\varphi$, with arc $e$ from $\psi$ to $\varphi$. Assume that $p_h$ has not deleted $\psi$, executes $e$, and sends to $p_k$ a $\varphi$ generated by $e$. The arrival of $\varphi$ at some stage $r > i$ makes $\psi$ irrelevant ($\pi_1(c_r^k(e) = 1)$, so that $fdist_{G_r^k}(t) \neq \infty$. There is irrelevance of contraction at $p_h$, because the clause(s) that contract $\psi$ do not arrive at $p_h$ fast enough to delete $\psi$ before it is used to generate $\varphi$. When $p_h$ finally deletes $\psi$, this deletion is irrelevant to $t$, because $p_h$ has already executed $e$: we call this phenomenon *late contraction*. There is irrelevance of contraction at $p_k$, because the arrival of $\varphi$ from $p_h$ makes the deletion of $\psi$ irrelevant: we call this phenomenon *contraction undone*. Distributed global contraction guarantees that $p_h$ will delete $\psi$ eventually, so that $s_j^h(\psi) = -1$ for some $j > i$. It is sufficient that $p_h$ executes $e$ and generates $\varphi$ at a stage $l < j$, and $\varphi$ arrives at $p_k$ at a stage $r > i$, for this situation to occur. Thus, distributed global contraction is not sufficient to prevent late contraction and contraction undone.

The following theorems integrate all our observations on subdivision, contraction and communication:

1. *If $S_i^k \vdash S_{i+1}^k$ generates $\psi$, then $\forall j > 0$, $space(G_{i+1}^k, j) \preceq_{mul} space(G_i^k, j)$.* When $\psi$ is generated, the subdivision function $\alpha$ may become defined on a tuple of premises $\bar{x}$ including $\psi$. If $\alpha$ decides that an arc $e$ with premises $\bar{x}$ is forbidden, ancestor-graphs including $e$ become forbidden, so that the bounded search spaces become smaller.
2. *If $S_i^k \vdash S_{i+1}^k$ replaces $\psi$ by $\psi'$, then $\forall j > 0$, $space(G_{i+1}^k, j) \preceq_{mul} space(G_i^k, j)$.* A contraction step replacing $\psi$ by $\psi'$ prunes those ancestor-graphs whose distance becomes infinite because of the deletion of $\psi$, and those ancestor-graphs which become forbidden as a consequence of the generation of $\psi'$.
3. *If $S_i^k \vdash S_{i+1}^k$ sends $\psi$, then $\forall j > 0$, $space(G_{i+1}^k, j) = space(G_i^k, j)$. If $S_i^k \vdash S_{i+1}^k$ receives $\psi$, $\forall j > 0$, $\exists\, l \leq i$, $space(G_{i+1}^k, j) \preceq_{mul} space(G_l^k, j)$.* When $p_k$ receives $\psi$, there may be three kinds of consequences: allowed ancestor-graphs may become forbidden (subdivision), reducing the multiplicity of some clauses; forbidden ancestor-graphs may become allowed (subdivision undone) and relevant deleted ancestors may become irrelevant (contraction undone), increasing the multiplicity of some clauses. However, since communication cannot expand the bounded search spaces, but only undo previous reductions, the resulting bounded search spaces are limited by the bounded search spaces at some previous stage.

These theorems show that the bounded search spaces capture all relevant phenomena: pruning by contraction, subdivision and cost of communication. While in a sequential derivation the bounded search spaces may either remain the same (expansion) or decrease (contraction), in a parallel derivation the bounded search spaces of a process may oscillate *non-monotonically* because of communication. The faster is communication, however, the lesser is the incidence of late contraction and contraction undone; at the limit, if the strategy has instantaneous propagation of clauses up to redundancy, they disappear:

**Lemma 62** *In a derivation with local eager contraction and instantaneous propagation of clauses up to redundancy, let $e$ be an arc of $t \in at_G(\varphi)$ which uses $\psi$ and generates $\psi' \in S_\infty - R(S_\infty)$. If $s_i^k(\psi) = -1$ and $\psi \in Rev_{G_i^k}(t)$ for some $p_k$:*

