Indefinite Information in Modal Logic Programming[1]

Matthew Stone
Department of Computer Science and Center for Cognitive Science
Rutgers University
110 Frelinghuysen Road, Piscataway NJ 08854-8019
mdstone@cs.rutgers.edu

Summary

We develop a modal logic programming language DIALUP in which programs can make disjunctive and existential assertions. Such assertions play an important role in specifications of agents for reasoning about planning and interaction, where it is essential to describe agents' partial information. More broadly, such assertions allow programmers to describe the modular structure of any specification. DIALUP is designed so that disjunctions indicate local ambiguities in search. By forcing ambiguities to be considered independently, these modular disjunctions can be used to construct efficiently executable specifications in reasoning tasks involving partial information that otherwise might require prohibitive search. To achieve this behavior in DIALUP requires prior proof-theoretic justifications of logic programming to be extended, strengthened, and combined with proof-theoretic analyses of modal deduction in a novel way.

Contents

[1]Thanks to Mark Steedman, Rich Thomason, L. Thorne McCarty and Michael Fourman for extensive comments. This work was supported by an NSF graduate fellowship, an IRCS graduate fellowship, and a postdoctoral fellowship from RUCCS, as well as NSF grant IRI95-04372, ARPA grant N6601-94-C6043, and ARO grant DAAH05-94-G0426. April 30, 2001.

1 Overview

Like all programming languages, logic programming languages need structuring constructs to describe the modularity of programs and thereby to facilitate their design and reuse. Modal logic provides a declarative setting to develop such structuring constructs [Miller, 1989, Giordano and Martelli, 1994, Baldoni et al., 1993, Baldoni et al., 1996]. A necessary goal $\Box G$ can be seen as a *modular* goal because, in modal logic, only program clauses of the form $\Box P$ can contribute to its proof. With the right modal semantics, modularity also brings locality: a goal $\Box(P \supset G)$ introduces a *local* assumption P that can only contribute to the proof of G. This paper extends this approach to modularity and locality to indefinite logic programs. We develop a modal logic programming language DIALUP in which programs can establish disjunctions and existentially quantified sentences.

Modal logic already provides a general, declarative formalism for specifying change over time, the knowledge of agents, and other special-purpose domains [Prior, 1967, Hintikka, 1971, Schild, 1991]. DIALUP, like the languages proposed in [Fariñas del Cerro, 1986, Debart et al., 1992, Baldoni et al., 1993, Baldoni et al., 1996], supports such specifications—indeed, the indefinite assertions that DIALUP makes available play an important role in applying modal formalisms to planning, information-gathering and communication [Stone, 2000]. Nevertheless, DIALUP derives broad applicability even beyond these domains, thanks to a crucial and direct connection between modularity and search.

In DIALUP, modal operators not only describe the content and structure of a specification, but also resolve ambiguities in execution in an intuitive and useful way. DIALUP extends the Near-Horn approach to disjunctive logic programming [Loveland, 1991]. This approach implements disjunction by a restart rule that can be highly ambiguous from the point of view of search. When using a program clause $A \vee B$ in a derivation, we may know to use a privileged disjunct (say A) to contribute to the proof of the current goal G; however, the other disjunct (B, here) may contribute to any earlier goal in the derivation of G. Thus, after the overall proof is completed using A as an assumption, we must be prepared to restart the entire proof using the assumption B instead. By making a program clause $A \vee B$ modular using a modal specification, we can constrain this restart process. With the modular program, A must contribute to the proof of the current goal G; but now B may contribute only toward earlier goals in its modular context. By constraining the scope of restarts, modular disjunction limits the size of proofs and the kinds of interactions that must be considered in proof search. Such constraints are crucial to the use of logical techniques in applications that require automatic assessment of incomplete information, such as planning and natural language generation.

The correct design of our extended language requires a variety of proof-theoretic ideas about logic programming to be extended, strengthened, and combined with proof-theoretic results about modal logic in a novel way. To describe logic programming, we start with the idea of uniform proof search described in [Miller et al., 1991] and extended to different kinds of disjunctions in [Miller, 1994, Nadathur and Loveland, 1995]. In uniform proof search, proof rules alternately decompose goals to atoms and then match these atoms against program clauses; this allows logical connectives to be viewed as instructions for search.

To derive a uniform proof system in the presence of existential quantifiers, however, we can no longer use the familiar quantifier rules used in previous logic programming research, which simply introduce fresh parameters; we must apply a generalization of Herbrand's theorem [Lincoln and Shankar, 1994] and work with quantifier rules that introduce structured terms. Moreover, to handle modal operators in uniform proof, we use a path-based sequent calculus for modal logic that assimilates modal proof to classical proof. Path-based techniques for reasoning in modal logic, pioneered by Fitting [Fitting, 1972, Fitting, 1983], have been extensively studied in recent years [Wallen, 1990, Ohlbach, 1991, Nonnengart, 1993, Auffray and Enjalbert, 1992]; a combined path-based Herbrand calculus for multi-modal logic is developed in detail in [Stone, 1999a]. The key property of this calculus is that inferences can be freely interchanged. This allows arbitrary proofs to be transformed easily into uniform proofs. But the very same property prevents this calculus from enforcing modularity in the *structure* of proofs. For example, with free interchange, inferences can be pulled outside the region of a proof where some local assumption is introduced, even when those inferences use information provided by that assumption.

To guarantee the modular behavior of the uniform system, we require a new, detailed analysis of *cancellation* in disjunctive logic programming. Modular restarts depend on the invariant that any disjunct assumed in case analysis is used in proof, or *cancelled*, before another restart is attempted. In some cases such cancellation requires the second disjunct of a clause to be used before the first. The DIALUP proof system achieves the invariant without reporting duplicate proofs using a new discipline that temporarily suppresses the cancelled assumption on first restarting for a disjunct analyzed out of order. This DIALUP regime is justified by transformations restricting uniform proofs.

At the same time, the modular behavior of the uniform system depends on proof-theoretic analyses of path-based sequent calculi adapted, in part, from [Stone, 1999b]. These analyses establish that path representations enforce modularity and locality in the uses of formulas in proofs, even with otherwise classical reasoning. Hence, although path-based calculi obscure the natural modularity of modal inference, they do not eliminate it. The operational rules of DIALUP are obtained by streamlining the uniform proof system to take advantage of these results; as a consequence, the interpreter can dynamically exploit the local use of modular assumptions.

The remainder of this paper is organized as follows. We outline the concrete motivations and intuitions behind DIALUP in Section 2. Next, we present an extended, precise description of DIALUP's behavior in Section 3. We give the logic that justifies DIALUP in Section 4; some additional observations that streamline the implementation are sketched in Section 5. We provide an extended example of the system in use in Section 6.

2 Motivation and examples

DIALUP accepts program statements of the syntactic category D and goals of the category G defined recursively as follows.

(1)
$$G ::= A \mid [\textsc{m}]G \mid G \wedge G \mid G \vee G \mid [\textsc{m}](\forall x G) \mid \exists x G \mid [\textsc{m}](D \supset G)$$
$$D ::= A \mid [\textsc{m}]D \mid D \wedge D \mid D \vee D \mid \forall x D \mid \exists x D \mid G \supset D$$

In (1), A schematizes an atomic formula; atomic formulas take the form $p(a_1, \ldots, a_k)$ where p is a predicate symbol of arity k and each a_i is either a variable or an atomic constant in some set C.

In (1), $[\textsc{m}]$ schematizes a modal operator of necessity; intuitively, such modal operators allow a specification to manipulate constrained sources of information. That is, a program statement $[\textsc{m}]D$ explicitly indicates that D holds in the constrained source of information associated with the operator $[\textsc{m}]$. Conversely, a goal $[\textsc{m}]G$ can be satisfied only when G is established by using information from the constrained source associated with $[\textsc{m}]$.

We will work in a multi-modal logic, in which any finite number of distinct necessity operators may be admitted. (Necessity operators will also be written as $\Box$ or $\Box_i$.) In addition to ordinary program clauses, a DIALUP specification may contain any of the following axiom schemes describing the modalities to be used in program clauses and goals:

(2)

	$\Box_i p \supset p$	*veridicality* (VER)
	$\Box_i p \supset \Box_i \Box_i p$	*positive introspection* (PI)
	$\Box_i p \supset \Box_j p$	*inclusion* (INC)

These axioms describe the nature of the information that an operator provides, and spell out relationships among the different sources of information in a specification. (VER) is needed for information that correctly reflects the world; (PI), for information that provides a complete picture of how things might be; and (INC), for a source of information, j, that elaborates on information from another source, i.[2] Because DIALUP uses this explicit axiomatization, we can take the names of the modal operators as arbitrary.

We can appeal to modal operators in specifications both for their expressive power in characterizing domains, and for their operational force in constraining logic programming inference. As an illustration of DIALUP's expressive resources, we consider a specification of communicating agents in Section 2.1. In this specification, modal operators represent the sources of information available to any agent individually, and the sources of information that groups of agents share. With these expressive resources, the specification accommodates the basic fact that each agent's knowledge is limited but may increase as messages are exchanged.

As an illustration of the the operational force of DIALUP's modal reasoning, we consider a specification of a planning problem in Section 2.2. In this specification, modal operators identify restricted sources of information that can be used to construct separate solutions for the subproblems of the overall task. Because it invokes these operators to characterize dependencies in the problem-solving task, the specification can be executed without considering interactions in subtasks that might otherwise potentially arise.

