Towards a unified model of search in theorem proving: subgoal-reduction strategies ⋆

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15, I-37134 Verona, Italy

Abstract

This paper advances the design of a unified model for the representation of search in first-order clausal theorem proving, by extending to *tableau-based subgoal-reduction strategies* (e.g., model-elimination tableaux), the *marked search-graph model*, already introduced for *ordering-based strategies*, those that use (ordered) resolution, paramodulation/superposition, simplification, and subsumption. The resulting *analytic marked search-graphs* subsume AND-OR graphs, and allow us to represent those *dynamic* components, such as *backtracking* and *instantiation of rigid variables*, that have long been an obstacle to model subgoal-reduction strategies properly. The second part of the paper develops for analytic marked search graphs the *bounded search spaces* approach to the analysis of *infinite* search spaces. We analyze how tableau inferences and backtracking affect the bounded search spaces during a derivation. Then, we apply this analysis to measure the effects of *regularity* and *lemmatization by folding-up* on search complexity, by comparing the bounded search spaces of strategies with and without these features. We conclude with a discussion comparing the marked search-graphs for tableaux, linear resolution and ordering-based strategies, showing how this search model applies across these classes of strategies.

Key words: Automated theorem proving, subgoal-reduction strategies, tableau-based strategies, strategy analysis, search model

⋆ Supported in part by the Ministero per l'Istruzione, l'Università e la Ricerca with grant no. 2003-097383.

Email address: mariapaola.bonacina@univr.it (Maria Paola Bonacina).
URL: http://www.sci.univr.it/∼bonacina/ (Maria Paola Bonacina).

1 Introduction

1.1 Motivation and background

The objective of this research is the construction of mathematical models of the search spaces generated by theorem-proving strategies, with the aim of improving our understanding of their complexity. Classical complexity analysis is concerned with decidable problems, and therefore does not apply to first-order theorem proving, which is semi-decidable. The lack of analytical tools to compare theorem-proving strategies makes it harder to assess progress in the field, appreciate new ideas before implementation, and understand why theorem provers succeed or fail. As the growth of methods and systems has increased the need for classification, evaluation and comparison, various efforts have clustered along four lines of research that we regard as complementary.

The most traditional one **(1)** is to consider *decidable cases* (e.g., propositional logics, the guarded fragment, decidable modal logics) and apply classical complexity analysis. In the general case, this kind of investigation involves showing that certain first-order strategies terminate, if inputs satisfy specific syntactic constraints (e.g., Fermüller et al., 1993; Fermüller and Leitsch, 1998; Fermüller et al., 2001; Armando et al., 2003; Peltier, 2003; Comon-Lundh and Courtier, 2003). A second direction **(2)** is to study the *relative complexity* of first-order calculi, usually adopting *proof length* as complexity measure. We mention (Eder, 1992), as a central reference, (Letz, 1993; Letz and Stenz, 2001b), for the application of this approach to tableau-based strategies, and (Baaz et al., 1994), on how different reductions to clausal form may cause non-elementary differences in proof length, because non-elementary differences in proof length are relevant also to proof search. A more recent pursuit **(3)** is to make *the experimental comparison* of theorem provers more systematic (e.g., Pelletier et al., 2002; Donini and Massacci, 2000a), so that it can give more information on the strengths and weaknesses of the methods implemented. A fourth approach **(4)** is to develop *mathematical models of the behavior of strategies*, that allows one to capture their essential features and establish some comparative results also in the general semi-decidable case (e.g., Plaisted and Zhu, 1997; Bonacina and Hsiang, 1998b; Bonacina, 1999a; Leitsch, 1997). The emphasis is on *search* and *search space pruning*, rather than proof length, and on differences that occur after reduction to clausal form. This paper contributes to this fourth direction by extending to *tableau-based subgoal-reduction strategies* the methodology developed in (Bonacina and Hsiang, 1998b) and (Bonacina, 1999a) for sequential and distributed ordering-based strategies, respectively.

Among refutational deduction methods, we distinguish *instance-based*, *ordering-based* and *subgoal-reduction* strategies. For simplicity, we refer to their

clausal versions, even when the methods are not necessarily clausal. *Instance-based strategies* are perhaps the oldest, as they date back to *Gilmore's multiplication method* (Chang and Lee, 1973). The idea is to implement Herbrand's theorem directly by generating ground instances of the clauses in the set to be refuted, and detecting inconsistencies at the propositional level. Contemporary strategies include *hyperlinking* (Lee and Plaisted, 1992) and *ordered semantic hyperlinking* (Plaisted and Zhu, 2000). The motivation is to avoid the *"duplication by combination"* (Plaisted and Zhu, 1997) represented by the repetition of most of the parents' literals in resolvents, and use efficient SAT checkers to guide the instance-generation process. A recent treatment can be found in (Ganzinger and Korovin, 2003).

Ordering-based methods generate clauses by *expansion inference rules*, such as resolution and paramodulation, and delete them by *contraction inference rules*, such as subsumption and simplification. They keep clauses in a database, formally a multiset, and succeed when the empty clause $\Box$ is derived. By deducing and keeping clauses, *they build implicitly many proof attempts*, and the generation of $\Box$ signals that one of them has been completed into a *proof*, given by the graph of ancestors of $\Box$. This class includes strategies that have been called at various times *resolution-based*, *rewriting-based*, *completion-based* and *saturation-based*. We proposed *"ordering-based"* since (Bonacina, 1999b) to encompass them all and underline the role of *well-founded orderings* in the definition of contraction, refinements of expansion and completeness proofs. Contraction-based strategies are ordering-based methods that apply contraction *eagerly* (e.g., Bonacina, 1999a).

Subgoal-reduction strategies select a *goal clause*, say φ_0, from the input set S, and work by reducing the goal to subgoals. In *linear resolution* (Kowalski and Kuehner, 1971; Loveland, 1972; Chang and Lee, 1973), the subgoal-reduction is done by generating resolvents; the sequence of goal clauses thus generated forms a *linear deduction* (i.e., a resolution tree shaped like a comb), and a *linear refutation* is a linear deduction of $\Box$. Linear deductions are *proof attempts* and linear refutations are *proofs*. *Model elimination* (Loveland, 1969, 1978) builds a survey of interpretations starting from φ_0 and seeks to show that none is a model of S. It can be defined on *chains* (Loveland, 1969, 1972; Chang and Lee, 1973) or *tableaux* of literals (e.g., Baumgartner and Furbach, 1993; Baumgartner and Brüning, 1997; Letz, 1998). Although independent of resolution, the chain-based version was understood as the improvement of linear resolution that made subgoal-reduction theorem-proving practical for first-order logic, by eliminating the need to keep all generated goals for ancestor-resolution. In the tableau-based presentation, the subgoal-reduction is done by steps that extend or close the branches of a tableau. Thus, open tableaux are *proof attempts* and closed tableaux are *proofs*.

Model elimination belongs to the class of *clause normal form tableaux* strate-

gies (e.g., Letz, 1998; Baumgartner and Furbach, 1998; Letz and Stenz, 2001b), that are subgoal-reduction strategies, inheriting on one hand from natural deduction methods, such as semantic or analytic tableaux (e.g., Smullyan, 1995), and on the other hand from mating-based (Andrews, 1981) and connection-based, or matrix-based, (Bibel, 1981) calculi. Among the rules of analytic tableaux, only the β-rule (for disjunction) and the γ-rule (for universally quantified variables) apply to clauses. In *ground tableaux*, the γ-rule replaces universally quantified variables by ground terms: it is not a mechanical rule, because it requires to enumerate ground terms or to guess the "right" ones. In *free-variable tableaux*, the γ-rule replaces universally quantified variables by free variables, also called parameters, that can be instantiated by most general unification. Since universal quantifiers in clauses are implicit, the γ-rule boils down to extending a branch of the tableau with a fresh copy of a clause. *Clause normal form tableaux* are free-variable tableaux, where γ-rule and β-rule are merged into a single rule called *extension*. Variables are instantiated when a branch is closed, because it contains two unifiable complementary literals, and their most general unifier (mgu) is applied to the tableau. In this context, the expression *rigid variables*, from unification theory, refers precisely to the fact that mgu's apply to the whole tableau.

In natural deduction (e.g., Smullyan, 1995), rules are seen as *analytic*, when applied bottom-up (i.e., for problem-analysis or subgoal-reduction), and as *synthetic*, when applied top-down (i.e., to generate consequences from premises). If we extend this classification to theorem-proving strategies, ordering-based strategies are *synthetic*, whereas subgoal-reduction strategies are *analytic*. The term *analytic* is also used for a finer distinction: subgoal-reduction rules are *analytic* if all formulae in the result of a reduction are instances of subformulae of existing formulae, *non-analytic* otherwise. Thus, tableau-based subgoal-reduction strategies are still *analytic*, whereas resolution-based subgoal-reduction strategies are not. We consider tableau-based strategies as *analytic subgoal-reduction strategies*, and we venture to use *synthetic subgoal-reduction strategies* for linear-resolution strategies. We shall see that this choice makes sense in terms of the respective search spaces.

1.2 Outline of contributions

1.2.1 A model of the search process for subgoal-reduction tableaux

In the first part of this paper we study the problem of *modelling the search space* of analytic subgoal-reduction strategies. Consistently with much literature in artificial intelligence and theorem proving, at least since (Kowalski, 1969), we use *search space* for the space of data, e.g., clauses, that the strategy may generate, and reserve *state space* for the space of all derivations. For

instance, for ordering-based strategies, a state of a derivation is a multiset of clauses, the state space is a graph with vertices labelled by multisets of clauses (e.g., the *I-tree* of (Bonacina and Hsiang, 1995)), and the search space is a graph with vertices labelled by clauses (e.g., the *marked search-graph* of (Bonacina and Hsiang, 1998b)). We will instantiate these notions for subgoal-reduction strategies as well. Clearly, these can be considered views of the same at different abstraction levels, with different advantages and disadvantages, that will be discussed in the paper.

It is important to observe that modelling the search space is intertwined with modelling the *search process*. This is all the more true in theorem proving, because theorem-proving strategies *modify* the search space during the search. Take again as possibly better-known example ordering-based strategies: deleting a clause by contraction modifies the search space given by the graph of all deducible clauses prior to the search. For tableau-based strategies, it is not even possible to adopt a concept of the search space, where substitutions are applied prior to the search (e.g., all deducible clauses as above), because substitutions are not applied locally to clauses, but globally to proof attempts. Thus, the strategy *modifies* the search space even by applying substitutions. For such reasons, one needs to model search space and search process together.

We begin in Section 2 by presenting tableau-based subgoal-reduction strategies, emphasizing those notions, such as *search plan*, *backtracking*, *iterative deepening*, *closure*, *fairness*, that are most relevant to modelling search. In Section 3, we develop a *modular* approach that distinguishes between *static* and *dynamic* aspects of the search. We model the former by the *structure* (i.e., vertices and hyperarcs) and the latter by the *marking* of a *marked search-graph*. We define *analytic marked search-graphs* for first-order clause normal form tableaux, including model elimination, and we give several examples showing how this model covers the various features of the strategies. The structure of the graph is determined by the analytic decomposition of clauses, while the marking represents all dynamic components, including application of substitutions to *rigid variables*, *closures* and *backtracking*. We establish a correspondence between stages of a derivation and markings of the underlying search graph, in such a way that the analytic marked search-graph associated to a stage offers a complete and accurate picture of the corresponding state of the search.

1.2.2 An approach to the measure of infinite search spaces

In Section 4, we move from modelling to *analyzing the search process* of tableaux-based subgoal-reduction strategies. What characterizes validity as a semi-decidable problem is that the search space is *infinite*. During the search, a strategy keeps in memory a finite amount of data (e.g., a set of clauses, a

tableau, one of its branches), with the possibility of generating any other such data that the inference rules can derive from the input. Our methodology is to reason in terms of both the *present* of the search (e.g., the finite amount of data held in memory), and its *future*, the infinite space of all its possible continuations. The marking of the marked search-graph allows us to define these concepts properly.

Since the future is infinite, the second major ingredient of our approach is a way of finitizing it. We define a notion of *dynamic path length*, that reflects not only the structure of the graph but also its marking, hence the actions of the strategy (e.g., backtracking). Then, we define the *bounded search space* within a certain distance, that is, the search space of all paths of the marked search-graph whose length is smaller than (or equal to) the bound. In this way, an infinite search space is reduced to an infinite succession of bounded search spaces, which are finite and can be compared in a well-founded ordering. As a basis for the comparison of strategies, we study how the different types of steps that a tableau-based strategy may perform, including both inference steps and backtracking, affect the bounded search spaces, making them smaller or larger.

1.2.3 Analysis of regularity and folding-up

In Section 5, we consider tableau-based strategies with and without *regularity check* and *lemmatization by folding-up*, and compare how their bounded search spaces evolve during derivations from the same input problem. The regularity check, also known as *identical ancestor pruning* (Astrachan and Stickel, 1992), or *equal predecessor fail* (Wallace and Wrightson, 1995), excludes tableaux with repeated literals on branches. This does not necessarily help from the point of view of proof length, since minimal closed tableaux may not be regular (Letz et al., 1994; Letz and Stenz, 2001b), but it is considered indispensable in practice for its strong search pruning effect, that was observed in experiments at least since (Letz et al., 1992). Here we contribute to explain this phenomenon by showing that applying the regularity check reduces the bounded search spaces. Such a result is relevant to comparing eventually refinements of model elimination such as (Plaisted, 1990; Baumgartner and Brüning, 1997; Baumgartner and Furbach, 1998), since not all of them are compatible with regularity.

Lemmatization, or lemmaizing, was introduced with model elimination itself (Loveland, 1969) to counter the redundancy due to repeated subgoals. The intuition is to avoid redundant search by memorizing that certain goals have already been solved. Let S be the input set of clauses, φ_0 the selected input goal clause, and T the set of all other clauses, i.e., $S = T \cup \{\varphi_0\}$. Operationally, lemmatization consists of turning solved goals into lemmas, and adding them to the consistent set T, so that they can be applied to subsequent goals.

Conceptually, lemmatization is a *meta-level rule*, because a lemma is inferred based on a fragment of the derivation, that adds *unsupported inferences*, since lemmas are logical consequences of T only (φ_0 and its descendants can be considered as the set of support) (Bonacina and Hsiang, 1998a). A different feature is *static lemmatization*, that consists of generating and adding lemmas to T by, e.g., UR-resolution, *before* the subgoal-reduction derivation starts (Schumann, 1994; Fuchs, 2000).

In early implementations, lemmatization did not help (Fleisig et al., 1974). This led to investigate various techniques inspired by lemmatization, such as *C-reduction* (Shostak, 1976), *folding-up* (Letz et al., 1994; Wallace and Wrightson, 1995; Fuchs, 2000; Letz and Stenz, 2001b), also called *backward* or *regressive merging* (Wallace and Wrightson, 1995), and *success caching* (e.g., Astrachan and Stickel, 1992; Letz et al., 1994; Bonacina and Hsiang, 1998a), or *success substitutions* (Letz and Stenz, 2001b), that, together with other features such as *failure caching* (e.g., Astrachan and Stickel, 1992; Letz et al., 1994; Bonacina and Hsiang, 1998a), or *failure substitutions* (Letz and Stenz, 2001b), have contributed to the growth of Prolog-technology (Stickel, 1992) and tableau-based theorem proving (e.g., Letz et al., 1992). While in most cases lemma generation must be heuristically controlled in order to be helpful, the analysis of search complexity in (Plaisted and Zhu, 1997) showed that unit lemmaizing and caching reduce from exponential to linear the amount of duplication in the search spaces of model elimination for problems in propositional Horn logic, and the discussion of experiments in (Astrachan and Loveland, 1997) reversed the negative judgement of (Fleisig et al., 1974). Here, we study the limited form of lemmatization traditionally called *folding-up*, or, more recently, *context unit lemmas* (Letz and Stenz, 2001b). This mechanism trades generality for efficiency, by allowing the strategy to apply non-unit lemmas like unit lemmas, provided they are applied in a restricted context. Then, we prove that folding-up reduces the bounded search spaces. Nevertheless, since the strategies under study employ backtracking, a given refinement – regularity, folding-up or other – reduces the search space only if its application is not undone. This is an intrinsic weakness of strategies that enumerate proof attempts by backtracking, and the analysis cannot but reflect it.

1.2.4 A unified framework for representing search

While it is desirable to cover all classes of strategies eventually, in this paper we study primarily *tableau-based subgoal-reduction strategies* based on clausal normal form. These strategies have "come of age," as exemplified, for instance, by the Setheo prover (Letz et al., 1992; Goller et al., 1994; Schumann, 2001), much like how the Otter prover (McCune, 1994) marked the maturity of ordering-based strategies. In Section 6, we extend our model of search to resolution-based subgoal-reduction strategies, defining *synthetic*

marked search-graphs for linear resolution. Section 6.3 compares analytic and synthetic marked search-graphs. Section 6.4 compares the synthetic marked search-graphs for linear resolution with those for ordering-based strategies from (Bonacina and Hsiang, 1998b). Thus, we have a unified framework for ordering-based strategies, analytic subgoal-reduction strategies and synthetic subgoal-reduction strategies. Section 7 compares the marked search-graph approach with other ways to model search (e.g., *AND-OR graphs*, *state space*). This leads us to discuss hybrid strategies that combine subgoal reduction and instance generation (e.g., Billon, 1996; Bierwald and Käufl, 1997; Baumgartner, 1998; Baumgartner et al., 1999; Beckert, 2003; Giese, 2001; van Eijck, 2001). The analysis of these strategies, or those based on non-normal form tableaux, is a direction for future work, considered with others in Section 8.

2 Analytic subgoal reduction: tableau-based strategies

2.1 Inference mechanism

We assume the usual basic notions in theorem proving, such as literal, clause, substitution, application of substitution, unifier and most general unifier (mgu). Let Θ be the given signature, and let $\mathbf{Lit}_\Theta$, $\mathbf{L}_\Theta$, $\mathbf{Seq}(\mathbf{Lit}_\Theta)$, and $\mathbf{T}_\Theta$, denote, respectively, the sets of literals, clauses, finite sequences of literals, and tableaux, on signature Θ. For $\Gamma = \langle L_1, \ldots, L_n \rangle \in \mathbf{Seq}(\mathbf{Lit}_\Theta)$, we use $leaf(\Gamma)$ to denote L_n, and $ancestors(\Gamma)$ to denote $\{L_1, \ldots, L_{n-1}\}$, meaning that $\{L_1, \ldots, L_{n-1}\}$ are the ancestors of L_n in Γ. With a slight abuse of notation we may also write, e.g., $L_i \in \Gamma$ for an i, $1 \leq i \leq n$. Θ includes the symbol *Goal* to be used as a dummy literal.

Definition 2.1 (theorem-proving problem) *A* (subgoal-reduction) theorem-proving problem *is given by a set of clauses* $S = T \cup \{\varphi_0\}$*, where* $\varphi_0 = Q_1 \vee \ldots \vee Q_n$ *is selected as* input goal clause *and rewritten as* $\neg Goal \vee Q_1 \vee \ldots \vee Q_n$.

Simply put, clause normal form tableaux are trees, with nodes labelled by literals. It is convenient to identify a branch with the sequence of literals that labels it, and view a tableau as a *multiset of sequences of literals* (Baumgartner and Furbach, 1994). As usual, a sequence $\Gamma \in \mathbf{Seq}(\mathbf{Lit}_\Theta)$ represents the partial Herbrand interpretation (or, equivalently, a set of Herbrand interpretations), that makes all instances of all literals in Γ true.

Definition 2.2 (initial tableau) *Given a theorem-proving problem* $S = T \cup \{\varphi_0\}$*, where* $\varphi_0 = Q_1 \vee \ldots \vee Q_n$*, the* initial tableau *for* $S = T \cup \{\varphi_0\}$*, is the multiset* $X_{\varphi_0} = \{\langle Goal, Q_i \rangle | 1 \leq i \leq n\}$.

The two basic rules to manipulate tableaux are *extension* and *mgu atomic closure*. We characterize them as inference rules that work on pairs $(S; X)$, where S is a set of clauses, and X is a tableau:

Unrestricted Extension (uExt)

$$\frac{(S \cup \{F_1 \vee \ldots \vee F_k\}; X \cup \{\langle L_1, \ldots, L_n\rangle\})}{(S \cup \{F_1 \vee \ldots \vee F_k\}; X \cup \{\langle L_1, \ldots, L_n, F_i\rangle | 1 \leq i \leq k\})}$$

A branch is *closed* if it contains two complementary literals, and *open* otherwise. A tableau is *closed* if all its branches are, and *open* otherwise. In the view of tableaux as multisets, closed branches are removed, so that a closed tableau is *empty* (denoted by $\emptyset$). We assume that applying a substitution to a sequence of literals means applying it to all literals in the sequence, and applying it to a multiset of sequences means applying it to all sequences in the multiset:

Mgu atomic closure (aClo)

$$\frac{(S; X \cup \{\langle L_1, \ldots, L_n\rangle\})}{(S; X\sigma)} \quad L_i\sigma = \neg L_j\sigma$$

where $1 \leq i, j \leq n$, and σ is the mgu of L_i and L_j.