1. *$\forall p_h$, $\forall j$, $s_j^h(\psi) = -1$ implies $\psi \in Rev_{G_j^h}(t)$ (what is relevant to one process is relevant to all: no late contraction).*
2. *$\forall j \geq i$, $\psi \in Rev_{G_j^k}(t)$ (what is relevant at a stage remains relevant at all following stages: no contraction undone).*

The approximation $fdist_{G_i^k}(t) = \infty \Rightarrow \forall j > i \; fdist_{G_j^k}(t) = \infty$ can be made:

**Theorem 61** *In a derivation with local eager contraction and instantaneous propagation of clauses up to redundancy, if $fdist_{G_i^k}(t) = \infty$ and $fdist_{G_j^k}(t) \neq \infty$ for some $0 < i < j$, there exists a $q > j$ such that $fdist_{G_q^k}(t) = \infty$.*

Next, we turn our attention to the overlap. We observe that two overlapping processes may generate variants of the same clause. The following property prevents different processes from generating variants of the same clause:

**Definition 61** *A subdivision function $\alpha$ has* no clause-duplication *if for all vertices $u \neq \mathsf{T}$, for any two hyperarcs into $u$, $e_1$ with inference rule $f$ and premises $\bar{x}$, and $e_2$ with inference rule $g$ and premises $\bar{y}$, $\forall i \geq 0$, if $\alpha((S_0, \ldots S_i), n, k, f, \bar{x}) = true$ and $\alpha((S_0, \ldots S_i), n, h, g, \bar{y}) = true$, then $k = h$.*

This property is compatible with fairness, for which one allowed process is sufficient. We show next that the combination of no clause-duplication and local eager contraction minimizes the overlap. There are two kinds of overlap: one caused by the subdivision function itself when it allows the same arc to more than one process, and one caused by communication (e.g., $\pi_2(c^k(e)) = true$ and $\pi_2(c^h(e)) = false$ but $\pi_1(c^h(e)) \neq 0$). No clause-duplication avoids the first kind of overlap by definition. For the second one, assume that $p_k$ is the only process authorized to generate all variants of $\varphi$. By local eager contraction, if $p_k$ generates more than one variant of $\varphi$, all but one are deleted before being sent to any other process. Thus, $p_k$ may send to another process only one variant, and the same variant to all processes, so that:

**Lemma 63** *In a derivation with local eager contraction and no clause-duplication, for any clause $\varphi$, if $p_h$ is the only process allowed to generate $\varphi$, $\exists r$ such that $\forall k \neq h$, $\forall i \geq r$, $\forall j > 0$, $mul_{G_i^k}(\varphi, j) \leq 1$ (i.e, communication may make* at most one *forbidden ancestor-graph allowed).*

We have all the elements to compare a contraction-based, uniformly fair strategy $\mathcal{C} = \langle I, \Sigma \rangle$ with a parallelization by subdivision $\mathcal{C}' = \langle I, M, \Sigma' \rangle$, which is uniformly fair and contraction-based. Since $\mathcal{C}$ and $\mathcal{C}'$ have the same inference system, the initial search space is the same (i.e., $G_0^k = G_0$ and $space(G_0^k, j) = space(G_0, j)$ for all $k$ and $j$). We compare first the behaviour on redundant clauses, next on ancestor-graphs including redundant inferences, and then on the remaining ancestor-graphs. We begin by proving three preliminary lemmas:

1. *If $\varphi \in S_i$ for some $i$, then $\exists p_k\ \exists j$ such that either $\varphi \in S_j^k$ or $\varphi \in R(S_j^k)$.*
2. *If $\varphi \in R(S_i)$ for some $i$, then $\forall p_k\ \exists j$ such that $\varphi \in R(S_j^k)$.*
3. *If $fdist_{G_i}(\varphi) = \infty$ for some $i$, then $\forall p_k\ \exists j$ s. t. $\forall l \geq j$, either $fdist_{G_l^k}(\varphi) = \infty$, or all $t \in at_G(\varphi)$ are forbidden for $p_k$ at stage $l$.*

