Annals of Mathematics and Artificial Intelligence 27 (1999) 149–199

A model and a first analysis of distributed-search contraction-based strategies *

Maria Paola Bonacina [a,**]

[a] *Department of Computer Science – The University of Iowa*
Iowa City, IA 52242-1419, USA
E-mail: bonacina@cs.uiowa.edu

While various approaches to parallel theorem proving have been proposed, their usefulness is evaluated only empirically. This research is a contribution towards the goal of machine-independent analysis of theorem-proving strategies. This paper considers clausal contraction-based strategies and their parallelization by distributed search, with subdivision of the search space and propagation of clauses by message-passing (e.g., à la Clause-Diffusion). A model for the representation of the parallel searches produced by such strategies is presented, and the bounded-search-spaces approach to the measurement of search complexity in infinite search spaces is extended to distributed search. This involves capturing both its advantages, e.g., the subdivision of work, and disadvantages, e.g., the cost of communication, in terms of search space. These tools are applied to compare the evolution of the search space of a contraction-based strategy with that of its parallelization in the above sense.

1. Introduction

The difficulty of fully-automated theorem proving has led to investigate ways of enhancing theorem-proving strategies with parallelism. Many approaches have been proposed, and the interested reader can find a description of the state of the art up to the early Nineties in [26,17,44], and a more recent overview in [15]. Since theorem-proving strategies are complex objects, parallelism can be introduced at different levels. A distinction among *parallelism at the term level*, *parallelism at the clause level*, and *parallelism at the search level* was introduced in [17]. In brief, parallelism at the term level means parallelizing the inner algorithms of the strategy (e.g., parallel term rewriting), parallelism at the clause level means parallel inferences within a single search (e.g., parallel resolution steps), and parallelism at the search level means parallel search. This paper considers *parallel search*:

* This paper is a significantly extended and revised version of "Analysis of distributed-search contraction-based strategies," *Proc. of JELIA-98*, Springer, LNAI 1489, 107–121, 1998, and "On the representation of parallel search in theorem proving," *Proc. of FTP-97*, Technical Report 97-50, RISC, J. Kepler Universität, Linz, 22–28, 1997.

** Supported in part by National Science Foundation grants CCR-94-08667 and CCR-97-01508.

concurrent deductive processes search in parallel the search space of the theorem-proving problem; each process executes a theorem-proving strategy, develops its own derivation and builds its own set of data (e.g., clauses or equations); the parallel search halts successfully as soon as one of the processes finds a proof.

In systems with parallel search, the deductive processes maintain separate databases (i.e., logically distributed memory), and communicate by message-passing. Since a shared memory is not needed, parallel search can be implemented on networks of workstations or multiprocessors, with either distributed or shared memory, and if there is a shared memory, it is used for message passing. Libraries for parallel programming (e.g., MPI [29]) allow the developer to write a parallel theorem prover with message passing, and then compile it on either a network or a multiprocessor.

There are many approaches to parallel search, which differ in how they differentiate and combine the activities of the deductive processes. In some methods the processes have the same inference system but *different search plans* (e.g., Team-Work [5,24] and see also [45] for combination of search plans in a sequential prover). In other methods the processes may have also *different inference systems* (e.g., [48,28,27]). Yet other methods let all processes have the same inference system and search plan, but *subdivide the search space* among the processes (e.g., Clause-Diffusion [18,12]); and this classification is not meant to be exhaustive. Furthermore, these principles are not mutually exclusive: for instance, Clause-Diffusion allows both subdivision of the search space and multiple search plans [19], and the method of [27] also features problem decomposition by splitting clauses. In this paper, we use *distributed search* for the methods that subdivide the search space, using the word "distributed" in its literal meaning of "giving each a share of something." For methods with multiple search plans, we suggest *multi-search*. In all these approaches, the processes need to *communicate*: distributed search needs communication to preserve completeness and load-balance; multi-search needs communication to merge the results of the processes, and also for completeness, if unfair search plans are used, and load-balance, if the problem is decomposed. This paper begins an analysis of distributed search; an analysis of multi-search is a direction for future work.

Parallel search may be applied to theorem-proving strategies in general. This paper studies *contraction-based* strategies, such as those originated from the Knuth-Bendix and term-rewriting paradigm on one hand, and the resolution-paramodulation paradigm on the other (e.g., [25,41,42,20,8] for general treatments and references). These strategies work mostly by *forward reasoning* (i.e., by deriving consequences from the assumptions and the negation of the target theorem), and therefore operate on a database of generated clauses. Their defining characteristics are a *well-founded ordering* on the data domain, a notion of *redundancy* of data based on the ordering, *contraction inference rules* to delete redundant data, *restrictions of expansion inference rules* to prevent the generation of redundant data, and an *eager-contraction* search plan. Examples of contrac-

tion rules are rewriting by equations and subsumption. Examples of restrictions of expansion are critical pair criteria (e.g., [6]), ordered inference rules (e.g., [31]) and basic inference rules (e.g., [9]). Among the features of these strategies, this paper is concerned with contraction and eager contraction, leaving the analysis of restrictions of expansion, as well as reasoning modulo a theory, to future work. Eager contraction means that the search plan selects an expansion inference rule only when no contraction rule is applicable. The purpose is to maintain the database maximally reduced, so that the strategy does not run out of memory. This requires to insert in the database only normalized clauses (*forward contraction*), and to keep the clauses already in the database normalized with respect to the insertions (*backward contraction*).

There has been a lot of interest in parallelizing contraction-based strategies (e.g., [18,24,23,12,33,22,27] and [26,17,44] for earlier references), because they behave well sequentially. Many theorem provers developed in recent years (e.g., Otter [37], Reveal [2], RRL [49], EQP [38], SPASS [47], Barcelona [40], and Gandalf [45]) implement these strategies, and also thanks to them succeeded in solving challenge problems (e.g., [2,32,1,39]). However, their parallelization is difficult, because of the size and variability of the database. Backward contraction means that the clauses used as premises are subject to contraction. It follows that the database is highly dynamic, and eager backward contraction is in conflict with eager parallel inference, because eager backward contraction imposes to inter-reduce before expanding, and this reduces the degree of concurrency of the inferences. The qualitative analysis in [17] pointed out how these factors may affect adversely approaches based on parallelism at the term or clause level.

One of the ideas behind the approach of subdividing the search space is that if each parallel process is allowed to consider only part of the search space and ignore the rest (because it is taken care of by others), it may find a proof sooner than a sequential process, which has to deal with the whole search space. The downside is that in order to preserve completeness (and eager contraction) in the presence of subdivision, the processes need to communicate. Furthermore, the subdivision may not be effective, so that the processes overlap and duplicate work. The problem then is: how can we *represent* and *measure* the advantages (e.g., subdivision) and disadvantages (e.g., communication and overlap) of distributed search, in order to balance them and see whether one dominates the other? It should be emphasized that building the mathematical tools to carry out this type of analysis is probably at least as important as an answer to the question. This is because the search spaces of theorem proving are *infinite*. Therefore, one cannot analyze subdivision and overlap in terms of total size of the search space. Neither can one 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. For the same reason, the standard approach in algorithm analysis – compare the worst-case time complexity of the fastest known sequential algorithm with the worst-case time complexity of

the parallel algorithm – does not apply.

In recent work [21], we proposed an approach to the analysis of sequential forward-reasoning strategies with contraction, 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*. These tools were applied to compare contraction-based strategies of different contraction power. In this paper we extend this approach to distributed search, in order to begin addressing the question outlined above. We emphasize that this paper aims at providing tools for the analysis, and demostrate their applicability by obtaining a first set of results, rather than giving a final answer. Also, this type of analysis is not supposed to replace, but *to complement* the experimental evaluation of strategies (e.g., [13,14] for most recent Clause-Diffusion experiments), like algorithm analysis complements experiments in other areas of computer science.

1.1. Organization and contents of the paper

Section 2 gives the basic definitions for the type of parallel strategies under analysis. Since parallelism belongs to the control part of a strategy, a central notion is that of *distributed-search plan* with a *subdivision function* to distribute the inferences. Section 3 studies three properties of distributed-search plans: *monotonicity of the subdivision*, *fairness* and *eager contraction*. For fairness, we give sufficient conditions for the parallelization of a fair strategy to be fair. For eager contraction, we point out that it is not obvious that a parallelization of a contraction-based strategy is contraction-based. We clarify what it means for a distributed strategy to be contraction-based, and we give sufficient conditions for this property also.

Section 4 presents the model for the representation of distributed search. The approach of [21] to the modelling of search was motivated by the need of extending the classical sequential model (e.g., [34]) with *contraction*. They key point is that a strategy with contraction not only visits, but also *modifies* the search space. In distributed search the modelling problem becomes more complicated, because not only contraction, but also *subdivision* and *communication* modify the search space. Furthermore, many processes are active in parallel. Our solution is based on distinguishing the search space and the dynamics of the search, 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 dynamics of the search, including contraction, subdivision and communication for all the processes.

Section 5 turns to the problem of measuring benefits and costs of distributed search. The methodology of [21] is based on the observation that for infinite search spaces it is not sufficient to consider 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 a clause may prevent the generation of others. These two domains are called the *present* and the *future* of the derivation. The future is the problematic part, because it is infinite. Our approach is to enrich the search space with a notion of *distance*. Once such a metric is in place, one can 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 bounded search spaces. Since they are *finite*, they can be compared, eliminating the obstacle of the impossibility of comparing infinite spaces. More precisely, they are defined as multisets of clauses, so that they can be compared in the multiset extension of a well-founded ordering on clauses. The second key property of the bounded search spaces is that they are *dynamic*: they depend on the steps selected by the strategy. Contraction affects the bounded search spaces by making clauses unreachable (infinite distance); subdivision excludes paths from the bounded search spaces; on the other hand, communication may *undo* in part the effect of subdivision, because the strategy uses communication to preserve fairness (hence completeness) in the presence of a subdivision. In addition to studying the bounded search spaces of the parallel processes independently, we give a formal definition of *overlap*, and we introduce *parallel bounded search spaces* for the distributed derivation as a whole.

Section 6 contains the analysis. We study first *eager contraction*: its essence is to delete a redundant clause *before* it is used to generate other clauses. In distributed search, eager contraction needs communication (e.g., to bring to a process a needed simplifier). We discover two patterns of behaviour, called *late contraction* and *contraction undone*, where the interaction of communication and contraction causes eager contraction to fail. The subtlety of this analysis is that it requires to study not only redundant clauses, but also redundant inferences. Second, we analyze the *overlap*: there are two kinds of overlap, overlap due to inaccurate subdivision, and overlap due to communication. Sufficient conditions to avoid the former and minimize the latter are given. Third, we put all the previous observations together by showing how the different kinds of actions in a distributed derivation (expansion, contraction, subdivision, communication) modify the bounded search spaces. The last task is to compare a sequential contraction-based strategy $\mathcal{C}$ with its parallelization $\mathcal{C}'$. We analyze the evolution of their respective bounded search spaces, and we prove that while all clauses deleted or made unreachable by $\mathcal{C}$ are deleted or made unreachable also by $\mathcal{C}'$, it is possible to exclude that the parallel bounded search spaces may be greater than the sequential ones, only if the overlap were minimized and propagation of clauses were immediate. This result is based on a worst-case analysis of the relation of communication and contraction. It may be interpreted on one hand as a negative result on the parallelizability of contraction-based strategies, and on the other hand as a limit that strategies may approximate: for instance, it justifies formally the intuition of improving performance by devising subdivision criteria that reduce the overlap (e.g., the ancestor-graph oriented criteria of [13]).

1.2. Related work

Most studies of complexity in deduction analyze the length of propositional proofs as part of the $NP \neq co\text{–}NP$ quest (e.g., see [46] for a survey and references), or work with Herbrand complexity and proof length to obtain lower bounds for sets of clauses (e.g., see [35] for a survey and references). The study in [43] analyzes *measures of duplication* in the search spaces generated by sequential theorem-proving strategies (see [21] for further discussion of these lines of research and more references). Our problem is how distributed search affects the complexity of searching for a proof.

The terms "distributed search" and "multi-search" are used in parallel algorithms (e.g., [3,4,10,11]). In algorithms, the search problem in its basic formulation is: given a universe U partitioned into n segments $\sigma_1 \ldots \sigma_n$, and a point q, determine which segment q belongs to. This notion of search problem and the search problem in theorem proving are very different. For instance, the universe U is finite, while the search space of theorem proving is infinite in general. The partition is statically given and it is part of the definition of the problem (e.g., U may be a dictionary divided into sections), whereas in theorem proving there is no partition. A subdivision (usually not a partition) may be built dynamically by a distributed strategy as part of the method to solve the problem in parallel. Furthermore, the partition of U is ordered in the sense that there are elementary operations to decide whether $q \in \sigma_i$, $q \in \bigcup_{j=1}^{i-1} \sigma_j$, $q \in \bigcup_{j=i+1}^{n} \sigma_j$. There is no such structure in the search spaces of theorem proving. The search problem in theorem proving can be considered an instance of the search problem in Artificial Intelligence, with the infinity of the space as a key characteristic.

2. Parallelization by subdivision

2.1. Inference rules and communication operators

Let Θ be a first-order signature, $\mathcal{L}_\Theta$ the language of all the clauses on Θ, and $\mathcal{P}(\mathcal{L}_\Theta)$ its powerset. A *theorem-proving problem* in clausal form consists in deciding whether $S \models \varphi$, where $S \in \mathcal{P}(\mathcal{L}_\Theta)$ is a set of assumptions, and $\varphi \in \mathcal{L}_\Theta$ is the target theorem. Refutationally, the problem is to determine whether $S \cup \{\neg\varphi\}$ is inconsistent. Inconsistency is shown if a derivation

$S \cup \{\neg\varphi\} = S_0 \vdash S_1 \vdash S_2 \vdash \ldots$

generates the empty clause (denoted by $\Box$). S_i represents the state of the derivation after i steps, usually the set (or multiset) of clauses that have been generated and are presently retained by the strategy. Depending on the strategy, S_i may be replaced by a tuple of sets of clauses with various meanings. We use $States$ for the set of states and $States^*$ for sequences of states.

The first component of a *theorem-proving strategy* is the *inference system*. *Expansion inference rules* generate new clauses, while *contraction inference rules*

delete existing clauses and possibly replace them by smaller clauses in a well-founded ordering $\succ$. For example, resolution and paramodulation are expansion rules, while tautology deletion, proper subsumption and simplification via equations are contraction rules. A typical choice for $\succ$ is the multiset extension of a complete simplification ordering [30] on the atoms. The following definition captures both types of rules, by characterizing an inference rule 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 2.1. Given a signature Θ, an inference rule f^n of arity n is a function $f^n: \mathcal{L}_\Theta^n \rightarrow \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$.

An inference rule f^n may not apply to a tuple $\bar{x}$, and in such case $f^n(\bar{x}) = (\emptyset, \emptyset)$. In the following, we may treat sets as multisets, and, given a tuple $\bar{x} = (\varphi_1, \ldots \varphi_n)$, denote by X the multiset $\{\varphi_1, \ldots \varphi_n\}$. Using the projection functions $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$ we define:

Definition 2.2. Given a well-founded ordering on clauses $(\mathcal{L}_\Theta, \succ)$, an inference rule f^n is an *expansion inference rule* if for all premises $\bar{x} \in \mathcal{L}_\Theta^n$, $\pi_2(f^n(\bar{x})) = \emptyset$. It is a *contraction inference rule with respect to* $\succ$ 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 $X - \pi_2(f^n(\bar{x})) \cup \pi_1(f^n(\bar{x})) \prec_{mul} X$, where $\succ_{mul}$ is the multiset extension of $\succ$.

Definition 2.3. An expansion rule is sound if $X \models \pi_1(f^n(\bar{x}))$, a contraction rule is sound if $X \models \pi_1(f^n(\bar{x}))$ and $X - \pi_2(f^n(\bar{x})) \cup \pi_1(f^n(\bar{x})) \models \pi_2(f^n(\bar{x}))$.

Given a set of clauses S and an inference system I, the *closure* of S with respect to I, written S_I^*, contains all clauses derivable from S by I: $S_I^* = \bigcup_{k \geq 0} I^k(S)$, where $I^0(S) = S$, $I^k(S) = I(I^{k-1}(S))$ for $k \geq 1$ and $I(S) = S \cup \{\varphi |\ \varphi \in \pi_1(f(\varphi_1, \ldots \varphi_n)),\ f \in I,\ \varphi_1, \ldots \varphi_n \in S\}$.

Clauses deleted by contraction are *redundant*, in the sense that they are not necessary to prove the theorem (see, e.g., [20,8] for the development of the notion of redundancy and references). A *redundancy criterion* is a mapping R on sets of clauses, such that $R(S)$ is the set of clauses that are redundant with respect to S according to R. The following definition is from [7]:

Definition 2.4. A mapping $R: \mathcal{P}(\mathcal{L}_\Theta) \rightarrow \mathcal{P}(\mathcal{L}_\Theta)$ is a *redundancy criterion* if

1. $S - R(S) \models R(S)$ (soundness),
2. if $S \subseteq S'$, then $R(S) \subseteq R(S')$ (monotonicity) and
3. if $(S' - S) \subseteq R(S')$, then $R(S') \subseteq R(S)$ (irrelevance of redundant clauses to establish redundancies).

A reason for defining redundancy criteria and not only contraction rules is that a redundancy criterion also captures the redundancy of clauses that are not in the existing set, since $R(S)$ does not have to be a subset of S. For instance, a clause that may be generated from a redundant clause in S may also be redundant, so that it is in $R(S)$ but not in S. A strategy can delete redundant clauses by contraction, and prevent the generation of redundant clauses, also by contraction, and by restrictions of expansion. The notion of redundancy can be generalized from clauses to inferences: an inference that uses a redundant clause without deleting it is a *redundant inference*[1].

Definition 2.5. A redundancy criterion R and a set of contraction rules, denoted I_R, *correspond* if, for all sets S:

1. for all $f^n \in I_R$ and $\bar{x} \in S^n$, $\pi_2(f^n(\bar{x})) \subseteq R(X - \pi_2(f^n(\bar{x})) \cup \pi_1(f^n(\bar{x})))$ (whatever is deleted by I_R is redundant according to R), and
2. for all $\varphi \in S \cap R(S - \{\varphi\})$, there exist $f^n \in I_R$ and $\bar{x} \in S^n$, such that $\pi_1(f^n(\bar{x})) = \emptyset$ and $\pi_2(f^n(\bar{x})) = \{\varphi\}$ (if a clause in S is redundant with respect to S, I_R can delete it without adding other clauses (to make it redundant)).

I_R and R are based on the same well-founded ordering $\succ$, and $S_i \vdash_I S_{i+1}$ implies $R(S_i) \subseteq R(S_{i+1})$ by Conditions 2 and 3 in Definition 2.4. In the following, we write $I = I_E \cup I_R$ to distinguish expansion and contraction rules in I.

A distributed strategy also features a system M of *communication operators*. We define the communication operators *send* and *receive* as functions, which take as argument the tuple of clauses being communicated, and return a set of clauses to be added and a set of clauses to be deleted:

Definition 2.6. Given a signature Θ, the communication operators *send* and *receive* are two functions $send\colon \mathcal{L}_\Theta^* \to \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$ and $receive\colon \mathcal{L}_\Theta^* \to \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$, such that for all $\bar{x} \in \mathcal{L}_\Theta^*$, $send(\bar{x}) = (\emptyset, \emptyset)$ and $receive(\bar{x}) = (\bar{x}, \emptyset)$.