The *weak link condition* allows an extension only if it makes it possible to close at least a branch. The *strong link condition* allows an extension only if it makes it possible to close a branch with adjacent complementary literals:

Extension with link condition

$$\frac{(S \cup \{F_1 \vee \ldots \vee F_k\}; X \cup \{\langle L_1, \ldots, L_n\rangle\})}{(S \cup \{F_1 \vee \ldots \vee F_k\}; X\sigma \cup \{\langle L_1, \ldots, L_n, F_i\rangle\sigma | 1 \leq i \leq k, i \neq m\})} L_j\sigma = \neg F_m\sigma$$

where $1 \leq m \leq k$, σ is the mgu of L_j and F_m, and $j = n$ in *extension with strong link condition (sExt)*, while $1 \leq j \leq n$, in *extension with weak link condition (wExt)*. *Model elimination tableaux* assume the strong link condition: extension with strong link condition is called *ME-extension*, while mgu atomic closure applied to non-adjacent literals is called *ME-reduction*.

Factoring, also called *forward merging* (Wallace and Wrightson, 1995), and usually implemented as *folding-down* (Goller et al., 1994; Letz and Stenz, 2001b), can be added as a refinement. Assume that the leaves of two open branches Γ and Γ' unify (i.e., $leaf(\Gamma)\sigma = leaf(\Gamma')\sigma$), and all ancestors of

$leaf(\Gamma')$ are ancestors of $leaf(\Gamma)$ (e.g., $leaf(\Gamma')$ is a sibling of an ancestor of $leaf(\Gamma)$). Then, if a closed tableau can be built under $leaf(\Gamma')\sigma$, the same closed tableau can be built under $leaf(\Gamma)\sigma$. Thus, factoring closes Γ and applies σ to the tableau:

Factoring (fClo)

$$\frac{(S; X \cup \{\Gamma, \Gamma'\})}{(S; X\sigma \cup \{\Gamma'\sigma\})} leaf(\Gamma)\sigma = leaf(\Gamma')\sigma,\ ancestors(\Gamma') \subseteq ancestors(\Gamma)$$

where σ is the mgu of $leaf(\Gamma)$ and $leaf(\Gamma')$.

In the rest of the paper, we consider the *tableau inference systems* $I_{TAB} = \{wExt, aClo\}$, $I_{MET} = \{sExt, aClo\}$, and $I_{MEF} = \{sExt, aClo, fClo\}$, and we use $I = \{ext, clo\}$ to stand for anyone of them, where ext stands for either $wExt$ or $sExt$, and clo covers $aClo$ or $fClo$.

2.2 Derivation and refutation

Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, its initial tableau $X_0 = X_{\varphi_0}$ can be reduced to different tableaux in general. A *depth-first search plan* will pick one to be X_1, and ignore the others for the time being, proceeding next to reduce X_1. A *breadth-first search plan*, on the other hand, will generate all tableaux which X_0 can be reduced to, before reducing any of them. Therefore, in depth-first search, a *state* of a derivation features *one tableau* – the most recently generated one, whereas in breadth-first search (likewise in *best-first search*) a state of a derivation features *a set of tableaux*.

Subgoal-reduction strategies implemented in state-of-the-arts provers typically use depth-first search, precisely because it keeps in memory only one proof attempt at a time. Accordingly, we choose to define the notion of *derivation* as a sequence of tableaux, as opposed to a sequence of sets of tableaux:

Definition 2.3 (derivation) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$ and a tableau inference system I, a* derivation *by I is a sequence $(S; X_0) \vdash_I \ldots (S; X_i) \vdash_I \ldots$ such that $X_0 = X_{\varphi_0}$, and $\forall i \geq 0$, X_{i+1} is generated from $(S; X_i)$ by applying a rule in I.*

Definition 2.4 (refutation) *A finite derivation $(S; X_0) \vdash_I \ldots (S; X_k)$ is a* refutation *of S by I if $X_k = \emptyset$.*

Definition 2.5 (refutational completeness) *A tableau inference system I is* refutationally complete *if, whenever $S = T \cup \{\varphi_0\}$ is unsatisfiable, and T is satisfiable, there exists a refutation of S by I.*

The above definitions can be easily generalized to sequences of sets of tableaux, if needed.

2.3 Backtracking and iterative deepening

If $X_i \neq \emptyset$ and no rule in I applies to $(S; X_i)$, there is a failure. Precisely because it develops one proof attempt at a time, depth-first search requires *backtracking* to switch to another proof attempt when the current one fails, and *iterative deepening* to retain completeness. For the purpose of modelling search, we need to make the notion of derivation more concrete by adding these features. In order to feature backtracking in the derivation, we add a counter d, whose value is incremented, whenever an inference is performed, and reset to the value of the stage the derivation is backtracking to, whenever backtracking occurs:

Definition 2.6 (derivation with backtracking) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$ and a tableau inference system I, a* derivation with backtracking *is a sequence $(S; X_0; d_0) \vdash_I \ldots (S; X_i; d_i) \vdash_I \ldots$, such that $X_0 = X_{\varphi_0}$, $d_0 = 0$, and $\forall i \geq 0$:*

- *either X_{i+1} is generated from $(S; X_i)$ by applying a rule in I, and $d_{i+1} = i{+}1$,*
- *or $X_{i+1} = X_{d_i - 1}$ and $d_{i+1} = d_i - 1$.*

The following example illustrates the usage of the counter d to represent backtracking:

Example 1 *Assume that a derivation starts with five inference steps, followed by two backtracking steps, three more inferences, and another backtracking step. The first five steps yield the state $(S; X_5; 5)$. The first backtracking step brings us back to $(S; X_4; 4)$. However, this is stage 6 in the derivation, that is, $i = 6$, $X_6 = X_4$ and $d_6 = 4$: the i counter is increased, precisely because the derivation is defined to include backtracking steps. The second backtracking step takes us back to $(S; X_3; 3)$ for $i = 7$, $X_7 = X_3$ and $d_7 = 3$. Notice how it would be wrong to capture backtracking by writing $X_{i+1} = X_{i-1}$ in Definition 2.6: such a definition would not work properly for consecutive backtracking steps, since we would have, for instance, $X_7 = X_5$! At this point the strategy selects another inference that generates $(S; X_8; 8)$: the d counter is reset to agree with the index i, because the strategy is no longer backtracking. Two more inferences lead us to $(S; X_{10}; 10)$, and the following backtracking step to $(S; X_9; 9)$ for $i = 11$, $X_{11} = X_9$ and $d_{11} = 9$.*

If the inference system included unrestricted extension, backtracking would not be necessary, even with a depth-first search plan, because if it is possible to extend any branch with a fresh copy of any clause, it is unnecessary to

undo instantiations, and therefore it is unnecessary to backtrack. However, unrestricted extension is too non-deterministic to be practical, and tableau-based strategies adopt an inference system with at least the weak link condition and a depth-first search plan with backtracking.

Iterative deepening (Korf, 1985) assumes a heuristic evaluation function: backtracking occurs if no rule applies to the current tableau (*natural failure*), or the value of the evaluation function on the current tableau is equal to the limit (*unnatural failure*):

Definition 2.7 (derivation with iterative deepening) *Given a problem* $S = T \cup \{\varphi_0\}$*, a tableau inference system* I*, a tableau evaluation function* h*, an initial limit* $k^* > 0$*, and an increment* $m > 0$ *for the limit, a* derivation with backtracking and iterative deepening *on* h *is a sequence* $(S; X_0; d_0; k_0) \vdash_I \ldots (S; X_i; d_i; k_i) \vdash_I \ldots$*, such that* $X_0 = X_{\varphi_0}$*,* $d_0 = 0$*,* $k_0 = k^*$*, and* $\forall i \geq 0$*:*

- *if* $h(X_i) < k_i$*, and at least a rule of* I *applies to* X_i*:* X_{i+1} *is generated from* $(S; X_i)$ *by applying a rule in* I*,* $d_{i+1} = i + 1$*, and* $k_{i+1} = k_i$*;*
- *otherwise:* $X_{i+1} = X_{d_i - 1}$*,* $d_{i+1} = d_i - 1$*, and*
  - $k_{i+1} = k_i$*, if* $d_i - 1 \neq 0$*,*
  - $k_{i+1} = k_i + m$*, if* $d_i - 1 = 0$*.*

The last subcase covers the situation when the search returns to the initial stage and increments the limit, because all tableaux whose heuristic value is below the limit have been tried. Possible choices of h include number of steps, number of extension steps, depth of the current tableau. For instance, Stickel's PTTP (e.g., Stickel, 1992) used IDA^*, or depth-first search with iterative deepening on an evaluation function defined, as in A^*-search [1], as the sum of a cost function and a heuristic function. The cost function captures the cost of the search effort already done, while the heuristic function estimates the cost of the search effort to be done. In PTTP, the cost function was the size of the current tableau, and the heuristic function was the number of its (open) branches, since at least this number of extensions is needed. The initial limit k^* and the increment m are fixed within a derivation, but different derivations may have different initial limit and increment. In the next section, we shall make h, k^* and m part of the search plan. This can be generalized to let the search plan change the increment during a derivation, based on some heuristic criterion. The property that ensures that all legal tableaux are explored eventually is called *fairness* and will be defined formally in Section 2.6.

[1] A^*-search itself is a best-first search procedure with no iterative deepening and no backtracking.

In order to determine uniquely the derivation generated from a given input, we need to define the notions of *search plan* and *derivation generated by a search plan*. As a search plan makes decision based on the *state* of the derivation, its definition will involve the set of all possible states, that we call *States*. Since in Definition 2.7, the *state* of a derivation is given by a set of clauses, a tableau and two natural numbers, we stipulate that *States* stand for $\mathbf{P}(\mathbf{L}_\Theta) \times \mathbf{T}_\Theta \times \mathbb{N} \times \mathbb{N}$, where $\mathbf{P}$ is powerset:

Definition 2.8 (search plan) *Given a tableau inference system I, a* depth-first search plan with iterative deepening and branch-selection function *is a tuple $\Sigma = \langle h, k^*, m, \xi_1, \zeta, \xi_2, \omega \rangle$, where:*

- $h\colon \mathbf{T}_\Theta \to \mathbb{N}$*, $k^* > 0$, and $m > 0$, are, respectively, the evaluation function, the initial limit, and the increment of the limit, for iterative deepening;*
- $\xi_1\colon States \to \mathbf{Seq}(\mathbf{Lit}_\Theta)$ *is the* branch-selection function*: $\xi_1((S; X; d; k)) = \Gamma \in X$;*
- $\zeta\colon States \times \mathbf{Seq}(\mathbf{Lit}_\Theta) \to I \cup \{backtrack\}$ *is the* rule-selection function*, which decides whether to backtrack, and returns an applicable rule $r \in I$ otherwise:*

$$\zeta((S; X; d; k), \Gamma) = \begin{cases} backtrack & \text{if no rule of } I \text{ applies to } \Gamma \\ & \text{or } h(X) = k, \\ r & \text{where } r \in I \text{ applies to } \Gamma, \text{ otherwise.} \end{cases}$$

- $\xi_2\colon States \times \mathbf{Seq}(\mathbf{Lit}_\Theta) \times I \to \mathbf{L}_\Theta \times \mathbf{Lit}_\Theta \times \mathbf{Lit}_\Theta \times \mathbf{Seq}(\mathbf{Lit}_\Theta)$ *is the* premise-selection function*:*

$$\xi_2((S; X; d; k), \Gamma, r) = \begin{cases} (\psi, F, L, \bot) & \text{where } \psi \in S, F \in \psi, L \in \Gamma, L \text{ and} \\ & \neg F \text{ unify, if } r = wExt; \\ (\psi, F, \bot, \bot) & \text{where } \psi \in S, F \in \psi, leaf(\Gamma) \text{ and} \\ & \neg F \text{ unify, if } r = sExt; \\ (\bot, L, L', \bot) & \text{where } L, L' \in \Gamma, L \text{ and } \neg L' \text{ unify,} \\ & \text{if } r = aClo; \\ (\bot, \bot, \bot, \Gamma') & \text{where } \Gamma' \in X, leaf(\Gamma) \text{ and } leaf(\Gamma') \\ & \text{unify, } ancestors(\Gamma') \subseteq ancestors(\Gamma), \\ & \text{if } r = fClo; \\ (\bot, \bot, \bot, \bot) & \text{otherwise.} \end{cases}$$

- ω: *States* $\rightarrow$ *Bool is the* termination-detection function*:*

$$\omega((S; X; d; k)) = \begin{cases} true & \text{if } X = \emptyset, \\ false & \text{otherwise.} \end{cases}$$

Searching the state space by depth-first search does not imply selecting branches in the current tableau in depth-first order; this happens under an additional hypothesis:

Definition 2.9 (depth-first branch-selection function) *A branch-selection function ξ_1 is a* depth-first branch-selection function*, if it selects the leftmost (or rightmost) longest branch.*

Definition 2.10 (tableau-based strategy) *A* tableau-based strategy *is a pair $\mathbf{C} = \langle I, \Sigma \rangle$, where I is a tableau inference system, and Σ is a depth-first search plan with iterative deepening and branch-selection function.*

Definition 2.11 (derivation generated by a strategy) *Given a problem $S = T \cup \{\varphi_0\}$ and a tableau-based strategy $\mathbf{C} = \langle I, \Sigma \rangle$, with $\Sigma = \langle h, k^*, m, \xi_1, \zeta, \xi_2, \omega \rangle$, the* derivation generated by $\mathbf{C}$ *from $S = T \cup \{\varphi_0\}$ is the sequence $(S; X_0; d_0; k_0) \vdash_{\mathbf{C}} \ldots (S; X_i; d_i; k_i) \vdash_{\mathbf{C}} \ldots$ such that $X_0 = X_{\varphi_0}$, $d_0 = 0$, $k_0 = k^*$, and $\forall i \geq 0$: if $\omega((S; X_i; d_i; k_i)) = false$, and $\xi_1((S; X_i; d_i; k_i)) = \Gamma$, then*

- *if $\zeta((S; X_i; d_i; k_i), \Gamma) = r \in I$, X_{i+1} is the tableau generated by applying r to Γ and the clause, literals or branch selected by ξ_2, $d_{i+1} = i + 1$, and $k_{i+1} = k_i$;*
- *if $\zeta((S; X_i; d_i; k_i), \Gamma) = backtrack$, $X_{i+1} = X_{d_i - 1}$, $d_{i+1} = d_i - 1$, and $k_{i+1} = k_i$, if $d_i - 1 \neq 0$, $k_{i+1} = k_i + m$, if $d_i - 1 = 0$.*

Note how, with branch selection, natural failure occurs when no rule applies to the selected branch.

2.5 Refinements: regularity and lemmatization by folding-up

Regularity affects backtracking, and therefore it is a feature of the search plan:

Definition 2.12 (irregularity) *A branch $\Gamma = \langle L_1, \ldots, L_n \rangle$ is* irregular *if $L_i = L_j$, for some i and j, $1 \leq i \neq j \leq n$,* regular*, otherwise. A tableau X is* irregular *if some $\Gamma \in X$ is,* regular*, otherwise.*

Definition 2.13 (regularity check) *A search plan Σ with rule-selection function ζ features* the regularity check*, if $\zeta((S; X; d; k), \Gamma) = backtrack$ whenever X is irregular.*

If a sub-tableau with root L is closed without using ME-reduction or factoring, it means that no model of T contains L, or $T \models \neg L$, and $\neg L$ is a *lemma*. If the sub-tableau was closed by ME-reduction steps with ancestors $A_1, \ldots, A_n$ of L, it means that no model of T containing $A_1, \ldots, A_n$ also contains L, that is, $T \models \neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$. Less frequently, if a factoring step with another leaf B was applied to close a descendant of L, no model of T contains L and $\neg B$, or $T \models \neg L \vee B$.

If a unit lemma $\neg L$ is generated, added to T, and subsequently applied to extend and close a branch containing a literal L', such that $L\sigma = L'\sigma$, no subgoals are generated. On the other hand, extension with a non-unit lemma $\neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$ would generate subgoals $\neg A_1\sigma, \ldots, \neg A_n\sigma$. Furthermore, once a non-unit lemma has been generated and added to T, it can be used also to extend any A', such that $A_i\sigma = A'\sigma$, generating subgoals $\neg L\sigma, \neg A_1\sigma \ldots, \neg A_{i-1}\sigma, \neg A_{i+1}\sigma \ldots, \neg A_n\sigma$.

Folding-up avoids the explicit addition of lemmas to T, hence preserving the subgoal-reduction character of the strategy. Lemmas are encoded within the tableau ("folded"). If a unit lemma $\neg L$ is generated, $\neg L$ is attached as an additional label to the root of the whole tableau. If a non-unit lemma $\neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$ is generated, $\neg L$ is attached as an additional label to the node of label A_n, if A_n occupies the deepest position among the $A_1, \ldots, A_n$. This encoding is consistent with reading branches as partial interpretations: if $T \models \neg L$, $\neg L$ is attached to the root because it is true in all models of T; if $T \models \neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$, $\neg L$ is attached to the node of label A_n, because it is true in all models of T where $A_1, \ldots, A_n$ are true. Then, the strategy may use the additional label $\neg L$ to close branches by mgu atomic closure or ME-reduction. This amounts to restricting the application of non-unit lemmas to a context where no subgoals need to be generated. Indeed, assume $T \models \neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$ and $\neg L$ is attached to the node of label A_n: a branch Γ that contains $A_1, \ldots, A_n$, and an L', such that $L\sigma = L'\sigma$, can be closed by using the additional label $\neg L$. If, instead of folding the lemma, Γ had been extended with $\neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$, the $\neg A_1, \ldots, \neg A_n$ would have been closed by n closure steps with the literals $A_1, \ldots, A_n$ in Γ. Since folding-up does not introduce a new inference rule, but an extended usage of ME-reduction or mgu atomic closure, we treat it as a feature of the search plan.

2.6 Closure and fairness

In order to define the search space of a theorem-proving strategy $\mathbf{C} = \langle I, \Sigma \rangle$ on an input problem S, one needs to define the domain of all the data that I can derive from S. This concept is traditionally called *deductive closure*,

and is obtained as a fixed point of an operator on sets induced by I. For instance, assume an ordering-based strategy with an inference system I made only of expansion rules, e.g., ordered resolution: then one defines the operator $\bar{I}(S) = S \cup \{f(X) \mid X \subseteq S, f \in I\}$, and the closure of S with respect to I is the fixed point $S^* = \bigcup_{k \geq 0} \bar{I}^k(S)$, where $\bar{I}^0(S) = S$ and $\bar{I}^k(S) = \bar{I}(\bar{I}^{k-1}(S))$ (e.g., Bonacina and Hsiang, 1998b). For instance, for ordered resolution, the closure contains all ordered resolvents. Such an operator is *monotonic* ($A \subseteq B$ implies $\bar{I}(A) \subseteq \bar{I}(B)$) and *increasing* ($A \subseteq \bar{I}(A)$, e.g., page 54 of (Winskel, 1994)).

This definition of $\bar{I}$ is not sufficient if the strategy features contraction, because deletions jeopardize both monotonicity and increasingness. The approach of (Leitsch, 1997) handles deletions by subsumption by a notion of "replacement sequences" in the closure (Def. 4.2.10 page 174). That of (Bonacina and Hsiang, 1998b) treats contraction in general, including both subsumption and equational simplification, by extending $\bar{I}$ to include also clauses generated by contraction, e.g, equational simplification, and handling deletions in the marked search-graph.

For subgoal-reduction strategies, the notion of closure needs to be different, because subgoal reduction does not produce the closure of a set of clauses with respect to an inference system, but rather of a goal with respect to an inference system and a set of input clauses. The operator associated to a tableau inference system, given a set S of clauses and a set M of tableaux, produces the set of tableaux derivable from those in M by using the rules in I and the clauses in S:

Definition 2.14 (operator induced by inference system) *A tableau inference system I induces the operator $\hat{I}\colon \mathbf{P}(\mathbf{L}_\Theta) \times \mathbf{P}(\mathbf{T}_\Theta) \to \mathbf{P}(\mathbf{T}_\Theta)$ such that, for any set of clauses S and set of tableaux M, $\hat{I}(S, M) = \{X \mid X$ is generated by applying some $f \in I$ to some $X' \in M$ and possibly some $\psi \in S\}$.*

Then, the closure yields the set of all tableaux that can be generated:

Definition 2.15 (deductive closure) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$ and a tableau inference system I, the* deductive closure *of φ_0 w.r.t. I and S is the set $\mathbf{T}^*(S, I, \varphi_0) = \bigcup_{k \geq 0} \hat{I}^k(S, \{X_{\varphi_0}\})$, where $\hat{I}^0(S, M) = M$ and $\hat{I}^k(S, M) = \hat{I}(S, \hat{I}^{k-1}(S, M))$ for all $k \geq 1$.*

The notion of closure allows us to define *fairness*:

Definition 2.16 (fairness) *A derivation $(S; X_0) \vdash_{\mathbf{C}} \ldots (S; X_i) \vdash_{\mathbf{C}} \ldots$ is* fair *if and only if either it is a refutation, or $\forall i \geq 0$, $\forall$ regular $X \in \hat{I}(S, \{X_i\})$, $\exists j$ such that $X_j = X$. (If $\mathbf{C}$ does not feature the regularity check, the word "regular" is dropped.) A search plan is* fair *if all the derivations that it generates are.*

Fairness ensures that all legal tableaux will be considered eventually.