These allow us to show that all redundant clauses eliminated by $\mathcal{C}$ will be excluded by $\mathcal{C}'$ as well:

**Theorem 62** *If $fdist_{G_i}(\varphi) = \infty$ for some $i$, then there exists an $r$ such that for all $i \geq r$ and $j > 0$, $pmul_{G_i}(\varphi, j) = 0$.*

To show that all ancestor-graphs pruned by $\mathcal{C}$ are pruned by $\mathcal{C}'$, we need to use Lemma 62 to prevent late contraction and contraction undone, and this can be done only under the hypothesis of instantaneous propagation of clauses:

**Lemma 64** *Assume that $\mathcal{C}'$ has instantaneous propagation of clauses up to redundancy. If $fdist_{G_i}(t) = \infty$ for some $i$, then for all $p_k$ there exists a $j$ such that for all $l \geq j$, either $fdist_{G_l^k}(t) = \infty$, or $t$ is forbidden for $p_k$ at stage $l$.*

The final lemma covers ancestor-graphs that are not pruned. Thus, we need a hypothesis on subdivision, and we assume that $\mathcal{C}'$ has no clause-duplication:

**Lemma 65** *Assume that $\mathcal{C}'$ has instantaneous propagation of clauses up to redundancy and no clause-duplication. If $fdist_{G_i}(\varphi) \neq \infty$ for all $i$, there exists an $r$ such that for all $i \geq r$ and $j > 0$, $pmul_{G_i}(\varphi, j) \leq mul_{G_i}(\varphi, j)$.*

**Theorem 63** *If $\mathcal{C}'$ has instantaneous propagation of clauses up to redundancy and no clause-duplication, $\forall j\ \exists m$ s. t. $\forall i \geq m$ $pspace(G_i, j) \preceq_{mul} space(G_i, j)$.*

Intuitively, a value $j$ of the bound may represent the search depth required to find a proof. If the problem is hard enough that the sequential strategy does not succeed before stage $m$, the parallel strategy faces a smaller bounded search space beyond $m$, and therefore may succeed sooner.

Theorem 63 is a limit theorem, in a sense similar to other theoretical results obtained under an ideal assumption. On one hand, it explains the nature of the problem, by indicating in the overlap and the communication-contraction node its essential aspects. On the other hand, it represents a limit that concrete strategies may approximate by improving overlap control and communication.

## 7 Discussion

If it had been possible to prove that a contraction-based parallelization has smaller bounded search spaces without assuming instantaneous communication, there would have been a ground to expect a generalized success of distributed search, at least to the extent to which smaller bounded search search spaces mean shorter search. However, we found that this type of result does not hold. Therefore, a distributed-search contraction-based strategy may do better than its sequential counterpart, but it is not guaranteed to. When adopting distributed search, one expects that communication will have a cost, and contraction may be delayed. The trade-off is to accept these disadvantages in order to avoid synchronization (a method where parallel processes have to synchronize on every inference in order to enforce eager contraction would be hopeless). Also, one may conjecture that the advantage of subdivision will offset the cost of communication in terms of delayed contraction. Our analysis showed that this conjecture does not hold on the bounded search spaces. In summary, this analysis contributes to explain why the parallelization of efficient forward-reasoning strategies has been an elusive target. Furthermore, the explanation is analytical, rather than based solely on empirical observations.

So little is known about complexity in theorem proving, and strategy analysis, however, that these findings should be regarded as a beginning, not a conclusion. In this paper we have tried essentially to determine whether distributed search may make the search space *smaller* by doing at least as much contraction as the sequential process and adding the effect of the subdivision. Accordingly, we have compared bounded search spaces by comparing the multiplicities of each clause. It remains the question of whether distributed search may take advantage by performing steps in different order, especially contraction steps, hence producing *different* search spaces. Thus, a first direction for further research may be to find other ways to compare the bounded search spaces, which may shed light on other aspects, and possibly other advantages, of distributed search. Another direction for future work is to apply the bounded search spaces to analyze *multi-search contraction-based strategies*. These issues may be connected to, or even require, the continuation of the analysis of sequential contraction-based strategies. In [10], we compared strategies with the same search plan and inference systems different in contraction power. The complementary problem of analyzing sequential strategies with the same inference system but different search plans still needs to be addressed. Finally, we have considered only forward-reasoning strategies; another line of research is to extend our methodology to subgoal-reduction strategies, such as those based on model elimination.