This definition captures the effect of *send* and *receive* on the database of the process that executes the operation: $receive(\bar{x}) = (\bar{x}, \emptyset)$, because a *receive* operation adds the received clauses to the database of the receiver, and $send(\bar{x}) = (\emptyset, \emptyset)$, because a *send* operation does not modify the database of the sender. Alternatively, one may define $send(\bar{x}) = (\emptyset, \bar{x})$, which implies that sent data are discarded by the sender. This behavior can be rendered by a send operation as in Definition 2.6 followed by the deletion of $\bar{x}$. In some cases, $send(\bar{x}) = (\emptyset, \emptyset)$ can be simulated by $send(\bar{x}) = (\emptyset, \bar{x})$ followed by $receive(\bar{x}) = (\bar{x}, \emptyset)$, e.g., the case of a broadcast operation where the sender also receives the message. We adopt $send(\bar{x}) = (\emptyset, \emptyset)$, because sending a datum does not imply that the sender deletes it.

[1] Note that a clause generated by a redundant inference may not be redundant, since the same clause may also be generated by a non-redundant inference.

Definitions 2.1 and 2.6 characterize inference rules and communication operators, respectively, in terms of additions and deletions, because we are interested in the effect of the acts of a process on its database. Accordingly, Definition 2.6 abstracts from other information, such as the destination of sent clauses. This can be added by refining the definition of *send* into $send\colon \mathcal{L}_\Theta^* \times \mathcal{P}(\mathbb{N}) \rightarrow \mathcal{P}(\mathcal{L}_\Theta) \times \mathcal{P}(\mathcal{L}_\Theta)$, where $send(\bar{x}, Q)$ means that $\bar{x}$ is sent to the subset Q of processes. (We assume that processes are denoted by natural numbers, e.g., if there are n processes, they are identified by the numbers $0, \ldots n-1$.) This covers sending to one process (Q is singleton), to many (*multicast*) and to all (*broadcast*). The next section uses Definitions 2.1 and 2.6 to build a notion of derivation including both inference and communication steps.

2.2. Distributed-search plans

The other major component of a theorem-proving strategy is the *search plan*, whose task is to choose the inference rule and the premises at each step of a derivation. In distributed search, it also controls the *communication* and the *subdivision of the search space* among the deductive processes.

For the selection of the next inference, the search plan features a *rule-selecting function* and a *premise-selecting function*. The selections are made based on the partial history of the derivation (e.g., $S_0 \vdash_I \ldots S_i$), the number of deductive processes, and the identifier of the process that is executing the selection. Therefore, the domain is $States^* \times \mathbb{N} \times \mathbb{N}$. The rule-selecting function is a function $\zeta\colon States^* \times \mathbb{N} \times \mathbb{N} \rightarrow I \cup M$, where I is the set of inference rules, and M is the set of communication operators of the strategy. The premise-selecting function is a function $\xi\colon States^* \times \mathbb{N} \times \mathbb{N} \times (I \cup M) \rightarrow \mathcal{L}_\Theta^*$, which also takes as argument the inference rule or operator chosen by ζ. If ζ selects an inference rule f and ξ selects a tuple of premises $\bar{x}$, the inference step consists of applying f to $\bar{x}$.

For communication, at each step of a derivation the search plan may select a communication operator, rather than an inference rule. If ζ selects *send*, the clauses chosen by ξ will be sent. If ζ selects *receive*, whichever clauses are pending to be received will be received. There is no selection of clauses, because every process receives whatever it is sent.

A *subdivision of the search space* is a subdivision of the inferences in the closure S_I^*. Since S_I^* is infinite and unknown, at each stage S_i of a derivation the search plan subdivides the inferences that can be done in S_i. Thus, the subdivision is built *dynamically* during the derivation. From the point of view of each process p_k, an inference is either *allowed* (it is assigned to p_k), or *forbidden* (it is assigned to others). Therefore, we model the subdivision of the search space by distinguishing between allowed and forbidden steps. For this purpose, the search plan includes a *subdivision function* $\alpha\colon States^* \times \mathbb{N} \times \mathbb{N} \times I \times \mathcal{L}_\Theta^* \rightarrow Bool$, whose arguments also include the tuple of premises chosen by ξ. A subdivision function is partial in general: we use $\perp$ to denote "undefined". Where defined, α

returns *true* if the process is allowed to perform the step, and $false$ otherwise. The ability of detecting termination completes the definition of search plan:

Definition 2.7. A *distributed-search plan* Σ for a set I of inference rules and a set M of communication operators is a 4-tuple $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, where

1. $\zeta\colon States^* \times \mathbb{N} \times \mathbb{N} \rightarrow I \cup M$ is the *rule-selecting function*,
2. $\xi\colon States^* \times \mathbb{N} \times \mathbb{N} \times (I \cup M) \rightarrow \mathcal{L}^*_\Theta$ is the *premise-selecting function*: $\xi((S_0, \ldots S_i), n, k, f^m) \in S_i^m$ (i.e., it selects clauses in the current state),
3. $\alpha\colon States^* \times \mathbb{N} \times \mathbb{N} \times I \times \mathcal{L}^*_\Theta \rightarrow Bool$ is the *subdivision function*, and
4. $\omega\colon States \rightarrow Bool$ is the *termination-detecting function*, such that $\omega(S) = true$ if and only if S contains the empty clause.

This definition generalizes that of sequential search plan in [21]: a sequential search plan only has the components ζ, ξ and ω, with $\zeta\colon States^* \rightarrow I$ and $\xi\colon States^* \times I \rightarrow \mathcal{L}^*_\Theta$.

We have all the elements to define a *distributed strategy* as a triple $\langle I, M, \Sigma \rangle$, with inference system I, communication operators M, and distributed-search plan Σ. The execution of a distributed strategy by n processes $p_0, \ldots p_{n-1}$ yields a *distributed derivation*. All processes have the same inference system and search plan, and their activities are differentiated by the subdivision function, which allows every process to execute certain steps and forbids others:

Definition 2.8. Given a theorem-proving problem S, the *distributed derivation* generated by strategy $\mathcal{C} = \langle I, M, \Sigma \rangle$, with $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, for processes $p_0, \ldots p_{n-1}$ $(n > 0)$, is a collection of n *local derivations* $S = S_0^k \vdash_\mathcal{C} S_1^k \vdash_\mathcal{C} \ldots S_i^k \vdash_\mathcal{C} \ldots$, for $k \in [0, n-1]$, such that for all k and $i \geq 0$, if

- $\omega(S_i^k) = false$,
- $\zeta((S_0^k, \ldots S_i^k), n, k) = f$,
- either $f = receive$ and there is a tuple $\bar{x}$ pending to be received, or $f \neq receive$ and $\xi((S_0^k, \ldots S_i^k), n, k, f) = \bar{x}$, and
- either $f \in M$ or $\alpha((S_0^k, \ldots S_i^k), n, k, f, \bar{x}) = true$,

then $S_{i+1}^k = S_i^k \cup \pi_1(f(\bar{x})) - \pi_2(f(\bar{x}))$.

Ideally, all local derivations start in parallel; in practice, there is typically a process that starts first and initiates all the others. Subsequently, the local derivations are *asynchronous* in general: any two processes p_j and p_k are not expected to be in stage i simultaneously, and if p_k sends a clause φ to p_j at stage i of p_k's derivation, there is no assumption on which stage of p_j's derivation φ is received. All decisions are made locally, with no central control: the functions ω, ζ, ξ, and α apply to the partial history of the local derivations.

Definition 2.9. A distributed-search plan $\Sigma' = \langle \zeta', \xi', \alpha, \omega \rangle$ is a *parallelization by subdivision* of a sequential search plan $\Sigma = \langle \zeta, \xi, \omega \rangle$ for inference system I, if for all $(S_0, \ldots S_i) \in States^*$, n and k,

1. if $\zeta'((S_0, \ldots S_i), n, k) \in I$, then $\zeta'((S_0, \ldots S_i), n, k) = \zeta(S_0, \ldots S_i)$ (whenever ζ' selects an inference rule, it selects the rule that ζ would select if given the same partial derivation), and
2. $\forall f \in I$, $\xi'((S_0, \ldots S_i), n, k, f) = \xi((S_0, \ldots S_i), f)$ (ξ' selects the premises that ξ would select if given the same partial derivation and inference rule).

A distributed strategy $\mathcal{C}' = \langle I, M, \Sigma' \rangle$ is a *parallelization by subdivision* of a sequential strategy $\mathcal{C} = \langle I, \Sigma \rangle$, if Σ' is a parallelization by subdivision of Σ. The local derivations controlled by Σ' differ from the sequential derivation controlled by Σ, because of the subdivision function, which for every process forbids some choices, forcing the process to choose something else, and the communication steps. Since subdivision is one of the reasons for communication, the subdivision is the main principle that differentiates the derivations.

3. Properties of distributed-search plans

3.1. Monotonicity of the subdivision

If ζ and ξ can select a certain f and $\bar{x}$, α needs to be defined on their selection. For this purpose it would be sufficient to require that α is total on existing clauses. However, we do not want the subdivision function to "forget" its decision when a clause is deleted by contraction. Thus, we require that α is total on generated clauses:

Definition 3.1. A subdivision function α is *total on generated clauses* if for all $S_0 \vdash \ldots S_i \vdash \ldots$, k, n, f^m, $\bar{x}$, if $\bar{x} \in (\bigcup_{j=0}^{i} S_j)^m$ then $\alpha((S_0 \ldots S_i), n, k, f^m, \bar{x}) \neq \perp$.

Next, since it is undesirable that the subdivision function changes the status of a step after it has been decided, we require that it is *monotonic* with respect to the partial ordering $\perp \sqsubset false$ and $\perp \sqsubset true$ on its range $Bool$:

Definition 3.2. A subdivision function α is *monotonic* if for all $S_0 \vdash \ldots S_i \vdash \ldots$, n, k, f, $\bar{x}$, $i \geq 0$, $\alpha((S_0 \ldots S_i), n, k, f, \bar{x}) \sqsubseteq \alpha((S_0 \ldots S_{i+1}), n, k, f, \bar{x})$.

In the rest of the paper we assume that the subdivision functions are total on generated clauses and monotonic.

Example 3.1. In Clause-Diffusion [18,12] the search space is subdivided by assigning clauses to processes and allowing inferences accordingly (e.g., a process

may paramodulate into a clause only if it owns it). Such a mechanism corresponds to a partial subdivision function, because it is not known whether a step is allowed until after its premises have been generated and assigned. It is a total function on generated clauses, because clauses are assigned right after generation and forward contraction. (Raw clauses – clauses before forward contraction – cannot be selected as premises, so that the subdivision function does not need to be defined on them. When describing the state of a derivation more in detail, we use a separate component for raw clauses, so that they are not in S_i.) On the other hand, the subdivision functions of Clause-Diffusion are not monotonic. The cause is subsumption of variants. Consider a process p_k which generates a clause ψ, and receives a variant ψ' which subsumes ψ. If ψ belongs to p_k and ψ' does not, or vice versa, the assignment of the clause in the database of p_k changes. However, it cannot change indefinitely. Since variants are equivalent in the subsumption ordering, variant subsumption in Clause-Diffusion is controlled by a well-founded ordering which combines lexicographically the subsumption ordering and some other well-founded ordering on attributes of the clauses (e.g., their identifiers) [16]. Since this ordering is well-founded, there cannot be an infinite sequence of subsumptions of variants. Thus, the assignment of a clause stabilizes eventually in the database of all processes. It follows that Clause-Diffusion satisfies a weaker form of monotonicity: if $\alpha((S_0, \ldots S_i), n, k, f, \bar{x})$ becomes defined at some stage i, it remains defined at all following stages, and there is a stage $j \geq i$ beyond which the decision does not change, i.e., $\forall r > j$, $\alpha((S_0, \ldots S_r), n, k, f, \bar{x}) = \alpha((S_0, \ldots S_j), n, k, f, \bar{x})$.

3.2. Fairness

A derivation is *successful* if $\omega(S_i^k) = true$ for some p_k and i; a strategy is *complete* if it succeeds whenever the input is inconsistent; completeness is made of *refutational completeness* of the inference system (there exist successful derivations whenever the input is inconsistent), and *fairness* of the search plan (the search plan drives the strategy to generate a successful derivation whenever successful derivations exist). A sufficient condition for fairness is *uniform fairness*: whatever can be inferred from persistent non-redundant premises is generated eventually. This notion of fairness is widely used for forward-reasoning theorem proving. We report here its formulation in [7] and refer to [20] for more references and a discussion of this concept of fairness:

Definition 3.3. Given an inference system I and a redundancy criterion R, a derivation $S_0 \vdash \ldots S_i \vdash \ldots$ is *uniformly fair* with respect to I and R if $I(S_\infty - R(S_\infty)) \subseteq \bigcup_{j \geq 0} S_j$, where $S_\infty = \bigcup_{j \geq 0} \bigcap_{i \geq j} S_i$ is the set of persistent clauses.

A search plan Σ is *uniformly fair* w. r. t. I and R if all its derivations are[2].

In a distributed derivation, $S^k_\infty = \bigcup_{j\geq 0}\bigcap_{i\geq j} S^k_i$ is the set of persistent clauses at p_k, and $S_\infty = \bigcup_{k=0}^{n-1} S^k_\infty$. A process only needs to be fair with respect to what is allowed to do by the subdivision function α. Since α is monotonic, being allowed (i.e., $\exists j \geq 0$ such that $\alpha((S^k_0, \ldots S^k_j), n, k, f^m, \bar{x}) = true$) is equivalent to being persistently allowed after a certain stage (i.e., $\exists i \geq 0$ such that $\forall j \geq i$, $\alpha((S^k_0, \ldots S^k_j), n, k, f^m, \bar{x}) = true$).

Definition 3.4. A local derivation $S^k_0 \vdash S^k_1 \vdash \ldots S^k_i \vdash \ldots$ is *uniformly fair* with respect to I, R and α, if $\varphi \in I(S^k_\infty - R(S^k_\infty))$, and $\exists i \geq 0$, such that $\forall j \geq i$, $\alpha((S^k_0, \ldots S^k_j), n, k, f^m, \bar{x}) = true$, where $\varphi \in \pi_1(f^m(\bar{x}))$, imply $\varphi \in \bigcup_{j\geq 0} S^k_j$.

For fairness of the distributed derivation, the search plan needs to ensure that for every inference from a tuple of persistent non-redundant premises, there is a process which is allowed to do it (*fairness of subdivision*), and has all its premises (*fairness of communication*):

Definition 3.5. A distributed derivation $S^k_0 \vdash_{\mathcal{C}} S^k_1 \vdash_{\mathcal{C}} \ldots S^k_i \vdash_{\mathcal{C}} \ldots$, for $k \in [0, n-1]$, controlled by $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, is *uniformly fair* with respect to I and R, if:

1. $\forall f^m \in I$, $\forall \bar{x} \in (S_\infty - R(S_\infty))^m$, such that $\pi_1(f^m(\bar{x})) \neq \emptyset$, $\exists\, p_k$ such that
   (a) $\bar{x} \in (S^k_\infty - R(S_\infty))^m$ (*fairness of communication*), and
   (b) $\exists i \geq 0\ \forall j \geq i$, $\alpha((S^k_0 \ldots S^k_j), n, k, f^m, \bar{x}) = true$ *(fairness of subdivision).*
2. $\forall p_k$, $S^k_0 \vdash_{\mathcal{C}} \ldots S^k_i \vdash_{\mathcal{C}} \ldots$ is uniformly fair w. r. t. I, R and α *(local fairness).*

Definition 3.5 is adequate in the sense that a distributed derivation which is uniformly fair according to Definition 3.5 satisfies Definition 3.3:

Theorem 3.1. If a distributed derivation $S^k_0 \vdash_{\mathcal{C}} S^k_1 \vdash_{\mathcal{C}} \ldots S^k_i \vdash_{\mathcal{C}} \ldots$ (for $k \in [0, n-1]$) is uniformly fair with respect to I and R, then $I(S_\infty - R(S_\infty)) \subseteq \bigcup_{k=0}^{n-1}\bigcup_{j\geq 0} S^k_j$.

Proof: for all $\varphi \in I(S_\infty - R(S_\infty))$, φ is generated by an inference rule $f^m \in I$ applied to a tuple of persistent premises $\bar{x} \in (S_\infty - R(S_\infty))^m$. By fairness of communication, there exists a process p_k such that $\bar{x} \in (S^k_\infty - R(S_\infty))^m$. Since $S^k_\infty \subseteq S_\infty$, $\psi \notin R(S_\infty)$ implies $\psi \notin R(S^k_\infty)$ by monotonicity of R. Thus, $\bar{x} \in (S^k_\infty - R(S^k_\infty))^m$. By fairness of the subdivision function, $\alpha((S^k_0, \ldots S^k_j), n, k, f^m, \bar{x}) = true$ for all $j \geq i$ for some $i \geq 0$. By local fairness, the derivation by p_k is uniformly fair. Since $\varphi \in I(S^k_\infty - R(S^k_\infty))$, and the step generating φ is allowed, $\varphi \in \bigcup_{j\geq 0} S^k_j$. Therefore, $\varphi \in \bigcup_{k=0}^{n-1}\bigcup_{j\geq 0} S^k_j$. □

[2] For all properties defined for derivations, we say that a search plan has the property if all its derivations do.

Then, Definition 3.5 gives the sufficient conditions for the parallelization by subdivision of a fair search plan to be fair:

Theorem 3.2. Let Σ be a uniformly fair search plan with respect to I and R, and Σ' a parallelization by subdivision of Σ. If Σ' has fairness of communication and fairness of subdivision, Σ' is uniformly fair with respect to I and R.

Proof: Σ' inherits local fairness from the fact that Σ is uniformly fair, so that it satisfies all the conditions of Definition 3.5. □

A sufficient condition for fairness of communication is *propagation of clauses up to redundancy*:

Definition 3.6. A distributed derivation has *propagation of clauses up to redundancy* if for all $\varphi \in S_\infty - R(S_\infty)$, for all p_k, there exists a j such that $\varphi \in S_j^k$.

In turn, sufficient conditions for propagation of clauses up to redundancy are that for all $\varphi \in S_\infty - R(S_\infty)$, there are p_k and i such that p_k broadcasts φ at stage i, and all sent messages are received eventually[3]. Contraction affects neither fairness of communication nor propagation of clauses up to redundancy, because both requirements apply only to persistent non-redundant clauses. The asynchronous nature of the derivation does not affect them either, because it is sufficient that sent messages are received eventually. In practice, a strategy does not know which clauses are persistent, and therefore satisfies Definition 3.6 by propagating more clauses. For example, Clause-Diffusion strategies [18] let each process broadcast the clauses it owns, and Modified Clause-Diffusion strategies [12] let each process broadcast the non-trivial normal forms of the clauses it generates (i.e., clauses are broadcast after generation and forward contraction).

3.3. Eager contraction

Let $\mathcal{C} =< I_E \cup I_R, \Sigma >$ be a strategy such that I_R contains only contraction rules that delete clauses without generating any, Σ is uniformly fair with respect to I_E and R, and $\mathcal{C}$ is complete. If I'_R is a set of contraction rules that replace clauses redundant according to R by smaller clauses, the strategy $\mathcal{C}' =< I_E \cup I_R \cup I'_R, \Sigma >$ is also complete [7]. This means that uniform fairness with respect to I_E and R is sufficient for completeness. This result of [7] assumes implicitly that for all input sets S, $S_I^* = S_{I_E}^*$, that is, everything that can be generated can be generated by expansion. This assumption is traditionally left implicit, because the known inference systems satisfy it. We make this assumption in the following, together with uniform fairness with respect to I_E and R. However, it is very well-known from both theoretical studies (e.g., [30,7,20,17,8,21])

[3] This is the characterization of fairness of communication given in [12] for specific strategies.

and experiments with theorem provers (e.g., [2,32,1,37,47,38–40,45]), that contraction is crucial for the practical effectiveness of forward reasoning, and not only applicable contractions should be done, but it is better to do them eagerly, in order to prevent redundant clauses from generating other redundant clauses. Eager-contraction can be defined as follows:

Definition 3.7. A derivation $S_0 \vdash_I S_1 \vdash_I \ldots S_i \vdash_I \ldots$ 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 there exists an $l \geq i$ such that $S_l \vdash S_{l+1}$ deletes φ, and $\forall j, i \leq j \leq l$, $S_j \vdash S_{j+1}$ is not an expansion inference, unless the derivation succeeds sooner.