Definition 2.17 (completeness) *A tableau-based strategy* $\mathbf{C} = \langle I, \Sigma \rangle$ *is* complete *if* I *is* refutationally complete *and* Σ *is* fair.

Note how $\hat{I}$ is monotonic in its second argument ($M_1 \subseteq M_2$ implies $\hat{I}(S, M_1) \subseteq \hat{I}(S, M_2)$), but *not increasing* ($M \not\subseteq \hat{I}(S, M)$). In model elimination and tableau-based strategies, at level k of the construction of the closure, $\hat{I}$ needs to add only data that can be derived from data generated at level $k-1$. Therefore, $\hat{I}$ does not need to be increasing. On the other hand, for ordering-based strategies, the operator $\bar{I}$ needs to be increasing, in order to add at level k also clauses derivable from a premise generated at level $k-1$ and a premise generated at any level $0, \ldots k-2$.

3 A model of search for tableau-based strategies

In this section we introduce the marked search-graphs for tableau-based strategies. In order to distinguish them from those for ordering-based strategies, we call them *analytic (marked) search-graphs*, and use *synthetic (marked) search-graphs* for those from (Bonacina and Hsiang, 1998b).

3.1 Analytic search-graphs

Since a tableau inference system works by decomposing clauses into literals, we represent its search space as a *hypergraph*, with vertices labelled by literals, and hyperarcs for all the extensions in the deductive closure. We give first an abstract notion of *analytic search-graph* and then we define *the analytic search-graph induced* by a tableau inference system for a theorem-proving problem.

Definition 3.1 (analytic search-graph) *An* analytic search-graph *is a hypergraph* (V, E, v_0, l)*, where* V *is the set of vertices,* E *is the set of hyperarcs,* $v_0 \in V$ *is the root, and* $l: V \rightarrow \mathbf{Lit}_\Theta$ *is a vertex-labelling function from vertices to literals.*

Definition 3.2 (induced analytic search-graph) *Given a theorem-proving problem* $S = T \cup \{\varphi_0\}$*, and a tableau inference system* I*, the* analytic search-graph induced by I for S*, denoted* $G(S, I, \varphi_0)$*, is the analytic search-graph* (V, E, v_0, l) *that satisfies the following properties:*

- $l(v_0) = Goal$*, and*
- V *and* E *are the smallest sets such that: for all tableaux* $X, X' \in T^*(S, I, \varphi_0)$*, if* X' *can be generated from* $(S; X)$*, by applying an extension inference rule*

in I to some $\psi = F_1 \vee \ldots \vee F_k \in S$ and $\Gamma = \langle L_1, \ldots, L_n \rangle \in X$, V contains vertices $v, u_1, \ldots, u_k$, such that $l(v) = L_n$, $l(u_j) = F_j$ for $1 \leq j \leq k$, and E contains a hyperarc $e = (v; u_1, \ldots, u_k)$.

For $e = (v; u_1, \ldots, u_k)$, $root(e) = v$ and $children(e) = \{u_1, \ldots, u_k\}$. Since $G(S, I, \varphi_0)$ has a hyperarc if an extension applies, link conditions on the extension rule in I affect the structure of $G(S, I, \varphi_0)$, as the next example shows:

Example 2 *Assume $S = \{\neg P \vee \neg Q, Q \vee B, Q \vee \neg P, \neg B \vee R\}$ with $\varphi_0 = \neg P \vee \neg Q$. Figure 1 shows two fragments of $G(S, I_{MET}, \varphi_0)$ (strong link condition) and $G(S, I_{TAB}, \varphi_0)$ (weak link condition) on the left and on the right, respectively. $G(S, I_{TAB}, \varphi_0)$ includes additional arcs such as* a *and* b*, closing*

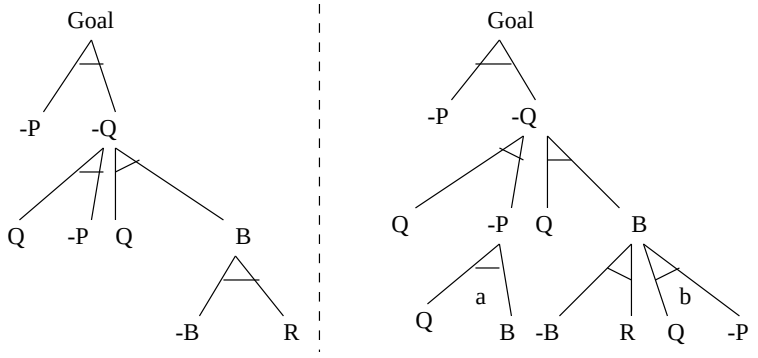

Fig. 1. Fragments of analytic search-graph with strong and weak link condition, respectively.

branches with non-adjacent complementary literals Q and $\neg Q$.

$G(S, I, \varphi_0)$ contains the structure of all possible tableaux starting with φ_0, but does not account for the dynamics of the search, namely *most general unifiers*, *closure steps* and *backtracking*. In the next section we shall define a marking for analytic search-graphs, such that the analytic marked search-graph captures these aspects as well.

3.2 Analytic marked search-graphs

Unlike in synthetic strategies, where unifiers apply locally, and the search graph contains the resolvents *prior to the search process*, in analytic strategies unifiers apply globally, so that their application is *part of the search process*. The following example illustrates this difference:

Example 3 *Assume $S = \{P(a), \neg P(x) \vee \neg Q(x), P(b), Q(b)\}$ and $\varphi_0 = \neg P(x) \vee \neg Q(x)$. Figure 2 shows the synthetic search-graph for resolution, on the left, and $G(S, I, \varphi_0)$, on the right. On the left, the substitutions $\{x \leftarrow a\}$ and $\{x \leftarrow b\}$ are applied to the resolvents $\neg Q(a)$ and $\neg Q(b)$, which label distinct*

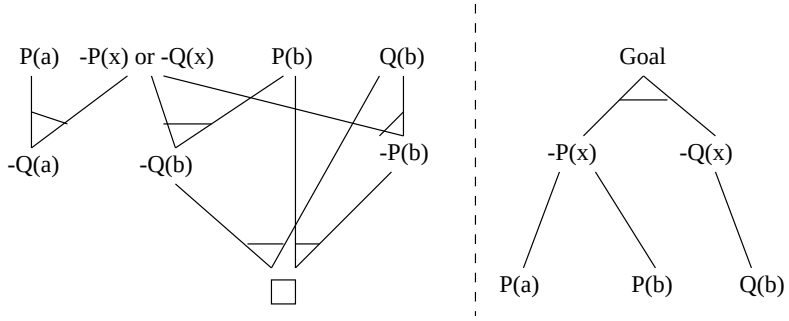

Fig. 2. Synthetic and analytic search-graphs.

vertices. In $G(S, I, \varphi_0)$, the vertices labelled by $\neg P(x)$ and $\neg Q(x)$ cannot be instantiated prior to the search, because the same vertex cannot be labelled simultaneously with, e.g., $\neg P(a)$ and $\neg P(b)$. The search spaces of synthetic and analytic strategies need to be represented in different ways.

Thus, the first component of the *marking* of an analytic search-graph will be a substitution:

Definition 3.3 (marking substitution) *A* marking substitution *for an analytic search-graph (V, E, v_0, l), with $l\colon V \to \mathbf{Lit}_\Theta$, is a substitution on signature Θ.*

Once a substitution is associated to an analytic search-graph, *only one active proof attempt* can be represented:

Example 4 *Reconsider Example 3 and Figure 2: the synthetic marked search-graph allows us to simulate a depth-first derivation that generates $\neg Q(a)$, fails, backtracks, generates $\neg Q(b)$, and then the empty clause, as well as a breadth-first derivation that generates $\neg Q(a)$, $\neg Q(b)$, $\neg P(b)$, and the empty clause, keeping in memory both proof attempts and doing no backtracking. This is impossible in the analytic marked search-graph, because it is either $\{x \leftarrow a\}$ or $\{x \leftarrow b\}$. An analytic strategy with a breadth-first search plan would generate and keep multiple tableaux, hence it would need multiple markings of the analytic search-graph.*

To model closure steps, we define first *ancestor-paths* and then *path-marking functions*:

Definition 3.4 (ancestor-path) *Given an analytic search-graph $G = (V, E, v_0, l)$, and a vertex $v \in V$, an* ancestor-path *of v is a sequence of vertices $P = \langle v_0, \ldots, v_n \rangle$, such that $\forall i$, $0 \leq i \leq n-1$, $\exists e \in E$ with $root(e) = v_i$ and $v_{i+1} \in children(e)$, and $v_n = v$. Such an e* contributes *to P, written as $e \in_c P$. $\mathbf{AP}(G)$ denotes the set of ancestor-paths in G. For $P, P' \in \mathbf{AP}(G)$, P is a* prefix *of P', written as $P <_p P'$, if $P = \langle v_0, \ldots, v_n \rangle$ and $P' = \langle v_0, \ldots, v_n, v_{n+1}, \ldots, v_m \rangle$ for vertices $v_{n+1}, \ldots, v_m$ and $m > n$. We write*

$P' >_p P$, *if* $P <_p P'$.

Definition 3.5 (path-marking function) *Given an analytic search-graph* $G = (V, E, v_0, l)$, *a* path-marking function *is a function* $b\colon \mathbf{AP}(G) \to \{open, closed\}$, *such that for all* $P, P' \in \mathbf{AP}(G)$, *if* $b(P) = closed$ *and* $P <_p P'$, *then* $b(P') = closed$.

After substitutions and closures, the third *dynamic* component is *backtracking*: in the presence of backtracking, modelling a state of the search requires to capture which proof attempt is being pursued, and which have been tried and undone. For this purpose we use a *hyperarc-marking function*:

Definition 3.6 (hyperarc-marking function) *Given an analytic search-graph* (V, E, v_0, l), *a* hyperarc-marking function *is a function* $c\colon E \to \{-1, 0, 1\}$.

We shall see that $c(e) = 1$, if e was executed, and therefore is included in the active proof attempt, $c(e) = -1$, if e was executed and undone, and $c(e) = 0$ otherwise.

Lemmatization by folding-up is also part of the dynamics of the search, and in order to model it, we add to the marking a function sl, which labels vertices with sets of literals, that will be interpreted as folded-up literals. Clearly, when we model a strategy that does not feature folding-up, $sl(v) = \emptyset$ for all vertices, so that it can be ignored. We now have all ingredients to define a *marking*, and hence an *analytic marked search-graph*:

Definition 3.7 (analytic marking) *Given an analytic search-graph* (V, E, v_0, l) *with* $l\colon V \to \mathbf{Lit}_\Theta$, *an* analytic marking *is a tuple* (σ, b, c, sl), *where* σ *is a marking substitution,* b *a path-marking function,* c *a hyperarc-marking function, and* sl *a second vertex-labelling function* $sl\colon V \to \mathbf{P}(\mathbf{Lit}_\Theta)$.

Definition 3.8 (analytic marked search-graph) *An* analytic marked search-graph *is a tuple* $(V, E, v_0, l, \sigma, b, c, sl)$, *where* (V, E, v_0, l) *is an analytic search-graph, and* (σ, b, c, sl) *is an analytic marking.*

In the next section, we shall define the analytic marked search-graph induced by a derivation.

3.3 Modelling the search generated by a derivation

An analytic marked search-graph $(V, E, v_0, l, \sigma, b, c, sl)$, where (V, E, v_0, l) is $G(S, I, \varphi_0)$, for some S, I, and φ_0, represents a state of the search for a refutation of S by I starting from φ_0. In this section we make this notion precise by defining the analytic marked search-graph corresponding to a state

of a derivation. Thus, a derivation will yield a succession of analytic marked search-graphs, that represents the search process.

Given a marking, a hyperarc can be selected if its root has not been extended otherwise, and its extension step applies under the marking substitution:

Definition 3.9 (label of ancestor-path) *Given an analytic marked search-graph* $G = (V, E, v_0, l, \sigma, b, c, sl)$*, for all* $P = \langle v_0, v_1, \ldots, v_n \rangle \in \mathbf{AP}(G)$*, let* $label_G(P) = \langle Goal, l(v_1), \ldots, l(v_n) \rangle \sigma = \langle Goal, l(v_1)\sigma, \ldots, l(v_n)\sigma \rangle$*.*

Definition 3.10 (enabled hyperarc) *Given an analytic marked search-graph* $G = (V, E, v_0, l, \sigma, b, c, sl)$*, a hyperarc* $e = (v; u_1 \ldots, u_k) \in E$*, under ancestor-path* $P = \langle v_0, \ldots, v \rangle \in \mathbf{AP}(G)$*, is* enabled*, if for all* $a \in E$*, such that* $root(a) = v$ *and* $a \neq e$*,* $c(a) \neq 1$*, and the inference of* e *applies to* $label_G(P)$*.*

The second requirement is due to the link condition and makes the status of a hyperarc dependent on the marking substitution, as the following example shows. If an enabled hyperarc is no longer such, we write that it is "disabled:"

Example 5 *Let* $I = I_{MET}$ *and* $S = \{P(x) \vee Q(x, z) \vee C(z), \neg P(f(u)) \vee C(f(u)), \neg P(y) \vee E(y), \neg Q(d, d), \ldots\}$*, where* $\varphi_0 = P(x) \vee Q(x, z) \vee C(z)$*, and* d *is a constant. In Figure 3,* $\sigma = \varepsilon$ *(the empty substitution), all hyperarcs are enabled, and all paths are open: when hyperarc* e *is fired, the only change is that*

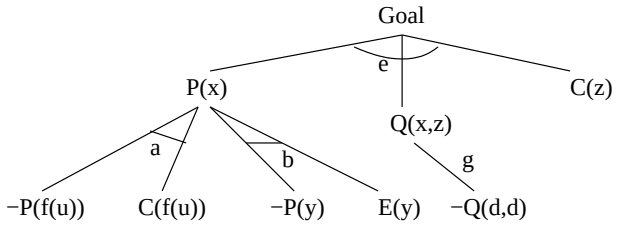

Fig. 3. Initial marking of a fragment of analytic marked search-graph.

$c(e)$ *becomes 1. If hyperarc* a *is selected next, the marking has* $\sigma = \{x \leftarrow f(u)\}$ *and* $c(a) = 1$*, the leftmost ancestor-path becomes closed, and hyperarcs* b *and* g *are disabled: hyperarc* b *is disabled, because* $P(x)$ *has been extended in another way; hyperarc* g *is disabled, because* $Q(x, z)\sigma = Q(f(u), z)$ *and* $Q(d, d)$ *do not unify, so that ME-extension does not apply. Note how* σ *affects at each step the appropriate variables, because each clause has its own variables: if hyperarc* b *were selected instead of* a*, it would be* $\sigma = \{y \leftarrow x\}$ *(the substitution may be simply a variable renaming),* $c(b) = 1$*, the third ancestor-path from the left would be closed,* a *would be disabled, but* g *would not be, because* $Q(x, z)\sigma = Q(x, z)$ *and* $Q(d, d)$ *unify.*

In the propositional case, the static structure of the graph is sufficient to capture the impact of a link condition (e.g., Example 2). In the first-order case, the applicability of an extension step depends on the substitution, which

belongs to the marking; therefore, a link condition affects both the structure of the graph and the dynamic status (enabled or not) of its hyperarcs. The strong link condition may disable more arcs than the weak link condition, and no hyperarc would become disabled because of the substitution if there were no link condition.

By adding to the marking the counter d_i of Definition 2.11, we have now all the elements to define the succession of markings induced by a derivation:

Definition 3.11 (analytic marked search-graphs: induced succession) *Let $S = T \cup \{\varphi_0\}$ be a theorem-proving problem, $\mathbf{C} =< I, \Sigma >$ a tableau-based strategy with search plan $\Sigma = \langle h, k^*, m, \xi_1, \zeta, \xi_2, \omega \rangle$, and $G(S, I, \varphi_0) = (V, E, v_0, l)$ the analytic search-graph induced by I for S. The derivation generated by $\mathbf{C}$ from $S = T \cup \{\varphi_0\}$* induces the succession of analytic marked search-graphs $\{G_i = (V, E, v_0, l, \sigma_i, b_i, c_i, sl_i, d_i)\}_{i \geq 0}$ *defined as follows. Initially, $\sigma_0 = \varepsilon$, $b_0(P) = open$ for all ancestor-paths $P \in \mathbf{AP}(G(S, I, \varphi_0))$, $c_0(a) = 0$ for all hyperarcs $a \in E$, $sl_0(v) = \emptyset$ for all vertices $v \in V$, and $d_0 = 0$. Then, for all stages $i \geq 0$:*

(1) Assume ξ_1 selects Γ, ζ selects wExt and ξ_2 returns $(\psi, F_m, L, \bot)$, where $\psi = F_1 \vee \ldots \vee F_k$, $1 \leq m \leq k$, and θ is the mgu of L and $\neg F_m$. Similarly, assume ξ_1 selects $\Gamma \in X_i$, ζ selects sExt and ξ_2 returns $(\psi, F_m, \bot, \bot)$, where $\psi = F_1 \vee \ldots \vee F_k$, $1 \leq m \leq k$, and θ is the mgu of $leaf(\Gamma)$ and $\neg F_m$. In either case, let $P' = \langle v_0, \ldots, v \rangle$ be the ancestor-path such that $label_{G_i}(P') = \Gamma$, let $e = (v; u_1, \ldots, u_k)$ be the enabled hyperarc under P' such that $l(u_j) = F_j$, for $1 \leq j \leq k$, and let $P^* = \langle v_0, \ldots, v, u_m \rangle$. Then: $\sigma_{i+1} = \sigma_i \circ \theta$,

$$b_{i+1}(P) = \begin{cases} closed \text{ if } P = P^*, \\ b_i(P) \quad otherwise, \end{cases} \qquad c_{i+1}(a) = \begin{cases} 1 \quad \text{if } a = e, \\ c_i(a) \text{ otherwise}, \end{cases}$$

and $d_{i+1} = i + 1$. Furthermore:
(a) If no lemma is derived, then $sl_{i+1} = sl_i$.
(b) If a unit lemma $\neg L$ is derived by closing P^*:

$$sl_{i+1}(v) = \begin{cases} sl_i(v) \cup \{\neg L\} \text{ if } v = v_0, \\ sl_i(v) \qquad\quad otherwise. \end{cases}$$

(c) If a non-unit lemma $\neg L \vee \neg A_1 \vee \ldots \vee \neg A_n$ is derived by closing P^*:

$$sl_{i+1}(v) = \begin{cases} sl_i(v) \cup \{\neg L\} \text{ if } l_i(v) = A_n, \\ sl_i(v) \qquad\quad otherwise, \end{cases}$$

where A_n is the deepest among the $A_1, \ldots, A_n$.

(2) *Assume* ξ_1 *selects* $\Gamma = \langle L_1, \ldots, L_n \rangle$, ζ *selects* $aClo$, ξ_2 *returns* $(\bot, L_j, L_q, \bot)$, $1 \leq j, q \leq n$, *and* θ *is the mgu of* L_j *and* $\neg L_q$. *Similarly, assume* ξ_1 *selects* Γ, ζ *selects* $fClo$, ξ_2 *returns* $(\bot, \bot, \bot, \Gamma')$, *and* θ *is the mgu of* $leaf(\Gamma)$ *and* $leaf(\Gamma')$. *In either case, let* P^* *be the ancestor-path such that* $label_{G_i}(P^*) = \Gamma$. *Then:* $\sigma_{i+1} = \sigma_i \circ \theta$, $c_{i+1} = c_i$, $d_{i+1} = i + 1$, *and*

$$b_{i+1}(P) = \begin{cases} closed \text{ if } P = P^*, \\ b_i(P) \quad otherwise. \end{cases}$$

Furthermore, sl_{i+1} *is defined as in Case 1.*

(3) *If* Σ *backtracks undoing an extension step represented by hyperarc* e:

$$c_{i+1}(a) = \begin{cases} -1 \quad \text{if } a = e, \\ c_i(a) \text{ otherwise.} \end{cases}$$

If Σ *backtracks undoing a closure,* $c_{i+1} = c_{d_i - 1}$. *For the other components,* $d_{i+1} = d_i - 1$, $\sigma_{i+1} = \sigma_{d_i - 1}$, $b_{i+1} = b_{d_i - 1}$, *and* $sl_{i+1} = sl_{d_i - 1}$ *regardless.*

In Case 1, Σ may select a hyperarc that was never tried before, whose marking goes from 0 to 1, or one that was tried and undone, and in such a case its marking goes from -1 to 1. Indeed, during a derivation, the strategy may re-consider steps that were undone in a different state of the search.