What these examples have in common is their dependence on the *modularity* of modal logic. Modal formulas describe not only what needs to be derived in a proof, but also how derivations should be broken down into parts and what information should be taken into account in each. As we shall see in Sections 3, 4 and 5, some subtleties are involved in describing and justifying a correct inference procedure that does break down derivations into modular parts and does consider restricted information in each. With these examples, then, we also motivate our inference procedure and survey the modularity it can offer.

2.1 Characterizing Domains with Indefinite Modal Specifications: Communicating Agents

In collaborative tasks, agents coordinate their decisions in pursuit of a common goal. To succeed in such collaboration, agents may need to communicate. For example, one agent may need to supply its partner with key information which will allow the partner to decide what to do next. Planning a contribution to conversation therefore requires agents to draw inferences about their partners'

[2]For those used to thinking of modality semantically, in terms of accessibility relations among possible worlds, (VER) can be captured by a reflexive accessibility relation, (PI) can be captured by a transitive accessibility relation, and (INC) can be captured by if j-accessibility entails i-accessibility.

changing information. Of course, agents generally cannot characterize what their partners know specifically. More frequently, each agent has an incomplete, abstract description of the other.

This section shows how communicating agents in such circumstances can assess the effects of communicative actions, and can thereby plan contributions to a collaborative exchange, by posing queries against DIALUP specifications. These specifications explicitly model the partial and increasing body of information that communicating agents enjoy. To do so, they crucially exploit nested implications and existential assertions.

Our example involves a patron P accomplishing transactions with a bank teller T. We begin by adopting a modal perspective on the information of these agents. We use the operator $[\textsc{p}]$ to represent the knowledge of the patron P, and we use the operator $[\textsc{t}]$ to represent the knowledge of the teller T. In addition, we appeal to a restricted body of shared information that the two agents maintain as part of their collaboration, a *conversational record* [Clark and Marshall, 1981, Thomason, 1990], represented by the operator $[\textsc{cr}]$. The eight formal rules governing these modalities, given in (3), represent a reasonable idealization of conversation [Stone, 1998b].

(3)
$$\begin{array}{cr}
[\textsc{p}]p \supset p \quad [\textsc{t}]p \supset p \quad [\textsc{cr}]p \supset p & (\textsc{ver}) \\
[\textsc{p}]p \supset [\textsc{p}][\textsc{p}]p \quad [\textsc{t}]p \supset [\textsc{t}][\textsc{t}]p \quad [\textsc{cr}]p \supset [\textsc{cr}][\textsc{cr}]p & (\textsc{pi}) \\
[\textsc{cr}]p \supset [\textsc{p}]p \quad [\textsc{cr}]p \supset [\textsc{t}]p & (\textsc{inc})
\end{array}$$

We motivate DIALUP specifications for the patron P and the teller T by considring the DIALUP queries that these agents might use to infer opportunities for efficient communication. To begin, suppose that P has asked T whether P has sufficient funds to withdraw \$50 without penalty. Suppose further that T decides to convey two facts in response: that P does have the required funds, which we abbreviate to q; and that in fact P's account contains \$600, which we abbreviate to r.

T might realize these facts efficiently, by reasoning as follows. T knows that P will be drawing conclusions from T's communication as T produces it. Once T communicates r, q will be clear to P by inference. Accordingly, it suffices for T to streamline the response from both r and q to r alone. As described by [Green and Carberry, 1994], such inferences—reducing a compound communicative act to a simpler one when the other effects are clear from context—are characteristic of indirect answers in discourse.

We can formalize this inference as a modal query by drawing on the link between sources of information and modal operators. To start, T envisages the consequences of communicating r. That is, T restricts attention to developments of the situation compatible with what T knows, as represented by the content of $[\textsc{t}]$, and T assumes further that r is put on the record, so that $[\textsc{cr}]r$ is true. Then, in that hypothetical context, T tests whether q can also be taken as part of the conversational record. These operations correspond to the query in (4), a statement formulated in terms of modal operators and an implication nested within a query.

(4) $[\textsc{t}]([\textsc{cr}]r \supset [\textsc{cr}]q)$

So proving (4) would justify the use of r as an indirect answer, abbreviating r and q.

Now, in an evenly matched dialogue, it will be just as important to *ask* questions as to answer them. For a more interesting example, then, consider a similar information-gathering exchange from the patron P's point of view. In formulating a question, P may have had to provide background information to T in order for T to be able to provide an answer, as in (5).

(5) I have account 42. What is my balance?

It is natural to assume that *P*'s background assertion, like *T*'s reply, is calculated for its effect on the conversational record. Indeed, by making use of existential specifications and queries, we can formalize this calculation in terms of the same schema as (4).

Here is how. If *P* aims for *T* to provide an answer, then *P* must hope to establish that *T* *knows* the answer. Following [Hintikka, 1971], we can formalize this condition as $\exists b[\mathrm{S}]bal(P,b)$: there is a specific real-world value b which *T* knows to be *P*'s balance. The contribution *P* makes—$acct(P,42)$—is intended to put this condition on the conversational record. By an analogous argument to that which suggested (4), this leads to a modal query with nested implications by which *P* might justify or select the proper background to the question:

(6) $[\mathrm{P}]([\mathrm{CR}]acct(P,42) \supset [\mathrm{CR}]\exists b[\mathrm{T}]bal(P,b))$

For this query to represent a fruitful model of *P*'s conversational reasoning, however, it must access an *indefinite specification* outlining shared information about *T*. The specification must entail that *T* would know the answer given the background, but it cannot entail that *P* would know the answer. Otherwise, *P* will have no reason ever to ask a question. The existential statement in (7) provides such a specification.

(7) $[\mathrm{CR}]\forall i(\exists x[\mathrm{T}]acct(i,x) \supset \exists b[\mathrm{T}]bal(i,b))$

In effect, (7) registers the teller's ability to look up the balance of any account as part of the common ground; formally, it says it is common knowledge that if *T* knows what *i*'s account is, then *T* knows what *i*'s balance is.

Informally, we can see why (7) entails (6), given the modal theory of (3). The query considers *P*'s hypothetical view of the shared context, and asks us to obtain a specific conclusion about what *T* knows according to this view. Now both the shared background—(7)—and hypothetical additions to it—$[\mathrm{CR}]acct(P,42)$—retain their shared status in the queried view. Given $[\mathrm{CR}]acct(P,42)$, it follows that $\exists x[\mathrm{T}]acct(i,x)$ for $i = P$ (and $x = 42$); *T* knows what is shared. Thus (7) applies and the needed conclusion follows.

In this paper, I will show how this reasoning is captured straightforwardly by a logic programming search strategy for reasoning with indefinite modal specifications. Of course, the strategy also respects the restricted logical scope of existential quantifiers—thus, with (7) we can infer that *P* has a balance, but cannot infer that *P* knows what that balance is.

While (4), (6) and (7) are surely toy examples, I will suggest that indefinite specifications and hypothetical queries provide an attractive perspective for reasoning about contributions to dialogue generally. A generation system can produce a concise and precise discourse only by drawing inferences about how the hearer will interpret that discourse as dependent on, and as an update to, the conversational record. Such inference is naturally founded on a declarative framework, like the modal specification presented above, for describing the knowledge of the participants in the conversation.

In order to give a more accurate picture of the dynamics of dialogue and the dependencies of utterances on shared context, the logical content of utterances can be represented more precisely, in terms of *presuppositions* and *assertions* [van der Sandt, 1992, Stone and Webber, 1998]. Moreover, a treatment of dialogues with multiple utterances can be obtained by appealing to AI formalisms of knowledge and ability [Davis, 1994, Stone, 1998a], and introducing a nested implication for each step of action. As outlined in Section 6, these two features allow a much more detailed version of the teller's and patron's exchange above to be specified and validated in DIALUP.

The strength of this inferential model is that the presuppositions and other facts that this discourse relies on can be explicitly represented without being explicitly communicated. In a robust system, such facts cannot be ignored altogether; for instance, they are needed to answer questions of clarification [Moore and Paris, 1993]. However, they cannot be uttered either—imagine the implicatures ensuing from: "I have an account. It's number 42. My account has a balance. What is it?". Such distracting restatements of the obvious have plagued earlier conversational agents, such as [Power, 1977, Houghton, 1986, Cassell et al., 1994].

2.2 The Operational Force of Indefinite Modal Reasoning: Modular Search

A record of the possible interactions that may arise in problem-solving can be an important part of a specification of how to reason in a domain. Here is a simple example.

In planning a trip, it is important to determine before you begin that you will be able to complete the trip successfully. To be stranded midway would be a real disaster. Often, however, many details about the trip cannot be resolved in advance. For such situations, showing that the trip will be successful means showing that you will be able to negotiate these details when the time comes, no matter how they turn out. What makes it possible to quickly derive confidence in a planned journey is the knowledge that such details cannot conspire together to require global revisions of the plan.

Thus, suppose one leg of a journey involves taking an early train. At the station, you have to get a ticket for the train and (if you're like me) get a cup of coffee. Tickets can be purchased from a teller at a window or from automatic machines; the windows can have prohibitive lines and the automatic machines can be out of order, but the station management always makes sure that one quick and easy method is available. Similarly, there are a couple of places to get coffee at the station; you can be sure at least one will be open at any time trains are leaving, but since their hours vary, you may not know which. Using the abbreviations in Figure 3, we might formalize this situation by the logic program of Figure 1. (The representation of these abilities by atomic propositions is a harmless abbreviation that allows the *modularity* of the example—our main focus—to shine through. As we will see in Section 6, we could also represent abilities in DIALUP using complex formulas in a logic of knowledge and action, formulas more like those seen already in the example of Section 2.1.)