A sequential strategy is *contraction-based*, if its inference system includes contraction rules and its search plan has eager contraction. The parallelization of eager contraction is a difficult problem. First, each local derivation needs to have eager contraction:

Definition 3.8. A distributed derivation has *local eager contraction*, if for all p_k, $i \geq 0$ and $\varphi \in S_i^k$, if there are $f^m \in I_R$ and $\bar{x} \in (S_i^k)^m$, such that $\pi_2(f^m(\bar{x})) = \{\varphi\}$, then there exists an $l \geq i$ such that $S_l^k \vdash S_{l+1}^k$ deletes φ, and $\forall j, i \leq j \leq l$, $S_j^k \vdash S_{j+1}^k$ is neither an expansion nor a *send* step, unless p_k halts sooner.

Note that the definition has "p_k halts sooner" instead of "succeeds" as in the sequential case, because p_k halts if it succeeds or another process succeeds. A local property, however, is not sufficient; since in a distributed derivation multiple processes generate clauses independently, we need to consider the behavior of the strategy with respect to the situation where a clause φ at p_k can be contracted by premises generated elsewhere:

Definition 3.9. A distributed 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 there exists an $l \geq i$ such that $S_l^k \vdash S_{l+1}^k$ deletes φ, 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 *send* step.

Global eager contraction is the generalization of eager contraction to distributed derivations, while distributed global contraction[4] is a weaker requirement which guarantees contraction, but not priority over expansion. Since processes have separate databases, contraction with respect to $\bigcup_{h=0}^{n-1} S_i^h$ rests on communication. The following lemmas show how to combine local eager contraction with propagation of clauses up to redundancy to obtain global contraction properties.

[4] The name distributed global contraction was used in [18] to indicate techniques to keep a distributed database inter-reduced without centralizing contraction.

First, propagation of clauses alone implies that all processes know eventually whether a clause is redundant:

Definition 3.10. A distributed derivation has *propagation of redundancy*, if $\varphi \in R(\bigcup_{h=0}^{n-1} S_i^h)$ implies that for all p_h there exists a j such that $\varphi \in R(S_j^h)$.

Lemma 3.1. Propagation of clauses up to redundancy implies propagation of redundancy.

Proof: assume that $\varphi \in R(\bigcup_{h=0}^{n-1} S_i^h)$. By properties 2 and 3 in Definition 2.4, $\varphi \in R(S_\infty - R(S_\infty))$. Let $\{\psi_1, \ldots \psi_m\}$ be a minimal multiset of clauses in $S_\infty - R(S_\infty)$ such that $\varphi \in R(\{\psi_1, \ldots \psi_m\})$. By propagation of clauses up to redundancy, for all p_h, for all ψ_k, $1 \leq k \leq m$, there exists a j_k such that $\psi_k \in S_{j_k}^h$. Let $j = max\{j_k | 1 \leq k \leq m\}$. Since $\psi_1, \ldots \psi_m$ are persistent, $\psi_1, \ldots \psi_m \in S_j^h$ and $\varphi \in R(S_j^h)$. □

With local eager contraction the redundant clauses are indeed contracted:

Lemma 3.2. Local eager contraction and propagation of clauses up to redundancy imply distributed global contraction.

Proof: assume that for some $\varphi \in S_i^k$, $f^m \in I_R$ and $\bar{x} \in (\bigcup_{h=0}^{n-1} S_i^h)^m$, it is $\pi_2(f^m(\bar{x})) = \{\varphi\}$. We need to show that p_k deletes φ unless it halts sooner. Let $M = X - \pi_2(f^m(\bar{x})) \cup \pi_1(f^m(\bar{x}))$. By Definition 2.5, $\varphi \in R(M)$. Let $X - \pi_2(f^m(\bar{x})) = \{\psi_1, \ldots \psi_r\}$. There are two cases.

1. If for all ψ_q, $1 \leq q \leq r$, $\psi_q \in S_\infty - R(S_\infty)$, by propagation of clauses up to redundancy, for all ψ_q, there exists a j_q such that $\psi_q \in S_{j_q}^k$. Let $j = max\{j_q | 1 \leq q \leq r\}$. Since $\psi_1, \ldots \psi_r$ are persistent, $\psi_1, \ldots \psi_r \in S_j^k$. By local eager contraction, there exists an $l \geq j$ such that $S_l^k \vdash S_{l+1}^k$ deletes φ unless p_k halts sooner.
2. If for some ψ_q, $1 \leq q \leq r$, $\psi_q \notin S_\infty - R(S_\infty)$, this specific contraction step may not be done, but we show that p_k still deletes φ unless it halts sooner. If $\pi_1(f^m(\bar{x})) = \emptyset$, $\varphi \in R(M)$ means $\varphi \in R(\{\psi_1, \ldots \psi_r\})$, hence $\varphi \in R(\bigcup_{h=0}^{n-1} S_i^h)$ and $\varphi \in R(S_\infty)$ by the properties of R (Definition 2.4). If $\pi_1(f^m(\bar{x})) \neq \emptyset$, since everything that can be generated by contraction can be generated by expansion, by uniform fairness with respect to I_E and R, and again the properties of R, it is $\varphi \in R(S_\infty)$. By Definition 2.5, there are $f^m \in I_R$ and $\bar{y} = (\psi'_1, \ldots \psi'_m)$, $\psi'_1, \ldots \psi'_m \in S_\infty - R(S_\infty)$ such that $\pi_2(f^m(\bar{y})) = \{\varphi\}$ and $\pi_1(f^m(\bar{y})) = \emptyset$. Like in Case 1, by propagation of clauses up to redundancy and local eager contraction, p_k deletes φ unless it halts sooner. □

By local eager contraction, p_k performs no expansion or communication between stage j (when φ becomes reducible with respect to the local database of p_k) and

stage l (when p_k deletes φ). That is, p_k deletes φ eagerly with respect to the local database. However, p_k may perform expansion or communication, possibly using φ as premise, in the interval between stage i (when φ is reducible with respect to the global database) and stage j. Thus, local eager contraction and propagation of clauses up to redundancy imply distributed global contraction, not global eager contraction, which requires a stronger property:

Definition 3.11. Assume $\varphi \in S_\infty - R(S_\infty)$ and let i be the earliest stage such that $\varphi \in \bigcup_{k=0}^{n-1} S_i^k$. A distributed derivation has *immediate propagation of clauses up to redundancy* if $\forall p_k \ \varphi \in S_i^k$.

Lemma 3.3. Local eager contraction and immediate propagation of clauses up to redundancy imply global eager contraction.

Proof: the proof is the same as for Lemma 3.2 with the following modification of Case 1: if for all ψ_q, $1 \leq q \leq r$, $\psi_q \in S_\infty - R(S_\infty)$, then, by immediate propagation of clauses up to redundancy, $\psi_1, \ldots \psi_r \in S_i^k$. By local eager contraction there exists an $l \geq i$ such that $S_l^k \vdash S_{l+1}^k$ deletes φ, and $\forall j, i \leq j \leq l$, $S_j^k \vdash S_{j+1}^k$ is neither an expansion nor a *send* step, unless p_k halts sooner. □

Distributed strategies with asynchronous communication cannot have immediate propagation. Therefore, we say that a distributed strategy is *contraction-based*, if its inference system includes contraction rules, and its search plan has local eager contraction and distributed global contraction. We use immediate propagation as a limit property to study problems and express results.

In order to establish sufficient conditions for the parallelization by subdivision of a contraction-based strategy to be contraction-based, we consider next the compatibility of subdivision and eager contraction:

Theorem 3.3. Let $\mathcal{C} = \langle I, \Sigma \rangle$ be a contraction-based strategy, and $\mathcal{C}' = \langle I, M, \Sigma' \rangle$, with $\Sigma' = \langle \zeta', \xi', \alpha, \omega' \rangle$, a parallelization by subdivision of $\mathcal{C}$. If Σ' propagates clauses up to redundancy and for all $f \in I_R$, $\alpha((S_0^k \ldots S_i^k), n, k, f, \bar{x}) = \mathit{true}$, for all i, n, k and $\bar{x}$, then $\mathcal{C}'$ is a distributed contraction-based strategy.

Proof: $\mathcal{C}'$ has contraction rules, because it has the inference system of $\mathcal{C}$. Since Σ' is a parallelization by subdivision of Σ, Σ' selects the same inferences as Σ. Since contraction inferences are allowed to all processes, the subdivision has no effect on contraction, and local eager contraction follows from the hypothesis that Σ is eager-contraction. Distributed global contraction follows from local eager contraction and propagation of clauses up to redundancy by Lemma 3.2. □

The requirement that all contraction inferences are allowed to all processes is a strong one, especially in equational problems, where contraction, e.g.,

simplification, is not only deletion but also deduction of new equations. Assume that the subdivision function α may forbid contraction inferences, e.g., $f(\bar{x}) = (\{\psi_1, \ldots \psi_m\}, \{\varphi_1, \ldots \varphi_p\})$. Consider a strategy that lets a process delete the φ_j, but not generate the ψ_i, when ζ selects f, ξ selects $\bar{x}$, and α forbids the step. It is sufficient to guarantee that at least one process generates and propagates the ψ_i to preserve completeness. The proof of Theorem 3.3 applies also to such a strategy; let $\mathcal{C}$ and $\mathcal{C}'$ be as in Theorem 3.3:

Theorem 3.4. If Σ' propagates clauses up to redundancy, and whenever $\zeta((S_0^k \ldots S_i^k), n, k) = f \in I_R$, $\xi((S_0^k \ldots S_i^k), n, k, f) = \bar{x}$, $\alpha((S_0^k \ldots S_i^k), n, k, f, \bar{x}) = false$, $S_{i+1}^k = S_i^k - \pi_2(f(\bar{x}))$, then $\mathcal{C}'$ is a distributed contraction-based strategy.

Clause-Diffusion strategies [18] are contraction-based by Theorem 3.3, because they propagate clauses up to redundancy and allow all contractions to all processes. Modified Clause-Diffusion strategies [12] are contraction-based by Theorem 3.4: they propagate clauses up to redundancy; forward contractions and contractions that do not generate clauses are allowed to all processes; backward simplifications may be forbidden, but when a process selects a forbidden backward-simplification step, it deletes the reducible equation without generating its normal form.

4. Search graphs for distributed search

4.1. The structure of the search space

The structure of the search space of a theorem-proving problem depends on the problem and the inference system, so that it is the same regardless of whether the space is searched in parallel or sequentially. Therefore, the representation of the structure of the search space is the same as in the sequential case [21].

Given a theorem-proving problem S on signature Θ and an inference system I, the search space is modelled as a *search graph*, more precisely a *hypergraph*. The vertices represent the clauses in the closure S_I^*, and the hyperarcs represent the inferences. Given a hypergraph (V, E), an *arc-labelling function* is a function $h\colon E \to I$ from hyperarcs to inference rules. A *vertex-labelling function* is an injective function $l\colon V \to \mathcal{L}_\Theta/\stackrel{\bullet}{=}$ from vertices to equivalence classes of clauses, where $\stackrel{\bullet}{=}$ is equivalence up to variable renaming. Thus, all variants of a clause are associated to a unique vertex. For simplicity, we refer to $l(v)$ as a clause, meaning a representative of a class of variants.

Definition 4.1. Given a theorem-proving problem S on signature Θ and a set of inference rules I, the *search space* induced by S and I is represented by the *search graph* $G(S_I^*) = (V, E, l, h)$, where V is the set of vertices, l is a vertex-labelling function $l\colon V \to \mathcal{L}_\Theta/\stackrel{\bullet}{=}$, h is an arc-labelling function $h\colon E \to I$, and

if $f^n(\varphi_1, \ldots \varphi_n) = (\{\psi_1, \ldots \psi_m\}, \{\gamma_1, \ldots \gamma_p\})$ for $f^n \in I$, E contains a hyperarc $e = (v_1, \ldots v_k; w_1, \ldots w_p; u_1, \ldots u_m)$, where $h(e) = f^n$, and

- $v_1 \ldots v_k$ are labelled by the premises that are not deleted: $l(v_j) = \varphi_j$ and $\varphi_j \notin \{\gamma_1, \ldots \gamma_p\}$, $\forall j$, $1 \leq j \leq k$, where $k = n - p$,
- $w_1 \ldots w_p$ are labelled by the deleted clauses: $l(w_j) = \gamma_j$, $\forall j$, $1 \leq j \leq p$, and
- $u_1 \ldots u_m$ are labelled by the generated clauses: $l(u_j) = \psi_j$, $\forall j$, $1 \leq j \leq m$.

In the sequel, we use interchangeably vertices and their labels. Without loss of generality, we consider hyperarcs in the form $(v_1, \ldots v_n; w; u)$, where at most one clause is added or deleted. For instance, a *resolution* arc has the form $(v_1, v_2; u)$, where u is a resolvent of v_1 and v_2; a *simplification* arc has the form $(v; w; u)$, where v reduces w to u; and a *normalization* arc has the form $(v_1, \ldots v_n; w; u)$, where u is a normal form of w with respect to the simplifiers $v_1, \ldots v_n$. Contraction inferences that purely delete clauses are represented as replacement by *true*, where *true* is a dummy clause such that $true \prec \varphi$ for all φ. A special vertex T in the search graph is labelled by *true*. We refer to [21] for the application of this representation to more inference rules and several examples of inference steps.

4.2. Evolution of the search graph during a distributed derivation

The search graph $G(S_I^*) = (V, E, l, h)$ represents the static structure of the search space. The representation of the search process during a distributed derivation needs to cover: (1) the selections by the search plan, (2) the deletions by contraction, (3) the subdivision of the search space, (4) the effect of communication. These four aspects are intertwined, because the search plan controls contraction, subdivision and communication, and these in turn affect the successive choices of the search plan.

The dynamics of the search during a derivation is described by *marking functions* for vertices and arcs. By using multiple marking functions we can represent the effects of all the processes on the same search graph, yielding a *parallel marked search-graph*:

Definition 4.2. A *parallel marked search-graph* $(V, E, l, h, \bar{s}, \bar{c})$ for processes $p_0, \ldots p_{n-1}$ is given by a search graph (V, E, l, h), and:

- An n-tuple $\bar{s}$ of *vertex-marking functions*, where each component $s^k \colon V \to Z$ (from vertices to integers) is defined as follows:

$$s^k(v) = \begin{cases} m & \text{if } m \text{ variants } (m > 0) \text{ of } l(v) \text{ are present for } p_k, \\ -1 & \text{if } p_k \text{ deleted all variants of } l(v) \text{ it generated or received,} \\ 0 & \text{otherwise.} \end{cases}$$

- An n-tuple $\bar{c}$ of *arc-marking functions*, where $c^k: E \rightarrow \mathbb{N} \times Bool$, and for $e = (v_1, \ldots v_n; w; u)$, $\pi_1(c^k(e))$ is the number of times p_k executed arc e or received a variant of $l(u)$ generated by e, and $\pi_2(c^k(e)) = true/false$, if p_k is allowed/forbidden to execute e.

If we want to concentrate on process p_k we consider the *marked search graph* (V, E, l, h, s^k, c^k). In addition to the generation of clauses, the vertex-marking function captures the dynamic effects of contraction (if a clause is deleted, its marking becomes negative) and communication (if a clause is received, its marking is incremented). The first component of the arc-marking will be used to keep track of relevance of ancestors, as shown in Section 5.1. The second component (*permission marking*) represents the subdivision of the space.

Let S be a theorem-proving problem, $\mathcal{C} = \langle I, M, \Sigma \rangle$ a distributed strategy with $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, $S = S_0^k \vdash_{\mathcal{C}} \ldots S_i^k \vdash_{\mathcal{C}} \ldots$ for $k \in [0, n-1]$ the distributed derivation generated by $\mathcal{C}$ from S, and $G(S_I^*) = (V, E, l, h)$ the underlying search graph. At each stage of a derivation a process can execute a hyperarc only if the hyperarc is *enabled*, that is, its premises are present and it is allowed:

Definition 4.3. A hyperarc $e = (v_1, \ldots v_n; w; u) \in E$ is *enabled* for process p_k at stage i, if $s_i^k(v_j) > 0$ for $1 \leq j \leq n$, $s_i^k(w) > 0$ ($s_i^k(w) > 1$ if $w \in \{v_1, \ldots v_n\}$, e.g., for a variant subsumption arc (v, v, T)), and $\pi_2(c_i^k(e)) = true$.

The following definitions show how to associate marking functions to the stages of a derivation.

Definition 4.4. A distributed derivation induces n successions (one per process) of vertex-marking functions $\{s_i^k\}_{i \geq 0}$ as follows. 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 $e = (v_1, \ldots v_n; w; u)$:

$$s_{i+1}^k(v) = \begin{cases} s_i^k(v) - 1 & \text{if } v = w \text{ and } s_i^k(v) > 1, \\ -1 & \text{if } v = w \text{ and } s_i^k(v) = 1, \\ s_i^k(v) + 1 & \text{if } v = u \text{ and } s_i^k(v) \geq 0, \\ 1 & \text{if } v = u \text{ and } s_i^k(v) = -1, \\ s_i^k(v) & \text{otherwise.} \end{cases}$$

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

$$s_{i+1}^k(v) = \begin{cases} s_i^k(v) + 1 & \text{if } v \in \{v_1, \ldots v_n\} \text{ and } s_i^k(v) \geq 0, \\ 1 & \text{if } v \in \{v_1, \ldots v_n\} \text{ and } s_i^k(v) = -1, \\ s_i^k(v) & \text{otherwise.} \end{cases}$$

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

Inference steps performed by p_k affect only the marking s^k, reflecting the fact that the databases of the processes are separate. Note that $s^k_0(v) = 0$ for all vertices, including input clauses, because reading or receiving the input clauses already modifies the search space, since the subdivision function applies also to input clauses. Therefore, also the steps of reading or receiving the input clauses need to be included in the derivation. (Read steps can be modelled as expansion steps, so that input clauses are treated like generated clauses.)

Definition 4.5. A distributed derivation induces n successions (one per process) of arc-marking functions $\{c^k_i\}_{i\geq 0}$ as follows. For all $a \in E$, $\pi_1(c^k_0(a)) = 0$ and $\pi_2(c^k_0(a)) = true$, and $\forall i \geq 0$:

$$\pi_1(c^k_{i+1}(a)) = \begin{cases} \pi_1(c^k_i(a)) + 1 \text{ if at stage } i \ p_k \text{ executes } a, \\ \pi_1(c^k_i(a)) + 1 \text{ if at stage } i \ p_k \text{ receives a clause generated by } a, \\ \pi_1(c^k_i(a)) \qquad \text{otherwise;} \end{cases}$$

$$\pi_2(c^k_{i+1}(a)) = \begin{cases} \alpha((S^k_0, \ldots S^k_{i+1}), n, k, f, \bar{x}) \text{ if } \alpha((S^k_0, \ldots S^k_{i+1}), n, k, f, \bar{x}) \neq \perp, \\ true \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise,} \end{cases}$$

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

The condition $\pi_2(c^k_0(a)) = true$ means that all arcs are allowed initially. As the derivation progresses, including the steps to read or receive the input clauses, the strategy dynamically applies the subdivision function to exclude certain inferences for each process. The permission marking inherits the properties of the subdivision function: if α is total on generated clauses, the marking of an arc is set to either $true$ or $false$ when its premises are generated, and if α is monotonic, it does not change afterwards.

The *active search space* is the part of the graph determined by the vertices with positive marking:

Definition 4.6. Given a parallel marked search-graph $G = (V, E, l, h, \bar{s}, \bar{c})$, the *active search space for process* p_k is $G^{k+} = (V^{k+}, E^{k+}, l, h, s^k, c^k)$, where $V^{k+} = \{v \mid v \in V,\ s^k(v) > 0\}$ and E^{k+} is the restriction of E to V^{k+}. The *active search space* is $G^+ = (V^+, E^+, l, h, \bar{s}, \bar{c})$, where $V^+ = \{v \mid v \in V,\ \exists k\ s^k(v) > 0\}$ and E^+ is the restriction of E to V^+.