In summary, each stage $(S; X_i; d_i)$ of a derivation has its associated analytic marked search-graph $G_i = (V, E, v_0, l, \sigma_i, b_i, c_i, sl_i, d_i)$, where the hyperarcs with positive marking are in the current proof attempt (X_i), and those with non-zero marking constitute the portion of the search space visited by the strategy:

Definition 3.12 (active search space) *Given an analytic marked search-graph* $G = (V, E, v_0, l, \sigma, b, c, sl)$, *the* active search space *is the analytic marked search-graph* $G^+ = (V^+, E^+, v_0, l^+, \sigma^+, b^+, c^+, sl^+)$, *where* $E^+ = \{a \mid a \in E,\ c(a) = 1\}$, $V^+ \subseteq V$ *is the subset of vertices touched by arcs in* E^+, *and* $l^+, \sigma^+, b^+, c^+, sl^+$ *are the appropriate restrictions of* l, σ, b, c, sl *to* V^+ *and* E^+.

Definition 3.13 (visited search space) *Given an analytic marked search-graph* $G = (V, E, v_0, l, \sigma, b, c, sl)$, *the* visited search space *is the analytic marked search-graph* $G^* = (V^*, E^*, v_0, l^*, \sigma^*, b^*, c^*, sl^*)$, *where* $E^* = \{a \mid a \in E,\ c(a) \neq 0\}$, $V^* \subseteq V$ *is the subset of vertices touched by arcs in* E^*, *and* $l^*, \sigma^*, b^*, c^*, sl^*$ *are the appropriate restrictions of* l, σ, b, c, sl *to* V^* *and* E^*.

For instance, if hyperarc e is fired at the i-th step, $c_i(e) = 1$ and e becomes part of G_i^+ and G_i^*; if $c_j(e) = -1$ for some $j > i$ because of backtracking, e is not in G_j^+ but still in G_j^*. Thus, G_i^* is the search space explored *up to*

stage i, which contains both the current tableau and those that were tried and undone.

The following two examples illustrate how the analytic marked search-graph models various features, e.g., backtracking and factoring in Example 6, and ME-reduction and lemmatization in Example 7.

Example 6 *Let* $S = \{\neg Q(x) \vee \neg P(x,y), Q(b) \vee \neg P(b,f(b)), Q(k) \vee \neg P(k,k), P(k,k)\}$*, where* b *and* k *are constants, with* $I = I_{MET}$ *and* $\varphi_0 = \neg Q(x) \vee \neg P(x,y)$*. Figure 4 shows on the left the initial state of the analytic marked search-graph, where* $\sigma_0 = \varepsilon$*, all hyperarcs are enabled, and all ancestor-paths are open: assume that hyperarcs* e *and then* a *are executed as the first two*

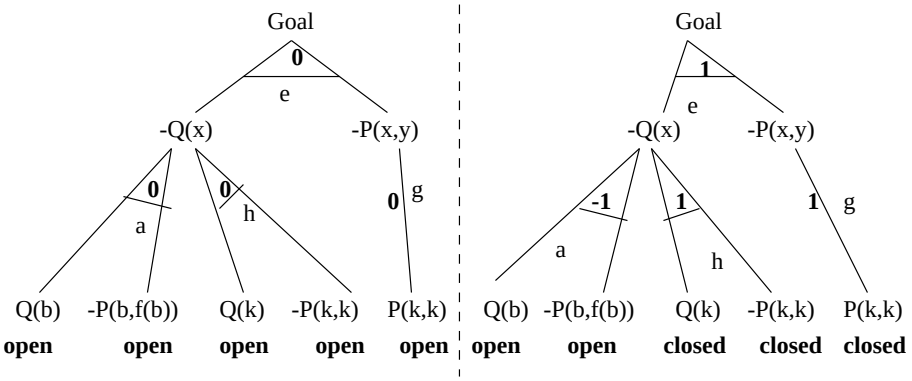

Fig. 4. Initial and final marking of $G(S, I, \varphi_0)$ for Example 6.

steps of the derivation: $\sigma_2 = \{x \leftarrow b\}$*,* $c_2(e) = c_2(a) = 1$*,* $d_2 = 2$*, the leftmost ancestor-path becomes closed, and hyperarcs* h *and* g *are disabled, because they are not compatible with* $\{x \leftarrow b\}$*. Next, a factoring step on literals* $\neg P(b, f(b))$ *and* $\neg P(x,y)\{x \leftarrow b\}$ *closes also the ancestor-path with leaf* $\neg P(b, f(b))$*, and* $\sigma_3 = \{x \leftarrow b, y \leftarrow f(b)\}$ *with* $d_3 = 3$*. At this point, however, nothing else can be done, and the strategy backtracks. First, it undoes the factoring step, so that* $d_4 = 2$*,* $\sigma_4 = \sigma_2 = \{x \leftarrow b\}$*, and the ancestor-path with leaf* $\neg P(b, f(b))$ *is reopened. Then, it undoes the extension step of hyperarc* a*, so that* $d_5 = 1$*,* $\sigma_5 = \sigma_1 = \varepsilon$*,* $c_5(a) = -1$*, and hyperarcs* h *and* g *are enabled again. After this, the strategy fires hyperarc* h*, so that* $\sigma_6 = \{x \leftarrow k\}$*,* $c_6(h) = 1$*, the ancestor-path terminating with* $Q(k)$ *is closed,* a *is disabled and* $d_6 = 6$*. The next step is a factoring inference that merges* $\neg P(k,k)$ *with* $\neg P(x,y)\sigma_6 = \neg P(k,y)$*, so that* $\sigma_7 = \{x \leftarrow k, y \leftarrow k\}$*, the ancestor-path terminating with* $\neg P(k,k)$ *is also closed and* $d_7 = 7$*. The derivation ends with the extension step of hyperarc* g *that closes the rightmost ancestor-path. The picture on the right in Figure 4 shows the final marking, where the closed tableau is made of the arcs marked* 1*.*

Example 7 *Let* $S = \{(C_1)\ L(x,a) \vee L(x,b) \vee \neg L(x,y);\ (C_2)\ L(x,f(x)) \vee \neg P(x,z);\ (C_3)\ \neg L(x,b) \vee \neg P(x,a);\ (C_4)\ \neg L(x,a) \vee \neg P(x,b);\ (C_5)\ P(x,y) \vee \neg M(y) \vee L(x,y);\ (C_6)\ M(a);\ (C_7)\ M(b);\ (\varphi_0)\ \neg L(c,a)\}$*, where* a*,* b *and* c

are constants, with $I = I_{MET}$ *and* $\varphi_0 = \neg L(c, a)$*. Figure 5 shows the marking produced by the following steps, where variables have been renamed only when necessary to keep things simple:*

(1) hyperarc e0 *with* ε*,*
(2) hyperarc e1 *(clause* C_1*) with* $\{x \leftarrow c\}$*,*
(3) hyperarc e2 *(clause* C_3*) with* $\{x \leftarrow c\}$*,*
(4) hyperarc e3 *(variant of clause* C_5*) with* $\{x \leftarrow c, w \leftarrow a\}$*,*
(5) hyperarc e4 *(clause* C_6*) with the substitution unchanged,*
(6) closure of the third leaf of e3 *(*$L(x, w)\{x \leftarrow c, w \leftarrow a\} = L(c, a)$*) by ME-reduction with* $\varphi_0 = \neg L(c, a)$*.*

At this point, the subtableau with root $\neg P(x, a)\{x \leftarrow c, w \leftarrow a\}$ *is closed, and the lemma* $P(c, a) \vee L(c, a)$ *is folded-up: this is represented by the marking* $\{P(c, a)\}$ *for the node with label* $\neg L(c, a)$ *(which happens to be* φ_0*). The third leaf of* e1 *can be extended in a few ways: assume that the strategy applies clause* C_2 *and fires arc* e5 *in Figure 5 with substitution* $\{x \leftarrow c, w \leftarrow a, y \leftarrow f(c)\}$*. Then, the second leaf of* e5 *(*$\neg P(x, z)\{x \leftarrow c, w \leftarrow a, y \leftarrow f(c)\} = \neg P(c, z)$*) can be closed by an ME-reduction step with the folded-up literal* $\{P(c, a)\}$ *from lemma* $P(c, a) \vee L(c, a)$*, yielding the final substitution* $\{x \leftarrow c, w \leftarrow a, y \leftarrow f(c), z \leftarrow a\}$*. In Figure 5 hyperarcs marked* 1 *and closed ancestor-paths identify the closed tableau, while hyperarcs with null marking (*e6 *with a variant of clause* C_1 *and* e7 *with a variant of clause* C_5*) show some alternatives. Other alternatives (e.g., under closed nodes) are not displayed for simplicity.*

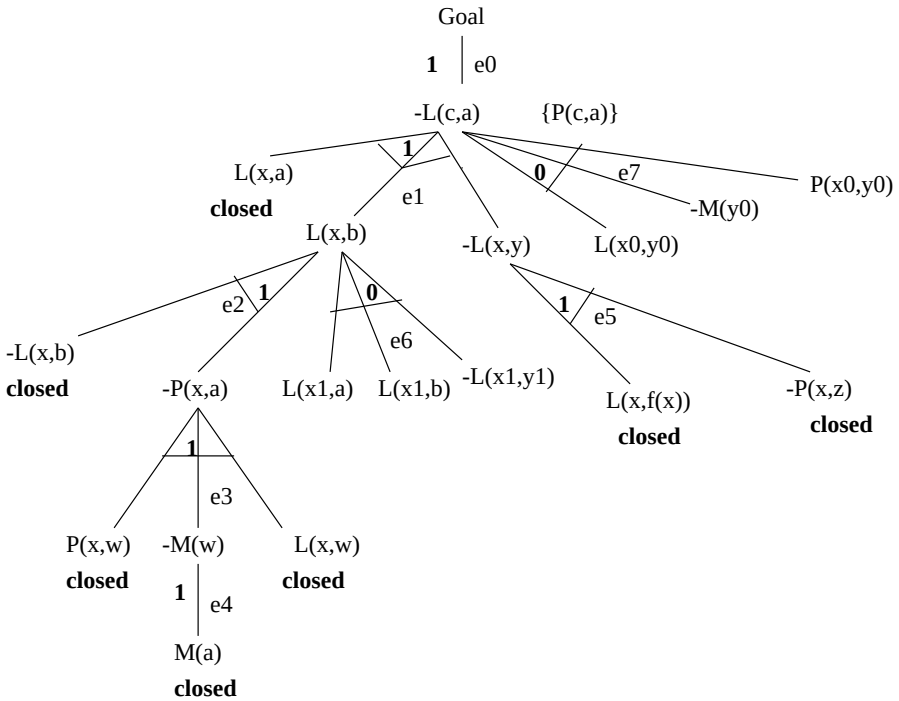

Fig. 5. Final marking of part of $G(S, I, \varphi_0)$ for Example 7.

4.1 Bounded search spaces

In order to finitize and compare search spaces, we define *bounded search spaces*, meaning the portion of the search space within a certain bound. Thus, we need to find a quantity that can be bounded. We resort to ancestor-paths, because they are *finite*, even if the analytic search-graph is infinite, as they are defined by taking a vertex in the graph and *"looking back"* from that vertex:

Definition 4.1 (length of ancestor-path) *Given an analytic marked search-graph $G = (V, E, v_0, l, \sigma, c, b, sl)$, for all ancestor-paths $P \in \mathbf{AP}(G)$, the* length of P in G *is defined as*

$$len_G(P) = \begin{cases} \infty & \text{if } \exists a \in E,\ a \in_c P,\ c(a) = -1, \\ \mid \{a \mid a \in E,\ a \in_c P\} \mid & \text{otherwise.} \end{cases}$$

The length is *infinite*, if a hyperarc in the ancestor-path has negative marking because of backtracking. Thus, infinite length captures the fact that hyperarcs below one undone by backtracking are excluded from consideration in the current state of the search. If the hyperarc failed naturally, either it has no descendants, or all its descendants already have negative marking; if the hyperarc failed unnaturally, its descendants have null marking but are ignored. If an undone hyperarc is re-considered by the strategy, its marking becomes positive and the length of the relevant ancestor-paths will be finite again.

Since tableau-based strategies work by generating and discarding interpretations, we define bounded search spaces as *multisets of partial interpretations*. They are *multisets*, because a sequence of literals may label multiple ancestor-paths:

Definition 4.2 (bounded search space) *Given an analytic marked search-graph $G = (V, E, v_0, l, \sigma, b, c, sl)$, for all $j > 0$, the* bounded search space within length j *is the multiset of sequences of literals*

$$space(G, j) = \sum_{\Gamma \in \mathbf{Seq}(\mathbf{Lit}_\Theta)} mul_G(\Gamma, j) \cdot \Gamma$$

where a multiset is written as a polynomial, with the multiplicities as coefficients, and $mul_G(\Gamma, j) = |\{P \mid P \in \mathbf{AP}(G),\ label_G(P) = \Gamma,\ b(P) = open,\ 0 \leq len_G(P) \leq j\}|$.

Thus, $space(G, j)$ contains all sequences of literals labelling open ancestor-paths within length j. It is safe to let Γ range in $\mathbf{Seq}(\mathbf{Lit}_\Theta)$, because $mul_G(\Gamma, j) = 0$ for all j, if there is no $P \in \mathbf{AP}(G)$ such that $label_G(P) = \Gamma$. If $c(e) = 1$ for all $e \in_c P$, $label_G(P)$ is a branch in the current tableau. For a sufficiently deep j, $space(G, j)$ includes the open branches of the current tableau, and all their possible continuations down to depth j, as well as the branches of all other possible tableaux, whose hyperarcs are currently disabled, because other steps have been selected, but could become enabled again upon backtracking. Ancestor-paths that have been already successfully closed or undone by backtracking are excluded, because they are not part of the search space to be considered under the current marking.

Intuitively, while G_i^+ represents the *present* and G_i^* the *present and past* of the search, $space(G_i, j)$ represents its *present and future*[2] within distance j. The "future" means the unexplored space of all future possibilities, which is infinite in general. In order to capture it finitely, we define *bounded* search spaces, and consider the succession $\{space(G_i, j)\}_{j>0}$.

4.2 Analysis of the impact of derivation steps on the bounded search spaces

We begin our analysis by studying how the cardinality of the bounded search spaces varies with the inference steps of a derivation[3]. As a preliminary remark, we note that $\mathbf{AP}(G_i) = \mathbf{AP}(G_{i+1})$ for all stages i of a derivation, since no step modifies the structure of the analytic marked search-graph. The first theorem shows that a closure makes the bounded search spaces smaller for all bounds deep enough to include the ancestor-path being closed. This matches the intuition that a closure represents a partial success that leaves a smaller search space for consideration:

Theorem 1 *If* $(S_i; X_i; d_i) \vdash_I (S_{i+1}; X_{i+1}; d_{i+1})$ *is a closure step, then* $\forall j > 0,\ |space(G_{i+1}, j)| \leq |space(G_i, j)|$, *and* $\exists n > 0,\ \forall j \geq n,\ |space(G_{i+1}, j)| < |space(G_i, j)|$.

Proof: let P^* be the ancestor-path being closed, with $len_{G_i}(P^*) = n$, and θ the substitution applied with the closure. First we show that the application of the substitution does not affect the cardinalities of the bounded search spaces, so that we need to consider only the effect of the closure. Let P be any ancestor-path such that $b_i(P) = b_{i+1}(P) = open$, and let $label_{G_i}(P) = \Gamma$. If $\Gamma \neq \Gamma\theta$, i.e., $label_{G_i}(P) \neq label_{G_{i+1}}(P)$, the multiplicity of Γ decreases by one, and that of $\Gamma\theta$ increases also by one, so that altogether there is no change. The only

[2] Note however that $space(G_i, j)$ does not count closed ancestor-paths, whereas G_i^+ includes them.

[3] Here and in the following cardinality means multiset cardinality.

effect of the step is to close and exclude from the bounded search spaces deep enough to contain them (i.e., for all $j \geq n$) all P's such that $P \geq_p P^*$. Thus, the thesis holds. □

On the other hand, when a closure is undone upon backtracking, the reduction of the bounded search spaces that it had caused is also undone:

Theorem 2 *If* $(S_i; X_i; d_i) \vdash_I (S_{i+1}; X_{i+1}; d_{i+1})$ *is a backtracking step undoing a closure, then* $\forall j > 0,\ |space(G_{i+1}, j)| \geq |space(G_i, j)|$ *and* $\exists n > 0,\ \forall j \geq n,\ |space(G_{i+1}, j)| > |space(G_i, j)|$.

Proof: by the same reasoning of the proof of Theorem 1 applied in the opposite direction. □

Theorems 1 and 2 cover *mgu atomic closure*, and hence *ME-reduction*, including steps that apply a folded-up lemma, and *factoring*. We consider next extension with link condition, either weak or strong:

Theorem 3 *Let* $(S_i; X_i; d_i) \vdash_I (S_{i+1}; X_{i+1}; d_{i+1})$ *be an extension with link condition and* e *the executed hyperarc.*

(1) If e *is executed for the first time, then* $\forall j > 0,\ |space(G_{i+1}, j)| \leq |space(G_i, j)|$, *and* $\exists n > 0,\ \forall j \geq n,\ |space(G_{i+1}, j)| < |space(G_i, j)|$.
(2) Otherwise, $\forall j > 0,\ |space(G_{i+1}, j)| \geq |space(G_i, j)|$, *where equality holds if* e *is extension by a unit clause.*

Proof: let $P = \langle v_0, \ldots, v \rangle$ be the ancestor-path being extended and $P^* = \langle v_0, \ldots, v, v' \rangle$ the ancestor-path being closed. Thus, $root(e) = v$, $v' \in children(e)$ and $e \in_c P^*$. Let $len_{G_i}(P) = n - 1$ and $len_{G_i}(P^*) = n$.
In Case 1, we have $c_i(e) = 0$ and $c_{i+1}(e) = 1$. This change of marking has no effect on the bounded search spaces, and the thesis follows from the proof of Theorem 1 for the closure.
In Case 2, the strategy re-considers e after having executed it and undone it at previous stages. Thus, it is $c_i(e) = -1$ and $c_{i+1}(e) = 1$. Let $A = \{Q \mid Q \in \mathbf{AP}(G_i) \wedge\ e \in_c Q \wedge\ (\forall a \in_c Q\ a \neq e \Rightarrow\ c_i(a) \neq -1)\}$. Recall that $c_{i+1}(a) = c_i(a)$ for all $a \neq e$. A is the set of all and only the ancestor-paths Q such that $len_{G_i}(Q) = \infty$ precisely because $c_i(e) = -1$. Since $c_{i+1}(e) = 1$, $len_{G_{i+1}}(Q) \neq \infty$ and the ancestor-paths in A get reinstated. Let $B = \{Q \mid Q \in \mathbf{AP}(G_i) \wedge\ Q \geq_p P^* \wedge\ (\forall a \in_c Q\ a \neq e \Rightarrow\ c_i(a) \neq -1)\}$. B contains the relevant ancestor-paths that get closed by closing P^*. Whether for A or B, it is irrelevant to consider a Q such that $c_{i+1}(a) = c_i(a) = -1$ for some $a \in_c Q,\ a \neq e$, because such a Q is excluded from both $space(G_i, j)$ and $space(G_{i+1}, j)$ regardless of e. Since $e \in_c P^*$, $Q \geq_p P^*$ implies $e \in_c Q$, and $B \subseteq A$. If e is an extension with a unit clause, $e = (v; v')$, $B = A$ and $\forall j > 0,\ |space(G_{i+1}, j)| = |space(G_i, j)|$. If e is an extension with a non-unit

clause, $e = (v; u_1, \ldots, u_k)$, $v' = u_m$ for some m, $1 \leq m \leq k$, $B \subset A$, and it cannot be $B = A$, because $A - B$ contains at least the ancestor-paths in the form $\langle v_0, \ldots, v, u_i \rangle$, for $1 \leq i \neq m \leq k$. Then $\forall j < n,\ |space(G_{i+1}, j)| = |space(G_i, j)|$, and $\forall j \geq n,\ |space(G_{i+1}, j)| > |space(G_i, j)|$. □

This theorem shows that when an extension with link condition is applied for the first time, the extension itself is neutral, and the overall effect is that of the associated closure (Case 1). On the other hand (Case 2), if an extension with a non-unit clause, that was formerly undone by backtracking, is executed, the bounded search spaces may grow, as ancestor-paths that had been excluded are included again. Thus, an extension with link condition does not necessarily enlarge or reduce the bounded search spaces. However, we find that such a result can be obtained for *undoing* an extension with link condition, because all ancestor-paths that are re-opened, by undoing the closure, get infinite distance and therefore remain excluded:

Theorem 4 *If* $(S_i; X_i; d_i) \vdash_I (S_{i+1}; X_{i+1}; d_{i+1})$ *undoes an extension step with link condition, then* $\forall j > 0,\ |space(G_{i+1}, j)| \leq |space(G_i, j)|$ *and, if the clause of the extension step is not a unit clause,* $\exists n > 0,\ \forall j \geq n,\ |space(G_{i+1}, j)| < |space(G_i, j)|$.