In general, without knowing more about a specification, we can expect a number of cases to be considered in proof search that is exponential in the number of ambiguities in it. Here, for example, searching automatically for proofs of *take-journey* is likely to require showing that *take-journey* holds independently in the four cases that the program specifies (cases in which we assume either *use-machine* or *use-window* and assume either *visit-donuts* or *visit-starbucks*). We will focus on the use of a search strategy based on a naive restart rule; this algorithm performs case analysis by completing proof search with one disjunct then attempting a fresh proof of the main query ("restarting") using the other disjunct as an assumption. Consider proving *take-journey* this way. The first subproof of *take-journey* takes care of the *use-machine* and *visit-donuts* case by using the first disjuncts. This subproof requires two restarts of the goal *take-journey*, once assuming *use-window* and once assuming *visit-starbucks*—leading to a proof of *take-journey* from *use-window* and *visit-donuts* and a proof of *take-journey* from *use-machine* and *visit-starbucks*. The last case, proving *take-journey* from *use-window* and *visit-starbucks*, arises as a restart goal in both of these subproofs. This is good restart behavior; in multiple case analyses, subproofs can reintroduce cases so search never terminates [Loveland, 1991]. Expanding the search space can delay the point where this happens, but cannot avoid difficulties if the full space must be explored—whether because all solutions are

$$\begin{array}{ll}
\multicolumn{2}{c}{\textit{get-ticket} \wedge \textit{get-coffee} \supset \textit{take-journey}.} \\
\textit{use-machine} \supset \textit{get-ticket}. & \textit{visit-donuts} \supset \textit{get-coffee}. \\
\textit{use-window} \supset \textit{get-ticket}. & \textit{visit-starbucks} \supset \textit{get-coffee}. \\
\textit{use-machine} \vee \textit{use-window}. & \textit{visit-donuts} \vee \textit{visit-starbucks}.
\end{array}$$

Figure 1: Unstructured logic program

$$\begin{array}{ll}
\multicolumn{2}{c}{[\textsc{ticket}]\textit{get-ticket} \wedge [\textsc{coffee}]\textit{get-coffee} \supset \textit{take-journey}.} \\
[\textsc{ticket}](\textit{use-machine} \supset \textit{get-ticket}). & [\textsc{coffee}](\textit{visit-donuts} \supset \textit{get-coffee}). \\
[\textsc{ticket}](\textit{use-window} \supset \textit{get-ticket}). & [\textsc{coffee}](\textit{visit-starbucks} \supset \textit{get-coffee}). \\
[\textsc{ticket}](\textit{use-machine} \vee \textit{use-window}). & [\textsc{coffee}](\textit{visit-donuts} \vee \textit{visit-starbucks}).
\end{array}$$

Figure 2: Modular logic program

Symbol	Content
take-journey	I can have a successful train-trip
get-ticket	I can obtain a ticket
get-coffee	I can obtain coffee
use-machine	I can use an automatic ticketing machine
use-window	I can use a teller's window
visit-donuts	I can get coffee from Dunkin Donuts
visit-starbucks	I can get coffee from Starbucks

Figure 3: Interpretations of symbols for the example

needed or because a dead end must fail before an alternative is tried.

In fact, the ambiguities in this problem are independent. Where one gets one's ticket doesn't affect where one gets one's coffee, and vice versa. The specification of Figure 1 omits this fact, and as a result an automatic system must search for different ways of proving *get-coffee*, depending on whether it has assumed *use-machine* or assumed *use-window*. If we provide a better specification, including the knowledge that these alternatives are independent, the search strategy will be able to break up the proof in advance. It will restrict the alternatives from $\textit{use-machine} \vee \textit{use-window}$ to proving *get-ticket*, and it will restrict the alternatives from $\textit{visit-donuts} \vee \textit{visit-starbucks}$ to proving *get-coffee*. In general, when alternatives are specified not to interact, worst-case proof size increases only linearly as new independent ambiguities are added. This makes for fast failure as well as fast success.

We will use modal specifications to indicate that alternatives do not interact. Such a specification is given for our train problem in Figure 2. The specification invokes two necessity operators, $[\textsc{ticket}]$ and $[\textsc{coffee}]$, to distinguish the goals and program clauses describing getting a ticket from those describing getting coffee. A simple metaphor for understanding why the ambiguities in the resulting specification must be interpreted independently derives from the well-known use of modal operators to describe the knowledge of agents. By this metaphor, Figure 2 describes how problem-solving tasks in catching a train can be assigned to separate problem-solving agents with specialized information—an agent *T* that knows just about tickets and an agent *C* that knows just about coffee.

This metaphor leads directly to intuitions about search. The problem of getting a ticket is assigned to agent *T* by the goal [TICKET]*get-ticket*. *T* has certain alternatives to consider in getting the ticket, *use-machine* or *use-window*. *T* considers these alternatives and no others in solving its task. Likewise, the problem of getting coffee is assigned to agent *C* by the goal [COFFEE]*get-coffee*. In solving this problem, *C* considers just its alternatives: *visit-donuts* or *visit-starbucks*. Since the two agents are reasoning separately about different goals and ambiguities, the record of their problem solving is just a combined record of their independent steps—not, as before, an interacting record with combined resolutions of ambiguities.

In this paper, we will see how this intuitive account of modularity in search can be realized formally in a logic programming interpreter. In brief, the interpreter must keep track of the uses of assumptions (such as *use-machine*), in order to determine which modular context its inferences currently contribute toward (for example, *T*'s information). The interpreter must keep track of alternative modular goals (for example, both *take-journey* and [TICKET]*get-ticket*), in order to reconsider only goals that match this context. Finally, the interpreter must orchestrate the subsequent uses of assumptions once goals are restarted (such as the assumption *use-window* for a restart of [TICKET]*get-ticket*), in order to ensure that the interpreter's subsequent inferences remain within the context it is committed to. Because any modal proof can be transformed to respect the interpreter's discipline, the interpreter can reason correctly about a specification while considering only a restricted set of proofs with this explicit modular structure.

3 The DIALUP interpreter

In Section 2, we have seen that modal operators can endow specifications with a notion of modularity that suggests intuitions about content and proof-search. In this section, we shall describe the data structures that DIALUP uses to realize this modularity, and then present a precise description of DIALUP's modular operational behavior. We begin by situating DIALUP within the program of *abstract logic programming languages* [Miller et al., 1991].

3.1 Abstract Logic Programming

Logic programming languages embody simple, specific search procedures for building proofs. At each step in logic programming search, the goal is to find a way to use the available assumptions to establish a specific query. If the query is complex, its logical structure directly determines the available alternatives for search. Thus, logical symbols in queries can be seen as instructions for decomposing and transforming the search problem that the interpreter faces. Similarly, the atomic formulas that an assumption can be used to derive—the *head* (or heads) of that assumption—serve as indexes that regulate whether an assumption can be applied. And the logical structure of the assumption provides an instruction for creating a set of new search problems whenever the assumption is used.

This general perspective on logic programming has been formalized and analyzed under the framework of *abstract logic programming languages* [Miller et al., 1991]; the framework has been applied to intuitionistic, linear and classical logics [Miller et al., 1991, Hodas and Miller, 1994, Miller, 1994, Nadathur and Loveland, 1995]. Mathematically, this formalization begins by establishing a correspondence between search problems posed to the logic programming interpreter and sequents in a proof calculus. The action of the interpreter in transforming search problems can then be seen as the construction of a proof in a restricted sequent calculus; the rules of this calculus are specialized so as to model the constrained search entertained by the logic programming interpreter. The key result required to show the correctness of a logic programming language in this framework is then to show that the restricted sequent calculus permits a derivation of a goal exactly when the goal is provable in a general sequent calculus for the logic.

In some cases, the logic programming proof system can be quite familiar. In [Miller et al., 1991], for instance, any goal of the interpreter is represented as an ordinary sequent $\Gamma \longrightarrow G$. The interpreter is distinguished as building *uniform* proofs, where ordinary sequent rules are used in a distinguished order. More generally, however, a logic programming proof system must be augmented with information to correctly capture the state of the interpreter. For example, the structure of sequent rules may have to distinguish the current goal, as well as the program clause currently being matched against that goal. This gives a *focusing proof system*, whose deductions are called *focusing proofs* [Andreoli, 1992]. Then the correctness of the logic programming language lies in showing that the restricted proof system is sound and complete—that every proof in the ordinary system can be transformed into a proof with the restricted form, and vice versa.

To describe DIALUP search in particular, sequents must be augmented with two kinds of information that enforce the modularity of DIALUP programs: specifications of *possible worlds* constraining the use of formulas; and records of *restarts*, *cancellations* and *suppressions* constraining case analysis for disjunctive assertions.

3.2 Possible Worlds

A modal query $[\mathrm{M}]G$ must be established solely from program statements that describe the content of $[\mathrm{M}]$. Some proof systems [Onishi and Matsumoto, 1957, Kanger, 1957, Fitting, 1983] and modular logic programming languages [Miller et al., 1991, Giordano and Martelli, 1994] restrict applicable formulas directly by syntactic criteria. However, DIALUP achieves the restriction by drawing on modal semantics [Kripke, 1963] and semantics-based path deduction for modal logic developed by Fitting [Fitting, 1972, Fitting, 1983] and investigated computationally in [Wallen, 1990, Ohlbach, 1991, Nonnengart, 1993, Auffray and Enjalbert, 1992].