The *generated search space* is represented by the portion of the search graph with non-zero marking:

Definition 4.7. Given a parallel marked search-graph $G = (V, E, l, h, \bar{s}, \bar{c})$, the *search space generated by* p_k is $G^{k*} = (V^{k*}, E^{k*}, l, h, s^k, c^k)$, where $V^{k*} = \{v \mid v \in V,\ s^k(v) \neq 0\}$ and E^{k*} is the restriction of E to V^{k*}. The *generated search space*

is $G^* = (V^*, E^*, l, h, \bar{s}, \bar{c})$, where $V^* = \{v \mid v \in V,\ \exists k\ s^k(v) \neq 0\}$ and E^* is the restriction of E to V^*.

In summary, if we fix k and let i vary we get the successions $\{s_i^k\}_{i\geq 0}$ and $\{c_i^k\}_{i\geq 0}$ which characterize the derivation by p_k. For each state S_i^k, the marked search graph $G_i^k = (V, E, l, h, s_i^k, c_i^k)$ represents the state of the search space at stage i at process p_k, G_i^{k+} represents its active part, and G_i^{k*} represents the portion of the search space generated by p_k up to stage i. If we fix i and let k vary, we get the tuples of markings $\bar{s}_i = (s_i^0, \ldots s_i^{n-1})$ and $\bar{c}_i = (c_i^0, \ldots c_i^{n-1})$ which characterize the distributed derivation. The parallel marked search graph $G_i = (V, E, l, h, \bar{s}_i, \bar{c}_i)$ represents the state of the search space after all processes executed i steps, G_i^+ represents its active part, and G_i^* represents the search space generated by all processes up to stage i.

Similar to contraction (e.g., see the discussion in [21]), subdivision and communication could not have been represented by structural modifications of the graph. When a process p_i receives a clause φ, all the inferences between the clauses already stored at p_i and φ become available. Thus, it would seem that this effect could be represented by adding the arcs for such inference steps. The problem with such an approach is that the structure of the search space would depend on the search plan, because communication steps are decided dynamically by the search plan. In our approach, all inference steps with φ are in the static structure of the search graph, and the effect of receiving φ is represented by incrementing its marking and enabling those steps. A positive consequence of this representation is a sort of *transparency*: from the point of view of the possible inferences it is irrelevant whether a clause is available at a process because it was input, generated, or received.

Similarly, if forbidden arcs were eliminated from the graph, the structure of the search space would depend on the search plan, because the subdivision is decided dynamically by the search plan. Furthermore, we could not represent the parallel searches on one graph, because what is forbidden for one process is allowed for the other. In summary, our approach can represent dynamic search spaces because it distinguishes what belongs to the structure of the space and what belongs to the dynamics of the search.

5. Measures of search complexity

In this section we extend our approach to measure search complexity to distributed search. We begin by introducing the notion of *distance* of a clause in the search space.

5.1. A notion of distance for search spaces

The distance of a clause is measured on an *ancestor-graph*, which represents a *generation path* of the clause:

Definition 5.1. 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 $e = (v_1, \ldots v_n; v_{n+1}; v)$ is a hyperarc in E and $t_1, \ldots t_n, t_{n+1}$ are ancestor-graphs of $v_1, \ldots v_n, v_{n+1}$, then the graph with root v connected by e to the subgraphs $t_1, \ldots t_n, t_{n+1}$ is an *ancestor-graph* of v, and it is denoted by the triple $(v; e; (t_1, \ldots t_n, t_{n+1}))$.

The set of the ancestor-graphs of v in G is denoted by $at_G(v)$ (or $at_G(\varphi)$).

Consider a clause φ and one of its ancestor-graphs t. If an ancestor of φ in t is deleted by contraction, it becomes impossible to reach φ by traversing t. However, this is true only if the ancestor is *relevant*:

Definition 5.2. Let $t = (v; e; (t_1, \ldots t_n, t_{n+1}))$ be an ancestor-graph of v, where $l(v) = \varphi$ and $e = (v_1, \ldots v_n; v_{n+1}; v)$. A vertex $w \in t$, $w \neq v$, is *relevant to v in t for p_k at stage i* if

- either $w \in \{v_1, \ldots v_n, v_{n+1}\}$ and $\pi_1(c_i^k(e)) = 0$,
- or $\exists j$, $1 \leq j \leq n+1$, such that w is relevant to v_j in t_j for p_k at stage i.

$Rev_{G_i^k}(t)$ denotes the set of vertices relevant to v in t for p_k at stage i.

The following examples show how Definition 5.2 and Definition 4.2 cooperate to capture the relevance of ancestors. Numbers in boldface in the figures represent the markings.

Example 5.1. Figure 1 shows the ancestor-graph t_1 of $R \vee D$. Initially all an-

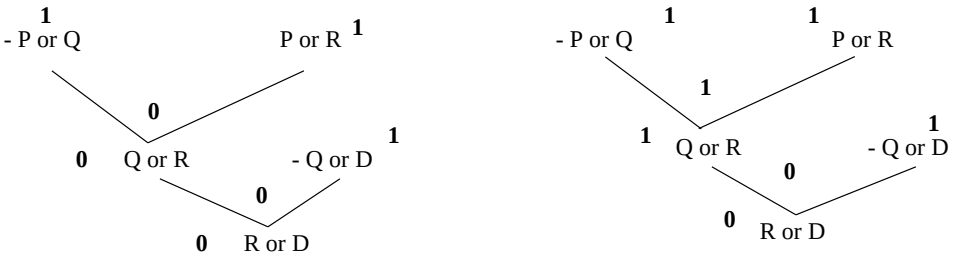

Figure 1. Ancestor graph t_1 before and after the generation of $Q \vee R$.

cestors are relevant. If the resolution arc that generates $Q \vee R$ is executed, the marking changes as shown in the right picture. After the step, the ancestors $\neg P \vee Q$ and $P \vee R$ are no longer relevant: if they were deleted, their deletion would not affect $Q \vee R$ and $R \vee D$, because $Q \vee R$ has been already generated. If $Q \vee R$ had been received, rather than generated, the resulting marking is the same, precisely to capture the fact that a subsequent deletion of $\neg P \vee Q$ or $P \vee R$ would not affect $Q \vee R$ and its descendants.

Example 5.2. Figure 2 illustrates the same concepts in the even more compelling case where a clause is used to delete its parents. Assume that p_k generates

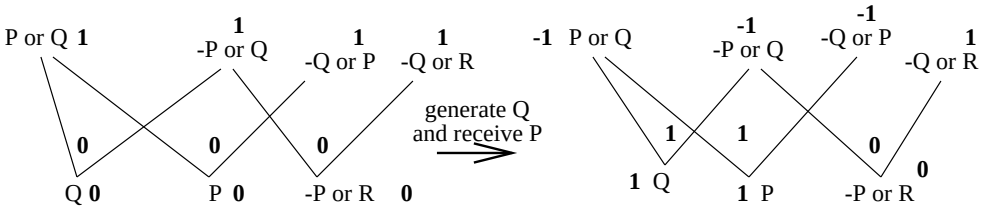

Figure 2. The evolution of the marking for Example 5.2.

Q by resolution from $P \vee Q$ and $\neg P \vee Q$, and then uses it to subsume $P \vee Q$ and $\neg P \vee Q$. Next, p_k receives P and uses it to subsume $\neg Q \vee P$. The marking resulting after all these steps is shown in the picture on the right in Figure 2: as desirable, $P \vee Q$ and $\neg P \vee Q$ are not relevant to Q, and $\neg Q \vee P$ and $P \vee Q$ are not relevant to P. On the other hand, $\neg P \vee Q$ is relevant to $\neg P \vee R$.

Example 5.3. The left picture in Figure 3 contains two ancestor-graphs: t_2 con-

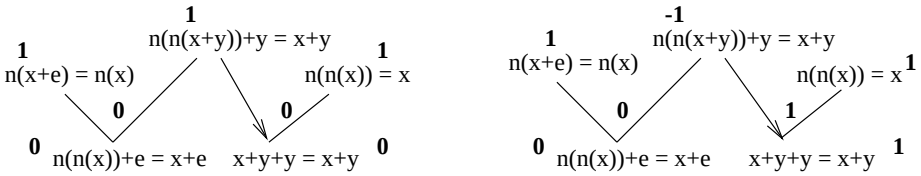

Figure 3. t_2 and t_3 before and after the simplification of $n(n(x+y))+y \simeq x+y$.

sists of a paramodulation step generating $n(n(x))+e \simeq x+e$ by paramodulating $n(x+e) \simeq n(x)$ into $n(n(x+y))+y \simeq x+y$; and t_3 consists of a simplification step which reduces $n(n(x+y))+y \simeq x+y$ to $x+y+y \simeq x+y$ by applying the simplifier $n(n(x)) \simeq x$. If the simplification step is executed (see the resulting marking on the right), $n(n(x+y))+y \simeq x+y$ and $n(n(x)) \simeq x$ are no longer relevant to $x+y+y \simeq x+y$. Indeed, the deletion of $n(n(x+y))+y \simeq x+y$ is not an obstacle to reach $x+y+y \simeq x+y$, rather, it has been part of a step

that generates it. On the other hand, $n(n(x+y))+y \simeq x+y$ is still relevant to $n(n(x))+e \simeq x+e$, and its absence prevents the process from generating $n(n(x))+e \simeq x+e$ through t_2.

Given a clause φ and one of its ancestor-graphs t, the p-distance, or *past-distance*, measures the portion of t that p_k has visited. The f-distance, or *future-distance*, measures the portion of t that p_k still needs to visit to reach φ by traversing t. The g-distance, or *global-distance*, is their sum. If a relevant ancestor of φ on t was deleted by p_k, the distance of φ on t for p_k is *infinite*:

Definition 5.3. For all φ, $t \in at_G(\varphi)$, p_k and stages i:

- The *p-distance of φ on t for p_k at i* is $pdist_{G_i^k}(t) = \mid \{w \mid w \in t, s_i^k(w) \neq 0\} \mid$.
- The *f-distance of φ on t for p_k at i* is

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

- The *g-distance of φ on t for p_k at i* is $gdist_{G_i^k}(t) = pdist_{G_i^k}(t) + fdist_{G_i^k}(t)$.

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

If a vertex has non-zero marking it is counted in the p-distance regardless of whether the clause was generated or received, because a process may advance on a path not only by generating clauses but also by receiving them. Note that it may happen that an ancestor-graph of φ has infinite distance and another one has finite distance. Accordingly, it may also happen that φ is generated by traversing an ancestor-graph, and therefore has positive marking, while another of its ancestor-graphs has infinite distance. If all ancestor-graphs of φ have infinite distance, the distance of φ in the graph is infinite, meaning that φ is unreachable.

Example 5.4. In the picture on the left in Figure 1, the p-distance of $R \vee D$ is 3 and the f-distance is 2. After $Q \vee R$ is generated, the p-distance is 4 and the f-distance is 1. If we consider $Q \vee R$ itself, its p-distance goes from 2 to 3, and its f-distance goes from 1 to 0. In the picture on the right in Figure 2, Q and P and their descendants have finite distance, whereas $\neg P \vee R$ and its descendants have infinite distance. In the picture on the right in Figure 3, $x+y+y \simeq x+y$ and its descendants have finite distance, whereas $n(n(x))+e \simeq x+e$ and its descendants have infinite distance.

We conclude with the correspondence between infinite distance and redundancy. If a relevant ancestor ψ in an ancestor-graph t of φ is deleted by con-

traction, it means that ψ and the arc of t that uses ψ are redundant. Thus, if $fdist_{G^k}(t) = \infty$, t includes redundant inference(s). If $fdist_{G_i^k}(\varphi) = \infty$, all paths reaching φ are redundant, and φ itself is redundant.

5.2. Subdivision, communication and overlap

If the subdivision function forbids an arc e to process p_k, p_k cannot continue the traversal of an ancestor-graph that contains e:

Definition 5.4. An ancestor-graph t is *forbidden* for process p_k at stage i if there exists an arc e in t such that $\pi_1(c_i^k(e)) = 0$ and $\pi_2(c_i^k(e)) = false$. It is *allowed* otherwise.

The next example illustrates the effect of subdivision. In the examples, we assume that if all the premises of an arc are absent, α is undefined and the arc is allowed by default (see Definition 4.5). In the figures, **T** and **F** are abbreviations for $true$ and $false$.

Example 5.5. In Figure 4, let t be the ancestor-graph made of arcs e and a. In

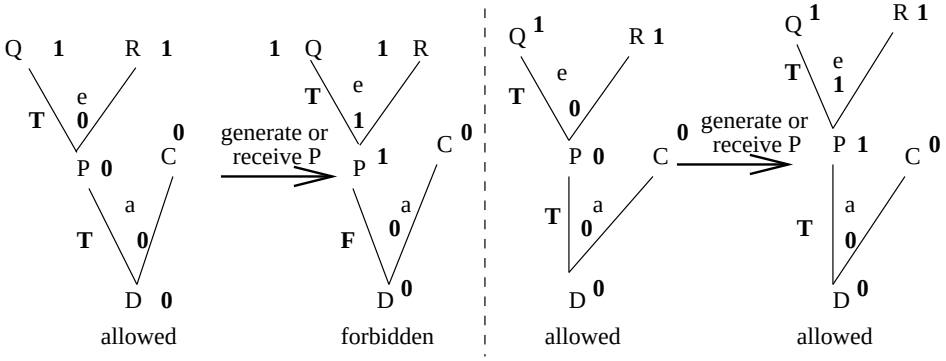

Figure 4. The effect of subdivision.

the left part of Figure 4, t is initially allowed for p_k, because e is allowed, and a is allowed by default. If p_k generates P by executing e, and α decides that a is forbidden, t becomes forbidden, and a subdivision of the search space occurs. In the right part of the figure, α decides that a is allowed, so that t remains allowed. The behaviour is the same if p_k receives P instead of generating it.

The next example shows that the given definitions distinguish properly among different ancestor-graphs of a clause.

Example 5.6. In Figure 5, let t be the ancestor-graph made of arcs b and a,

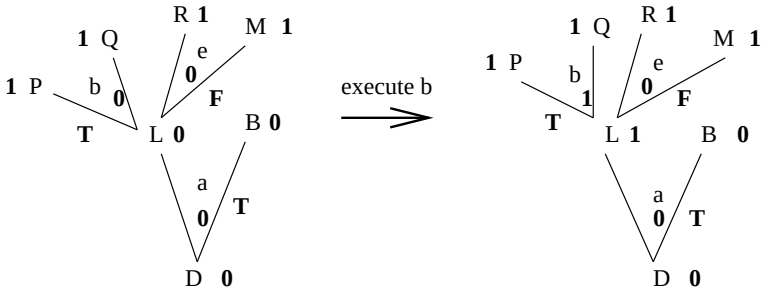

Figure 5. Ancestor-graphs t (made of b and a) and t' (made of e and a).

and t' the ancestor-graph made of arcs e and a. In the left picture, t is allowed, whereas t' is forbidden, because $\pi_2(c^k(e)) = false$ and $\pi_1(c^k(e)) = 0$. Assume that p_k executes b, generates L, and α decides that a remains allowed. The status of the two ancestor-graphs is unchanged: t' remains forbidden, because the marking of e has not changed. This is appropriate, because L was generated by executing b, not e, which means p_k is traversing t, not t'.

Subdivision excludes ancestor-graphs from consideration by a process. However, communication may undo its effect, as shown in the following example:

Example 5.7. Figure 6 shows the same ancestor-graph t of Figure 4 with a different permission marking. In the left part of Figure 6, t is initially forbidden

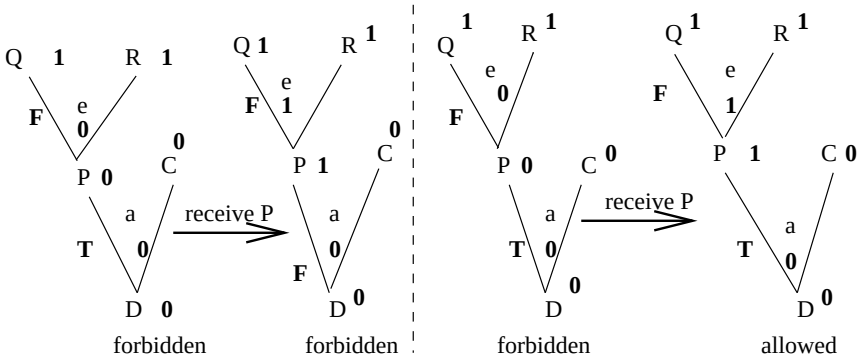

Figure 6. An effect of communication.

for process p_k, because $\pi_1(c_i^k(e)) = 0$ and $\pi_2(c_i^k(e)) = false$. If p_k receives P, and α decides based on P that a is forbidden, t remains forbidden, because $\pi_1(c_{i+1}^k(a)) = 0$ and $\pi_2(c_{i+1}^k(a)) = false$. If α decides that a is allowed (second half of Figure 6), t becomes allowed, because $\pi_1(c_{i+1}^k(e)) = 1$ and $\pi_2(c_{i+1}^k(a)) = true$, so that neither e nor a satisfy Definition 5.4. Note that the ancestor-graph made of e only goes from forbidden to allowed regardless of the permission of a.

By requiring $\pi_2(c_i^k(e)) = false$ and $\pi_1(c_i^k(e)) = 0$, Definition 5.4 takes communication into account. Note that $\pi_2(c_i^k(e)) = false$ and $\pi_1(c_i^k(e)) \neq 0$ may happen only as a consequence of communication, because a forbidden step cannot be executed. The idea is that a forbidden arc whose consequence has been received becomes irrelevant, because it is no longer true that forbidding that arc prevents the process from exploring that path.

In summary, our model captures here two aspects of distributed search. On one hand, the subdivision excludes search paths, thereby reducing the search effort of each process. On the other hand, the strategy employs communication to make sure that fairness (hence completeness) is preserved in presence of a subdivision. As a consequence, excluded search paths may be included again. 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 5.5. Two processes p_k and p_h *overlap* on ancestor-graph t at stage i, if t is allowed for both p_k and p_h at stage i.

Since an overlap represents a potential duplication of search effort, a goal of a distributed-search plan is to preserve fairness while minimizing the overlap.

5.3. The bounded search spaces for local and distributed derivations

We have all the elements to define a measure of search complexity that captures the effects of contraction, subdivision and communication on the search space. The following definition of *bounded search spaces* combines the application of the notion of distance to measure the effect of contraction, and the notion of allowed ancestor-graph to measure the effects of subdivision and communication. The bounded search space of bound j for p_k is the space of clauses whose g-distance for p_k is smaller than or equal to j. It is written as a multiset, with the multiplicity of a clause equal to the number of its *allowed* ancestor-graphs whose g-distance is at most j:

Definition 5.6. For all $j > 0$, the *bounded search space within distance j for process p_k at stage i* is the multiset of clauses $space(G_i^k, j) = \sum_{v \in V, v \neq \mathsf{T}} mul_{G_i^k}(v, j) \cdot l(v)$, where $mul_{G_i^k}(v, j) = |\{t \mid t \in at_G(v), t \; allowed \; for \; p_k \; at \; stage \; i, 0 < gdist_{G_i^k}(t) \leq j\}|$.

The bounded search spaces can be compared by using the multiset extension $\succ_{mul}$ of the ordering $\succ$ on clauses. The bounded search spaces for the sequential case, denoted $space(G_i, j)$, are defined in the same way, with multiplicity $mul_{G_i}(v, j) = |\{t \mid t \in at_G(v), 0 < gdist_{G_i}(t) \leq j\}|$. The sequential definition is a special case of Definition 5.6, because in the sequential case there is one process and all ancestor-graphs are allowed.