Proof: let e be the undone hyperarc, $P = \langle v_0, \ldots, v \rangle$ the ancestor-path whose extension is being undone, and $P^* = \langle v_0, \ldots, v, v' \rangle$ the ancestor-path whose closure is being undone. Thus, $root(e) = v$, $v' \in children(e)$ and $e \in_c P^*$. Let $len_{G_i}(P) = n - 1$ and $len_{G_i}(P^*) = n$.
We have $c_i(e) = 1$ and $c_{i+1}(e) = -1$. Let $A = \{Q \mid Q \in \mathbf{AP}(G_i) \wedge\ e \in_c Q \wedge\ (\forall a \in_c Q\ a \neq e \Rightarrow\ c_i(a) \neq -1)\}$. Recall that $c_{i+1}(a) = c_i(a)$ for all $a \neq e$. A is the set of all and only the ancestor-paths Q such that $len_{G_i}(Q) \neq \infty$ and $len_{G_{i+1}}(Q) = \infty$, precisely because $c_{i+1}(e) = -1$. Let $B = \{Q \mid Q \in \mathbf{AP}(G_i) \wedge\ Q \geq_p P^* \wedge\ (\forall a \in_c Q\ a \neq e \Rightarrow\ c_i(a) \neq -1)\}$. B contains the relevant ancestor-paths that get reopened by reopening P^*. Whether for A or B, it is irrelevant to consider a Q such that $c_{i+1}(a) = c_i(a) = -1$ for some $a \in_c Q,\ a \neq e$, because such a Q is excluded from both $space(G_i, j)$ and $space(G_{i+1}, j)$ regardless of e. Since $e \in_c P^*$, $Q \geq_p P^*$ implies $e \in_c Q$, and $B \subseteq A$. If e is an extension with a unit clause, $e = (v; v')$, $B = A$ and $\forall j > 0,\ |space(G_{i+1}, j)| = |space(G_i, j)|$. If e is an extension with a non-unit clause, $e = (v; u_1, \ldots, u_k)$, $v' = u_m$ for some m, $1 \leq m \leq k$, $B \subset A$, and it cannot be $B = A$, because $A - B$ contains at least the ancestor-paths in the form $\langle v_0, \ldots, v, u_i \rangle$, for $1 \leq i \neq m \leq k$. Then $\forall j > 0,\ |space(G_{i+1}, j)| \leq |space(G_i, j)|$ and $\forall j \geq n,\ |space(G_{i+1}, j)| < |space(G_i, j)|$. □

Theorems 1, 2 and 3 imply that the bounded search spaces of a tableau-based strategy are *non-monotonic* during a derivation, because of *backtracking*. This result is mitigated by the fact that when undoing a step that combines exten-

sion and closure, the bounded search spaces do not grow and some become smaller, because the reduction due to undoing the extension dominates over the growth due to undoing the closure (Theorem 4). Thus, undoing closures (e.g., ME-reduction) appears "more serious" than undoing extensions with closure (e.g., ME-extension). Informally, one can think of extension steps as developing the subgoal reduction, while closure steps are those that "wrap up" the proof by propagating constraints through unification. When the strategy extends a branch while closing one of its children, the closure is a consequence of a restriction on extension (the link condition), and the main content of the step is still a subgoal reduction. Then, intuitively, the failure of a closure could be more serious because it would represent the failure of a more advanced attempt to complete the proof. In the next section we apply these results to measure the impact of *regularity* and *lemmatization by folding-up*.

5 Comparison of tableau-based strategies

We compare the bounded search spaces of a strategy with *pruning by regularity*, and *lemmatization by folding-up*, respectively, with those of one without these features. Let $\mathbf{C}_1 = \langle I, \Sigma_1 \rangle$ and $\mathbf{C}_2 = \langle I, \Sigma_2 \rangle$ be two strategies with the same inference system I (either $I_{TAB} = \{wExt, aClo\}$ or $I_{MET} = \{sExt, aClo\}$ or $I_{MEF} = \{sExt, aClo, fClo\}$), and *fair* search plans Σ_1 and Σ_2. In the following, $(S; X_i^1; d_i^1)$ and $(S; X_i^2; d_i^2)$ denote the states of the derivations by $\mathbf{C}_1$ and $\mathbf{C}_2$, respectively, applied to the same input problem $S = T \cup \{\varphi_0\}$, and G_i^1 and G_i^2 their associated analytic marked search-graphs, respectively. Since the two strategies have the same inference system, $G_0^1 = G_0^2$.

5.1 Regularity

An ancestor-path P in marked search-graph G is irregular if $label_G(P)$ is (see Definitions 2.12 and 3.9). Let Σ_1 and Σ_2 differ only in one respect: Σ_2 features the regularity check, whereas Σ_1 does not (see Definition 2.13). Let r be the first stage of the derivations where an irregularity arises. That is, the two strategies execute the same steps up to stage r. At stage $r-1$, both strategies execute a step, either a closure or an extension with link condition, that applies some substitution σ and closes some ancestor-path P, in such a way that some other ancestor-path P^* becomes irregular, so that $X_r^1 = X_r^2$ is irregular. Then, Σ_1 ignores the irregularity and proceeds as usual, whereas Σ_2 undoes the step made to go from X_{r-1}^2 to X_r^2. If that step was a closure, either there is a different closure step, with a different substitution, that allows Σ_2 to close P, or else Σ_2 will have to backtrack the latest extension step with link condition that contributed to determining $label_{G_r^2}(P)$. If the step from X_{r-1}^2 to

X_r^2 that closed P was an extension with link condition, then that is precisely the latest extension step that contributed to determining $label_{G_r^2}(P)$. Either case, the following theorem applies:

Theorem 5 *If the regularity check induces Σ_2 to backtrack an extension step with link condition at stage m, and neither Σ_1 nor Σ_2 backtrack past stage m for the rest of the derivation (i.e., for no $k > m$, $d_k = m$), then $\exists\ k > m$ and $\exists\ n > 0$ such that $\forall i \geq k,\ \forall j \geq n,\ |space(G_i^2, j)| < |space(G_i^1, j)|$.*

Proof: let the undone extension be that of ancestor-path $P' = \langle v_0, \ldots, v \rangle$ with hyperarc e closing ancestor-path $P = \langle v_0, \ldots, v, v' \rangle$. Thus, $root(e) = v$, $v' \in children(e)$ and $e \in_c P$. Let $len_{G_i}(P') = n - 1$ and $len_{G_i}(P) = n$.
At stage m, Σ_2 undoes e, so that $c_{m+1}(e) = -1$ and $len_{G^2_{m+1}}(Q) = \infty$, for all Q such that $e \in_c Q$, including P itself. This reduces the bounded search spaces according to Theorem 4. On the other hand, Σ_1 executes some other inference step, closing some other ancestor-path Q', hence all Q such that $Q \geq_p Q'$, which may reduce the bounded search spaces according to Theorems 1 (if the step is a closure) or 3 (if the step is an extension with link condition). If the step executed by Σ_1 does *not* reduce the bounded search spaces, the thesis holds with $k = m + 1$. If it does, we need to compare the reductions. We distinguish two cases:

(1) If $e \in_c Q'$, it means that Σ_1 has executed a step in the portion of search space that Σ_2 excluded by backtracking. Although Q', and all the Q such that $Q >_p Q'$, are still open in G^2_{m+1}, they have infinite distance. Therefore, the ancestor-paths excluded by $\mathbf{C}_1$ are excluded also by $\mathbf{C}_2$, but not vice versa, so that $\forall j \geq n,\ |space(G_k^2, j)| < |space(G_k^1, j)|$ for $k = m + 1$.
(2) If $e \notin_c Q'$, it means that Σ_1 has executed a step in some other part of the search space. Thus, at stage $m + 1$, the bounded search spaces of the two strategies, for bounds large enough to include both P and Q', are uncomparable, because those of $\mathbf{C}_2$ exclude all Q such that $e \in_c Q$, and those of $\mathbf{C}_1$ exclude all Q such that $Q \geq_p Q'$. However, since Σ_1 and Σ_2 make the same choices, except for irregularity, Σ_2 closes Q' at some stage $m + l$: then $\forall j \geq n,\ |space(G_k^2, j)| < |space(G_k^1, j)|$ for $k = m + l + 1$.

The hypothesis that neither strategy backtracks past stage m means that Σ_2 will never undo the backtracking step that undid the generation of an irregular ancestor-path, and Σ_1 will never undo e and the step closing Q'. Under this hypothesis, if Σ_1 never succeeds in closing the irregular ancestor-path, the difference between the bounded search spaces at stage k will persist for all $i \geq k$. Assume that Σ_1 eventually closes the irregular ancestor-path, which may as well happen since closed irregular tableaux do exist. If this process takes x steps, Σ_2 executes other x steps, and closes at least x other ancestor-paths, since I satisfies a link condition. Thus, Σ_2 maintains the advantage it

had at stage k at all subsequent stages. □

If the search plans have a *depth-first branch-selection function* (see Definition 2.9), the difference between the two strategies appears immediately at stage $m+1$:

Theorem 6 *Assume that* Σ_1 *and* Σ_2 *have a* depth-first branch-selection function*. If the regularity check induces* Σ_2 *to backtrack an extension step with link condition at stage* m*, and neither* Σ_1 *nor* Σ_2 *backtrack past stage* m *for the rest of the derivation (i.e., for no* $k > m$*,* $d_k = m$*), then* $\exists\ n > 0$ *such that* $\forall i > m,\ \forall j \geq n,\ |space(G_i^2, j)| < |space(G_i^1, j)|$.

Proof: since the selection of branches is depth-first, at stage m, Σ_1 closes a Q' such that $e \in_c Q'$, so that Case 1 in the proof of Theorem 5 applies. □

5.2 Lemmatization by folding-up

Here we consider $\mathbf{C}_1 = \langle I, \Sigma_1 \rangle$ and $\mathbf{C}_2 = \langle I, \Sigma_2 \rangle$ with Σ_1 and Σ_2 identical, except that Σ_2 applies folding-up, whereas Σ_1 does not. Thus, they make the same choices, except when Σ_2 closes an ancestor-path by a folded-up lemma. As it is usually done in implementations, we assume that Σ_2 does not generate by folding-up a lemma that is subsumed by a unit clause in T, and gives lemma application lower priority than standard ME-reduction or mgu atomic closure.

Theorem 7 *If* Σ_2 *closes an ancestor-path by applying a folded-up lemma at stage* m*, and neither* Σ_1 *nor* Σ_2 *backtrack past stage* m *for the rest of the derivation (i.e., for no* $k > m$*,* $d_k = m$*), then* $\exists\ k > m$ *and* $\exists\ n > 0$*, such that* $\forall i \geq k,\ \forall j \geq n,\ |space(G_i^2, j)| < |space(G_i^1, j)|$.

Proof: let $P^* = \langle v_0, \ldots, v \rangle$, with $len_{G_m^1}(P^*) = len_{G_m^2}(P^*) = n$, be the ancestor-path closed by Σ_2 at stage m, by applying a folded-up lemma. This closes all ancestor-paths P such that $P \geq_p P^*$, and reduces the bounded search spaces for $\mathbf{C}_2$, according to Theorem 1. At stage m, Σ_1 executes an inference, closing some ancestor-path P', hence all P such that $P \geq_p P'$, and possibly reducing the bounded search spaces for $\mathbf{C}_1$, according to Theorem 1 (closure step) or 3 (extension step). If there is no reduction of bounded search spaces for $\mathbf{C}_1$, the thesis holds with $k = m+1$. Otherwise, we need to compare the two reductions. First, we observe that $P' \neq P^*$. If Σ_1 closes P' by a closure step, then it must be $P' \neq P^*$, because otherwise also Σ_2 would close it by a closure step without involving a lemma, since Σ_2 gives standard closure higher priority than lemma application. If Σ_1 closes P' by an extension with link condition, then either Σ_1 extends P^* and $P' >_p P^*$, or Σ_1 extends some other ancestor-path, and, in either case, it is $P' \neq P^*$. Then, we distinguish two cases:

(1) If $P^* <_p P'$, it means that Σ_1 has extended v, the leaf of P^*. By closing P^* by folding-up, Σ_2 has closed also P' and all P such that $P >_p P'$. On the other hand, we show that there is some $P >_p P^*$ that is closed in G^2_{m+1} but still open in G^1_{m+1}. Indeed, assume that v can be extended in more than one way, e.g., by two hyperarcs e_1 and e_2 and Σ_1 has fired e_1: at least all ancestor-paths P such that $e_2 \in_c P$ are open in G^1_{m+1} but closed in G^2_{m+1}. Assume that v can be extended in only one way, e.g., by executing an e with $root(e) = v$ and $children(e) = \{u_1, \ldots, u_n\}$. Say that Σ_1 has fired e and closed u_1 (i.e., $P' = \langle v_0, \ldots, v, u_1 \rangle$): all ancestor-paths with leaves $u_2, \ldots, u_n$ are open in G^1_{m+1}, but closed in G^2_{m+1}. Note that it cannot be $children(e) = \{u_1\}$, because that would mean that v is extended by a unit clause in T that subsumes the lemma applied by Σ_2, and in such a case Σ_2 would not have generated the lemma. Therefore, the ancestor-paths excluded by $\mathbf{C}_1$ are excluded also by $\mathbf{C}_2$, but not vice versa, so that $\forall j \geq n,\ |space(G^2_k, j)| < |space(G^1_k, j)|$ for $k = m + 1$.

(2) If $P^* \not<_p P'$, it means that Σ_1 has executed a step (either extension or closure) in some other part of the search space. Thus, at stage $m+1$, the bounded search spaces of the two strategies, for bounds large enough to include both P^* and P', are uncomparable, because those of $\mathbf{C}_2$ exclude all P such that $P \geq_p P^*$, and those of $\mathbf{C}_1$ exclude all P such that $P \geq_p P'$. However, since Σ_1 and Σ_2 make the same choices, except for folding-up, Σ_2 closes P' at some stage $m+l$, possibly after l applications of lemmas. Then, $\forall j \geq n,\ |space(G^2_k, j)| < |space(G^1_k, j)|$ for $k = m + l + 1$.

The hypothesis that neither strategy backtracks past stage m means that Σ_2 will never undo the closing of P^* by the folded-up lemma, and Σ_1 will never undo the step closing P'. Since Σ_1 is fair, and lemmatization does not change the power of the inference system, Σ_1 will close eventually P^*. However, Σ_1 will do so by closing a whole sub-tableau with root v. Assume that this process takes x steps. While Σ_1 closes this subtableau below v, Σ_2 executes other x steps, closing at least x other ancestor-paths, since I satisfies a link condition. Thus, Σ_2 maintains the advantage it had at stage k at all subsequent stages. □

Note how it makes no difference whether the applied lemma is a unit or non-unit lemma, because the essence of lemmatization by folding-up is precisely to restrict the usage of non-unit lemmas in such a way that they apply like unit lemmas.

Theorem 8 *Assume that Σ_1 and Σ_2 have a depth-first branch-selection function. If Σ_2 closes an ancestor-path by applying a folded-up lemma at stage m, and neither Σ_1 nor Σ_2 backtrack past stage m for the rest of the derivation (i.e., for no $k > m$, $d_k = m$), then $\exists\ n > 0$ such that $\forall i > m,\ \forall j \geq n,\ |space(G^2_i, j)| < |space(G^1_i, j)|$.*

Proof: since the selection of branches is depth-first, at stage m, Σ_1 closes a P' such that $P' >_p P^*$, so that Case 1 in the proof of Theorem 7 applies. □

Theorems 7 and 8 compare the bounded search spaces of the two strategies at the same stage. This formulation is advantageous, because the theorems apply regardless of whether the derivations terminate. If they do terminate, the comparison of the bounded search spaces at the same stage i makes sense as long as both strategies are running, that is, for $i \leq h$, if the strategy that terminates first does so at stage h. The following example shows an instance of the behavior analyzed in Theorem 7 in case of terminating derivations:

Example 8 *Let $I = I_{MET}$ and $S = \{P(x) \vee \neg Q(x), Q(a) \vee C, Q(b) \vee D, R(a,a) \vee O, \neg P(y) \vee \neg Q(z) \vee \neg R(y,z), \ldots\}$, where a and b are constants, C, D and O are disjunctions of literals, and φ_0 is $\neg P(y) \vee \neg Q(z) \vee \neg R(y,z)$. Figure 6 shows the markings induced by the two strategies $\mathbf{C}_1 = \langle I, \Sigma_1 \rangle$ and $\mathbf{C}_2 = \langle I, \Sigma_2 \rangle$. Both*

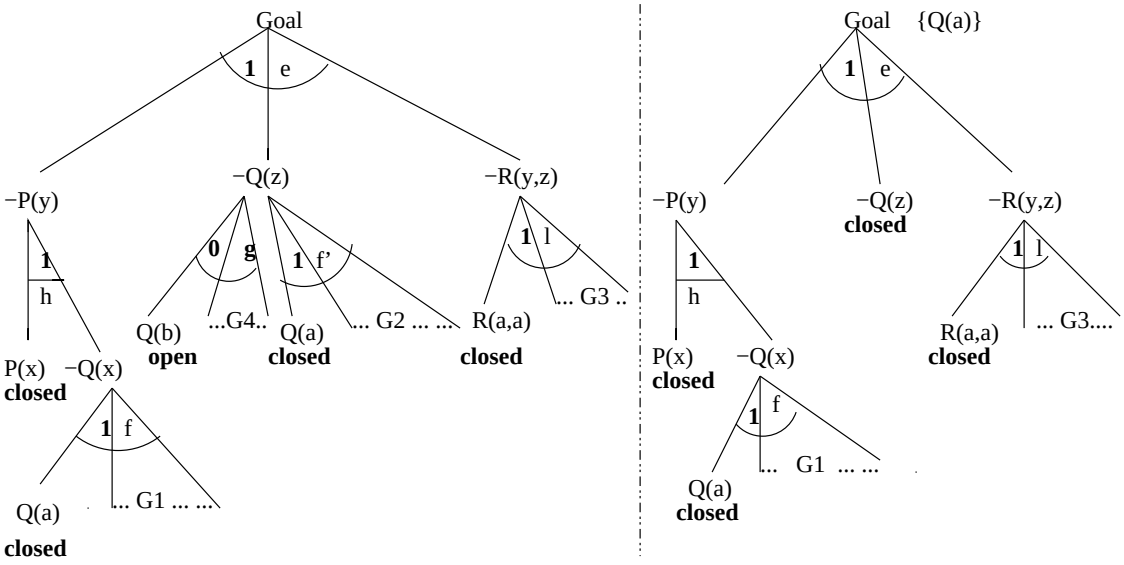

Fig. 6. The markings of $G(S, I, \varphi_0)$ for $\mathbf{C}_1$ (on the left) and $\mathbf{C}_2$ (on the right) in Example 8.

strategies $\mathbf{C}_1$ and $\mathbf{C}_2$ start by firing hyperarcs e, h, and f, in this order, closing the two leftmost ancestor-paths, and generating the marking substitution $\sigma_1 = \{y \leftarrow x, x \leftarrow a\}$. This corresponds to steps 1, 2 and 3 in both derivations. Then, both strategies explore subgraph $G1$ to solve the subgoals in C. Assume no lemmaizing occurs, so that $\mathbf{C}_1$ and $\mathbf{C}_2$ execute the same steps and find the same closed tableau for $\neg Q(x)\sigma_1 = \neg Q(a)$. Say this takes n_1 steps, so that we are at stage $3 + n_1$. Since this stage, the two strategies behave differently.
Upon closing the tableau for $\neg Q(x)\sigma_1 = \neg Q(a)$, $\mathbf{C}_2$ has folded-up lemma $Q(a)$, so that $sl(v_0) = \{Q(a)\}$ (we assume for simplicity that the tableau gets closed without ME-reductions, so that the lemma is a unit lemma). At stage $3 + n_1$, $\mathbf{C}_2$ closes the ancestor-path ending with $\neg Q(z)$ by using lemma $Q(a)$, yielding marking substitution $\sigma_2 = \{y \leftarrow x, x \leftarrow a, z \leftarrow a\}$. This excludes from the

bounded search spaces f' and $G2$, g and $G4$ (of course, they are in $G(S, I, \varphi_0)$: they are omitted in the right half of Figure 6 only to save space). Then, $\mathbf{C}_2$ executes hyperarc l and traverses $G3$ to solve the subgoals in O and build a closed tableau for $\neg R(y, z)\sigma_2 = \neg R(a, a)$. Assuming this takes n_3 steps, $\mathbf{C}_2$ terminates successfully at stage $h_2 = n_1 + n_3 + 5$ (n_1 steps in $G1$, n_3 steps in $G3$, and 5 steps for e, h, f, application of the lemma, and l).
Let us now consider $\mathbf{C}_1$. At stage $3+n_1$, $\mathbf{C}_1$ fires f', reaching the same marking substitution $\sigma_2 = \{y \leftarrow x, x \leftarrow a, z \leftarrow a\}$. Then, it searches $G2$ to solve again the subgoals in C and generate a closed tableau for $\neg Q(z)\sigma_2 = \neg Q(a)$. Assume this takes n_2 steps. Finally, it fires l and traverses $G3$ to build a closed tableau for $\neg R(y, z)\sigma_2 = \neg R(a, a)$ exactly like $\mathbf{C}_2$. Thus, $\mathbf{C}_1$ terminates successfully at stage $h_1 = n_1 + n_2 + n_3 + 5$ (n_1 steps in $G1$, n_2 steps in $G2$, n_3 steps in $G3$, and 5 steps for e, h, f, f', and l).
The stage m of Theorem 7 is $3+n_1$: for all stages i, such that $3+n_1 < i \leq h_2$, the bounded search spaces of $\mathbf{C}_2$ are strictly smaller than those of $\mathbf{C}_1$, because they do not include anything below $\neg Q(z)$ (i.e., f' and $G2$, g and $G4$), whereas those of $\mathbf{C}_1$ do.