In this scheme, the language of logical formulas is extended in proofs to permit explicit reference to possible worlds; in proofs, each program statement and each query is associated with a term recording the possible world where it holds. Instead of naming a world atomically, these terms name the sequence of transitions or *path* that is needed to reach a world from a designated starting point (the real world). Before using an atomic assertion to establish an atomic query, DIALUP must equate the possible-world paths associated with the two statements. The solution to such an equation must respect the types of logic variables and parameters for transitions, and must take into account an equational theory encoding the applicable (VER), (PI) and (INC) axiom schemes. Because of this, this equation implements the constraint that the program statement specifies the kind of information that the query describes. In writing path terms, I will use α and β etc. to name parametric transitions; x and y will represent logic variables over transitions; and μ, ν are metavariables over whole paths. I represent the association between a proposition P and a world μ by superscript: P^{μ}.

The logical treatment of connectives in DIALUP can be seen as alternating translation of modal formulas into classical formulas based on modal semantics with classical reasoning about the translations. For example, to reduce a query $[\text{M}]G^{\mu}$, DIALUP introduces a parametric transition of $[\text{M}]$-accessibility, α, and considers a new query $G^{\mu\alpha}$. Dually, to apply a statement $[\text{M}]P^{\mu}$, DIALUP introduces a fresh logic variable x over transitions of $[\text{M}]$-accessibility and considers applying the statement $P^{\mu x}$. DIALUP's other, classical connectives are interpreted by applying classical reasoning at the current world. For example, a query $(G_1 \wedge G_2)^{\mu}$ is broken down into the two queries G_1^{μ} and G_2^{μ}.

3.3 Restarts, Cancellations and Suppressions

DIALUP's approach to disjunctive assertions refines the restart rule of Near-Horn Prolog [Loveland, 1991]. In this regime, a logic programming derivation consists of a series of *blocks*. Each block analyzes an overall query in a single way for a single case described by the program. Thus, to apply a disjunctive assertion $A \vee B$ in a derivation, we first complete the current block, using one disjunct (say A) directly to establish the current goal G. Then we introduce a new block in which the other assumption is available (say B) and an appropriate new query H is posed. Recall that we cast any logic programming inference as the application of an appropriate sequent rule, so the logic programming derivation is a proof in a sequent calculus. We regard the restart treatment of disjunction in particular as a sequent rule, as schematized in (8):

(8) $$\frac{\ldots, A \longrightarrow G \qquad \ldots, B \longrightarrow H}{\ldots, A \vee B \longrightarrow G}\,(\vee \rightarrow)$$

The root of this sequent represents a state of the interpreter which applies $A \vee B$. The left subtree continues the block by reasoning from A; the right subtree starts a new block by reasoning from B to establish the new goal H. Thus, this formalism realizes a *block* by a maximal tree of contiguous inferences in which the right subtree of any $(\vee \rightarrow)$ inference in the block is omitted. See [Nadathur and Loveland, 1995, Nadathur, 1998] for more on the relationship between the restart rule and the structure of sequent calculus derivations.

The *restart* discipline in (8) is encoded by the new goal H which we prove using the new assumption B; the precise sequent rule that describes the interpreter must account for the choice of H. In classical logic, it suffices in the new block to consider the original, overall query given to the interpreter. To obtain modularity for modal logic, we need a more restrictive policy.

A key fact behind this policy is that the restart rule in (8) ultimately treats the assumptions A and B on a par, despite the apparent asymmetry in the goals G and H associated with them. Now, the symmetry must follow from the logical correctness of the rule, but we can motivate it more precisely by considering the schematic structure of a block in which (8) is used. The logic programming derivation which that block initiates can be represented as follows.

$$\begin{array}{c}
\begin{array}{cc} & [D_B] \\ D_A & D_H \\ \ldots A \longrightarrow G & \ldots B \longrightarrow H \end{array} \\ \hline
\ldots, A \vee B \longrightarrow G \\ \hline
D_\vee \\ \Sigma_G \\ \hline
D_G \\ \Sigma_H \\ \hline
D_0
\end{array} \tag{9}$$

At the root are performed inferences D_0, possibly empty, which introduce the restart goal H. At this point the state of the interpreter is represented by Σ_H, and H is in fact the goal the interpreter is considering (or is a goal the interpreter *could* consider at this restart step, if D_0 is empty). Further inferences D_G (again possibly empty) follow, until the interpreter reaches the goal G for which case analysis is introduced, at state Σ_G. Now any inferences $D_\vee$ are required to reach the disjunction, and the case analysis for it is introduced. Within this block, we perform inferences D_A to use the first disjunct. In the other block, we perform inferences D_H; at various stages here we will entertain *subproofs* which we schematize D_B, where inferences will be applied to B and further reasoning undertaken.

By rearraging these inferences, we can construct an alternative derivation in which case analysis for B is considered first. This derivation has the form schematized in (10).

$$\begin{array}{c}
\left[\begin{array}{c}
\begin{array}{cc} & D_A \\ & \ldots A \longrightarrow G \\ \cline{2-2} D_B & D_G \\ \ldots B \longrightarrow H' & \ldots A \longrightarrow H \end{array} \\ \hline
\ldots A \vee B \longrightarrow H' \\ \hline
D_\vee
\end{array}\right] \\ \hline
D_H \\ \Sigma_H \\ \hline
D_0
\end{array} \tag{10}$$

That is, we derive the goal H again using inferences D_0, but now we immediately apply any reasoning D_H that we had previously applied towards H in the B block. Within this reasoning, there are a number of places where access to B (using inferences D_B) is called for. At each such place, we provide this by first reasoning with $D_\vee$ to access the disjunction, and then handling the B case using the original reasoning. That now leaves an A case, which we treat using a restart to the goal H. The new block contains the reasoning D_G previously applied after H in the first block of (9) followed by the reasoning D_A previously applied to discharge the A case there.

The derivations in (9) and (10) show that it is possible to treat logic programming case analysis in either order, even using an asymmetrical restart rule. Of course, the alternative proofs may differ in size—(9) can be much smaller if B is used many times in D_H. More importantly, the alternative proofs differ in how they assign reasoning to blocks. Both proofs locate inferences D_0 and $D_\vee$ within the first, lower block. But whereas (9) locates D_A and D_G in the first block and locates D_H (and any D_B's) in the second, (10) locates D_H (and any D_B's) in the first and locates D_A and D_G in the second.

Now, let us use this symmetry to describe the order in which cases should be treated, and the restart goals we should consider in case analysis. We consider the inference figure of (8), and suppose that the assumption B is made at a particular ground path ν. So if we can guarantee that B^ν will be used in the new block, we can restrict the restart to goals H^μ where ν is a prefix of μ—to *modular* restarts. Adapting terminology from [Loveland, 1991] to this restricted context, I will refer to any use of a disjunctive premise in the first block of case analysis where it is assumed as a *cancellation*. If we have cancellations, we can enforce modular restarts.

Having cancellations is already very attractive from the point of view of implementation. It gives a simple invariant that drastically prunes the search space for the logic programming interpreter. We can illustrate this by observing that our hypothetical disjunct B^ν must be used in some block after it is assumed. Otherwise these later blocks form the basis of an independent proof of the query, which the interpreter must already consider elsewhere in its search space; the case analysis at $(A \vee B)^\nu$ is superfluous. Such redundancies in search must be pruned in practice.

Thus, by considering the right combination of disjuncts in the block where B^ν is introduced, we can and should guarantee the cancellation for B^ν in that block. We cannot say in advance, however, which disjuncts must be introduced in this block. For example, suppose the program contains another disjunctive assertion $C \vee D$ that we must use to establish the query. If B combines only with C in inference to prove the query, we must be prepared to consider the C case first to obtain a cancellation—this would mean that the derivation from the assumption B with case analysis for $C \vee D$ has the form schematized by (9). However, if B combines only with D, we must be prepared to consider the D case first to obtain a cancellation—the derivation from the assumption B with case analysis for $C \vee D$ should have the form schematized by (10). The danger this raises is that B might be required in cases both for C and D. The same proof could then be reported twice: once containing a subproof like (9), with C considered first, then D; and again containing a subproof like (10), with D considered first, then C.

Let us say that a disjunct D is *exceptional* in a block if D occurs as part of a disjunctive program statement in which other disjuncts precede D (textually), but D is the disjunct that is immediately analyzed in this block in a use of that program statement. Call the disjuncts that precede D *delayed*, and by extension, call the block in which D is treated an *exceptional* block and the blocks in which the delayed disjuncts are first treated *delayed* blocks. For example, then, B counts as an exceptional disjunct for any case analysis of $A \vee B$ in the lower block in (10), and that block is an exceptional block. Meanwhile, A is a delayed disjunct for any case analysis of $A \vee B$ in the lower block in (10) and the blocks in (10) where we restart by A are delayed.

The ambiguity just illustrated shows that the only occasions when we should consider an exceptional disjunct (and make the current block an exceptional one) are those which enable a cancellation for a premise introduced in the current block that would not get a cancellation here otherwise. We can refer to these as the *key* premise and the *key* cancellations of the exceptional block.