Definition 5.6 considers the bounded search spaces for the local derivation of a process. In order to capture the behavior of the distributed derivation as a whole, one needs to define a notion of *parallel bounded search spaces*. Assume that at stage i p_h and p_k overlap on an ancestor-graph t. Then t is counted in both $mul_{G_i^h}(v,j)$ and $mul_{G_i^k}(v,j)$. In order to reflect the overlap, one would like that t is counted twice in a suitable notion of parallel multiplicity. Thus, let $gmul_{G_i}(v,j) = \sum_{k=0}^{n-1} mul_{G_i^k}(v,j)$. However, p_h and p_k are active in parallel. Thus, let $pmul_{G_i}(v,j) = \lfloor gmul_{G_i}(v,j)/n \rfloor$. The following example shows how these notions apply to a couple of concrete cases.

Example 5.8. Assume that there are two processes p_1 and p_2 and clause $\varphi = l(v)$ has four ancestor-graphs whose g-distance is smaller than a given j. If at stage i two ancestor-graphs are allowed for p_1 only and the other two are allowed for p_2 only, $mul_{G_i^1}(v,j) = mul_{G_i^2}(v,j) = 2$, $gmul_{G_i}(v,j) = 4$ and $pmul_{G_i}(v,j) = 2$. In this case, there is no overlap. If all ancestor-graphs are allowed for p_1, and two are allowed also for p_2, $mul_{G_i^1}(v,j) = 4$, $mul_{G_i^2}(v,j) = 2$, $gmul_{G_i}(v,j) = 6$, $pmul_{G_i}(v,j) = 3$, and there is overlap.

We can then define the parallel bounded search spaces as follows:

Definition 5.7. For all $j > 0$, the *parallel bounded search space within distance j at stage i* is the multiset of clauses $pspace(G_i,j) = \sum_{v \in V, v \neq \mathsf{T}} pmul_{G_i}(v,j) \cdot l(v)$, where $pmul_{G_i}(v,j) = \lfloor gmul_{G_i}(v,j)/n \rfloor$ and $gmul_{G_i}(v,j) = \sum_{k=0}^{n-1} mul_{G_i^k}(v,j)$.

The sense of this definition is that in order to capture the behavior of the distributed derivation, we build an ideal representative process. It is an ideal process, because none of the p_k's has $mul_{G_i^k}(v,j) = pmul_{G_i}(v,j)$ for all v. It is a meaningful representative, because $pmul_{G_i}(v,j)$ is the *average multiplicity* of $l(v)$ at stage i. Building a representative process based on average multiplicity seems more significant than building a representative process which is the best on all clauses (i.e., the alternative definition $pmul_{G_i}(v,j) = min_{0 \leq k \leq n-1}\{mul_{G_i^k}(v,j)\}$) or the worst on all clauses (i.e., the alternative definition $pmul_{G_i}(v,j) = max_{0 \leq k \leq n-1}\{mul_{G_i^k}(v,j)\}$).

A minimal requirement for a suitable definition of parallel bounded search spaces is that if there are no subdivision and no communication, the parallel bounded search spaces should be equal to the sequential bounded search spaces. The following theorem shows that Definition 5.7 satisfies this requirement:

Theorem 5.1. If α is the constant function *true*, 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 $pspace(G_i, j) = space(G_i, j)$.

Proof: each p_k performs exactly the same steps as the sequential process (without subdivision Σ' is the same as Σ). In particular, the contraction steps are the same, so that for all stages i of the derivation, vertex v of the search graph, process p_k and value j of the bound, $mul_{G_i^k}(v,j) = mul_{G_i}(v,j)$. It follows that $space(G_i^k, j) = space(G_i, j)$. Also, $gmul_{G_i}(v,j) = n \cdot mul_{G_i}(v,j)$, and $pmul_{G_i}(v,j) = mul_{G_i}(v,j)$. It follows that $pspace(G_i, j) = space(G_i, j)$. □

The sequential and the distributed strategy have the same search complexity, so that the n parallel processes are wasted (by analogy with time complexity we could say that the speed-up is 1 and the efficiency is $1/n$).

6. The analysis

This section applies the framework developed so far to analyze distributed-search contraction-based strategies, and it is organized as follows: interaction of contraction and communication, properties of the subdivision that minimize the overlap, effect of each type of step on the bounded search spaces, and comparison of the evolution of sequential and parallel bounded search spaces.

6.1. Contraction and communication in contraction-based strategies

The possibility of receiving clauses creates the conditions for combinations of events that cannot occur in a sequential process. We begin by investigating what happens if a process receives a clause that it had already found redundant.

In [21] it was shown that if a φ is deleted at stage i and is regenerated via another ancestor-graph at some stage $j > i$, a contraction-based strategy will delete it again, and will do so before using φ to generate other clauses. This property is needed also in the distributed case, but it is not sufficient. First, we need to consider not only the possibility that p_k regenerates φ via another ancestor-graph, but also the possibility that p_k receives another variant of φ from another process, which may not be aware that φ is redundant. Second, we need to consider not only deleted clauses, but also clauses made unreachable by deleting a relevant ancestor on every ancestor-graph. In a sequential derivation, if $fdist_{G_i}(\varphi) = \infty$ because a relevant ancestor has been deleted on every $t \in at_G(\varphi)$, it is impossible for φ to appear at some $j > i$, because φ cannot be generated. In a distributed derivation, φ may still be received from another process at some $j > i$. Thus, we prove a more general result: if the distance of φ is infinite at some stage (regardless of whether φ is deleted or made unreachable by deleting a relevant ancestor on every ancestor-graph), and later φ appears (regardless of whether φ is received or regenerated), φ will be deleted and its distance will be infinite again.

Lemma 6.1. In a derivation with local eager contraction, for all processes p_k, stages i, and clauses φ, if $fdist_{G_i^k}(\varphi) = \infty$ and $s_j^k(\varphi) > 0$ for some $0 < i < j$, 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 φ to generate other clauses at any stage l, $j \leq l < q$.

Proof: if $fdist_{G_i^k}(\varphi) = \infty$, either $s_i^k(\varphi) < 0$ or $\forall t \in at_G(\varphi)$ there is an ancestor $\psi \in Rev_{G_i^k}(t)$ such that $s_i^k(\psi) < 0$. In the first case, it may happen that $s_j^k(\varphi) > 0$ for some $j > i$ if φ is generated on another ancestor-graph or received. In the second case, it may happen that $s_j^k(\varphi) > 0$ only if φ is received. In the first case, φ was deleted by contraction. In the second case, φ can be generated only from redundant clauses. In both cases, $\varphi \in R(S_i^k)$, and $\varphi \in R(S_j^k)$ for all $j > i$. If $s_j^k(\varphi) > 0$ for some $j > i$, we have $\varphi \in S_j^k \cap R(S_j^k - \{\varphi\})$. By Definition 2.5, there exist $f^m \in I_R$ and $\bar{x} \in (S_j^k)^m$, such that $\pi_1(f^m(\bar{x})) = \emptyset$ and $\pi_2(f^m(\bar{x})) = \{\varphi\}$. Since Σ satisfies local eager contraction, there exists a $q > j$ such that $S_{q-1}^k \vdash S_q^k$ deletes φ, so that $s_q^k(\varphi) < 0$, and p_k does not select an expansion step using φ at any stage l, $j \leq l < q$. Furthermore, since Σ is fair, it applies forward contraction before backward contraction, so that it will forward-contract φ before using it to backward-contract other clauses[5]. Therefore, p_k will not use φ to generate other clauses by either expansion or backward contraction at any stage l, $j \leq l < q$. □

Based on this lemma, we can make the following approximation: if a distance $fdist_{G_i^k}(\varphi)$ is infinite, then $fdist_{G_j^k}(\varphi)$ can be regarded as infinite for all $j > i$.

We consider next redundant inferences, rather than redundant clauses. This means we consider ancestor-graphs such that $fdist_{G_i^k}(t) = \infty$ rather than clauses such that $fdist_{G_i^k}(\varphi) = \infty$. A non-redundant clause has an ancestor-graph t such that $fdist_{G_i^k}(t) = \infty$, if t includes a redundant inference, i.e., a step using a redundant ancestor. In a sequential derivation, the result of Lemma 6.1 that φ is not used to generate other clauses at any stage l, $j \leq l < q$, implies that when φ is re-deleted at stage q, φ is still relevant on all ancestor-graphs where it was relevant at stage i. Therefore, a stronger approximation is supported: if $fdist_{G_i^k}(t)$ is infinite, $fdist_{G_j^k}(t)$ can be regarded as infinite for all $j > i$. In the distributed case, we cannot make the stronger approximation because communication may make deleted ancestors irrelevant. The following example shows how.

Example 6.1. Consider the leftmost picture in Figure 7, showing the marking at process p_k. Let t be the ancestor-graph made of e and b, and t' the ancestor-graph made of a and b. Since the relevant ancestor Q has been deleted, t has infinite distance, while t' has finite distance, so that P and R have finite distance. Note

[5] It is very well-known that applying backward contraction, e.g., subsumption, before forward contraction may violate fairness [36,16].

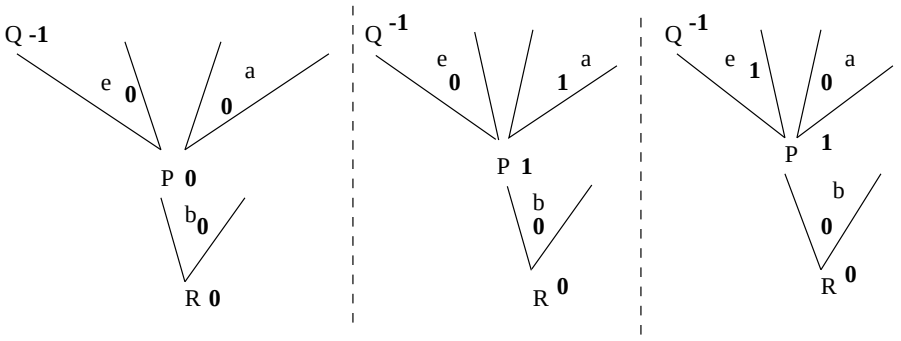

Figure 7. Communication may make contraction irrelevant.

that Q and therefore e are redundant, but we assume that a and P are not. If another process p_h executes a, generates P, and sends it to p_k, the marking at p_k evolves as in the picture in the middle: the distance on t is still infinite. Assume instead that p_h has not deleted Q, executes e, and sends to p_k a P generated by e. The resulting marking at p_k is shown in the rightmost picture in Figure 7: the distance of t is no longer infinite, because the arrival of P has made Q irrelevant ($\pi_1(c^k(e) = 1)$. Thus, the approximation $fdist_{G_i^k}(t) = \infty \Rightarrow \forall j > i \; fdist_{G_j^k}(t) = \infty$ may not hold for redundant ancestor-graphs of non-redundant clauses.

In this example there are two kinds of irrelevance of contraction. There is irrelevance of contraction at p_h, because the clause(s) that contract Q do not arrive at p_h fast enough to delete Q before it is used to generate P. When p_h finally deletes Q, 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 P from p_h makes the deletion of Q irrelevant: we call this phenomenon *contraction undone*.

Remark There is a special situation where it is expected that communication make a deleted ancestor irrelevant. Consider a distributed strategy which is contraction-based by Theorem 3.4: in such a strategy a process may delete a reducible clause φ without generating its reduced form, say φ'. Let e be the arc reducing φ to φ'. Any ancestor-graph t including e has infinite distance. When the φ' generated by another process arrives, the marking of e is increased, and φ becomes irrelevant to t (it is still relevant to all the ancestor-graphs not including e). This is expected, because the strategy executes the contraction step in two stages, delete φ and receive φ'.

Distributed global contraction does not prevent late contraction and contraction undone. Referring to Example 6.1, it only guarantees that Q will be deleted at p_h eventually, so that we may have $s_i^k(Q) = -1$ and $s_j^h(Q) = -1$ for some $j > i$. It is sufficient that p_h executes e and generates P at a stage $l < j$,

and P arrives at p_k at a stage $r > i$, for the situation of Example 6.1 to occur. The following lemma shows that local eager contraction and immediate propagation of clauses (hence global eager contraction) exclude late contraction and contraction undone; late contraction is excluded by showing that a deletion relevant for a process is relevant for all; contraction undone is excluded by showing that a deletion relevant at a stage remains relevant at all subsequent stages:

Lemma 6.2. Assume that the derivation has local eager contraction and immediate propagation of clauses up to redundancy. Consider $t \in at_G(\varphi)$ including an arc e, which uses premise ψ 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, then:

1. For all p_h and j, $s_j^h(\psi) = -1$ implies $\psi \in Rev_{G_j^h}(t)$ (no late contraction).
2. For all $j \geq i$, $\psi \in Rev_{G_j^k}(t)$ (no contraction undone).

Proof:

1. The proof is by way of contradiction. Assume that for some p_h, $s_j^h(\psi) = -1$ but $\psi \notin Rev_{G_j^h}(t)$, because p_h executes e at some stage l before deleting ψ. (Note that $\psi \in Rev_{G_i^k}(t)$ implies that e is not the arc that deletes ψ.) We distinguish two cases: $l \geq i$ and $l < i$.
   (a) Assume $l \geq i$ (i.e., p_h executes e after p_k has deleted ψ). From $s_i^k(\psi) = -1$, it follows $\psi \in R(S_i^k)$ and $\psi \in R(\bigcup_{h=0}^{n-1} S_i^h)$. Since p_h executes e at stage l, it is $\psi \in S_l^h$. Let g be the earliest stage such that $\psi \in S_g^h$ and let $r = max(i, g)$. Since $l \geq i$ and $l \geq g$, it is $l \geq r$. Since $r \geq i$, $\psi \in S_r^h \cap R(\bigcup_{h=0}^{n-1} S_r^h - \{\psi\})$. By Definition 2.5, there exist $f^m \in I_R$ and $\bar{x} \in (\bigcup_{h=0}^{n-1} S_r^h)^m$, such that $\pi_2(f^m(\bar{x})) = \{\psi\}$. By global eager contraction (Lemma 3.3), p_h deletes ψ at some stage $q \geq r$ without executing e at any stage between r and q, contradicting the assumption that p_h executes e at stage $l \geq r$.
   (b) Assume $l < i$ (i.e., p_h executes e before p_k deletes ψ). Since $\psi' \in S_{l+1}^h$ and $\psi' \in S_\infty - R(S_\infty)$, by immediate propagation of clauses up to redundancy, $\psi' \in S_{l+1}^k$. This together with $l < i$ (hence $l + 1 \leq i$) means that $\psi \notin Rev_{G_i^k}(t)$, contradicting the hypothesis.
2. By way of contradiction, assume that $\psi \notin Rev_{G_j^k}(t)$ for some $j \geq i$. The only event which may have made ψ irrelevant is that p_k has received from some other process p_h a variant of ψ' generated through e, but the first part of the proof showed that it is impossible that p_h executes e before deleting ψ. □

Under these hypotheses the approximation $fdist_{G_i^k}(t) = \infty \Rightarrow \forall j > i\ fdist_{G_j^k}(t) = \infty$ is justified:

Theorem 6.1. In a derivation with local eager contraction and immediate 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 $fdist_{G_q^k}(t) = \infty$.

Proof: let $t \in at_G(\varphi)$. There are three cases:

1. $fdist_{G_i^k}(t) = \infty$ because $s_i^k(\varphi) = -1$ and $fdist_{G_j^k}(t) \neq \infty$ because $s_j^k(\varphi) > 0$.
2. $fdist_{G_i^k}(t) = \infty$ because $s_i^k(\psi) = -1$ for $\psi \in Rev_{G_i^k}(t)$ and
   (a) $fdist_{G_j^k}(t) \neq \infty$ because $s_j^k(\psi) > 0$ while $\psi \in Rev_{G_j^k}(t)$.
   (b) $fdist_{G_j^k}(t) \neq \infty$ because $\psi \notin Rev_{G_j^k}(t)$.

In Cases 1 and 2a, Lemma 6.1 applies to φ and ψ, respectively, and there exists a $q > j$, such that $fdist_{G_q^k}(t) = \infty$. In Case 2b, let ψ' be the child of ψ in t and let e be the arc of t that generates ψ' from ψ. The only event which may have made ψ irrelevant is that p_k has received from some other process p_h a variant of ψ' generated through e. If $\psi' \in S_\infty - R(S_\infty)$, this is impossible by Lemma 6.2. If $\psi' \notin S_\infty - R(S_\infty)$, then $\psi' \in R(S_\infty)$. Let g be the earliest stage such that $\psi' \in S_g^k \cap R(\bigcup_{h=0}^{n-1} S_g^h - \{\psi'\})$. By Definition 2.5, there exist $f^m \in I_R$ and $\bar{x} \in (\bigcup_{h=0}^{n-1} S_g^h)^m$, such that $\pi_2(f^m(\bar{x})) = \{\psi'\}$. By global eager contraction, there exists a $q > j$ such that $s_q^k(\psi') = -1$ and $fdist_{G_q^k}(t) = \infty$. □

In summary, asynchronous communication may cause late contraction and contraction undone, which means that ancestor-graphs that could be pruned (infinite distance) are not pruned (finite distance). Since ancestor-graphs with finite distance are counted in the bounded search spaces, the bounded search spaces capture this cost of communication in terms of search space.

6.2. Minimization of the overlap

We analyze next the overlap. There are two kinds: first, there is the overlap caused by the subdivision function itself when it assigns the same arc to more than one process (e.g., $\pi_2(c^k(e)) = true$ and $\pi_2(c^h(e)) = true$); second, there is the overlap caused by communication (e.g., $\pi_2(c^k(e)) = true$ and $\pi_2(c^h(e)) = false$ but $\pi_1(c^h(e)) \neq 0$). In order to avoid the first type, the subdivision function should assign each arc to at most one process:

Definition 6.1. A subdivision function α has *no arc-duplication* if for all $e = (v_1, \ldots v_n; v_{n+1}; u)$ such that $u \neq \mathsf{T}$, $h(e) = f$ and $l(v_j) = \varphi_j$, $1 \leq j \leq n+1$, for all $i \geq 0$, there is at most one process p_k such that $\alpha((S_0^k, \ldots S_i^k), n, k, f, (\varphi_1, \ldots \varphi_{n+1})) = true$.

Since α is monotonic, the allowed process p_k is the same for all stages. The following example discusses duplication of expansion arcs.

Example 6.2. In Clause-Diffusion [18,12] only the process that owns an equation is allowed to paramodulate into it (see also Example 3.1). This is not sufficient to guarantee that there is no arc-duplication, because multiple variants of an equation may be generated. For instance, assume that p_k generates φ by traversing the ancestor-graph t of v, where $l(v) = \varphi$, and p_h generates a variant φ' of φ by traversing another ancestor-graph t' of v. Assume that φ and φ' are not forward-contracted. If p_k assigns φ to itself, and p_h assigns φ' to itself, the paramodulation hyperarcs into v are allowed for both p_k and p_h. Whether both processes will execute them depends on the continuation of the derivation: p_k broadcasts φ, p_h broadcasts φ', and each process compares them in the same well-founded ordering for variant subsumption. Assume that φ is smaller than φ' in this ordering. All processes delete φ' and keep φ, so that the paramodulation hyperarcs into v are enabled for p_k only. However, if p_h paramodulates into φ' before receiving φ and using it to subsume φ', and also p_k paramodulates into φ, a duplication of expansion inferences occurs. On the other hand, if the allocation criterion (the criterion that assigns equations to processes) guarantees that all variants of an equation are assigned to the same process, the property of no arc-duplication follows. Process p_k assigns φ to some p_j (possibly $j = k$ or $j = h$) and also p_h assigns φ' to p_j, so that the paramodulation hyperarcs into v are allowed to p_j only. For example, the *"syntax"* criterion [18] has this property, because it assigns clauses based on their non-variable symbols.