The reduction of the bounded search spaces is not guaranteed to occur for all derivations: it occurs for those derivations where the application of folding-up is not undone by backtracking. This is an hypothesis on the derivation, not on the strategy: backtracking does not happen, but it is not disallowed. If backtracking undoes the application of the lemma, the behavior of $\mathbf{C}_2$ collapses on that of $\mathbf{C}_1$, since the two strategies are otherwise identical:

Example 9 *Let us modify Example 8 by replacing clause $R(a, a) \vee O$ with clause $R(a, b) \vee O$: $S = \{P(x) \vee \neg Q(x), Q(a) \vee C, Q(b) \vee D, R(a, b) \vee O, \neg P(y) \vee \neg Q(z) \vee \neg R(y, z), \ldots\}$. After applying the lemma and generating marking substitution $\sigma_2 = \{y \leftarrow x, x \leftarrow a, z \leftarrow a\}$, as in Example 8, $\mathbf{C}_2$ finds that $R(a, b)$ and $\neg R(y, z)\sigma_2 = \neg R(a, a)$ do not unify. Thus, $\mathbf{C}_2$ undoes the application of the lemma and behaves like $\mathbf{C}_1$: fire f', close a tableau in $G2$, fail, backtrack, execute g, yielding marking substitution $\sigma_3 = \{y \leftarrow x, x \leftarrow a, z \leftarrow b\}$, visit subgraph $G4$, to find a closed tableau for $\neg Q(z)\sigma_3 = \neg Q(b)$, fire l, and traverse $G3$ to close a tableau for $\neg R(y, z)\sigma_3 = \neg R(a, b)$.*

6 Completing the picture: resolution-based subgoal-reduction

It is well-known that there exists a close correspondence between *linear resolution* (Kowalski and Kuehner, 1971) and model elimination with chains (Loveland, 1969), on one hand, and tableaux, on the other (Baumgartner and Furbach, 1993). The goal clauses of linear resolution and the chains of model elimination can be read as a special form of tableaux representation, where one branch at a time is represented. The boxed literals in ordered linear resolu-

tion (Chang and Lee, 1973), or the A-literals in model elimination, are exactly the literals from the branch represented by the clause, or chain. Symmetrically, these strategies can be simulated by a special kind of tableaux, called *ferns* in (Baumgartner and Furbach, 1993), where closed branches are omitted, leaves are labelled by B-literals (or plain literals) and inner nodes are labelled by A-literals (or boxed literals). The interested reader can find in (Baumgartner and Furbach, 1993) a full treatment.

The purpose of this section is not to pursue the above correspondence, e.g., by studying the analytic marked search-graphs for ferns. If we were to take this route, the result would be to have analytic marked search-graphs for tableaux, with linear resolution and model elimination on chains as special cases, on one hand, and synthetic marked search-graphs for ordering-based strategies (Bonacina and Hsiang, 1998b) on the other hand, with no connection. The purpose of this section is to use linear resolution to bridge to some extent the separation between synthetic and analytic marked search-graphs, and also illustrate their differences. Thus, we view linear resolution as a method that works *by generating clauses* – like ordering-based strategies – and therefore has a search space made of clauses, represented by a *synthetic search-graph*. However, only clauses that belong to linear deductions are admitted, and depth-first search and backtracking are represented in the same way as for tableaux-based strategies, as we shall see in the rest of the section.

6.1 Linear resolution

Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, linear-resolution strategies are subgoal-reduction strategies, that search for a linear refutation of S starting from φ_0. The basic rules to build linear refutations are *binary resolution* and *factoring*, applied to triples $(S; \varphi; A)$, where S is a set of clauses, φ is the goal clause and A the set of its goal ancestors:

Input Resolution (iRes)

$$\frac{(S \cup \{\psi\}; \varphi; A)}{(S \cup \{\psi\}; \varphi'; A \cup \{\varphi\})}$$

Ancestor Resolution (aRes)

$$\frac{(S; \varphi; A \cup \{\psi\})}{(S; \varphi'; A \cup \{\psi, \varphi\})}$$

where, for both rules, $\varphi' = ((\psi \setminus \{F\}) \cup (\varphi \setminus \{L\}))\sigma$ for literals $L \in \varphi,\ F \in \psi$ and mgu σ such that $L\sigma = \neg F\sigma$;

Factoring (fact)

$$\frac{(S; \varphi; A)}{(S; \varphi'; A \cup \{\varphi\})}$$

where $\varphi' = (\varphi \setminus \{L'\})\sigma$ for literals $L, L' \in \varphi$ and mgu σ such that $L\sigma = L'\sigma$.

Thus, the inference system for linear-resolution strategies is $I_{LR} = \{iRes, aRes, fact\}$. We assume, however, that all factors of clauses in S are added to S, and S is inter-reduced with respect to tautology deletion, subsumption and clausal simplification, during pre-processing.

The considerations on depth-first search made for tableaux apply to linear resolution as well: a search plan other than depth-first would involve generating and keeping many proof attempts (i.e., many linear deductions), and typical strategies employ depth-first search, building explicitly one linear deduction at a time:

Definition 6.1 (LR-derivation) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, an* LR-derivation *is a sequence $(S; \varphi_0; A_0) \vdash_{I_{LR}} \ldots (S; \varphi_i; A_i) \vdash_{I_{LR}} \ldots$ such that $A_0 = \emptyset$, and $\forall i \geq 0$, φ_{i+1} and A_{i+1} are generated from $(S; \varphi_i; A_i)$ by applying a rule in I_{LR}.*

Definition 6.2 (LR-refutation) *A finite LR-derivation $(S; \varphi_0; A_0) \vdash_{I_{LR}} \ldots (S; \varphi_k; A_k)$ is an* LR-refutation *if $\varphi_k = \Box$.*

I_{LR} is *refutationally complete* (e.g., Chang and Lee, 1973) since, whenever $S = T \cup \{\varphi_0\}$ is unsatisfiable, and T is satisfiable, there exists a refutation of S by I_{LR}. If φ_i has neither input-resolvents nor ancestor-resolvents nor factors, the strategy backtracks (*natural failure*):

Definition 6.3 (LR-derivation with backtracking) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, an* LR-derivation with backtracking *is a sequence $(S; \varphi_0; A_0; d_0) \vdash_{I_{LR}} \ldots (S; \varphi_i; A_i; d_i) \vdash_{I_{LR}} \ldots$, such that $A_0 = \emptyset$, $d_0 = 0$, and $\forall i \geq 0$:*

- *φ_{i+1} and A_{i+1} are generated from $(S; \varphi_i; A_i)$ by applying a rule in I_{LR}, and $d_{i+1} = i + 1$,*
- *or $\varphi_{i+1} = \varphi_{d_i - 1}$, $A_{i+1} = A_{d_i - 1}$ and $d_{i+1} = d_i - 1$.*

Under iterative deepening, backtracking also occurs for *unnatural failure*:

Definition 6.4 (LR-derivation with iterative deepening) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, a deduction evaluation function h, an initial limit $k^* > 0$, and an increment $m > 0$ for the limit, an* LR-derivation with backtracking and iterative deepening *on h is a sequence $(S; \varphi_0; A_0; d_0; k_0) \vdash_{I_{LR}} \ldots (S; \varphi_i; A_i; d_i; k_i) \vdash_{I_{LR}} \ldots$, such that $A_0 = \emptyset$, $d_0 = 0$, $k_0 = k^*$, and $\forall i \geq 0$:*

- *if* $h((A_i;\varphi_i)) < k_i$ *and at least a rule of* I_{LR} *applies to* φ_i*:* φ_{i+1} *and* A_{i+1} *are generated from* $(S;\varphi_i;A_i)$ *by applying a rule in* I_{LR}*,* $d_{i+1} = i+1$ *and* $k_{i+1} = k_i$*;*
- *otherwise:* $\varphi_{i+1} = \varphi_{d_i-1}$*,* $A_{i+1} = A_{d_i-1}$*,* $d_{i+1} = d_i - 1$*, and* $k_{i+1} = k_i$*, if* $d_i - 1 \neq 0$*,* $k_{i+1} = k_i + m$*, if* $d_i - 1 = 0$*.*

If φ_i is a tautology, or is subsumed by a clause in S or A_i, the strategy also backtracks. Unlike ordering-based strategies, where subsumption and tautology deletion are *contraction rules* that contracts a set of clauses and belong to the inference system, in linear resolution they represent *conditions for backtracking* implemented by the search plan. Let $States_{LR}$ stand for $\mathbf{P}(\mathbf{L}_\Theta) \times \mathbf{L}_\Theta \times \mathbf{P}(\mathbf{L}_\Theta) \times \mathbb{N} \times \mathbb{N}$:

Definition 6.5 (LR-search plan) *A* depth-first search plan with iterative deepening and literal-selection function *is a tuple* $\Sigma = \langle h, k^*, m, \xi_1, \zeta, \xi_2, \omega\rangle$*, where:*

- $h\colon \mathbf{L}_\Theta \to \mathbb{N}$*,* $k^* > 0$*, and* $m > 0$*, are, respectively, the evaluation function, the initial limit, and the increment of the limit, for iterative deepening;*
- $\xi_1\colon States_{LR} \to \mathbf{Lit}_\Theta$ *is the* literal-selection function*:* $\xi_1((S;\varphi;A;d;k)) = L \in \varphi$*;*
- $\zeta\colon States_{LR} \times \mathbf{Lit}_\Theta \to I_{LR} \cup \{backtrack\}$ *is the* rule-selection function*, which decides whether to backtrack, and returns an applicable rule* $r \in I_{LR}$ *otherwise:*

$$\zeta((S;\varphi;A;d;k),L) = \begin{cases} backtrack & \text{if } \varphi \text{ is a tautology,} \\ & \text{or } \varphi \text{ is subsumed by a } \psi \in S \cup A, \\ & \text{or no rule of } I_{LR} \text{ applies to } L, \\ & \text{or } h((A;\varphi)) = k, \\ r & \text{where } r \in I_{LR} \text{ applies to } L\text{, otherwise.} \end{cases}$$

- $\xi_2\colon States_{LR} \times \mathbf{Lit}_\Theta \times I_{LR} \to \mathbf{L}_\Theta \times \mathbf{Lit}_\Theta$ *is the* premise-selection function*:*

$$\xi_2((S;\varphi;A;d;k),L,r) = \begin{cases} (\psi, F) & \text{where } \psi \in S, F \in \psi, L \text{ and } \neg F \text{ unify,} \\ & \text{if } r = iRes; \\ (\psi, F) & \text{where } \psi \in A, F \in \psi, L \text{ and } \neg F \text{ unify,} \\ & \text{if } r = aRes; \\ (\bot, L') & \text{where } L' \in \varphi, L \text{ and } L' \text{ unify,} \\ & \text{if } r = fact; \\ (\bot, \bot) & \text{otherwise.} \end{cases}$$

- ω: *$States_{LR} \rightarrow Bool$ is the* termination-detection function*:*

$$\omega((S;\varphi;A;d;k)) = \begin{cases} true & \text{if } \varphi = \square, \\ false & \text{otherwise.} \end{cases}$$

Typical literal selection functions select the leftmost (or rightmost) literal.

Definition 6.6 (LR-strategy) *A* linear-resolution strategy *is a pair* $\mathbf{C}_{LR} = \langle I_{LR}, \Sigma \rangle$*, where* Σ *is a depth-first search plan with iterative deepening and literal-selection function.*

Definition 6.7 (LR-derivation generated by a strategy) *Given a problem* $S = T \cup \{\varphi_0\}$ *and an LR-strategy* $\mathbf{C}_{LR} = \langle I_{LR}, \Sigma \rangle$*, with* $\Sigma = \langle h, k^*, m, \xi_1, \zeta, \xi_2, \omega \rangle$*, the* LR-derivation generated by $\mathbf{C}_{LR}$ *from* $S = T \cup \{\varphi_0\}$ *is the sequence* $(S;\varphi_0;A_0;d_0;k_0) \vdash_{\mathbf{C}_{LR}} \ldots (S;\varphi_i;A_i;d_i;k_i) \vdash_{\mathbf{C}_{LR}} \ldots$ *such that* $A_0 = \emptyset$*,* $d_0 = 0$*,* $k_0 = k^*$*, and* $\forall i \geq 0$*: if* $\omega((S;\varphi_i;A_i;d_i;k_i)) = false$*, and* $\xi_1((S;\varphi_i;A_i;d_i;k_i)) = L$*, then*

- *if* $\zeta((S;\varphi_i;A_i;d_i;k_i),L) = r \in I_{LR}$*,* φ_{i+1} *and* A_{i+1} *are generated from* $(S;\varphi_i;A_i)$ *by applying* r *to* L *and the clause or literal selected by* ξ_2*,* $d_{i+1} = i+1$*, and* $k_{i+1} = k_i$*;*
- *if* $\zeta((S;\varphi_i;A_i;d_i;k_i),L) = backtrack$*,* $\varphi_{i+1} = \varphi_{d_i-1}$*,* $A_{i+1} = A_{d_i-1}$*,* $d_{i+1} = d_i - 1$*, and* $k_{i+1} = k_i$*, if* $d_i - 1 \neq 0$*,* $k_{i+1} = k_i + m$*, if* $d_i - 1 = 0$*.*

A search plan is *fair* if all the derivations that it generates are:

Definition 6.8 *A derivation* $(S;\varphi_0;A_0) \vdash_{\mathbf{C}_{LR}} \ldots (S;\varphi_i;A_i) \vdash_{\mathbf{C}_{LR}} \ldots$ *is* fair *if and only if* $\forall i \geq 0$*, for all non-tautological* φ *that can be generated from* $(S;\varphi_i;A_i)$ *by a rule in* I_{LR}*, there exists a* j *such that* φ_j *subsumes* φ*.*

Since I_{LR} is refutationally complete, an LR-strategy is complete if its search plan is fair.

6.2 Synthetic marked search-graphs for linear resolution

Given a theorem-proving problem, the search space for an LR-strategy will contain all linear deductions by resolution from that problem. Since linear deductions are made of clauses, it will be a hypergraph with vertices labelled by clauses:

Definition 6.9 (synthetic search-graph) *A* synthetic search-graph *is a hypergraph* (V, E, l)*, where* V *is the set of vertices,* E *is the set of hyperarcs, and* $l: V \rightarrow \mathbf{L}_\Theta$ *is a vertex-labelling function from vertices to clauses.*

Definition 6.10 (induced synthetic search-graph) *Given a theorem-proving problem $S = T \cup \{\varphi_0\}$, the* synthetic search-graph induced by I_{LR} for S with input goal φ_0, *denoted $SG(S, I_{LR}, \varphi_0)$, is the synthetic search-graph (V, E, l), such that v and E are the smallest sets satisfying the following properties:*

- *For all $\varphi \in S$, $\exists v \in V$ such that $l(v) = \varphi$;*
- *For all $v_1, v_2 \in V$ such that $l(v_1) = \varphi_1$, $l(v_2) = \varphi_2$, and ψ is a binary resolvent of φ_1 and φ_2, if*

(1) either $\varphi_1 \notin T$ (φ_1 is a goal clause) and $\varphi_2 \in S$ (φ_2 is an input clause),
(2) or $\varphi_1, \varphi_2 \notin T$ (both are goal clauses) and v_2 is ancestor of v_1,
then $\exists u \in V$ and $\exists e \in E$ such that $l(u) = \psi$ and $e = (v_1, v_2; u)$;

- *For all $v \in V$ such that $l(v) = \varphi \notin T$, if ψ is a factor of φ, then $\exists u \in V$ and $\exists e \in E$ such that $l(u) = \psi$ and $e = (v; u)$.*

Thus, $SG(S, I_{LR}, \varphi_0)$ contains all linear deductions from S with φ_0 as top clause. Hyperarcs $(v, w; u)$ represent binary resolution steps, where goal resolvent $l(u)$ is generated from goal parent $l(v)$ and side clause $l(w)$ (either input clause or ancestor). Hyperarcs $(v; u)$ represent factoring steps, where factor $l(u)$ is generated from goal parent $l(v)$.

The marking of synthetic search-graphs for linear resolution will feature a *hyperarc-marking function* c, defined as in Definition 3.6, and with the same interpretation: $c(e) = 1$, if e was executed, $c(e) = -1$, if e was executed and undone, and $c(e) = 0$ otherwise. On the other hand, the *path-marking function* of Definition 3.5 is replaced by:

Definition 6.11 (vertex-marking function) *Given a synthetic search-graph (V, E, l), a* vertex-marking function *is a function $q: V \to Z$.*

The marking of vertices is interpreted as follows: clauses in the current proof attempt (i.e., the linear deduction currently pursued) have *positive* marking; goal clauses that failed have *negative* marking; goal clauses that have not been reached have *null* marking. Furthermore, this marking identifies the current goal clause as the one with the *maximum marking*. Hence, for a vertex v, $q(v) = m + 1$, if $l(v)$ was generated, is active and has m active goal ancestors; $q(v) = -1$, if $l(v)$ was generated and failed (either naturally or unnaturally), or was deleted; $q(v) = 0$ otherwise. If $l(v)$ failed naturally, either v has no children, or all its children have negative marking as well. If it failed unnaturally, its children have marking 0, since they have not been reached.

Definition 6.12 (synthetic marking) *Given a synthetic search-graph (V, E, l), a* synthetic marking *is a pair (q, c), where q is a vertex-marking function and c is a hyperarc-marking function.*

Definition 6.13 (synthetic marked search-graph) *A* synthetic marked

search-graph *is a tuple* (V, E, l, q, c)*, where* (V, E, l) *is a synthetic search-graph, and* (q, c) *is a synthetic marking.*

Since hyperarcs represent clause generation, a hyperarc is enabled if its premises are present:

Definition 6.14 (enabled hyperarc) *Given a synthetic marked search-graph* $G = (V, E, l, q, c)$*, a hyperarc* $(v, w; u) \in E$ *or* $(v; u) \in E$ *is* enabled*, if* $q(v) > 0$ *and* $q(w) > 0$*, or* $q(v) > 0$*, respectively.*

Definition 6.15 (synthetic marked search-graphs: induced succession) *Let* $S = T \cup \{\varphi_0\}$ *be a theorem-proving problem,* $\mathbf{C}_{LR} =< I_{LR}, \Sigma >$ *an LR-strategy, and* $SG(S, I_{LR}, \varphi_0) = (V, E, l)$ *the synthetic search-graph induced by* I_{LR} *for* S *with input goal* φ_0*. The derivation generated by* $\mathbf{C}_{LR}$ *from* $S = T \cup \{\varphi_0\}$ induces the succession of synthetic marked search-graphs $\{(V, E, l, q_i, c_i)\}_{i \geq 0}$ *defined as follows. Initially, for all* $x \in V$*,* $q_0(x) = 1$ *if* $l(x) \in S$*,* $q_0(x) = 0$ *otherwise; and for all* $a \in E$*,* $c_0(a) = 0$*. Then, for all stages* $i \geq 0$*:*

(1) If Σ *selects an enabled hyperarc* $e = (v, w; u)$ *or* $e = (v; u)$*:*

$$q_{i+1}(x) = \begin{cases} q_i(v) + 1 & \text{if } x = u, \\ q_i(x) & \text{otherwise,} \end{cases} \qquad c_{i+1}(a) = \begin{cases} 1 & \text{if } a = e, \\ c_i(a) & \text{otherwise.} \end{cases}$$

(2) If Σ *backtracks, undoing* $e = (v, w; u)$ *or* $e = (v; u)$*,*

$$q_{i+1}(x) = \begin{cases} -1 & \text{if } x = u, \\ q_i(x) & \text{otherwise,} \end{cases} \qquad c_{i+1}(a) = \begin{cases} -1 & \text{if } a = e, \\ c_i(a) & \text{otherwise.} \end{cases}$$

Each state $(S_i; \varphi_i; A_i; d_i)$ of a derivation has its associated synthetic marked search-graph $SG_i = (V, E, l, q_i, c_i)$, and for $i > 0$ the current goal φ_i is the vertex with maximum marking in SG_i. Then, if the strategy performs a subgoal-reduction at stage i, φ_{i+1} is generated and $q_{i+1}(\varphi_{i+1})$ is the maximum marking in SG_{i+1} (Case 1). If the strategy backtracks because φ_i was deleted or failed (Case 2), $\varphi_{i+1} = \varphi_{d_i - 1}$ (i.e., the goal parent of φ_i), $q_{i+1}(\varphi_i) = -1$ and $q_{i+1}(\varphi_{i+1})$ is the maximum marking in SG_{i+1}.