These occasions can be identified by conditions on the inferences that constitute exceptional and delayed blocks. First, the *delayed* blocks cannot have given rise to cancellations for the key premise of the exceptional block. In terms of the schemas of (9) and (10), there can be no cancellations for the key premise in the inferences D_H and D_A which make up the delayed block and which would have appeared in the first block if the disjunct had not been delayed. We can ensure that no such inferences are entertained by *suppressing* that key premise—temporarily preventing this premise from being used—during any block where a delayed disjunct is first treated.

Second, the *exceptional* block cannot introduce a key cancellation inference in the reasoning it implicitly shares with the delayed block. In terms of the schemas of (9) and (10), there can be no cancellations for the key premise in the inferences D_0 and $D_\vee$ which appear in the lower block regardless of which case is analyzed first. We take separate strategies to protect D_0 and $D_\vee$. In protecting $D_\vee$, we need to avoid a key cancellation during the proof of the disjunction itself. To avoid this, we will again suppress the key premise—this time for the proof of goals introduced by backward chaining to establish exceptional disjuncts. There can be no suppression with D_0, because we have inferences that have perhaps already been performed when case analysis is considered. To protect D_0, we will restrict the set of restart goals we consider. A lower restart goal H will be associated with a smaller body of inferences D_0; thus, selection of a sufficiently low restart goal will eliminate any key cancellations from the corresponding D_0. We will call such a goal a *free* restart goal: we restart delayed blocks only to free restart goals. The stack-based data structures of logic programming proof search makes free restart goals easy to identify.

This reasoning accounts for the discipline of modular restarts, cancellation, and suppression that DIALUP follows. DIALUP records the initial query and any subsequent query where a new world parameter is first introduced as a possible restart goal. When case analysis is introduced for one disjunct, DIALUP records the need to restart to one of these goals with the remaining disjuncts (moreover, the disjunct analyzed immediately cannot be exceptional unless a cancellation is needed). At restart-time, DIALUP picks a restart goal G^μ such that the new assumption P^ν has ν a prefix of μ, for modularity; for delayed disjuncts, G^μ is also chosen to be free. P^ν is a key assumption that must be cancelled in the new block; previously suppressed assumptions are now exposed, and, if the block is delayed, the previous key assumption is suppressed. Finally, once the block is completed, DIALUP checks for a cancellation of the key premise P^ν. If there is none, the proof under construction is discarded as redundant.

3.4 Operational Rules

We can now describe DIALUP's operational behavior more precisely. We write a DIALUP task as a judgment $(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} G^\mu$, indicating that DIALUP has to derive the goal formula G at world-path μ using the program Γ. Within the program Γ we optionally distinguish a formula P^ν that describes the current state of the interpreter in applying a particular program clause to the current goal.

The records κ and κ_I provides information about the overall structure of the proof being constructed; DIALUP's computation in the proof search task determines additional structure of the proof, as returned in κ_O. In regarding the context for the proof search problem in terms of input and output records associated with it, we follow the technique adopted for example in [Lincoln and Shankar, 1994]. In these context records, we abstract a number of representations that the interpreter accumulates (according to a straightforward discipline) about values of variables and analysis of cases.

κ, records features of the proof that are already set; we represent κ as a tuple of the form $\langle G; K; s^?; F \rangle$. G gives the potential restart goals that have been introduced; K gives the key premise for cancellation in this block, if any; $s^?$ indicates whether any formulas are currently suppressed; F gives a pointer to the final list of restart goals that will count as free in the current block.

κ_I allows partial information to accumulate about the overall structure of the proof. We represent κ_I as a tuple of the form $\langle C; E; c^? \rangle$. C records what elements and logic variables have been have been introduced in the proof for paths and first-order terms and what equations on these variables must be solved to complete the proof; E records the restart goals that have not been ruled out as free given the contributions made so far to the current block; $c^?$ records whether any cancellation has taken place thus far in the block.

Finally, we represent κ_O with a tuple of the form $\langle C; E; c^?; (d^?)\rangle$. C, E and $c^?$ record values for the provisional constraints, the provisional free restart goals and the provisional cancellations, given the additional information accumulated during the current proof task; and $d^?$ indicates whether the use of the active current program clause (if any) involves the creation of delayed disjuncts (this is only meaningful when breaking down program clauses).

Suppose we have a specific program Γ consisting of modal formulas and a specific goal consisting of a modal formula G; We regard each of these expressions as a formula prefixed with the empty prefix; we formulate an initial state of proof of the form

$$\kappa = \langle G;;;F\rangle, \kappa_I = \langle ;G; \mathit{false}\rangle$$

This indicates that G is the only available restart goal; that there is no formula to cancel, no suppressions in force, no constraints that yet need to be solved, no cancellation that has yet occurred, and only G as a possibly free restart goal. We use the rules defined below in (11)–(15) to establish the judgment

$$(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} G$$

for some κ_0. In case κ_O takes the form $\langle C; F; c^?\rangle$, supplying constraints C on values of variables which can be satisfied by an appropriate substitution of values to variables and supplying goals F that fill the placeholder supplied for the free restart goals of the block, the result provides a DIALUP answer for the query G against the program Γ. Sections 4 and 5 argue that there is such an answer just in case G holds in all modal models at all worlds where Γ is true, according to the usual Kripke semantics.

We begin by specifying the instructions for search that break down complex goals into atomic ones. These cases apply in handling the task $(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} G^\mu$ whenever G is not an atomic formula; the rule selected is a function of the structure of G as specified in (11).

(11) a If G is of the form $B \wedge C$, search proceeds by first solving $(\kappa, \kappa_I, \kappa_{O1})\Gamma \xrightarrow{?} B^\mu$ and then continues by solving $(\kappa, \kappa_I', \kappa_O)\Gamma \xrightarrow{?} C^\mu$. Since the first task yields a record $\kappa_{O1} = \langle C_1; E_1; c_1^?; (d_1^?)\rangle$, we construct κ_I' as $\langle C_1; E_1; c_2^?\rangle$.

b If G is of the form $B \vee C$, search proceeds either by solving $(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} B^\mu$, or by solving $(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} C^\mu$.

c If G is of the form $\exists x A$, search proceeds by solving $(\kappa, \langle C, \text{NLVF}(X, \mu); E; c^?\rangle, \kappa_O)\Gamma \xrightarrow{?} A[X/x]^\mu$. C, E and $c^?$ name the elements of κ_I; the additional constraint $\text{NLVF}(X, \mu)$ characterizes substitutions that send the *new logic variable* X to some *first-order* term t defined at world μ.

d If G is of the form $[\text{M}]\forall x A$, we revise the context to $(\kappa', \kappa_I', \kappa_O)$ to record a *new parameter* α representing a *transition* of $[\text{M}]$-accessibility and a *new parameter* c representing a *first-order* individual defined (only) at world $\mu\alpha$. We solve $(\kappa', \kappa_I', \kappa_O)\Gamma \xrightarrow{?} A[c/x]^{\mu\alpha}$. Explicitly, we require constraints $\text{NPT}(\alpha, [\text{M}], \mu)$ and

$\text{NPF}(c, \mu\alpha)$ describing the new terms. If κ is $\langle G;K;s^?;F\rangle$, κ' is $\langle G, A[c/x]^{\mu\alpha};K;s^?;F\rangle$. If κ_I is $\langle C;E;c^?\rangle$ then if $c^?$ is true, then κ_I' is $\langle C, \text{NPT}(\alpha, [\text{M}], \mu), \text{NPF}(c, \mu\alpha);E;c^?\rangle$ and if $c^?$ is false then κ_I' is $\langle C, \text{NPT}(\alpha, [\text{M}], \mu), \text{NPF}(c, \mu\alpha);E, A[c/x]^{\mu\alpha};c^?\rangle$.

e If G is of the form $[\text{M}](B \supset C)$, we revise the context to $(\kappa', \kappa_I', \kappa_O)$ to record a *new parameter* α representing a *transition* of $[\text{M}]$-accessibility; search proceeds by solving $(\kappa', \kappa_I', \kappa_O)\Gamma, B^{\mu\alpha} \xrightarrow{?} C^{\mu\alpha}$. If κ is $\langle G;K;s^?;F\rangle$, κ' is $\langle G, C^{\mu\alpha};K;s^?;F\rangle$. If κ_I is $\langle C;E;c^?\rangle$ then if $c^?$ is true, then κ_I' is $\langle C, \text{NPT}(\alpha, [\text{M}], \mu);E;c^?\rangle$ and if $c^?$ is false then κ_I' is $\langle C, \text{NPT}(\alpha, [\text{M}], \mu);E, C^{\mu\alpha};c^?\rangle$.

f In other cases where G is of the form $[\text{M}]C$, search proceeds by solving $(\kappa', \kappa_I', \kappa_O)\Gamma \xrightarrow{?} C^{\mu\alpha}$ where α is a new parameter representing a transition of $[\text{M}]$-accessibility and κ' and κ_I' are constructed according to a rule which is textually identical to that of (11e).

The search instructions in (11) describe the processing DIALUP will do in breaking down any complex goal into a combination of atomic goals. Once this process is completed, the program itself is consulted; the interpreter performs an appropriate version of backward chaining. The interpreter chooses a clause that might match the goal nondeterministically from the program and dissects it—by rules dual to the ones above that dissect goals—to obtain an atomic fact and a sequence of new subgoals. Backward chaining is described by the decision rule of (12):

(12) Suppose G^μ is an atomic formula and P^ν is a program clause in Γ that is not suppressed (by $s^?$ in $\kappa = \langle G;K;s^?;F\rangle$). Then search for $(\kappa, \kappa_I, \kappa_O)\Gamma \xrightarrow{?} G^\mu$ may proceed by solving $(\kappa, \kappa_I, \kappa_O')\Gamma;P^\nu \xrightarrow{?} G^\mu$. Unless P^ν is the key premise K, κ_O and κ_O' are identical. Otherwise, say κ_O' is $\langle C;E;c^?;d^?\rangle$. κ_O is set to $\langle C;E \cap G;$ *true* $;d^?\rangle$; this records the inference as a cancellation that contributes towards the currently active goals.