For contraction arcs, the condition $u \neq \mathsf{T}$ in Definition 6.1 excludes contraction inferences that do not generate non-trivial clauses: for example, allowing subsumption steps to all processes does not violate the definition. For contraction inferences that generate non-trivial clauses, no arc-duplication and eager contraction may be conflicting requirements. Indeed, a strategy which is contraction-based by Theorem 3.3 has arc-duplication, since Theorem 3.3 requires that all contraction inferences are allowed to all processes. On the other hand, a strategy which is contraction-based by Theorem 3.4 may have no arc-duplication.

Remark Duplication and redundancy are two different concepts. In a distributed derivation a non-redundant clause may be duplicated, and such duplication is not a redundancy (e.g., in Example 6.2, it is redundant for a process to keep both φ and φ', while it is not redundant that each process has a copy of φ). Similarly, duplicated steps may not be redundant (e.g., consider two processes that do the same contraction step to eliminate their copies of a redundant clause).

We consider next the overlap due to communication: no arc-duplication does not avoid this kind of overlap, because it does not prevent different processes from generating variants of the same clause, as shown in the following example.

Example 6.3. Assume that φ can be generated by two arcs e and a, arc e is allowed only to p_h, and arc a is allowed only to p_k, so that the requirement of no arc-duplication is satisfied. Let t and t' be the ancestor-graphs of φ including e and a, respectively: t is forbidden for p_k ($\pi_2(c^k(e)) = false$ and $\pi_1(c^k(e)) = 0$) and t' is forbidden for p_h ($\pi_2(c^h(a)) = false$ and $\pi_1(c^h(a)) = 0$). Assume that p_h executes e, generates a variant φ_1 of φ and sends it to p_k, while p_k executes a, generates a variant φ_2 of φ and sends it to p_h. When p_h receives φ_2, t' becomes allowed for p_h, and when p_k receives φ_1, t becomes allowed for p_k, so that both ancestor-graphs are allowed for both processes.

In order to prevent different processes from generating variants of the same clause we need to specify a stronger requirement:

Definition 6.2. A subdivision function α has *no clause-duplication* if for all vertices $u \neq \mathsf{T}$, for any two hyperarcs $e_1 = (v_1, \ldots v_m; v_{m+1}; u)$ and $e_2 = (w_1, \ldots w_q; w_{q+1}; u)$, where $h(e_1) = f$, $l(v_j) = \varphi_j$, $1 \leq j \leq m+1$, $h(e_2) = g$, $l(w_j) = \psi_j$, $1 \leq j \leq q+1$, for all $i \geq 0$, if $\alpha((S_0^k, \ldots S_i^k), n, k, f, (\varphi_1, \ldots \varphi_{m+1})) = true$ and $\alpha((S_0^h, \ldots S_i^h), n, h, g, (\psi_1, \ldots \psi_{q+1})) = true$, then $k = h$.

No clause-duplication implies no arc-duplication (take $e_1 = e_2$ in the above definition). No clause-duplication is compatible with fairness, because fairness only requires that at least one process is allowed to do the step. It is also compatible with propagation of clauses up to redundancy, including the limit case of immediate propagation, because of the distinction between allowed arc and allowed ancestor-graph: assume that p_h is the only process allowed to generate φ; if p_h generates a variant of φ, by executing arc e, and sends it to all other processes, the effect at each receiver p_k is that the ancestor-graph of φ including e may become allowed (because $\pi_1(c^k(e)) \neq 0$), but arc e itself remains forbidden ($\pi_2(c^k(e)) = false$).

The next lemma shows that no clause-duplication limits the overlap due to communication to one ancestor-graph per clause: if p_h is the only process authorized to generate variants of φ, for all other processes communication may allow *at most one* forbidden ancestor-graph of φ, so that the multiplicity of φ at all processes other than p_h is at most 1.

Lemma 6.3. In a derivation with local eager contraction and no clause-duplication, for any clause φ, if p_h is the only process allowed to generate φ, there exists a stage r such that for all $k \neq h$, $i \geq r$ and $j > 0$, $mul_{G_i^k}(\varphi, j) \leq 1$.

Proof: let r be the earliest stage such that $\alpha((S_0^h, \ldots S_r^h), n, h, f, \bar{x}) \neq \perp$ for all f and $\bar{x}$ generating φ. In other words, r is the earliest stage when the subdivision of the arcs generating φ is completed. For all arcs e generating φ, $\pi_2(c_r^h(e)) = true$, and $\pi_2(c_r^k(e)) = false$, $\pi_1(c_r^k(e)) = 0$ for all $k \neq h$. For all $t \in at_G(\varphi)$, t includes

one of these arcs: it follows that all ancestor-graphs in $at_G(\varphi)$ are forbidden to all processes other than p_h at stage r. Thus, $mul_{G_r^k}(\varphi, j) = 0$ for all $k \neq h$. Assume that at some stage $q \geq r$, p_h executes one of the arcs generating φ, say $\hat{e}$, generates φ and sends it to p_k. It follows that $\pi_1(c_i^k(\hat{e})) = 1$ for some $i \geq r$, and the ancestor-graph including $\hat{e}$ may be allowed. If it becomes allowed, $mul_{G_i^k}(\varphi, j) = 1$. If p_h executes another of these arcs and generates a variant φ' of φ, by local eager contraction p_h uses φ to subsume φ', so that φ' is not sent. It follows that for all $k \neq h$, $i \geq r$, and $j > 0$, $mul_{G_i^k}(\varphi, j) \leq 1$. □

It is legitimate to ask whether the subdivision properties studied in this section may prevent late contraction and contraction undone. A simple reinspection of Example 6.1 shows that this is not the case.

Example 6.4. Reconsider Example 6.1 and Figure 7. Under no clause-duplication, both arcs e and a are allowed only to p_h, and are forbidden for p_k. As in Example 6.1, if p_h executes a there is no consequence. In fact, even if p_h executes a and then e there is no consequence, because local eager contraction is sufficient to ensure that the variant of P generated by a, say P_a, forward-subsumes the variant of P generated by e, say P_e. On the other hand, if p_h executes e first and sends P_e to p_k, there is late contraction (at p_h) and contraction undone (at p_k) as in Example 6.1. Note that even if p_h executes a before sending P_e to p_k, local eager contraction is not guaranteed to prevent the contraction undone phenomenon, because it is not guaranteed that P_a subsumes P_e. In fact, assuming a typical fair search plan where forward subsumption has priority over backward subsumption it is P_e that subsumes P_a.

However, it is possible to show that with no arc-duplication, and, a fortiori, with no clause-duplication, the effect of communication undoing subdivision and the effect of communication undoing contraction collapse into one:

Lemma 6.4. In a derivation with local eager contraction and no arc-duplication, if $fdist_{G_i^k}(t) = \infty$ and $fdist_{G_j^k}(t) \neq \infty$ for some $j > i$, then either there is a $q > j$, such that $fdist_{G_q^k}(t) = \infty$, or t is forbidden for p_k at both stages i and j, or t is forbidden for p_k at stage i and allowed at stage j.

Proof: the classification of cases is the same as in the proof of Theorem 6.1, assuming $t \in at_G(\varphi)$:

1. $fdist_{G_i^k}(t) = \infty$ because $s_i^k(\varphi) = -1$ and $fdist_{G_j^k}(t) \neq \infty$ because $s_j^k(\varphi) > 0$.
2. $fdist_{G_i^k}(t) = \infty$ because $s_i^k(\psi) = -1$ for $\psi \in Rev_{G_i^k}(t)$ and
   (a) $fdist_{G_j^k}(t) \neq \infty$ because $s_j^k(\psi) > 0$ while $\psi \in Rev_{G_j^k}(t)$.

(b) $fdist_{G_j^k}(t) \neq \infty$ because $\psi \notin Rev_{G_j^k}(t)$.

In Cases 1 and 2a, Lemma 6.1 applies to φ and ψ, respectively, and there exists a $q > j$, such that $fdist_{G_q^k}(t) = \infty$. In Case 2b, let ψ' be the child of ψ in t and let e be the arc of t that generates ψ' from ψ. The only event which may have made ψ irrelevant is that p_k has received from some other process p_h a variant of ψ' generated through e. If p_h has executed e, then e is allowed for p_h. Since the strategy has no arc-duplication, e is forbidden for p_k, so that $\pi_2(c_i^k(e)) = false$. Since $\psi \in Rev_{G_i^k}(t)$, it is $\pi_1(c_i^k(e)) = 0$. It follows that at stage i, t not only had infinite distance, it was also forbidden. If there is another arc $a \in t$ such that $\pi_1(c_j^k(a)) = 0$ and $\pi_2(c_j^k(a)) = false$, t is still forbidden at stage j. (For instance, a could be the arc below ψ' in t, whose status has been decided upon receiving ψ'.) Otherwise, t is allowed at stage j. □

In other words, either the ancestor-graph which is not pruned is forbidden, so that it is excluded anyway, or it was forbidden and it becomes allowed as an effect of the communication step, so that the effects of allowing the ancestor-graph and making its distance finite again coincide.

6.3. Effects of derivation steps on the complexity measures

This section integrates all our observations on subdivision, contraction and communication by showing how the different kinds of steps affect the local bounded search spaces during a derivation $S_0^k \vdash_{\mathcal{C}} S_1^k \vdash_{\mathcal{C}} \dots S_i^k \vdash_{\mathcal{C}} \dots$ by a distributed-search contraction-based strategy $\mathcal{C} =< I, M, \Sigma >$, with $I = I_E \cup I_R$, $\Sigma = \langle \zeta, \xi, \alpha, \omega \rangle$, and $\succ$ the well-founded ordering on clauses underlying I_R.

In a sequential derivation expansion steps do not change the bounded search spaces [21]. This would be sufficient for a distributed derivation as well, if we were considering only the inference itself, but we need to consider the effect of the subdivision. When ψ is generated, the subdivision function α may become defined on a tuple of premises $\bar{x}$ including ψ. If α decides that an arc e with premises $\bar{x}$ is forbidden, ancestor-graphs including e become forbidden (e.g., Example 5.5). Since we do not model the decisions by α as separate steps in the derivation, this phenomenon is captured as a possible effect of generating a clause:

Theorem 6.2. If $S_i^k \vdash S_{i+1}^k$ generates ψ, $\forall j > 0\ space(G_{i+1}^k, j) \preceq_{mul} space(G_i^k, j)$.

Proof: if $s_i^k(\psi) > 0$, p_k generates a variant of a clause that was already present, and no permission marking is affected. If $s_i^k(\psi) = 0$, $s_{i+1}^k(\psi) = 1$. If there are a tuple of premises $\bar{x}$ and an inference rule f, such that $\alpha((S_0^k \dots S_i^k), n, k, f, \bar{x}) = \perp$, and $\alpha((S_0^k, \dots S_{i+1}^k), n, k, f, \bar{x}) \neq \perp$, because of the generation of ψ, then let e be the arc of G with premises $\bar{x}$ and $h(e) = f$. By Definition 4.5, $\pi_2(c_i^k(e)) = true$. If $\alpha((S_0^k \dots S_{i+1}^k), n, k, f, \bar{x}) = true$, $\pi_2(c_{i+1}^k(e)) = true$ and nothing changes. If $\alpha((S_0^k \dots S_{i+1}^k), n, k, f, \bar{x}) = false$, $\pi_2(c_{i+1}^k(e)) = false$. If $\pi_1(c_i^k(e)) \neq 0$ (e.g., p_k

had already received the consequence of e), nothing changes. If $\pi_1(c_i^k(e)) = 0$, all ancestor-graphs including e that are allowed for p_k in G_i become forbidden. Let T denote the set of ancestor-graphs allowed for p_k in G_i and forbidden in G_{i+1}. For all φ, if $at_G(\varphi) \cap T = \emptyset$, $mul_{G_{i+1}^k}(\varphi, j) = mul_{G_i^k}(\varphi, j)$ for all $j > 0$. If $at_G(\varphi) \cap T \neq \emptyset$, let $p(\varphi) = min\{gdist_{G_{i+1}^k}(t) | t \in at_G(\varphi) \cap T\}$. That is, $p(\varphi)$ is the smallest value of the bound that is deep enough to include an ancestor-graph of φ whose status has changed. It follows that $mul_{G_{i+1}^k}(\varphi, j) = mul_{G_i^k}(\varphi, j)$ for all $j < p(\varphi)$ and $mul_{G_{i+1}^k}(\varphi, j) < mul_{G_i^k}(\varphi, j)$ for all $j \geq p(\varphi)$. Since for all clauses the multiplicity either diminishes or remains the same, we have $space(G_{i+1}^k, j) \preceq_{mul} space(G_i^k, j)$ for all $j > 0$. □

For a sequential derivation we proved that contraction reduces the bounded search spaces [21]. In addition, the proof of Theorem 6.2 applies also to a contraction inference that generates a clause other than the dummy clause *true*. A contraction step replacing ψ by ψ' prunes those ancestor-graphs whose distance becomes infinite because of the deletion of ψ, and those ancestor-graphs which become forbidden as a consequence of the generation of ψ'. The following theorem covers both phenomena. For the proof, we refer to [21] for the effect of contraction itself, and to the proof of Theorem 6.2 for the effect of subdivision.

Theorem 6.3. If $S_i^k \vdash S_{i+1}^k$ replaces ψ by ψ', $\forall j > 0$ $space(G_{i+1}^k, j) \preceq_{mul} space(G_i^k, j)$. If $s_i^k(\psi) = 1$ and there are clauses that have ψ as relevant ancestor at p_k at stage $i + 1$, then $\exists l > 0$, $\forall j \geq l$, $space(G_{i+1}^k, j) \prec_{mul} space(G_i^k, j)$.

A typical sequence of events is to generate a clause by either expansion or backward-contraction, forward-contract it, and then apply α to decide whether the steps involving this clause are allowed. If the clause is generated by expansion and forward-contraction does not apply, the subdivision decision is taken after the expansion step and Theorem 6.2 applies. Otherwise, the subdivision decision is taken after a contraction step and Theorem 6.3 applies.

We conclude with communication. When p_k sends a clause, there are no consequences on the bounded search spaces for p_k. When p_k receives a clause ψ, there may be three kinds of consequences. First, allowed ancestor-graphs may become forbidden (subdivision, e.g., Example 5.5), just like for the generation of a clause. Second, forbidden ancestor-graphs may become allowed (subdivision undone, e.g., Example 5.7). Third, relevant deleted ancestors may become irrelevant (contraction undone, e.g., Example 6.1). It follows that while in a sequential derivation the bounded search spaces may either remain the same (expansion) or decrease (contraction), in a distributed derivation the bounded search spaces of a process may oscillate *non-monotonically* because of communication. However, communication cannot expand the bounded search spaces, but only undo a previous reduction by subdivision or contraction. Therefore, the resulting bounded

search spaces are limited by the bounded search spaces at some previous stage:

Theorem 6.4. If $S_i^k \vdash S_{i+1}^k$ sends ψ, $\forall j > 0$, $space(G_{i+1}^k, j) = space(G_i^k, j)$. If $S_i^k \vdash S_{i+1}^k$ receives ψ, $\forall j > 0$, $\exists\, l \leq i$, $space(G_{i+1}^k, j) \preceq_{mul} space(G_l^k, j)$.

Proof: only the thesis about the *receive* step needs to be proved. We consider first the effect on the permission marking. If $s_i^k(\psi) > 0$, p_k receives a variant of a clause it already has, and no permission marking is affected. If $s_i^k(\psi) = 0$, $s_{i+1}^k(\psi) = 1$. If there is a tuple of premises $\bar{x}$ and an inference rule f such that $\alpha((S_0^k, \ldots S_i^k), n, k, f, \bar{x}) = \perp$, and $\alpha((S_0^k, \ldots S_{i+1}^k), n, k, f, \bar{x}) \neq \perp$, because of the arrival of ψ, let a be the arc of G with premises $\bar{x}$ and $h(a) = f$. By Definition 4.5, $\pi_2(c_i^k(a)) = true$. Let e be the arc that generated the received ψ.

1. If $\alpha((S_0^k, \ldots S_{i+1}^k), n, k, f, \bar{x}) = false$, $\pi_2(c_{i+1}^k(a)) = false$. If $\pi_1(c_i^k(a)) = 0$, $\pi_1(c_{i+1}^k(a))$ is still 0, and all ancestor-graphs including a that are allowed for p_k in G_i become forbidden in G_{i+1} (subdivision). Let T^- denote the set of these ancestor-graphs.

2. If $\alpha((S_0^k, \ldots S_{i+1}^k), n, k, f, \bar{x}) = true$, $\pi_2(c_{i+1}^k(a)) = true$. If $\pi_1(c_i^k(e)) = 0$ and $\pi_2(c_i^k(e)) = false$, we have now $\pi_1(c_{i+1}^k(e)) = 1$ and $\pi_2(c_{i+1}^k(e)) = false$, and all ancestor-graphs that e makes forbidden for p_k in G_i become allowed in G_{i+1} (subdivision undone). Let T_1^+ denote the set of these ancestor-graphs.

$T^- \cap T_1^+ = \emptyset$ because an ancestor-graph cannot be allowed and forbidden at the same time. Next, we consider contraction undone. If $\pi_1(c_i^k(e)) = 0$ and $s_i^k(\psi') = -1$ for a premise ψ' of e, we have $\pi_1(c_{i+1}^k(e)) = 1$ as a consequence of receiving ψ. It follows that for all ancestor-graphs t such that $fdist_{G_i^k}(t) = \infty$ because of ψ', it is $fdist_{G_{i+1}^k}(t) \neq \infty$. Let $T_2^+ = \{t \mid fdist_{G_i^k}(t) = \infty,\ fdist_{G_{i+1}^k}(t) \neq \infty,\ t\ allowed\ for\ p_k\ at\ i+1\}$ and $T^+ = T_1^+ \cup T_2^+$. If the strategy has global eager contraction, $T_2^+ = \emptyset$ and $T^+ = T_1^+$ by Theorem 6.1; if the strategy has no arc-duplication, $T_2^+ \subseteq T_1^+$ and $T^+ = T_1^+$ by Lemma 6.4; if the strategy has no clause-duplication, $T_2^+ \subseteq T_1^+$, $T^+ = T_1^+$ and $|T^+| = 1$ by Lemma 6.3. For all clauses φ:

- If $at_G(\varphi) \cap T^+ = \emptyset$ and $at_G(\varphi) \cap T^- = \emptyset$, $\forall j > 0$, $mul_{G_{i+1}^k}(\varphi, j) = mul_{G_i^k}(\varphi, j)$.
- If $at_G(\varphi) \cap T^- \neq \emptyset$ and $at_G(\varphi) \cap T^+ = \emptyset$, let $p(\varphi) = min\{gdist_{G_{i+1}^k}(t) | t \in at_G(\varphi) \cap T^-\}$. It follows that $mul_{G_{i+1}^k}(\varphi, j) = mul_{G_i^k}(\varphi, j)$ for all $j < p(\varphi)$ and $mul_{G_{i+1}^k}(\varphi, j) < mul_{G_i^k}(\varphi, j)$ for all $j \geq p(\varphi)$.
- If $at_G(\varphi) \cap T^+ \neq \emptyset$ and $at_G(\varphi) \cap T^- = \emptyset$, let $q(\varphi) = min\{gdist_{G_{i+1}^k}(t) | t \in at_G(\varphi) \cap T^+\}$. It follows that $mul_{G_{i+1}^k}(\varphi, j) = mul_{G_i^k}(\varphi, j)$ for all $j < q(\varphi)$ and $mul_{G_{i+1}^k}(\varphi, j) > mul_{G_i^k}(\varphi, j)$ for all $j \geq q(\varphi)$.