Definition 6.16 (active search space) *Given a synthetic marked search-graph* $SG = (V, E, l, q, c)$*, the* active search space *is the synthetic marked search-graph* $SG^+ = (V^+, E^+, l^+, q^+, c^+)$*, where* $V^+ = \{x \mid x \in V,\ q(x) > 0\}$*,* $E^+ = \{a \mid a \in E,\ c(a) > 0\}$*, and* l^+, q^+, c^+ *are the appropriate restrictions of* l, q, c *to* V^+ *and* E^+*.*

Definition 6.17 (visited search space) *Given a synthetic marked search-graph* $SG = (V, E, l, q, c)$*, the* visited search space *is the synthetic marked*

search-graph $SG^* = (V^*, E^*, l^*, q^*, c^*)$, *where* $V^* = \{x \mid x \in V,\ q(x) \neq 0\}$, $E^* = \{a \mid a \in E,\ c(a) \neq 0\}$, *and* l^*, q^*, c^* *are the appropriate restrictions of* l, q, c *to* V^* *and* E^*.

Example 10 *Let* $S = \{P(a);\ \neg P(x) \vee \neg Q(y) \vee \neg L(x,y);\ Q(f(z)) \vee \neg Q(z);\ Q(b);\ \neg D(y) \vee L(a, f(y));\ D(x) \vee L(y, f(x))\}$ *and* $\varphi_0 = \neg P(x) \vee \neg Q(y) \vee \neg L(x,y)$. *Figure 7 shows the part of* $SG(S, I_{LR}, \varphi_0)$ *with non-zero marking after a successful derivation. The negative marking of* $\neg L(a,b)$ *and its incoming arc indicates that the strategy tried that proof attempt and then backtracked.*

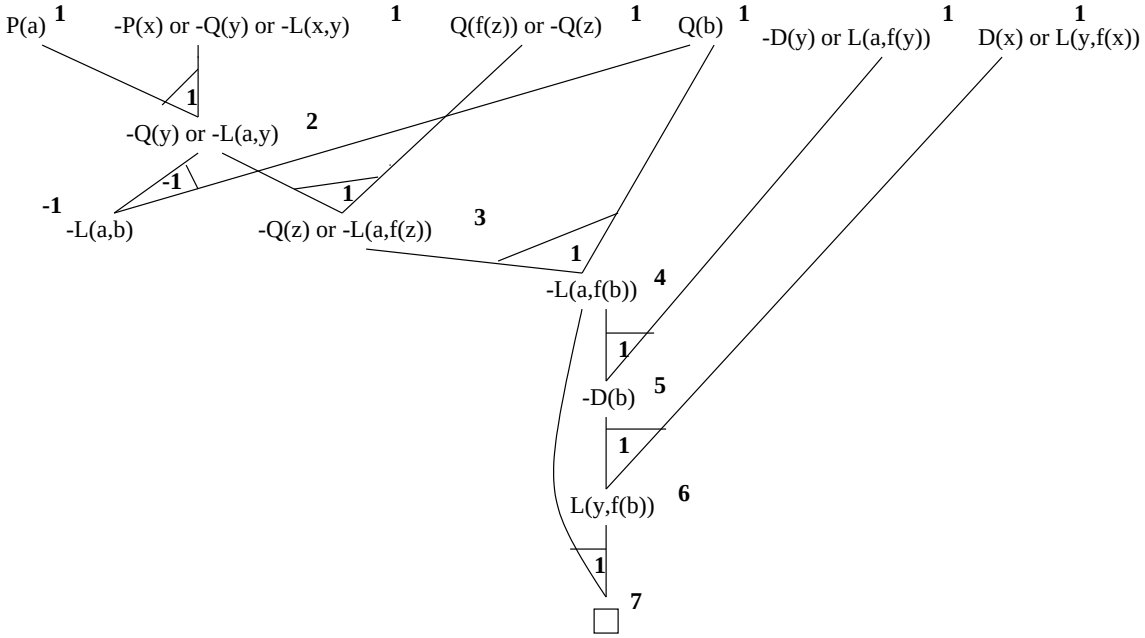

Fig. 7. The visited portion of $SG(S, I_{LR}, \varphi_0)$ for Example 10.

Thus, the framework of Sections 2 and 3 is extended to linear resolution. Then, one can apply to LR-strategies the notion of bounded search spaces of clauses as defined in (Bonacina and Hsiang, 1998b), to analyze, for instance, the pruning effect of tautology deletion and subsumption.

6.3 Comparison of analytic and synthetic marked search-graphs

The basic difference between analytic and synthetic search-graphs is right in the meaning of vertices and hyperarcs. In analytic search-graphs, vertices are labelled by literals, and hyperarcs have the form $e = (v; u_1, \ldots, u_k)$, which denotes extension of $l(v)$ with $l(u_1), \ldots, l(u_k)$. In synthetic search-graphs, vertices are labelled by clauses, and hyperarcs have the form $e = (v_1, \ldots, v_n; u)$, which denotes generation of $l(u)$ from $l(v_1), \ldots, l(v_n)$. Synthetic hyperarcs represent *synthesis* of objects from existing ones (e.g., clauses from clauses),

while analytic hyperarcs represent *decomposition* of goals into subgoals (e.g., clauses into literals). Accordingly, the synthetic search-graph has no root, or, all input clauses can be considered as roots, in the sense that they do not have parents. The analytic search-graph has a root that marks the beginning of the decomposition.

In terms of marking, the first difference is that the marking of a synthetic search-graph needs no substitution, since unifiers are applied to clauses (e.g., recall Examples 3 and 4). Second, synthetic marked search-graphs have a *vertex-marking function*, whereas analytic marked search-graphs have a *path-marking function*. Third, a synthetic hyperarc is enabled if the premises of the generation it represents are available; an analytic hyperarc is enabled if the extension it represents applies. Fourth, analytic hyperarcs form *ancestor-paths* that represent *interpretations*, while synthetic hyperarcs form *ancestor-graphs* (Bonacina and Hsiang, 1998b) that represent *proofs* of generated clauses. All these differences reflect the fact that synthetic strategies work by generating clauses, while analytic ones work by surveying interpretations.

On the other hand, active *search space*, *visited search space* and *backtracking* (the latter for linear resolution, not ordering-based strategies), are modelled uniformly in both synthetic and analytic graphs. Finiteness, of ancestor-paths in analytic marked search-graphs, and ancestor-graphs in synthetic marked search-graphs, is used in both contexts to introduce a measurable quantity in infinite search graphs: the *length* of an ancestor-path, and the *distance* of a vertex from the input along an ancestor-graph. In either kind of marked search-graph, the measurable quantity allows us to define *bounded search spaces*, as multisets of clauses (in the synthetic case) and multisets of partial interpretations (in the analytic case) within the given bound. The multiplicity of a clause φ in $space(G, j)$ is the number of ancestor-graphs of φ within distance j from the input. This is coherent with the fact that ordering-based strategies search for a refutation by searching for a proof of the empty clause. The multiplicity of a partial interpretation Γ in $space(G, j)$ is the number of open ancestor-paths with label Γ within distance j from the input. This is coherent with the fact that tableau-based strategies search for a refutation by eliminating candidate models.

6.4 Comparison of the synthetic marked search-graphs for subgoal-reduction with those for ordering-based strategies

Search graphs for linear resolution and ordering-based strategies have in common the basic interpretation of vertices and hyperarcs: vertices are labelled by clauses and hyperarcs represent generative inferences. A first simple difference is that synthetic search-graphs for ordering-based strategies feature

a *hyperarc-labelling function* (see (Bonacina and Hsiang, 1998b)), that label hyperarcs with the name of the applied inference rule, because ordering-based strategies often have multiple inference rules (e.g., resolution and paramodulation). This is unnecessary in the synthetic search-graphs for linear resolution, since the only inference rules are binary resolution and factoring, that can be distinguished on the basis of arity. Analytic search-graphs do not have hyperarc-labelling function either, because all hyperarcs represent instances of the same extension rule.

The handling of *contraction* is quite different. As already mentioned, in ordering-based strategies rules such as subsumption and tautology deletion are proper inference rules that contract the existing set of clauses, whereas in linear resolution strategies they apply only to the current goal and their application is merely a case for backtracking. Furthermore, ordering-based strategies feature not only contraction rules that delete clauses, but also contraction rules that *replace* clauses by clauses, such as *simplification* by equations. Therefore, for ordering-based strategies, the representation of contraction involves both structure and marking of the search-graph: the generation of clauses by contraction is represented by hyperarcs, while deletions are represented by the marking.

The interpretation of the *marking* of vertices has both analogies and differences. For ordering-based strategies, a positive marking means that the clause was generated and kept, a negative one, that it was generated and deleted by contraction. For linear-resolution strategies, a positive marking means that the clause was generated and is part of a linear proof attempt being pursued, a negative one, that it was generated but failed. Contracted clauses have a negative marking in both markings, but for linear resolution, the representation of contraction is subsumed by that of failure and backtracking. For both, non-zero marking corresponds to the visited search space.

The two models differ in the treatment of *variants*: in the marked search-graph of ordering-based strategies all variants of a clause are associated to the same vertex, whereas in that of synthetic subgoal-reduction strategies, different variants are associated to different vertices. The explanation lies in the difference in nature of the strategies. In an ordering-based strategy, if a clause φ, variant of an existing clause φ', is generated, φ is subsumed. Not only φ and φ' are logically equivalent, but multisets $S \cup \{\varphi\}$ and $S \cup \{\varphi, \varphi'\}$ contain implicitly the same proof attempts, so that φ and φ' are equivalent also from the proof search point of view. In linear resolution, if a goal φ, variant of an ancestor goal φ', is generated, φ is subsumed and the strategy backtracks to its predecessor, which is different than the predecessor of φ'. Even if φ and φ' are logically equivalent, the linear deductions rooted in φ and φ' are different in general.

In other words, the history of generated clauses is relevant to the proof search of subgoal-reduction strategies, whereas it is irrelevant for ordering-based strategies, where it matters only for *proof reconstruction*[4]. The reason is that ordering-based strategies generate many proof attempts *implicitly*, and do not restrict the search by the shape of the proof, so that the structure of the proof attempts is ignored during the search. On the other hand, subgoal-reduction strategies generate proof attempt(s) *explicitly*, and linear-resolution strategies admit only linear ones, so that their structure is part of the state of the search.

7 Related work

7.1 AND-OR graphs

AND-OR graphs have been used traditionally to represent the search space of subgoal-reduction strategies, especially in Horn logic. They have been used also as data structures in the implementation of specific systems (e.g., Konev and Jebelean, 2000). The following example shows AND-OR graph and marked search-graph for a given set of clauses:

Example 11 *Assume* $S = \{\neg P \vee Q, P \vee \neg R, \neg U \vee \neg R \vee Q, P \vee \neg B, \neg P \vee \neg Q\}$ *with* $\varphi_0 = \neg P \vee \neg Q$*. Figure 8 shows the AND-OR graph (with hyperarcs representing AND-arcs) and* $G(S, I, \varphi_0)$*.*

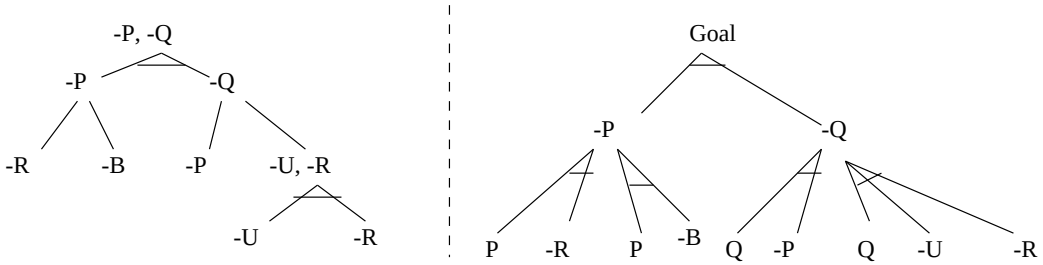

Fig. 8. AND-OR graph (on the left) and analytic search-graph $G(S, I, \varphi_0)$ (on the right).

AND-OR graphs are also analytic, because AND-arcs represent decomposition, and have rigid variables, because the literals of a clause are spread over different branches. Accordingly, substitutions cannot be applied prior to the search. However, AND-OR graphs have been traditionally used for SLD-resolution: in Horn logic, linear resolution lends itself to be interpreted

[4] When $\square$ is generated, the ordering-based strategy uses the history of clauses to extract the proof, i.e., the ancestor-tree of $\square$, from the record of generated clauses.

in terms of problem decomposition, because there is no need for ancestor-resolution and factoring, hence no need for vertices labelled by resolvents. Figure 9 shows $SG(S, I_{LR}, \varphi_0)$ for the problem of Example 11: one can see

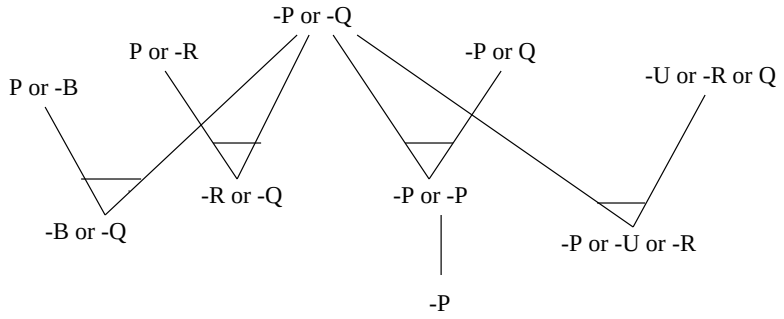

Fig. 9. Synthetic search-graph of linear resolution for Example 11.

how resolvents in the synthetic search graph correspond to frontiers in the analytic search-graph or in the AND-OR graph.

Marked search-graphs capture more features than AND-OR graphs, thanks to the idea of distinguishing what belongs to the static search space, and is represented by the graph, and what belongs to the search process, and is represented by the dynamic marking. The list includes first-order features (e.g., ME-reduction, factoring, ancestor-resolution), explicit closure of branches, link conditions (e.g., see Examples 2 and 5), lemmatization by folding-up (e.g., see Example 7), and *backtracking*, that in AND-OR graphs is left to the manual simulation by the reader. In our approach, backtracking is represented in a uniform way, regardless of its cause, in both synthetic and analytic search-graphs (e.g., see Examples 6 and 10).

7.2 State space

Search in theorem proving can be described at different abstraction levels. If we apply the classical AI notion of *state space* to the search problem given by a theorem-proving problem $S = T \cup \{\varphi_0\}$, and a tableau inference system I, the result is a tree, [5] with nodes labelled by states (e.g., (S, X, d)), and an arc from node (S, X, d) to node (S, X', d'), if there is an inference $(S, X, d) \vdash_I (S, X', d')$. If we omit the auxiliary components S and d, we get a tree with nodes labelled by tableaux, success nodes labelled by closed tableaux, and the root labelled by the initial tableau X_{φ_0}. Similarly, for I_{LR}, one gets a tree with nodes labelled by pairs $(A; \varphi)$, representing linear deductions. If all possible choices of initial goal are considered, the outcome is a *forest*. Applied to ordering-based strategies,

[5] A state may be reached in more than one way, so that the state space is a graph, but it can be represented as a tree by allowing different nodes to have the same label.

this model yields trees with nodes labelled by multisets of clauses. Such trees, termed *I-trees* to emphasize that their structure depends on the inference system, were used in (Bonacina and Hsiang, 1995) to develop a notion of target-oriented fairness for completion-based strategies.

Trees of proof attempts were used for subgoal-reduction strategies by several authors. In the framework of (Plaisted and Zhu, 1997), model elimination was modelled by using a forest with nodes labelled by chains. The study of static lemmatization in (Fuchs, 2000) refers to a tree with nodes labelled by ME-tableaux. The same model was adopted to describe many features and refinements of tableaux in (Letz and Stenz, 2001b). The purpose of these studies and the nature of their results are different from ours. For instance, the tree of tableaux is used in (Letz and Stenz, 2001b) to define refinements, to show that certain features are incompatible with others, to establish completeness theorems, or simulation results. The latter are essentially results of *relative complexity*, with *proof length*, e.g., some measure of tableau size, as the implicit complexity measure. The analysis consists in showing that for every closed tableau built by using a certain feature there exists a closed tableau generated using another feature, and one can be mapped (e.g., polynomially) into the other. Thus, they are ultimately proof-theoretic results on proof existence. The emphasis is on proof and proof length, not search space and search space size. On the other hand, our purpose was to study the evolution of the search space during the derivation, and capture the effect of refinements on search space size.

A disadvantage of the state space approach is that if a tableau becomes an atomic label, the actions of the strategy can be observed at most indirectly as "moves" from state to state, with much less detail than that offered by marked search-graphs. The same is true for ordering-based strategies, since the label of a node in an I-tree is a whole multiset of clauses. We emphasize that marked search-graphs and I-trees, or trees of tableaux, are complementary, because they represent different levels of abstraction. Furthermore, we took care of relating them: since a node of an I-tree is a state of a possible derivation, the notions of succession of marked search-graphs induced by a derivation, and of marked search-graph associated to a state (see Definitions 3.11 and 6.15 in this paper, Definition 3.7 in (Bonacina and Hsiang, 1998b), and Definitions 4.4 and 4.5 in (Bonacina, 1999a)) connect marked search-graphs and I-trees.

The above description of I-tree assumes depth-first search (DFS), because a state contains *one* tableau. If I were coupled with a breadth-first or best-first search plan (BFS), a state would contain a set of tableaux, and a node of the I-tree would be labelled by a set of tableaux. For linear resolution, we obtain an I-tree with nodes labelled by linear deductions under DFS, by sets of linear deductions under BFS. This dependency is characteristic of subgoal-reduction strategies: the I-tree of ordering-based strategies has nodes labelled by mul-

tisets of clauses, regardless of whether the search plan selects clauses by DFS or BFS. This difference is due to the fact that the state (multiset of clauses) of an ordering-based strategy contains *implicitly* the proof attempts that the strategy is developing, whereas the state of a subgoal-reduction strategy "is" the current proof attempt, or the current set of proof attempts, based on DFS vs. BFS.

The fact that the notion of state depends on the choice of search plan may be considered another disadvantage of the state-space model for subgoal-reduction strategies. In the marked search-graph approach, for both tableau-based and linear-resolution strategies, generalization to BFS can be done by replacing the induced marking with a set of markings, one per proof-attempt (one per tableau, e.g., Example 4, or one per linear deduction). This is relevant to study strategies that generate explicitly multiple tableaux by best-first search and apply *tableau subsumption* to prune them (Baumgartner and Brüning, 1997). Such strategies are described as expanding and contracting sets of tableaux, much like ordering-based strategies. Thus, one may also resort to analyze them on a synthetic marked search-graph with nodes labelled by (encodings of) tableaux, or on the I-tree of tableaux. It is fair to say, however, that methods generating explicitly multiple tableaux are mostly of theoretical interest, and modelling *failure caching* (e.g., Astrachan and Stickel, 1992; Letz et al., 1994; Bonacina and Hsiang, 1998a), or *failure substitutions* (Letz and Stenz, 2001b) is expected to be more relevant to the practice.

Trees of states and graphs of formulae are not the only formalisms for the search space of proof procedures. Another alternative is the tree of recursive calls of the procedure. This scheme, however, can be used only in decidable cases, in propositional (e.g., for the Davis-Putnam-Logeman-Loveland procedure in (Zhang et al., 1996)) or modal logics (e.g., Donini and Massacci, 2000b). Table 1 summarizes this discussion.

Classes of strategies	Closure	State space: I-tree	Search space: Synthetic MSG	Search space: Analytic MSG
Clausal normal form tableaux (I_{TAB}, I_{MET})	Set of tableaux	Tree of tableaux with arcs for moves	N.A.	MSG of literals with arcs for extensions
Linear resolution strategies (I_{LR})	Set of goal clauses	Tree of pairs $(A;\varphi)$ with arcs for moves	MSG of goal clauses with arcs for generations	AND-OR graph
Ordering-based strategies	Set of clauses	Tree of multisets of clauses with arcs for moves	MSG of clauses with arcs for generations	N.A.