Obviously, for implementation, the choice of clause P^μ in (12) can be restricted by typical heuristics such as a match between the predicate of G and a head predicate in P.

Backward chaining introduces a new kind of interpreter state in which the program clause P that the interpreter must apply to the current goal is distinguished. In such a state, the structure of P clause gives rise to instructions for search according to the specifications of (13).

(13) a If P is of the form $B \wedge C$, search proceeds either by solving $(\kappa, \kappa_I, \kappa_O)\Gamma;B^\nu \xrightarrow{?} G^\mu$, or by solving $(\kappa, \kappa_I, \kappa_O)\Gamma;C^\nu \xrightarrow{?} G^\mu$.

b If P is of the form $\forall xA$, search proceeds by constructing κ_I' to introduce a fresh logic variable X to leave open some first-order term t defined at world ν. Then we proceed with $(\kappa, \kappa_I', \kappa_O)\Gamma;A[X/x]^\nu \xrightarrow{?} G^\mu$. Explicitly, if κ_I is $\langle C;E;c^?\rangle$, κ_I' is $\langle C, \text{NLVF}(X, \mu);E;c^?\rangle$.

c If P is of the form $B \supset C$, search proceeds by first solving $(\kappa, \kappa_I, \kappa_O')\Gamma;C^\nu \xrightarrow{?} G^\mu$; we then construct appropriate κ' and κ_I' and solve $(\kappa', \kappa_I', \kappa_O'')\Gamma \xrightarrow{?} B^\nu$. Explicitly, suppose κ_O' is $\langle C;E;c^?;d^?\rangle$. Then κ_I' is $\langle C;E;c^?\rangle$. Meanwhile, if $d^?$ is true, κ' is exactly like κ except that the key formula K is now suppressed in $s^?$ in κ; otherwise κ' and κ are identical. Finally if κ_O'' is $\langle C';E';c'^?;d'^?\rangle$, then κ_O is $\langle C';E';c'^?;d^?\rangle$ (propagating the values accumulated throughout, except for passing down information about delayed disjuncts only from the first subproof).

d If P is of the form $[\mathrm{M}]B$, search proceeds by constructing κ'_I to introduce a fresh logic variable x to leave open a *new logic variable* representing a *transition* of $[\mathrm{M}]$-accessibility from ν. Then we proceed with $(\kappa,\kappa'_I,\kappa_O)\Gamma;B^{\nu x} \xrightarrow{?} G^{\mu}$. Explicitly, if κ_I is $\langle C;E;c^?\rangle$, κ'_I is $\langle C,\text{NLVT}(x,[\mathrm{M}],\mu);E;c^?\rangle$; the additional constraint $\text{NLVT}(x,[\mathrm{M}],\nu)$ characterizes substitutions that send the new logic variable x to an appropriate path.

e If P is of the form $B \vee C$, search proceeds in either of two ways. The ordinary case is to solve $(\kappa,\kappa_I,\kappa'_O)\Gamma;B^{\nu} \xrightarrow{?} G^{\mu}$ and to solve a further search problem $(\kappa,\kappa'_I,\kappa''_O)\Gamma,C^{\nu} \xrightarrow{?}$ by restarting ordinarily with key premise C^{ν}. We construct κ'_I from $\kappa'_O = \langle C;E;c^?;d^?\rangle$ as $\langle C;G;\ \mathit{false}\rangle$ (using G from κ). Assuming κ''_O is $\langle C';E';c'^?;d'^?\rangle$, we construct κ_O as $\langle C';E;c^?;d^?\rangle$. The restart problem is delayed until the block in progress is completed and the value for F in κ is determined.
The *exceptional* case, when κ contains a key premise R, is to solve $(\kappa,\kappa_I,\kappa'_O)\Gamma;C^{\nu} \xrightarrow{?} G^{\mu}$ and to solve a further search problem $(\kappa,\kappa'_I,\kappa''_O)\Gamma,B^{\nu} \xrightarrow{?}$ by *delayed* restart with key premise B^{ν}; this search problem is again delayed until the block in progress is completed and the value for F in κ is determined; we construct κ'_I from κ'_O as above. Finally, κ_O is defined from κ'_O and κ''_O as above except that here we use *true* in place of $d^?$.

f If P is of the form $\exists xA$, we update κ_I to κ'_I to leave open a fresh parameter c as a witness for the existential quantifier by Skolemizing; search proceeds by solving $(\kappa,\kappa'_I,\kappa_O)\Gamma;A[c/x]^{\nu} \xrightarrow{?} G^{\mu}$. Explicitly, we assume that the existential quantifier in the program is associated with a function f, and the sequence of logic variables that have been introduced during matching is given by the list V. We introduce a constraint $\text{SFP}(c,\nu,f,V)$ indicating that c must correspond to a term in which f is applied to arguments V to name an individual existing at world ν; if κ_I is $\langle C;E;c^?\rangle$ then κ'_I is $\langle C,\text{SFP}(c,\nu,f,V);E;c^?\rangle$.

The definitions outlined so far leave only two gaps in the specification of the interpreter. First, to match an atomic clause against an atomic goal, we must unify:

(14) Solve $(\kappa,\kappa_I,\kappa_O)\Gamma;A^{\nu} \xrightarrow{?} G^{\mu}$ where both A and G are atoms by constructing appropriate κ_O. If κ_I is $\langle C;E;c^?\rangle$ then κ_O is $\langle C,A=G,\nu=\mu;E;c^?;\ \mathit{false}\ \rangle$; if the new constraints will not be satisfiable the proof in progress may be rejected.

Second, to handle a disjunctive case, we must do an appropriate restart with new key premise P^{ν}:

(15) Solve $(\kappa,\kappa_I,\kappa_O)\Gamma \xrightarrow{?}$ selecting a restart goal G^{μ} from κ, constructing appropriate κ' and κ'_I, and solving $(\kappa',\kappa'_I,\kappa_O)\Gamma \xrightarrow{?} G^{\mu}$. Say κ is $\langle G;K;s^?;F\rangle$ and κ' is $\langle C;E;c^?\rangle$. Then G^{μ} must have $G^{\mu}\in G$, and, if this is a delayed restart, $G^{\mu}\in F$. If this is an ordinary restart, we define $s'^?$ so that no premises are suppressed; otherwise, we define $s'^?$ so that K is suppressed. We enforce the constraint that ν is a prefix of μ by constructing $\kappa'_I = \langle C,\nu\le\mu;E;c^?\rangle$. Finally, we require that κ_O takes the form $\langle C;F;\mathit{true};d^?\rangle$.

4 Why DIALUP is correct

In this section, we describe the design of DIALUP from a logical point of view. We begin by presenting a cut-free path-based sequent calculus for multi-modal deduction which uses Herbrand terms to reason correctly about parameterized instances of formulas. Since this calculus represents our basic *lifted sequent calculus* for modal logic, we refer to it as SCL here. SCL is derived explicitly in [Stone, 1999a], where it is proved that SCL provides a sound and complete characterization of Kripke models for first-order multi-modal logic. But SCL should offer few surprises to those familiar with prefixed tableaux [Fitting, 1983], the logical foundation of Herbrand terms [Lincoln and Shankar, 1994], and the possibilities for enforcing a proof-theoretic separation in modal deduction between constraints on accessibility and general first-order reasoning [Frisch and Scherl, 1991, Basin et al., 1998].

SCL has the advantage that inferences can be freely interchanged, allowing arbitrary proofs to be transformed easily into goal-directed proofs—we show in Theorem 1, presented in Section 4.2, how to obtain goal-directed proofs in this calculus. The very same flexibility of inference, however, means that this calculus neither respects nor represents the potential of modal inference to give proofs an explicitly modular structure.

We therefore rely on further proof-theoretic analyses of path-based sequent calculi to refine the uniform proof system and guarantee modular behavior. These analyses establish that path representations enforce modularity and locality in the uses of formulas in proofs, even with otherwise classical reasoning. The operational rules of DIALUP are obtained by transforming the uniform proof system to take advantage of these results; as a consequence, the interpreter can dynamically exploit locality in the use of modular assumptions. The transformation starts in Section 4.3 by dividing uniform proofs into separate *segments* which apply one axiom from the program. The transformation continues in 4.4 by dividing uniform proofs into separate *blocks* to analyze separate cases. As a preliminary to modularity, we organize these blocks so that each one contains a *cancellation* whereby the most-recently introduced case contributes to the goal being proved [Loveland, 1991]. Finally, in Section 4.5, we combine the presence of cancellations and the inherent ability of the modal language to modularly restrict the contributions premises can make (together with the uniformity of proof search and the independence of cases) to derive a final sequent calculus (in Figures 23 and 24) which can be regarded as a formal specification for the interpreter of a logic programming language. We relate this specification explicitly to the operational rules of Section 3.4 in Section 5.