- If $at_G(\varphi) \cap T^+ \neq \emptyset$ and $at_G(\varphi) \cap T^- \neq \emptyset$, let $p(\varphi)$ and $q(\varphi)$ be defined as above, and $U(\varphi, j) = \{t |\ t \in at_G(\varphi), gdist_{G^k_{i+1}}(t) \leq j\}$. If $p(\varphi) < q(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) = mul_{G^k_i}(\varphi, j)$ for all $j < p(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) < mul_{G^k_i}(\varphi, j)$ for all $p(\varphi) \leq j < q(\varphi)$, whereas for all $j \geq q(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) \gtreqless mul_{G^k_i}(\varphi, j)$ depending on whether $|U(\varphi, j) \cap T^+| \gtreqless |U(\varphi, j) \cap T^-|$. If $p(\varphi) \geq q(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) = mul_{G^k_i}(\varphi, j)$ for all $j < q(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) > mul_{G^k_i}(\varphi, j)$ for all $q(\varphi) \leq j < p(\varphi)$, whereas for all $j \geq p(\varphi)$, $mul_{G^k_{i+1}}(\varphi, j) \gtreqless mul_{G^k_i}(\varphi, j)$ depending on whether $|U(\varphi, j) \cap T^+| \gtreqless |U(\varphi, j) \cap T^-|$.

For all j, for all φ such that $mul_{G^k_{i+1}}(\varphi, j) > mul_{G^k_i}(\varphi, j)$, let $l(\varphi)$ be the highest index such that $l(\varphi) \leq i$ and $mul_{G^k_{i+1}}(\varphi, j) \leq mul_{G^k_{l(\varphi)}}(\varphi, j)$. Such a stage $l(\varphi)$ is guaranteed to exist, because receiving φ may only restore ancestor-graphs that are forbidden or unreachable at stage i but were allowed and reachable at some earlier stage. Note that for all those φ whose multiplicities remain the same or decrease, it is sufficient to take $l(\varphi) = i$. Let $l = min\{l(\varphi) \mid \varphi \in space(G^k_0, j)\}$. It follows that $space(G^k_{i+1}, j) \preceq_{mul} space(G^k_l, j)$. □

6.4. A limit theorem for distributed-search contraction-based strategies

In this section we lift the analysis from considering the parallel processes in isolation to comparing the distributed derivation as a whole with a sequential derivation. Let $\mathcal{C} = \langle I, \Sigma \rangle$ be a contraction-based strategy, with inference system $I = I_E \cup I_R$, such that for all input sets S, $S^*_I = S^*_{I_E}$, and search plan $\Sigma = \langle \zeta, \xi, \omega \rangle$ uniformly fair with respect to I_E and R (see Section 3.3). Let $\mathcal{C}' = \langle I, M, \Sigma' \rangle$ with $\Sigma' = \langle \zeta', \xi', \alpha, \omega' \rangle$ be a parallelization by subdivision of $\mathcal{C}$ such that α is total on generated clauses and monotonic; Σ' is uniformly fair (in the sense of Definition 3.5) with respect to I_E and R, and has propagation of clauses up to redundancy; $\mathcal{C}'$ satisfies the hypotheses of Theorem 3.3 or Theorem 3.4 and therefore is contraction-based (i.e., it has local eager contraction and distributed global contraction). We consider the derivation by $\mathcal{C}$, $S = S_0 \vdash_{\mathcal{C}} \ldots S_i \vdash_{\mathcal{C}} \ldots$, and the distributed derivation by $\mathcal{C}'$, $S = S^k_0 \vdash_{\mathcal{C}'} \ldots S^k_i \vdash_{\mathcal{C}'} \ldots$, generated by processes $p_0, \ldots p_{n-1}$. Since $\mathcal{C}$ and $\mathcal{C}'$ have the same inference system, the initial search space is the same, i.e., $G^k_0 = G_0$ and $space(G^k_0, j) = space(G_0, j)$ for all k and j.

We begin by showing that a clause redundant for $\mathcal{C}$ is redundant also for $\mathcal{C}'$. The first lemma follows from uniform fairness of $\mathcal{C}'$:

Lemma 6.5. If $\varphi \in S_i$ for some $i \geq 0$, then there exist p_k and $j \geq 0$ such that either $\varphi \in S^k_j$ or $\varphi \in R(S^k_j)$.

Proof: if φ is generated by $\mathcal{C}$, then, since $S^*_I = S^*_{I_E}$, it can be generated by expansion by $\mathcal{C}'$. Let S_∞ be the set of persistent clauses in the derivation by

$\mathcal{C}'$, i.e., $S_\infty = \bigcup_{k=0}^{n-1} S_\infty^k$. If $\varphi \in I_E(S_\infty - R(S_\infty))$, by uniform fairness of Σ' and Theorem 3.1, there is a p_k that generates φ: $\varphi \in S_j^k$ for some j. If $\varphi \notin I_E(S_\infty - R(S_\infty))$, it means that φ can be generated only by using clauses that are redundant or non-persistent, hence redundant, in the derivation by $\mathcal{C}'$. Thus, φ itself is redundant: $\varphi \in R(S_j^k)$ for some p_k and stage j. □

Lemma 3.1 (propagation of clauses up to redundancy implies propagation of redundancy) and Lemma 6.5 yield the following:

Lemma 6.6. If $\varphi \in R(S_i)$ for some $i \geq 0$, then for all p_k there exists a $j \geq 0$ such that $\varphi \in R(S_j^k)$.

Proof: let $\{\psi_1, \ldots \psi_m\} \subseteq S_i$ be the smallest subset of clauses such that $\varphi \in R(\{\psi_1, \ldots \psi_m\})$. By Lemma 6.5, for all r, $1 \leq r \leq m$, there exist a p_k and a j_r such that $\psi_r \in S_{j_r}^k \cup R(S_{j_r}^k)$. Because all clauses deleted by $\mathcal{C}'$ are redundant with respect to R, and R is monotonic, it follows that for all $l \geq j_r$, $\psi_r \in S_l^k \cup R(S_l^k)$. Thus, if we take $l = max\{j_r \mid 1 \leq r \leq m\}$, we have $\psi_r \in \bigcup_{k=0}^{n-1}(S_l^k \cup R(S_l^k))$ for all r, $1 \leq r \leq m$. The hypothesis $\varphi \in R(\{\psi_1, \ldots \psi_m\})$ implies $\varphi \in R(\bigcup_{k=0}^{n-1}(S_l^k \cup R(S_l^k)))$. Since redundant clauses are irrelevant to establish other redundancies, it follows that $\varphi \in R(\bigcup_{k=0}^{n-1} S_l^k)$. By propagation of redundancy, it follows that for all p_k there exists a j such that $\varphi \in R(S_j^k)$. □

A clause found redundant in the sequential derivation is either contracted ($s_i(\varphi) = -1$) or unreachable ($s_i(\varphi) = 0$ and $fdist_{G_i}(\varphi) = \infty$). In either case, $fdist_{G_i}(\varphi) = \infty$, and, by the approximation supported by the sequential version of Lemma 6.1, we may assume $fdist_{G_j}(\varphi) = \infty$ for all $j \geq i$. In the distributed derivation there is a third possibility: either φ is contracted by p_k ($s_i^k(\varphi) = -1$), or φ is unreachable for p_k ($s_i^k(\varphi) = 0$ and $fdist_{G_i^k}(\varphi) = \infty$), or all its ancestor-graphs are forbidden for p_k. In the first two cases, $fdist_{G_i^k}(\varphi) = \infty$ and, by the approximation supported by Lemma 6.1, we may assume $fdist_{G_j^k}(\varphi) = \infty$ for all $j \geq i$. In the third case, it is not possible to prove that all ancestor-graphs are permanently forbidden for p_k, because communication may undo subdivision. However, the key point is that this is not necessary for redundant clauses. If φ is redundant, it is possible to prove that for every process, for some stage j, for all $l \geq j$, either φ is deleted, or it is unreachable, or all its ancestor-graphs are forbidden. This condition is satisfied by a process where φ is permanently forbidden, but it is also satisfied by a process where a forbidden ancestor-graph of φ becomes allowed, φ is generated, and then deleted by contraction.

Lemma 6.7. If $fdist_{G_i}(\varphi) = \infty$ for some $i \geq 0$, then for all p_k there exists a $j \geq 0$ such that for all $l \geq j$, either $fdist_{G_l^k}(\varphi) = \infty$, or all $t \in at_G(\varphi)$ are forbidden for p_k at stage l.

Proof: if $fdist_{G_i}(\varphi) = \infty$, it means either $s_i(\varphi) = -1$ (φ deleted) or for all $t \in at_G(\varphi)$ there is a $\psi \in Rev_{G_i}(t)$ such that $s_i(\psi) = -1$ (φ unreachable). Since deleted clauses are redundant, and clauses that can be generated only by redundant clauses are redundant, $\varphi \in R(S_i)$. By Lemma 6.6, for all p_k, there exists a $g \geq 0$ such that $\varphi \in R(S_g^k)$. Let g_k be the smallest such index for p_k: $\varphi \in R(S_r^k)$ for all $r \geq g_k$. We distinguish two cases.

- Some process in the distributed derivation generates φ. If p_k generates or receives φ at some stage (smaller or greater than g_k is irrelevant), there exists a $q \geq g_k$ such that $\varphi \in S_q^k \cap R(S_q^k - \{\varphi\})$. By Definition 2.5, there exist $f^m \in I_R$ and $\bar{x} \in (S_q^k)^m$, such that $\pi_2(f^m(\bar{x})) = \{\varphi\}$. By local eager-contraction of Σ', there exists a $j \geq q$ such that $s_j^k(\varphi) = -1$. Hence $fdist_{G_j^k}(\varphi) = \infty$ and $fdist_{G_l^k}(\varphi) = \infty$ for all $l \geq j$ by the approximation supported by Lemma 6.1. If p_k neither generates nor receives φ, there exists a $j \geq 0$ such that for all $l \geq j$, $s_l^k(\varphi) = 0$ and either $fdist_{G_l^k}(\varphi) = \infty$ or all $t \in at_G(\varphi)$ are forbidden for p_k at stage l.
- No process generates φ. Since Σ' is uniformly fair, this means that for all processes allowed to generate φ, φ is made unreachable by deleting relevant ancestors on all its ancestor-graphs. In other words, φ is either unreachable or forbidden everywhere, and the thesis holds. □

Thus, all redundant clauses eliminated by $\mathcal{C}$ will be excluded by $\mathcal{C}'$ as well:

Theorem 6.5. If $fdist_{G_i}(\varphi) = \infty$ for some $i \geq 0$, then there exists an $r \geq 0$ such that for all $i \geq r$ and $j > 0$, $pmul_{G_i}(\varphi, j) = 0$.

Proof: by Lemma 6.7 for all p_k there is a stage l_k such that for all $i \geq l_k$ and $j > 0$, $mul_{G_i^k}(\varphi, j) = 0$. Let $r = max\{l_k \mid 0 \leq k \leq n-1\}$. It follows that $pmul_{G_i}(\varphi, j) = 0$ for all $i \geq r$ and $j > 0$. □

The next lemma moves the analysis from redundant clauses to ancestor-graphs including redundant inferences.

Lemma 6.8. Assume $\mathcal{C}'$ has immediate propagation of clauses up to redundancy. If $fdist_{G_i}(t) = \infty$ for some $i \geq 0$, then for all p_k there exists a $j \geq 0$ such that for all $l \geq j$, either $fdist_{G_l^k}(t) = \infty$, or t is forbidden for p_k at stage l.

Proof: let $t \in at_G(\varphi)$. If $fdist_{G_i}(t) = \infty$, either $s_i(\varphi) = -1$ or $s_i(\psi) = -1$ for some $\psi \in Rev_{G_i}(t)$.

- If $s_i(\varphi) = -1$, $fdist_{G_i}(\varphi) = \infty$ and Lemma 6.7 applies to φ: for all p_k there exists a $j \geq 0$ such that for all $l \geq j$, either $fdist_{G_l^k}(\varphi) = \infty$ (hence $fdist_{G_l^k}(t) = \infty$), or all $t' \in at_G(\varphi)$ are forbidden for p_k at stage l (hence t is forbidden).

- If $s_i(\psi) = -1$, $fdist_{G_i}(\psi) = \infty$ and Lemma 6.7 applies to ψ: for all p_k there exists a $j \geq 0$ such that for all $l \geq j$, either $fdist_{G_l^k}(\psi) = \infty$, or all $t' \in at_G(\psi)$ are forbidden for p_k at stage l.
  - If all $t' \in at_G(\psi)$ are forbidden for p_k at stage l, let $\hat{t}$ be the ancestor-graph of ψ which is a subgraph of t: since $\hat{t}$ is forbidden, t is also forbidden.
  - If $fdist_{G_l^k}(\psi) = \infty$, either $s_l^k(\psi) = -1$, or for all $t' \in at_G(\psi)$ there is an ancestor ψ' of ψ such that $s_l^k(\psi') = -1$ and $\psi' \in Rev_{G_l^k}(t')$.
    - Assume $s_l^k(\psi) = -1$. Since $\psi \in Rev_{G_i}(t)$, the sequential process deletes ψ when it is relevant to t. Σ' is the same as Σ except for the subdivision, but the subdivision does not affect deletions according to Theorem 3.3 or Theorem 3.4. Thus, ψ is deleted when it is relevant to t also in the distributed derivation. Let p_h and g be a first process and earliest stage such that $s_g^h(\psi) = -1$ and $\psi \in Rev_{G_g^h}(t)$. By Lemma 6.2, for all $k \neq h$, $s_l^k(\psi) = -1$ implies $\psi \in Rev_{G_l^k}(t)$ (no late contraction), and for p_h itself $\psi \in Rev_{G_q^h}(t)$ for all $q \geq g$ (no contraction undone). Thus, for all p_k and for all $l \geq j$, $\psi \in Rev_{G_l^k}(t)$ and $fdist_{G_l^k}(t) = \infty$.
    - Otherwise, let $\hat{t}$ be the ancestor-graph of ψ which is a subgraph of t: we have $s_l^k(\psi') = -1$ and $\psi' \in Rev_{G_l^k}(\hat{t})$. Thus, $\psi' \in Rev_{G_l^k}(t)$ and $fdist_{G_l^k}(t) = \infty$. □

Therefore, all ancestor-graphs pruned by $\mathcal{C}$ will be pruned by $\mathcal{C}'$ as well. This lemma does not hold without the hypothesis of immediate propagation of clauses: using the notation in the proof, we could have $fdist_{G_i}(t) = \infty$, and $fdist_{G_g^h}(t) = \infty$, but $fdist_{G_l^k}(t) \neq \infty$, because a contraction relevant at p_h is not relevant at p_k (late contraction), or $fdist_{G_l^h}(t) \neq \infty$ for some $l > g$, because ψ becomes irrelevant at p_h at a later stage (contraction undone).

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

Lemma 6.9. Assume $\mathcal{C}'$ has immediate propagation of clauses up to redundancy and no clause-duplication. If $fdist_{G_i}(\varphi) \neq \infty$ for all $i \geq 0$, there exists a stage r such that for all $i \geq r$ and $j > 0$, $pmul_{G_i}(\varphi, j) \leq mul_{G_i}(\varphi, j)$.

Proof: let $|at_G(\varphi)| = m$, and q be the number of ancestor-graphs of φ pruned by contraction in the sequential derivation. Since $fdist_{G_i}(\varphi) \neq \infty$ for all $i \geq 0$, it is $0 \leq q < m$. Let l be the stage when such contraction is completed: for all $i \geq l$ and $j > 0$, it is $0 \leq mul_{G_i}(\varphi, j) \leq m - q$. In the derivation by $\mathcal{C}'$, since the strategy has no clause-duplication, all arcs generating φ are allowed to only one process, say p_h. Let r_1 be the stage when the subdivision of the ancestor-graphs of

φ is completed: by Lemma 6.3, for all $k \neq h$, $i \geq r_1$, and $j > 0$, $mul_{G_i^k}(\varphi, j) \leq 1$. By Lemma 6.8, each of the q ancestor-graphs of φ pruned by contraction in the sequential derivation is eliminated also by p_h after a certain stage. Let r_2 be the earliest stage when all q ancestor-graphs are excluded by p_h: for all $i \geq r_2$ and $j > 0$ it is $0 \leq mul_{G_i^h}(\varphi, j) \leq m - q$. It follows that there exists a stage $r = max(r_1, r_2)$ such that for all $i \geq r$ and $j > 0$, $gmul_{G_i}(\varphi, j) \leq m - q + n - 1$, recalling that $gmul_{G_i}(\varphi, j) = \sum_{k=0}^{n-1} mul_{G_i^k}(\varphi, j)$, and counting $m-q$ as the upper bound for the multiplicity at p_h and 1 as the upper bound for the multiplicity at the remaining $n - 1$ processes. From $pmul_{G_i}(\varphi, j) = \lfloor gmul_{G_i}(\varphi, j)/n \rfloor$, we have $pmul_{G_i}(\varphi, j) \leq \lfloor (m - q + n - 1)/n \rfloor$ for all $j > 0$ and $i \geq r$. Since $m - q \geq 1$ (not all ancestor-graphs of φ are pruned), and $n \geq 2$ (there are at least two parallel processes), $(m - q + n - 1)/n < m - q + 1$, hence $\lfloor (m - q + n - 1)/n \rfloor \leq m - q$. It follows that $pmul_{G_i}(\varphi, j) \leq mul_{G_i}(\varphi, j)$ for all $j > 0$ and $i \geq r$. □

The strict disequality $pmul_{G_i}(\varphi, j) < mul_{G_i}(\varphi, j)$ does not hold in general: if $m-q = 1$ (all but one ancestor-graph are pruned by contraction), $pmul_{G_i}(\varphi, j) = mul_{G_i}(\varphi, j) = 1$. On the other hand, if $m-q > 1$, it is $(m-q+n-1)/n < m-q$, hence $\lfloor (m - q + n - 1)/n \rfloor < m - q$, and $pmul_{G_i}(\varphi, j) < mul_{G_i}(\varphi, j)$. Since the number of ancestor-graphs grows exponentially, $m \gg n$ for most clauses. Unless contraction prunes almost all ancestor-graphs of φ, $m-q$ is the dominating term in $m - q + n - 1$, and the parallel multiplicity is approximately the sequential multiplicity divided by the number of processes.

The main theorem then follows:

Theorem 6.6. If $\mathcal{C}'$ has immediate propagation of clauses up to redundancy and no clause-duplication, then $\forall j > 0$, $\exists m \geq 0$ such that $\forall i \geq m$ $pspace(G_i, j) \preceq_{mul} space(G_i, j)$.

Proof: consider all the clauses (finitely many) in $space(G_0, j)$ for any given j. For all clauses φ such that $fdist_{G_i}(\varphi) = \infty$ for some $i \geq 0$, apply Theorem 6.5 to find a stage r_φ such that for all $i \geq r_\varphi$, $pmul_{G_i}(\varphi, j) = 0$. For all clauses φ such that $fdist_{G_i}(\varphi) \neq \infty$ for all $i \geq 0$, apply Lemma 6.9 to find a stage r_φ such that $pmul_{G_i}(\varphi, j) \leq mul_{G_i}(\varphi, j)$ for all $i \geq r_\varphi$. Let $m = max\{r_\varphi \mid \varphi \in space(G_0, j)\}$. For all φ, $pmul_{G_i}(\varphi, j) \leq mul_{G_i}(\varphi, j)$ for all $i \geq m$. It follows that $pspace(G_i, j) \preceq_{mul} space(G_i, j)$ for all $i \geq m$. □

We conclude with a counterexample showing that immediate propagation of clauses is necessary to get Lemma 6.9, hence Theorem 6.6, because no clause-duplication alone does not prevent late contraction.

Example 6.5. A non-redundant clause P has three ancestor-graphs, t_1, t_2 and t_3. The sequential process prunes t_1 and t_2 by deleting relevant ancestors Q_1 and Q_2 respectively, so that the sequential multiplicity of P reduces from 3 to 1. There are two parallel processes p_1 and p_2, where only p_1 is allowed to generate P.