Table 1
Models of the search space: MSG stands for marked search-graph.

The above remark on DFS vs. BFS in the definition of the I-tree is better understood in terms of *proof confluence*. A strategy is *proof confluent* if it is never the case that committing to an inference step prevents it from finding a proof. For proof confluent strategies, the derivation generated by the search plan is a *path* in the I-tree: the strategy can go down one path with no need of backtracking. If a strategy is not proof confluent, it goes down one path of the I-tree, backtracks and tries another one, so that a derivation corresponds to a *subtree* of the I-tree, because also the steps undone by backtracking are included. Fair ordering-based strategies are trivially proof confluent, because they accumulate generated data without ever undoing steps. Analytic subgoal-reduction strategies with link condition, whether weak or strong, and depth-first search, are not proof confluent. Analytic subgoal-reduction strategies without link condition are trivially proof confluent but not practical, as already discussed in Section 2.3. Similarly trivially proof confluent, but unpractical, would be synthetic subgoal-reduction strategies and analytic subgoal-reduction strategies with link condition and a breadth-first search plan.

Since breadth-first search is not practical, the problem of designing proof-confluent, analytic, subgoal-reduction strategies with weak link condition and depth-first search has been investigated by several authors (e.g., Billon, 1996; Bierwald and Käufl, 1997; Baumgartner, 1998; Baumgartner et al., 1999; Beckert, 2003; Giese, 2001; van Eijck, 2001). The *disconnection method* (Billon, 1996) and *hypertableaux* (Baumgartner, 1998; Stolzenburg, 1999) achieve proof confluence by two modifications. On one hand, they restrict extension in such a way that variables in the tableau get renamed but not properly instantiated: e.g., *hyperextension*[6] extends branch Γ with $P_1 \vee \ldots \vee P_k \vee \neg N_1 \vee \ldots \vee \neg N_m$, if Γ contains positive literals $L_1, \ldots, L_m$, such that $L_i\sigma = N_i\sigma$ and $L_i\sigma \stackrel{\bullet}{=} L_i$ for $1 \leq i \leq m$, where the latter condition means that $L_i\sigma$ is a variant of L_i. On the other hand, they enrich the inference system with *link inferences*: roughly speaking, a *binary link* (Lee and Plaisted, 1992; Billon, 1996) is similar to a binary resolution step where the instances of the parents are added instead of the resolvent; a *hyperlink* (Lee and Plaisted, 1992) resembles a hyperresolution step (without polarity requirement on the electrons), where the instance of the nucleus (Lee and Plaisted, 1992) or the instances of the electrons (Baumgartner, 1998; Stolzenburg, 1999) are added instead of the hyperresolvent[7].

[6] Hyperextension satisfies the weak link condition, but not the strong one, because it does not require that one of the L_j's is the leaf of Γ.

[7] In (Lee and Plaisted, 1992), all literals in the nucleus are linked; in (Baumgartner, 1998; Stolzenburg, 1999), all negative literals.

These two moves are complementary: the strategy adds instances to S (by the link inferences), in place of instantiating the tableau, in order to avoid backtracking to undo instantiations in the tableau. Since S is expanded, *contraction rules* become desirable, but only unit subsumption, clausal simplification, variant subsumption and tautology deletion are admissible, because proper subsumption would remove all instances. The implementation of the disconnection method in (Letz and Stenz, 2001a) attaches instances to the tableau instead of adding them to S; it still avoids backtracking, because the substitutions are not applied to the whole tableau, but only locally to the attached instances. This variant requires to test for *ground closure* instead of using mgu atomic closure: all variables in the tableau are replaced by a new constant, and the procedure checks if every branch has a pair of complementary literals. The ground closure test reminds one of the *hyperlinking method* (Lee and Plaisted, 1992), that interleaves instance generation by hyperlinking with testing for propositional contradiction by DPLL (the Davis-Putnam-Logeman-Loveland procedure, e.g., (Chang and Lee, 1973)) after replacing all variables in the clauses by a new constant. A different approach was presented in (Giese, 2001; van Eijck, 2001). It simulates breadth-first search, i.e., developing all proof attempts, while keeping in memory only one, by generating a tableau decorated with sets of *constraints*. Specialized data structures for this purpose were presented in (Giese, 2001).

The resulting methods interleave *model generation*[8] (e.g., by hyperextension) with *instance generation* (e.g., by hyperlinking). They can be considered *hybrids* between *subgoal-reduction* and *instance-based* strategies, such as the above mentioned hyperlinking method, its successor *Ordered Semantic HyperLinking (OSHL)* (Plaisted and Zhu, 2000), or (Ganzinger and Korovin, 2003). In OSHL, DPLL is no longer merely a test for contradiction, but a model generator applied at the ground level, so that the survey of interpretations is represented by the *semantic tree* built implicitly by the splitting rule of DPLL. In all three approaches – disconnection method, hypertableaux and OSHL – the model-generation part *controls* the instance-generation part in such a way that it generates instances that close the selected branch in the tableau or semantic tree. This development of hybrid strategies draws on an intrinsic similarity between analytic subgoal-reduction and instance-based strategies: both work by generating instances, rather than consequences like ordering-based strategies. The fact that instantiations are part of the marking for analytic strategies (e.g., Examples 3 and 4) is an evidence of this similarity. Table 2 summarizes this discussion.

[8] Model elimination, from a refutational point of view.

	Search plan	Proof confluent	Derivation	Backtracking	Proof attempts	Admitted proof	Used in practice
Ordering-based	any	yes	path of I-tree	no	many implicit	any	yes
Synthetic subgoal-reduction (I_{LR})	BFS	yes	path of I-tree	no	many explicit	linear (s. linked)	no
	DFS	no	subtree of I-tree	yes	one explicit	linear (s. linked)	yes
Analytic subgoal-red. no link cond.	any	yes	path of I-tree	no	one explicit	any	no
Analytic subgoal-red. w. link cond. (I_{MET}, I_{TAB})	BFS	yes	path of I-tree	no	many explicit	linked	no
	DFS	no	subtree of I-tree	yes	one explicit	linked	yes
Hybrid (e.g. hyper-tableaux)	any	yes	path of I-tree	no	one explicit	weakly linked	yes

Table 2
Summary of strategies: "linked" refers to a proof built respecting a link condition, and "s. linked" stands for "strongly linked."

8 Discussion

Much research in theorem proving consists in defining rules or control mechanisms that can counter the explosion of the search space. Analyzing formally the impact of these features is very challenging, because the search space of first-order theorem proving is infinite in general. A proper notion of "size" is not defined, and since pruning mechanisms cannot turn in general an infinite search space into a finite one, it is problematic to compare infinite graphs and say that one is "smaller" than the other.

Our approach consists in building appropriate notions of *bounded search spaces*, that are finite and can be compared in a well-founded ordering. (Bonacina and Hsiang, 1998b) compared ordering-based strategies of different contraction power. (Bonacina, 1999a) compared distributed-search contraction-based strategies with their sequential bases. In this paper we compared tableau-based strategies with and without *regularity check* and *lemmatization by folding-up*. The main difference in terms of the analysis is that the bounded search spaces of ordering-based strategies are *monotonic*, whereas those of tableau-based subgoal-reduction strategies are not. This depends on properties of *proof confluence* and *redundancy*. Fair ordering-based strategies are proof confluent, and their contraction mechanisms satisfy the property *"once redundant, always redundant"* (Bonacina and Dershowitz, 2005): if something becomes redundant, it remains redundant for the rest of the derivation. Thus, considering that expansion steps visit the search space, and therefore do not modify the bounded search spaces, while contraction steps contract them by removing redundancies, it follows that the bounded search spaces are essentially monotonic. On

the other hand, analytic subgoal-reduction strategies, with link condition and depth-first search, are neither proof confluent, nor have a notion of persistent redundancy. It follows that bounded search spaces oscillate non-monotonically, and the analytic results that one can obtain are intrinsically weaker.

Since lack of proof confluence is problematic, a main direction for future work is to model and analyze the *proof-confluent hybrid strategies* of Section 7.3 to measure their potential advantage with respect to plain subgoal-reduction strategies. This may entail modelling pure instance-based strategies as well. Other topics include analyzing and comparing other refinements of ME-tableaux, clause normal form tableaux, and tableaux in general also beyond clause normal form (e.g., Baumgartner and Furbach, 1998; Letz and Stenz, 2001b; Haehnle, 2001). A feature that reduces the bounded search spaces for all bounds $j \geq n$ may not help if there are proofs within smaller distance, i.e., if some $space(G, i)$ for $i < n$ contains a proof. Indeed, refinements of strategies may not help if the problem is too easy. If we knew that i is the smallest bound such that $space(G, i)$ contains a proof, we could work with $space(G, i)$ only, but knowing i amounts to having a proof already, so that the difficulty of searching for one has disappeared. However, a possible direction for future work is to aim at *relative* results, namely assume a perfect strategy that will find a proof by searching only in $space(G, i)$, where i remains a parameter, and seek results on the behavior of concrete strategies relative to this oracle.

Acknowledgements Part of this work was done while the author was with the Department of Computer Science of The University of Iowa. In addition to support from the Department, College and University, the author acknowledges support from the National Science Foundation with grants CCR-94-08667, CCR-97-01508, and EIA-97-29807, and from the College of Liberal Arts and Sciences with a Dean Scholar Award. The author thanks Alessandro Armando of the Dipartimento di Informatica, Sistemistica e Telematica, of the Università degli Studi di Genova, where part of this work was done in the summer of 2001, and the referees for their constructive and helpful comments.

References

Andrews, P. B., 1981. Theorem proving via general matings. J. ACM 28 (2), 193–214.

Armando, A., Ranise, S., Rusinowitch, M., 2003. A rewriting approach to satisfiability procedures. Information and Computation 183 (2), 140–164.

Astrachan, O. L., Loveland, D. W., 1997. The use of lemmas in the model elimination procedure. J. Autom. Reason. 19 (1), 117–141.

Astrachan, O. L., Stickel, M. E., 1992. Caching and lemmaizing in model elimination theorem provers. In: Kapur, D. (Ed.), Proc. of CADE-11. Vol. 607 of LNAI. Springer, pp. 224–238, full version: Tech. Rep. 513, SRI International, Dec. 1991.

Baaz, M., Fermüller, C., Leitsch, A., 1994. A non-elementary speed-up in proof length by structural clause form transformation. In: Proc. of LICS–94. IEEE Press, pp. 213–219.

Baumgartner, P., 1998. Hyper tableaux – the next generation. In: de Swart, H. (Ed.), Proc. of TABLEAUX–98. Vol. 1397 of LNAI. Springer, pp. 60–76.

Baumgartner, P., Brüning, S., 1997. A disjunctive positive refinement of model elimination and its application to subsumption deletion. J. Autom. Reason. 19, 205–262.

Baumgartner, P., Eisinger, N., Furbach, U., 1999. A confluent connection calculus. In: Ganzinger, H. (Ed.), Proc. of the 16th CADE. Vol. 1632 of LNAI. Springer, pp. 329–343.

Baumgartner, P., Furbach, U., 1993. Consolution as a framework for comparing calculi. J. Symbolic Comput. 16 (5), 445–477.

Baumgartner, P., Furbach, U., 1994. Model elimination without contrapositives and its application to PTTP. J. Autom. Reason. 13, 339–359.

Baumgartner, P., Furbach, U., 1998. Variants of clausal tableaux. In: Bibel, W., Schmitt, P. H. (Eds.), Automated Deduction - A Basis for Applications. Vol. I: Foundations - Calculi and Methods. Kluwer Academic Publishers, Ch. 3, pp. 73–102.

Beckert, B., 2003. Depth-first proof search without backtracking for free-variable clausal tableaux. J. Symbolic Comput. 36 (1–2), 117–138.

Bibel, W., 1981. On matrices with connections. J. ACM 28, 633–645.

Bierwald, C., Käufl, T., 1997. Tableau prover Tatzelwurm: hyper-links and UR-resolution. In: Bonacina, M. P., Furbach, U. (Eds.), Proc. of FTP-1997. No. 97-50 in Tech. Rep. of RISC. Johannes Kepler Universität, pp. 22–28.

Billon, J.-P., 1996. The disconnection method. In: Miglioli, P., Moscato, U., Mundici, D., Ornaghi, M. (Eds.), Proc. of TABLEAUX-96. Vol. 1071 of LNAI. Springer, pp. 110–126.

Bonacina, M. P., 1999a. A model and a first analysis of distributed-search contraction-based strategies. Annals of Mathematics and Artificial Intelligence 27 (1–4), 149–199.

Bonacina, M. P., 1999b. A taxonomy of theorem-proving strategies. In: Wooldridge, M. J., Veloso, M. (Eds.), Artificial Intelligence Today. Vol. 1600 of LNAI. Springer, pp. 43–84.

Bonacina, M. P., Dershowitz, N., 2005. Abstract canonical inference. ACM Transactions on Computational Logic to appear.

Bonacina, M. P., Hsiang, J., 1995. Towards a foundation of completion procedures as semidecision procedures. Theoretical Computer Science 146, 199–242.

Bonacina, M. P., Hsiang, J., 1998a. On semantic resolution with lemmaizing and contraction and a formal treatment of caching. New Generation Com-

puting 16 (2), 163–200.
Bonacina, M. P., Hsiang, J., 1998b. On the modelling of search in theorem proving – towards a theory of strategy analysis. Information and Computation 147, 171–208.
Chang, C.-L., Lee, R. C.-T., 1973. Symbolic Logic and Mechanical Theorem Proving. Academic Press.
Comon-Lundh, H., Courtier, V., 2003. New decidability results for fragments of first-order logic and application to cryptographic protocols. In: Nieuwenhuis, R. (Ed.), Proc. of RTA-14. Vol. 2706 of LNAI. Springer, pp. 148–164.
Donini, F. M., Massacci, F., 2000a. Design and results of TANCS–2000 non–classical (modal) systems comparison. In: Dyckhoff, R. (Ed.), Proc. of TABLEAUX–2000. Vol. 1847 of LNAI. Springer, pp. 52–56.
Donini, F. M., Massacci, F., 2000b. EXPtime tableaux for ALC. Artificial Intelligence 124 (1), 87–138.
Eder, E., 1992. Relative Complexities of First-order Calculi. Artificial Intelligence. Vieweg, Wiesbaden.
Fermüller, C., Leitsch, A., 1998. Decision procedures and model-building in equational clause logic. J. IGPL 6 (1), 17–41.
Fermüller, C., Leitsch, A., Hustadt, U., Tammet, T., 2001. Resolution decision procedures. In: Robinson, A., Voronkov, A. (Eds.), Handbook of Automated Reasoning. Vol. 2. North Holland, Ch. 25, pp. 1793–1849.
Fermüller, C., Leitsch, A., Tammet, T., Zamov, N., 1993. Resolution Methods for the Decision Problem. LNAI 679. Springer.
Fleisig, S., Loveland, D. W., Smiley, A., Yarmush, D., 1974. An implementation of the model elimination proof procedure. J. ACM 21, 124–139.
Fuchs, M., 2000. Controlled use of clausal lemmas in connection tableau calculi. J. Symbolic Comput. 29 (2), 299–342.
Ganzinger, H., Korovin, K., 2003. New directions in instantiation-based theorem proving. In: Proc. of LICS 2003. IEEE Computer Society Press, pp. 55–64.
Giese, M., 2001. Incremental closure of free-variable tableaux. In: Gore, R. P., Leitsch, A., Nipkow, T. (Eds.), Proc. of the First IJCAR. Vol. 2083 of LNAI. Springer, pp. 545–560.
Goller, C., Letz, R., Mayr, K., Schumann, J., 1994. SETHEO v3.2: recent developments. In: Bundy, A. (Ed.), Proc. of the 12th CADE. Vol. 814 of LNAI. Springer, pp. 778–782.
Haehnle, R., 2001. Tableaux and related methods. In: Robinson, A., Voronkov, A. (Eds.), Handbook of Automated Reasoning. Elsevier Science Publishers B. V., Ch. 3, pp. 100–178.
Konev, B., Jebelean, T., 2000. Using meta-variables for natural deduction in THEOREMA. In: Kerber, M., Kohlhase, M. (Eds.), Proc. of Calculemus-2000. A. K. Peters.
Korf, R. E., 1985. Depth-first iterative deepening: an optimal admissible tree search. Artificial Intelligence 27 (1), 97–109.
Kowalski, R., 1969. Search strategies for theorem proving. In: Meltzer, B.,

Michie, D. (Eds.), Machine Intelligence. Vol. 5. Edinburgh University Press, pp. 181–201.

Kowalski, R., Kuehner, D., 1971. Linear resolution with selection function. Artificial Intelligence 2, 227–260.

Lee, S.-J., Plaisted, D. A., 1992. Eliminating duplication with the hyperlinking strategy. J. Autom. Reason. 9, 25–42.

Leitsch, A., 1997. The Resolution Calculus. Springer.

Letz, R., July 1993. First-order calculi and proof procedures for automated deduction. Ph.D. thesis, Technische Hochschule Darmstadt, Darmstadt, Germany.

Letz, R., 1998. Clausal tableaux. In: Bibel, W., Schmitt, P. H. (Eds.), Automated Deduction - A Basis for Applications. Vol. I: Foundations - Calculi and Methods. Kluwer Academic Publishers, Ch. 2, pp. 43–72.

Letz, R., Mayr, K., Goller, C., 1994. Controlled integration of the cut rule into connection tableau calculi. J. Autom. Reason. 13 (3), 297–338.

Letz, R., Schumann, J., Bayerl, S., Bibel, W., 1992. SETHEO: a high performance theorem prover. J. Autom. Reason. 8 (2), 183–212.

Letz, R., Stenz, G., 2001a. DCTP – a disconnection calculus theorem prover. In: Gore, R. P., Leitsch, A., Nipkow, T. (Eds.), Proc. of the First IJCAR. Vol. 2083 of LNAI. Springer, pp. 381–385.

Letz, R., Stenz, G., 2001b. Model elimination and connection tableau procedures. In: Robinson, A., Voronkov, A. (Eds.), Handbook of Automated Reasoning. Elsevier Science Publishers B. V., Ch. 28, pp. 2015–2114.

Loveland, D. W., 1969. A simplified format for the model elimination procedure. J. ACM 16 (3), 349–363.

Loveland, D. W., 1972. A unifying view of some linear Herbrand procedures. J. ACM 19 (2), 366–384.

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

McCune, W. W., 1994. Otter 3.0 reference manual and guide. Tech. Rep. 94/6, Mathematics and Computer Science Division, Argonne National Laboratory.

Pelletier, F. J., Sutcliffe, G., Suttner, C. B., 2002. The development of CASC. AI Communications 15 (2–3), 79–90.

Peltier, N., 2003. A resolution-based model-building algorithm for a fragment of $\mathcal{OCC}1\mathcal{N}_{=}$. In: Dahn, I., Vigneron, L. (Eds.), Proc. of FTP-2003. No. DSCI–II710/03 in Technical Reports. Universidad Politécnica de Valencia, pp. 91–103.

Plaisted, D. A., 1990. A sequent-style model elimination strategy and a positive refinement. J. of Autom. Reason. 6 (4), 389–402.

Plaisted, D. A., Zhu, Y., 1997. The Efficiency of Theorem Proving Strategies. Friedr. Vieweg & Sohns.

Plaisted, D. A., Zhu, Y., 2000. Ordered semantic hyper linking. J. Autom. Reason. 25, 167–217.

Schumann, J., 1994. Delta: a bottom-up pre-processor for top-down theorem

provers. In: Bundy, A. (Ed.), Proc. of the 12th CADE. Vol. 814 of LNAI. Springer, pp. 774–777.

Schumann, J., 2001. Automated Theorem Proving in Software Engineering. Springer.

Shostak, R. E., 1976. Refutation graphs. Artificial Intelligence 7, 51–64.

Smullyan, R. M., 1995. First-Order Logic. Dover, (Republication of Vol. 43, *Ergebnisse der Mathematik und ihrer Grenzgebiete* Series, Springer, 1968).

Stickel, M. E., 1992. A Prolog technology theorem prover: new exposition and implementation in Prolog. Theoretical Computer Science 104, 109–128.

Stolzenburg, F., 1999. Loop-detection in hypertableaux by powerful model generation. J. Universal Computer Science 5 (3), 135–155.

van Eijck, J., 2001. Model generation from constrained free variable tableaux. In: Gore, R. P., Leitsch, A., Nipkow, T. (Eds.), Short papers presented at the First IJCAR. No. DII 11/2001 in Technical Reports. Università di Siena, pp. 160–169.

Wallace, K., Wrightson, G., 1995. Regressive merging in model elimination tableau-based theorem provers. J. IGPL 3 (6), 921–937.

Winskel, G., 1994. The Formal Semantics of Programming Languages. Foundations of Computing Series. MIT Press.

Zhang, H., Bonacina, M. P., Hsiang, J., 1996. PSATO: a distributed propositional prover and its application to quasigroup problems. J. Symbolic Comput. 21, 543–560.