4.1 Modal sequent calculus

All the proof systems in this paper are parameterized by a *modal regime*, which describes the relationships among the modal operators of the language. This regime is derived from the specification input to DIALUP. DIALUP assumes increasing first-order domains across worlds, and offers four kinds of modal operators: T, subject just to (VER); K4, subject just to (PI); S4, subject to both; and K, subject to neither. Further, the relationships among operators are characterized by a relation $i \leq j$ that holds when we have a schema $\Box_i P \supset \Box_j P$. Thus, for DIALUP, we set up a regime as a tuple $\langle A, N, \mathit{increasing} \rangle$, where A is a function associating each modal operator with a type from K, K4, T and S4; N is a (strict) partial order on the modal operators obtained by taking the transitive closure of $\leq$; and *increasing* codes the relationships among first-order domains across worlds. The regime is a convenient structure for identifying classes of modal frames and classes of Kripke modal models where the accessibility relations and the domains of quantification respect intended constraints, in the usual way (see for example [Fitting, 1983, Auffray and Enjalbert, 1992, Debart et al., 1992, Stone, 1999a]).

The basic constituent in the proof system is a *tracked, prefixed formula*. The formulas extend the basic languages $D(C)$ and $G(C)$ of definitions and goals (parameterized by atomic constants C and collectively identified as $L(C)$) defined in (1) by allowing additional terms—representing arbitrary witnesses of first order quantifiers, and arbitrary transitions of modal accessibility among possible worlds—to be introduced into formulas for the purposes of proof. We begin by assuming two countable sets of symbols: a set H of *first-order Herbrand functions* and Υ of *modal Herbrand functions*. We can now define sets P_H of *first-order Herbrand terms*, κ_Υ of *modal Herbrand terms*, and $\Pi(\kappa_\Upsilon)$ of *Herbrand prefixes* by mutual recursion:

Definition 1 (Herbrand terms and prefixes) *Assume that t_0 is a Herbrand prefix and let $t_1,\dots,t_n$ be a sequence (possibly empty), where each t_i is either an element of C, a first-order Herbrand term, or a Herbrand prefix. Then if h is a first-order Herbrand function then $h(t_0,t_1,\dots,t_n)$ is a* first-order Herbrand term. *If η is a modal Herbrand function then $\eta(t_0,t_1,\dots,t_n)$ is a* modal Herbrand term. *A* Herbrand prefix *is any finite sequence of modal Herbrand terms.*

A *prefixed formula* is now an expression of the form A^μ with A a formula and μ a Herbrand prefix—we use $D(C\cup P_H)^{\Pi(\kappa_\Upsilon)}$ and $G(C\cup P_H)^{\Pi(\kappa_\Upsilon)}$ to refer to prefixed definitions and prefixed goals. For Herbrand calculi, formulas must also be *tracked* to indicate the sequence of instantiations that has taken place in the derivation of the formula.

Definition 2 (tracked expressions) *If E denotes the expressions of some class, then the* tracked expressions *of that class are expressions of the form e_I where e is an expression of E and I is a finite sequence (possibly empty) of elements of $C\cup P_H\cup\Pi(\kappa_\Upsilon)$.*

We say that a tracked expression e_I *tracks* a term t just in case t occurs as a subterm of some term in I.

In order to reason correctly about multiple modal operators, we need to keep track of the kinds of accessibility that any modal transition represents. To endow the system with correct first-order reasoning on increasing domains, we also need to keep track of the worlds where first-order terms are introduced. We use the following notation to record these judgments: $\mu/\nu : i$ indicates that world ν is accessible from world μ by the accessibility relation for modality i; and $t : \mu$ indicates that the entity associated with term t exists at world μ.

It is convenient to keep track of this information by anticipating the restricted reasoning required for the DIALUP fragment $L(C)$ and exploiting the structure of Herbrand terms, as follows. It is clear that there are countably many first-order Herbrand terms, Herbrand prefixes, and formulas in $L(C\cup P_H)$. We can therefore describe a correspondence as follows. If A is a formula of the form $\forall xB$ or $\exists xB$, we define a corresponding first-order Herbrand function h_A so that each first-order Herbrand function is h_A for some A and no first-order Herbrand function is h_A and h_B for distinct A and B. Likewise, if A is a formula of the form $\Box_i B$ and u is a natural number, we define a corresponding modal Herbrand function η_A^u so that each modal Herbrand function is η_A^u for some A and no modal Herbrand function is η_A^u and η_B^v for distinct A and B or distinct u and v. Now we have:

Definition 3 (Herbrand Typings) *A* Herbrand typing for the language $L(C\cup P_H)$ *(under a correspondence as just described) is a set Ξ of statements, each of which takes one of two forms:*

1. $\mu/\mu\eta : i$ *where:* μ *is a Herbrand prefix and* η *is a modal Herbrand term of the form* $\eta^u_A(\mu, I)$ *and* A *is* $\Box_i B$[3]

2. $t : \mu$ *where* t *is a first-order Herbrand term of the form* $h(\mu, I)$.

A sequence of modal and first-order Herbrand terms X *determines a Herbrand typing* Ξ_X*, consisting of the appropriate* $\mu/\mu\eta : i$ *for each modal Herbrand term* η *that occurs in* X *(possibly as a subterm) and the appropriate* $h : \mu$ *for each first-order Herbrand term* h *that occurs in* X *(possibly as a subterm).*

This definition of Herbrand terms specializes the definition of [Stone, 1999a] to the DIALUP language by eliminating cases for inference rules that are not required in DIALUP; it also anticipates our arguments about transformations between Herbrand proofs by indexing Herbrand terms by natural numbers; in [Stone, 1999a] Herbrand terms are indexed only by formulas. This move is correct and complete since we can translate back and forth by erasing the numerical index (in one direction) and decorating with the index 0 (in the other).

Definition 4 (Typings) *Suppose that* Ξ *is a Herbrand typing over a language* $L(C \cup P)^{\Pi(\kappa)}$*, and that* $S = \langle A, N, increasing \rangle$ *is a modal regime. We define the relation that* E *is a* derived typing *from* Ξ *with respect to* S*, written* $S, \Xi \triangleright E$*, as the smallest relation satisfying the following conditions:*

- (K). $S, \Xi \triangleright \mu/\nu : i$ *if* $\mu/\nu : i \in \Xi$.
- (T). $S, \Xi \triangleright \mu/\mu : i$ *if* $A(i)$ *is T or S4, and* μ *occurs in* Ξ.
- (4). $S, \Xi \triangleright \mu/\nu : i$ *if* $\mu/\mu' : i \in \Xi$, $S, \Xi \triangleright \mu'/\nu : i$*, and* $A(i)$ *is K4 or S4.*
- (Inc). $S, \Xi \triangleright \mu/\nu : j$ *if* $S, \Xi \triangleright \mu/\nu : i$ *and* $i \leq j$ *according to* N.
- (V). $S, \Xi \triangleright t : \mu$ *if* $t : \mu \in \Xi$.
- (I). $S, \Xi \triangleright t : \nu$ *if* $S, \Xi \triangleright \mu/\nu : i$ *for some* i *and* $S, \Xi \triangleright t : \mu$.

Inspection of these rules shows that $S, \Xi \triangleright \mu/\nu : i$ only if ν and μ occur in Ξ. Moreover, given these rules, an easy induction on the length of typing derivations gives that $S, \Xi \triangleright \mu/\nu : i$ only if $\nu = \mu\nu'$ for some prefix ν'. Thus, suppose that $S, \Xi \triangleright \mu/\nu : i$ for some Herbrand typing Ξ: each step in the derivation must concern some prefix of ν and thus $S, \Xi_\nu \triangleright \mu/\nu : i$. These invariants permit some simplifications in reasoning in the fragment $L(C \cup P)$ over more expressive modal regimes containing other modal operators and other uses of connectives. (In particular, we can streamline the formulation of axiom inferences and the tracking of terms at modal and quantifier rules over the general case; see [Stone, 1999a, Section 5].)

These definitions allow us to describe the modal Herbrand sequent calculus precisely. For the DIALUP fragment of modal logic, it suffices to consider sequents of the form $\Delta \longrightarrow \Gamma$, where Δ is a multiset of prefixed definitions (from $D(C \cup P_H)^{\Pi(\kappa_\Upsilon)}$), and Γ is a multiset of prefixed goals (from $G(C \cup P_H)^{\Pi(\kappa_\Upsilon)}$). We can then specialize a cut-free modal sequent calculus—for example the one presented in [Stone, 1999a]—to the DIALUP fragment by omitting unneeded inference figures, and by exploiting the DIALUP invariant that $S, \Xi \triangleright \mu/\nu : i$ only if ν is of the form $\mu\nu'$. Such a calculus, SCL, is given in Definition 5.

[3] *including as a special case* $\Box_i(B \supset C)$*, which we will abbreviate to* $B >_i C$.