## References

1. S. Anantharaman and M. P. Bonacina. An application of automated equational reasoning to many-valued logic. In M. Okada and S. Kaplan, editors, *CTRS-90*, volume 516 of *LNCS*, pages 156–161. Springer Verlag, 1990.

2. S. Anantharaman and J. Hsiang. Automated proofs of the Moufang identities in alternative rings. *J. of Automated Reasoning*, 6(1):76–109, 1990.
3. L. Bachmair and H. Ganzinger. Non-clausal resolution and superposition with selection and redundancy criteria. In A. Voronkov, editor, *LPAR-92*, volume 624 of *LNAI*, pages 273–284. Springer Verlag, 1992.
4. L. Bachmair and H. Ganzinger. A theory of resolution. Technical Report MPI-I-97-2-005, Max Planck Institut für Informatik, 1997.
5. M. P. Bonacina. On the reconstruction of proofs in distributed theorem proving: a modified Clause-Diffusion method. *J. of Symbolic Computation*, 21:507–522, 1996.
6. M. P. Bonacina. Experiments with subdivision of search in distributed theorem proving. In M. Hitz and E. Kaltofen, editors, *PASCO-97*, pages 88–100. ACM Press, 1997.
7. M. P. Bonacina. Distributed contraction-based strategies: model and analysis. Technical Report 98-02, Dept. of Computer Science, University of Iowa, 1998.
8. M. P. Bonacina and J. Hsiang. Parallelization of deduction strategies: an analytical study. *J. of Automated Reasoning*, 13:1–33, 1994.
9. M. P. Bonacina and J. Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theoretical Computer Science*, 146:199–242, 1995.
10. M. P. Bonacina and J. Hsiang. On the modelling of search in theorem proving – Towards a theory of strategy analysis. *Information and Computation*, forthcoming, 1998.
11. R. Bündgen, M. Göbel, and W. Küchlin. Strategy-compliant multi-threaded term completion. *J. of Symbolic Computation*, 21:475–506, 1996.
12. J. Denzinger and S. Schulz. Recording and analyzing knowledge-based distributed deduction processes. *J. of Symbolic Computation*, 21:523–541, 1996.
13. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier, Amsterdam, 1990.
14. D. Kapur and H. Zhang. An overview of RRL: rewrite rule laboratory. In N. Dershowitz, editor, *3rd RTA*, volume 355 of *LNCS*, pages 513–529. Springer Verlag, 1989.
15. C. Kirchner, C. Lynch, and C. Scharff. Fine-grained concurrent completion. In H. Ganzinger, editor, *7th RTA*, volume 1103 of *LNCS*, pages 3–17. Springer Verlag, 1996.
16. A. Leitsch. *The Resolution Calculus*. Springer, Berlin, 1997.
17. W. McCune. Otter 3.0 reference manual and guide. Technical Report 94/6, MCS Div., Argonne Nat. Lab., 1994.
18. W. McCune. Solution of the Robbins problem. *J. of Automated Reasoning*, 19(3):263–276, 1997.
19. D. A. Plaisted. Equational reasoning and term rewriting systems. In D. Gabbay and J. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 273–364. Oxford University Press, New York, 1993.
20. D. A. Plaisted and Y. Zhu. *The Efficiency of Theorem Proving Strategies*. Friedr. Vieweg & Sohns, 1997.
21. C. B. Suttner and J. Schumann. Parallel automated theorem proving. In L. Kanal, V. Kumar, H. Kitano, and C. B. Suttner, editors, *Parallel Processing for Artificial Intelligence*. Elsevier, Amsterdam, 1994.
22. A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1:425–467, 1995.