Assume that p_2 deletes Q_1 and Q_2 when they are relevant, while p_1 deletes them only after having used them, so that they are no longer relevant (late contraction, as in Example 6.1). By no clause-duplication, the multiplicity of P at p_2 is at most 1, but the multiplicity at p_1 remains 3, so that the parallel multiplicity is at most $(3 + 1)/2 = 2$.

On the other hand, note that immediate propagation of clauses without a sufficiently strong hypothesis on the subdivision is not sufficient either, because communication causes overlap.

Intuitively, in Theorem 6.6, a value j of the bound may represent the search depth that includes a proof. If the problem is hard enough that the sequential strategy does not succeed before stage m, the distributed strategy may face a smaller bounded search space beyond m, and therefore succeed sooner.

7. Discussion

We have applied the bounded-search-spaces approach to analyze distributed-search contraction-based strategies. This work is part of a larger effort towards a theory of strategy analysis. Such a theory would provide us with an understanding of what is complexity in theorem proving, and with mathematical tools to evaluate strategies independent of implementation.

Complexity in theorem proving is related neither to the length of the input (e.g., a shorter input set may require a longer search) nor to the length of the output (e.g., a shorter proof may require a longer search). Therefore, one needs to study the search process itself. Since first-order theorem proving is only semi-decidable, the search process is potentially infinite. Thus, the tools of classical complexity theory and algorithm analysis, which are concerned with decidable problems, are not applicable.

Studying the search process means studying a possibly non-terminating process that visits, and also modifies, an infinite search space of formulae and inferences. This problem has two facets: modelling and measuring. Our methodology employs the marked search graph for the first, and the bounded search spaces for the second one. In order to analyze distributed search we have enriched them with *subdivision* and *communication*.

For subdivision, we have used at all levels the idea of distinguishing between *allowed* and *forbidden* inferences: a distributed-search plan features a subdivision function to decide dynamically the status of the inferences, a process' derivation only includes allowed steps, the parallel marked search graph is augmented with the permission marking, and the bounded search spaces of a process only include allowed ancestor-graphs.

For communication, we have equipped a distributed strategy with communication operators, and extended the marking of the marked search graph to reflect communication steps. This aspect is more delicate than it may seem: defining

properly how receiving a clause modifies the marked search graph is critical to make the bounded search spaces sensitive to communication.

The objective of this analysis has been distributed search for contraction-based clausal strategies. The *eager contraction* property of these strategies is a problem for parallelization. A first issue is the compatibility of eager contraction and subdivision, since the subdivision may forbid contraction inferences. A suitable compromise is to subdivide generations by contraction while leaving deletions unrestricted (Theorems 3.3 and 3.4). A second issue is the relation of asynchronous communication and eager contraction. We have formalized the problem in terms of properties of the derivation, distinguishing between the local property (*local eager contraction*) and the global properties: *distributed global contraction*, supported by *propagation of clauses up to redundancy*, and *global eager contraction*, supported by *immediate propagation of clauses up to redundancy*. Since the latter is a limit property, a distributed contraction-based strategy is required to have local eager contraction and distributed global contraction.

We have analyzed first the evolution of the *bounded search spaces* of a parallel process. In a sequential derivation, expansion inferences do not modify the bounded search spaces (the strategy visits the search space without changing it), while contraction inferences reduce them (the strategy prunes the search space). For a sequential contraction-based strategy, the reductions of the bounded search spaces by contraction are permanent because contraction is *monotonic* (whatever was contracted can be contracted again) and *eager* (whatever was contracted is contracted again before being used for expansion, so that deleting it is still relevant to prune ancestor-graphs). For a distributed-search contraction-based strategy, it is still true that whatever was contracted can be contracted again, and eagerly relative to local data (Lemma 6.1), but it is not guaranteed to be eager relative to global data (*late contraction*), so that the pruning of redundant ancestor-graphs of non-redundant clauses may be undone (*contraction undone*).

Our measure may seem too sensitive here: if a clause is not redundant, why bother about whether its redundant ancestor-graphs are pruned? In order to appreciate this aspect, consider a strategy that visits all the ancestor-graphs of a non-redundant clause, generate m variants of it, and delete all but one variant by subsumption. Compare with a strategy which prunes half of the ancestor-graphs early on, without visiting them, generates $m/2$ variants, and deletes all but one of them by subsumption. The final result is the same: one variant of a non-redundant clause. However, the first strategy has done more work, hence should have higher search complexity. Our measure captures this difference, because the bounded search spaces for the first strategy will include all m ancestor-graphs, whereas those for the second strategy will include $m/2$ ancestor-graphs.

Therefore, the bounded search spaces oscillate non-monotonically in a distributed derivation, because they reflect on one hand the advantage of subdivision and contraction, and on the other hand the cost of communication. The latter emerges in two ways: undoing some of the subdivision, which is the cost of

preserving completeness in a distributed setting, and causing failures of eager contraction. Note that since it is not known how to define "time" for theorem-proving strategies, it is very important to be able to capture the cost of communication in the bounded search spaces, which are essentially a measure of space.

The next level of the analysis has been to consider all the parallel processes together. For this purpose, we have defined a notion of *overlap* of parallel searches, introduced the *parallel bounded search spaces*, and shown that *no clause-duplication* and local eager contraction minimize the overlap. Then we have compared the parallel bounded search spaces of a distributed-search contraction-based strategy with the bounded search spaces of its sequential counterpart. We have proved that distributed search preserves contraction (Theorem 6.5), but would preserve eager contraction only if propagation of clauses up to redundancy were immediate (Lemma 6.9). Under this hypothesis, and no clause-duplication, the parallel bounded search spaces are bounded by those of the sequential strategy (Theorem 6.6).

More precisely, for any depth of the search space there is a number of steps m after which the parallel bounded search space of that depth is guaranteed to be smaller than or equal to the sequential bounded search space of same depth, and will remain such for the rest of the derivation. Therefore, the distributed strategy may succeed sooner. On the other hand, if the sequential strategy solves the problem in less than m steps, it may mean that the problem is too easy for the parallelization to pay off. Similar to algorithms, also for strategies a parallel strategy or a more sophisticated strategy may not be rewarded if the input problem is too easy. In algorithm analysis, problems that are too easy are excluded from consideration because the analysis is asymptotic, that is, it captures the complexity of the algorithm as the input problem becomes harder (e.g., longer). This is not possible in this form in theorem proving, because we do not have a measure of the input instance that expresses its difficulty. Therefore, it is important to have a formulation of the theorem that may exclude instances that are too easy.

We feel that the results of our analysis are significant in a few ways:

1. Theorem 6.6 may be interpreted as a limit theorem, in a sense similar to other theoretical results obtained under an ideal assumption. It explains the nature of the problem, by indicating in the overlap and the communication-contraction node its essential aspects, and it represents a limit that strategies may approximate by improving overlap reduction and communication.

2. Since we have shown that immediate propagation is necessary to guarantee eager contraction, Theorem 6.6 may be interpreted also as a negative result on the parallelizability of contraction-based strategies. If it had been possible to prove that a contraction-based parallelization has smaller bounded search spaces without assuming immediate propagation, 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. Since this type of result does not hold, it means that a distributed-search contraction-based strategy may do better than its sequential counterpart, but it is not guaranteed to.

3. When adopting distributed search, one expects that contraction may be delayed. The trade-off is to accept this disadvantage in order to avoid synchronization (a method where parallel processes have to synchronize on every inference in order to enforce global eager contraction would be hopeless). Also, one may conjecture that the advantage of subdivision will offset the disadvantage of delayed contraction. Our analysis has shown that this conjecture does not hold (at least with the bounded search spaces as a measure), because even if the subdivision minimizes the overlap, there is a worst-case scenario where eager contraction fails (Examples 6.1 and 6.4).

In practice, this analysis is relevant to those theorem-proving problems where eager contraction is key to find a proof. While this class has never been characterized formally, the experimental record with forward-reasoning provers indicates that it is certainly not small. All contemporary forward-reasoning provers employ contraction, and while on one hand their search plans usually only approximate eager contraction as defined in Definition 3.7, on the other hand they succeed by using far more contraction than allowed by the theoretical definitions of complete inference systems. For instance, the popular theorem prover Otter [37] only approximates eager contraction, because it does backward-contraction only after all expansion children of a given clause have been generated, but it uses systematically *deletion by weight*[6], which is obviously not complete. The more recent Argonne prover EQP [38], which solved the Robbins conjecture [39], has more eager contraction than Otter, because it does backward-contraction after every generation of a clause. Because the negative result in our analysis is based on a worst-case scenario, it is not in contradiction with the fact that distributed provers implementing distributed-search contraction-based strategies may behave well, and even show super-linear speed-up, on some problems (e.g., [13,14]).

So little is known about complexity in theorem proving, and how to analyze strategies, 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. Several other issues remain to be addressed.

First, distributed search may take advantage of performing steps in different order, especially contraction steps, hence producing *different* search spaces and different proofs. For example, because of the subdivision, a process may generate

[6] Deletion by weight means deleting all clauses whose size is above a given threshold.

an important clause (e.g., a very useful simplifier) earlier than in the sequential derivation, and the early presence of this clause may change the search considerably. Thus, a direction for further research is to find other ways to compare the bounded search spaces, in order to capture this *reordering of search*.

Second, because we have chosen to analyze eager contraction, the cost of communication has been in the forefront. However, communication delays may also be beneficial, by letting a process ignoring certain clauses longer (e.g., see the considerations on the experiments in [13]). This is another direction to integrate this analysis, perhaps by using the active search space as a complementary measure.

Third, in this paper we have pursued the approach of defining parallel bounded search spaces as representatives of the distributed derivation, and compare them with the sequential ones. However, for the distributed derivation to succeed, it is sufficient that one of the parallel processes succeeds. Therefore, another direction of investigation is to study conditions such that at least one of the parallel processes does better than the sequential one. This may also lead to explore alternative definitions of parallel bounded search spaces (see the discussion in Section 5.3).

Successful applications of theorem provers, whether sequential or parallel, usually stem from multiple features, such as eager contraction, reasoning modulo a theory, restrictions of expansion, and indexing mechanisms (e.g., [39]). All such features and their interactions represent as many directions of future work in strategy analysis, including analysis of restrictions of expansion inferences, analysis of reasoning modulo a theory, analytical comparison of search plans (assuming the same inference system), and analysis of parallel approaches with multi-search.

For the latter, our parallel marked search graph is sufficiently general to cover multi-search as well, since it is sufficient to drop the permission marking and assign the processes different search plans to obtain a model of multi-search. The analysis of multi-search will require studying a *reordering of search*, similar to the one mentioned above for distributed search: for example, if Σ_1 generates φ_1 early on and φ_2 much later, and Σ_2 generates φ_2 early on and φ_1 much later, a multi-plan including both Σ_1 and Σ_2 may take advantage of generating both clauses early. On the other hand, the multi-plan may fail to reorder the search, generating one very similar to what either sequential Σ_1 or sequential Σ_2 would produce, but with the additional overhead of parallelism (e.g., communication and duplication). The extension of our framework of definitions to encompass both distributed search and multi-search has begun in [15].

Another line of research is the analysis of subgoal-reduction strategies, such as those based on model elimination, which requires to model searches with backtracking. Some of these directions may require to consider data more general than clauses (e.g., constrained clauses or other formulae), and larger inference steps (our treatment already allows to consider normalization as a single step). In this

paper we have assumed clauses, because they are common in fully-automated theorem provers, but the analysis does not depend on the clausal form, so that it can be generalized to more general formulae.

Acknowledgements

I would like to thank Claude Kirchner for inviting me to present an early version of this paper at INRIA-Lorraine and for the fruitful conversation that followed.

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, *Proc. of CTRS-90*, volume 516 of *LNCS*, pages 156–161. Springer-Verlag, 1991.

[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] M. J. Atallah, F. Dehne, R. Miller, A. Rau-Chaplin, and J. J. Tsay. Multisearch techniques: parallel data structures on a mesh-connected computer. *J. of Parallel and Distributed Computing*, 20(1):1–13, 1994.

[4] M. J. Atallah and A. Fabri. On the multisearch problem for hypercubes. *Computational Geometry: Theory and Applications*, 5, 1996.

[5] J. Avenhaus, J. Denzinger, and M. Fuchs. DISCOUNT: a system for distributed equational deduction. In J. Hsiang, editor, *Proc. of RTA-95*, volume 914 of *LNCS*. Springer, 1995.

[6] L. Bachmair and N. Dershowitz. Critical pair criteria for completion. *J. of Symbolic Computation*, 6(1):1–18, 1988.

[7] L. Bachmair and H. Ganzinger. Non-clausal resolution and superposition with selection and redundancy criteria. In A. Voronkov, editor, *Proc. of LPAR-92*, volume 624 of *LNAI*, pages 273–284. Springer-Verlag, 1992.

[8] L. Bachmair and H. Ganzinger. A theory of resolution. Technical Report MPI-I-97-2-005, Max Planck Institut für Informatik, 1997. To appear in J. A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*, Elsevier Science.

[9] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation. *Information and Computation*, 121(2):172–192, 1995.

[10] A. Bäumker, W. Dittrich, and F. Meyer auf der Heide. Truly efficient parallel algorithms: 1-optimal multisearch for and extension of the BSP model. Technical Report tr-rsfb-96-008, Dept. of Math. and Comp. Sci., Univ. of Paderborn, 1996. See also *Proc. of the 3rd European Symp. on Algorithms*, LNCS 979, pages 17–30, 1995.

[11] A. Bäumker, W. Dittrich, and A. Pietracaprina. The complexity of parallel multisearch on coarse grained machines. *Algoritmica*, 1998. Special Issue on Coarse Grained Parallel Algorithms.

[12] 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.

[13] M. P. Bonacina. Experiments with subdivision of search in distributed theorem proving. In M. Hitz and E. Kaltofen, editors, *Proc. of PASCO-97*, pages 88–100. ACM Press, 1997.

[14] M. P. Bonacina. Mechanical proofs of the Levi commutator problem. In P. Baumgartner et al., editor, *Notes of the CADE-15 Workshop on Problem Solving Methodologies with Automated Deduction*, pages 1–10, 1998.

[15] M. P. Bonacina. Ten years of parallel theorem proving: a perspective. In B. Gramlich, H. Kirchner, and F. Pfenning, editors, *Notes of the FLoC-99 Workshop on Strategies in Automated Deduction*, pages 3–15, 1999. Full version: A taxonomy of parallel strategies for deduction, Tech. Rep. 99-07, Dept. of Comp. Sci., Univ. of Iowa, May 1999.

[16] M. P. Bonacina and J. Hsiang. On subsumption in distributed derivations. *J. of Automated Reasoning*, 12:225–240, 1994.

[17] M. P. Bonacina and J. Hsiang. Parallelization of deduction strategies: an analytical study. *J. of Automated Reasoning*, 13:1–33, 1994.

[18] M. P. Bonacina and J. Hsiang. The Clause-Diffusion methodology for distributed deduction. *Fundamenta Informaticae*, 24:177–207, 1995.

[19] M. P. Bonacina and J. Hsiang. Distributed deduction by Clause-Diffusion: distributed contraction and the Aquarius prover. *J. of Symbolic Computation*, 19:245–267, 1995.

[20] M. P. Bonacina and J. Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theoretical Computer Science*, 146:199–242, 1995.

[21] M. P. Bonacina and J. Hsiang. On the modelling of search in theorem proving – towards a theory of strategy analysis. *Information and Computation*, 147:171–208, 1998.

[22] R. Bündgen, M. Göbel, and W. Küchlin. A master-slave approach to parallel term-rewriting on a hierarchical multiprocessor. In J. Calmet and C. Limongelli, editors, *Proc. of the 4th DISCO*, volume 1128 of *LNCS*, pages 184–194. Springer, 1996.

[23] R. Bündgen, M. Göbel, and W. Küchlin. Strategy-compliant multi-threaded term completion. *J. of Symbolic Computation*, 21:475–506, 1996.

[24] J. Denzinger and S. Schulz. Recording and analyzing knowledge-based distributed deduction processes. *J. of Symbolic Computation*, 21:523–541, 1996.

[25] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier, 1990.

[26] B. Fronhöfer and G. Wrightson, Eds. *Parallelization in Inference Systems*. Number 590 in LNAI. Springer-Verlag, 1990.

[27] D. Fuchs. Requirement-based cooperative theorem proving. In J. Dix, L. Fariñas del Cerro, and U. Furbach, editors, *Proc. of JELIA-98*, volume 1489 of *LNAI*, pages 139–153. Springer, 1998.

[28] M. Fuchs and A. Wolf. Cooperation in model elimination: CPTHEO. In C. Kirchner and H. Kirchner, editors, *Proc. of CADE-15*, volume 1421 of *LNAI*, pages 42–46. Springer, 1998.

[29] W. Gropp and E. Lusk. User's guide for mpich, a portable implementation of MPI. Technical Report 96/6, MCS Div., Argonne Nat. Lab., 1996.

[30] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In Th. Ottman, editor, *Proc. of the 14th ICALP*, volume 267 of *LNCS*, pages 54–71. Springer-Verlag, 1987.

[31] J. Hsiang and M. Rusinowitch. Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. *J. ACM*, 38(3):559–587, 1991.

[32] D. Kapur and H. Zhang. A case study of the completion procedure: proving ring commutativity problems. In J.-L. Lassez and G. Plotkin, Eds., editors, *Computational Logic – Essays in Honor of Alan Robinson*, pages 360–394. The MIT Press, 1991.

[33] C. Kirchner, C. Lynch, and C. Scharff. Fine-grained concurrent completion. In H. Ganzinger, editor, *Proc. of RTA-96*, volume 1103 of *LNCS*, pages 3–17. Springer, 1996.

[34] R. Kowalski. Search strategies for theorem proving. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 181–201. Edinburgh University Press, 1969.

[35] A. Leitsch. *The Resolution Calculus*. Springer, 1997.

[36] D. W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland, 1978.

[37] W. W. McCune. Otter 3.0 reference manual and guide. Technical Report 94/6, MCS Div., Argonne Nat. Lab., 1994.

[38] W. W. McCune. 33 Basic test problems: a practical evaluation of some paramodulation strategies. In R. Veroff, editor, *Automated Reasoning and its Applications: Essays in Honor of Larry Wos*, pages 71–114. MIT Press, 1997.

[39] W. W. McCune. Solution of the Robbins problem. *J. of Automated Reasoning*, 19(3):263–276, 1997.

[40] R. Niewenhuis, Rivero J. M., and M. A. Vallejo. The Barcelona prover. *J. of Automated Reasoning*, 18(2), 1997.

[41] D. A. Plaisted. Mechanical theorem proving. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence*. Elsevier, 1990.

[42] D. A. Plaisted. Equational reasoning and term rewriting systems. In D. Gabbay et al., editor, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 274–367. Oxford University Press, 1993.

[43] D. A. Plaisted and Y. Zhu. *The Efficiency of Theorem Proving Strategies*. Friedr. Vieweg & Sohns, 1997.

[44] C. B. Suttner and J. Schumann. Parallel automated theorem proving. In L. Kanal et al., editor, *Parallel Processing for Artificial Intelligence*. Elsevier, 1994.

[45] T. Tammet. Gandalf. *J. of Automated Reasoning*, 18(2):199–204, 1997.

[46] A. Urquhart. The complexity of propositional proofs. *Bull. of Symbolic Logic*, 1:425–467, 1995.

[47] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER. In M. McRobbie and J. Slaney, editors, *Proc. of CADE-13*, volume 1104 of *LNAI*, pages 141–145. Springer, 1996.

[48] A. Wolf and R. Letz. Strategy parallelism in automated theorem proving. In *Proc. of FLAIRS-98*, 1998.

[49] H. Zhang. Herky: high performance rewriting in RRL. In D. Kapur, editor, *Proc. of CADE-11*, volume 607 of *LNAI*, pages 696–700. Springer-Verlag, 1992.