Definition 5 (Herbrand sequent calculus) *For basic first-order multi-modal Herbrand deductions in the* DIALUP *fragment over a regime* S*, we will use the sequent rules defined here, which comprise the system SCL. The rules consist of an axiom rule and recursive rules—each recursive rule relates a* base *sequent below to one or more* spur *sequents above; it applies to the base in virtue of an occurrence of a distinguished tracked, prefixed formula in the sequent; we refer to this as the* principal expression *or simply the* principal *of the inference. Similarly, each of the sequent rules introduces new expressions onto each spur, which we refer to as the* side expressions *of the rule. We will also refer to the two named expression occurrences at axioms as the* principal expressions *or* principals *of the axiom. Now we have:*

1. *axiom—A atomic:*

$$\Delta, A^{\mu}_{X} \longrightarrow \Gamma, A^{\mu}_{Y}$$

2. *conjunctive:*

$$\frac{\Delta, A \wedge B^{\mu}_{X}, A^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Gamma}{\Delta, A \wedge B^{\mu}_{X} \longrightarrow \Gamma}$$

$$\frac{\Delta \longrightarrow \Gamma, A \vee B^{\mu}_{X}, A^{\mu}_{X}, B^{\mu}_{X}}{\Delta \longrightarrow \Gamma, A \vee B^{\mu}_{X}}$$

$$\frac{\Delta, A^{\mu}_{X} \longrightarrow \Gamma, A \supset B^{\mu}_{X}, B^{\mu}_{X}}{\Delta \longrightarrow \Gamma, A \supset B^{\mu}_{X}}$$

3. *disjunctive:*

$$\frac{\Delta \longrightarrow \Gamma, A \wedge B^{\mu}_{X}, A^{\mu}_{X} \qquad \Delta \longrightarrow \Gamma, A \wedge B^{\mu}_{X}, B^{\mu}_{X}}{\Delta \longrightarrow \Gamma, A \wedge B^{\mu}_{X}}$$

$$\frac{\Delta, A \vee B^{\mu}_{X}, A^{\mu}_{X} \longrightarrow \Gamma \qquad \Delta, A \vee B^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Gamma}{\Delta, A \vee B^{\mu}_{X} \longrightarrow \Gamma}$$

$$\frac{\Delta, A \supset B^{\mu}_{X} \longrightarrow A^{\mu}_{X}, \Gamma \qquad \Delta, A \supset B^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Gamma}{\Delta, A \supset B^{\mu}_{X} \longrightarrow \Gamma}$$

4. *possibility—where* η *is* $\eta^{u}_{\Box_i A}(\mu, X)$ *for some u:*

$$\frac{\Delta \longrightarrow \Gamma, \Box_i A^{\mu}_{X}, A^{\mu\eta}_{X,\mu\eta}}{\Delta \longrightarrow \Gamma, \Box_i A^{\mu}_{X}}$$

5. *necessity—subject to the side condition* $S, \Xi_{\nu} \triangleright \mu/\mu\nu : i$*:*

$$\frac{\Delta, \Box_i A^{\mu}_{X}, A^{\mu\nu}_{X,\mu\nu} \longrightarrow \Gamma}{\Delta, \Box_i A^{\mu}_{X} \longrightarrow \Gamma}$$

6. *existential—subject to the side condition that h is* $h_B(\mu, X)$ *for* B^{μ}_{X} *the principal of the rule*

(either $\exists xA$ or $\forall xA$):

$$\frac{\Delta, \exists xA^{\mu}, A[h/x]^{\mu}_{X,h} \longrightarrow \Gamma}{\Delta, \exists xA^{\mu}_{X} \longrightarrow \Gamma} \quad \frac{\Delta \longrightarrow \Gamma, \forall xA^{\mu}_{X}, A[h/x]^{\mu}_{X,h}}{\Delta \longrightarrow \Gamma, \forall xA^{\mu}_{X}}$$

7. *universal—subject to the side condition $S, \Xi_{t,\mu} \triangleright t : \mu$:*

$$\frac{\Delta, \forall xA^{\mu}_{X}, A[t/x]^{\mu}_{X,t} \longrightarrow \Gamma}{\Delta, \forall xA^{\mu}_{X} \longrightarrow \Gamma} \quad \frac{\Delta \longrightarrow \Gamma, \exists xA^{\mu}_{X}, A[t/x]^{\mu}_{X,t}}{\Delta \longrightarrow \Gamma, \exists xA^{\mu}_{X}}$$

A S-proof or S-derivation for a sequent $\Delta \longrightarrow \Gamma$ is a tree built by application of these inference figures (in such a way that any side conditions are met for regime S), with instances of the axiom as leaves and with the sequent $\Delta \longrightarrow \Gamma$ at the root. A tree similarly constructed except for containing some arbitrary sequent S as a leaf is a *derivation from S*. In [Stone, 1999a] it is shown that there is an S-proof for a sequent using these rules just in case that sequent is valid in all Kripke models that respect the regime S.

Our syntactic methods for reasoning about derivations exploit *permutability of inference*—the general ability to transform derivations so that inferences are interchanged [Kleene, 1951]. To develop the notion of permutability of inference, we need to make some observations about the SCL sequent rules. First, the reasoning that is performed in subderivations is reasoning about subformulas (and vice versa). That is, in any spur sequent, the occurrence of the principal expression and the side expression all correspond to—or as we shall say, *are based in*—the occurrence of the principal in the base sequent. Likewise, each of the remaining expressions in the spur *are based in* an occurrence of an identical expression in the base. Here, as in [Kleene, 1951], we are assuming an *analysis* of each inference to specify this correspondence in the case where the same expression has multiple occurrences in the base or in a spur. Thus, our proof techniques, where they involve copying derivations, sometimes involve (implicit) reanalyses of inferences.

Now, in any derivation, the spur of one inference serves as the base for an *adjacent* inference or an axiom. We can therefore associate any tracked prefixed formula occurrence E in any sequent in the derivation with the occurrence in the root (or *end-sequent*) which E is based in. A similar notion can relate inferences, as follows. Suppose O is the inference at the root of a (sub)derivation, and L is another inference in the (sub)derivation. Then *L is based in* an expression E in the spur of O if the principal expression of L is based in E; *L is based in O* itself if E is a side expression of O. An important special case is that of an axiom based in an inference O. In effect, such an axiom marks a contribution that inference O contributes to completing the deduction.

To define interchanges of inference, we appeal to the two basic operations of *contraction* and *weakening*, which we cast as transformations on proofs. (In other proof systems, contraction and weakening may be introduced as explicit *structural rules*.)

Lemma 1 (Weakening) *Let D be an SCL proof, let Δ_0 be a finite multiset of tracked prefixed definitions and let Γ_0 be a finite multiset of tracked prefixed goals (in the same language as D). Denote by $\Delta_0 + D + \Gamma_0$ a derivation exactly like D, except that where any node in D carries $\Delta \longrightarrow \Gamma$, the corresponding node in $\Delta_0 + D + \Gamma_0$ carries $\Delta, \Delta_0 \longrightarrow \Gamma, \Gamma_0$. (When Δ_0 or Γ_0 is empty, we drop the corresponding $+$ from the notation.) Then $\Delta_0 + D + \Gamma_0$ is also an SCL proof.*

Lemma 2 (Contraction) *Let $\mathcal{D}$ be an SCL proof whose root carries $\Delta \longrightarrow \Gamma, E, E$. Then we can construct an SCL proof $\mathcal{D}'$ whose root carries $\Delta \longrightarrow \Gamma, E$, whose height is at most the height of $\mathcal{D}$ and where there is a one-to-one correspondence (also preserving order of inferences) that takes any inference of $\mathcal{D}'$ to an inference with the same principal and side expressions in $\mathcal{D}$. We can likewise transform an SCL proof $\mathcal{D}$ whose root carries $\Delta, E, E \longrightarrow \Gamma$ into an SCL proof $\mathcal{D}'$ whose root carries $\Delta, E \longrightarrow \Gamma$.*

These lemmas follow from straightforward induction on the structure of derivations. These consequences continue to hold, suitably adapted, for the intermediate proof systems that we will construct from SCL in later sections (including SCLU and SCLV).

Now consider two adjacent inferences in a derivation, a base inference R and an inference S (whose base is a spur of R). If S is not based in R, we may replace the derivation rooted at the base of R by a new derivation of the same end-sequent in which S applies at the root, R applies adjacent, and the remaining subderivations are copied from the original derivation (but possibly weakened to reflect the availability of additional logical premises). Performing such a replacement constitutes an interchange of rules R and S and demonstrates the permutability of R over S; see [Kleene, 1951]. SCL is formulated so that any such pair of inferences may be exchanged in this way.

We also observe that we can correctly introduce an abbreviation for goal occurrences of $\Box_i(A \supset B)$ by a single formula $(A >_i B)$ and the consolidation of corresponding inferences $(\rightarrow \Box_i)$ and $(\rightarrow\supset)$ into a single figure $(\rightarrow >_i)$. Again when the inference applies to principal A^{μ}_{X}, the figure is formulated using η for $\eta^{u}_{A}(\mu, X)$ as:

$$\frac{\Gamma, A^{\mu\eta}_{X,\mu\eta} \longrightarrow B^{\mu\eta}_{X,\mu\eta}, A >_i B^{\mu}_{X}, \Delta}{\Gamma \longrightarrow A >_i B^{\mu}_{X}, \Delta} \rightarrow >_i$$

We will refer to the calculus using $(\rightarrow >_i)$ in place of $(\rightarrow \Box_i)$ and $(\rightarrow\supset)$ as SCLI.

To transform an SCLI deduction into an SCL deduction, we can eliminate $(\rightarrow >_i)$ figures inductively as follows. Take a derivation of this form:

$$\frac{\begin{array}{c}\mathcal{D}\\ \Gamma, A^{\mu\eta}_{X} \longrightarrow B^{\mu\eta}_{X}, A >_i B^{\mu}_{X}, \Delta\end{array}}{\Gamma \longrightarrow A >_i B^{\mu}_{X}, \Delta} \rightarrow >_i$$

Transform $\mathcal{D}$ inductively into an SCL derivation $\mathcal{D}'$; then construct:

$$\frac{\dfrac{\begin{array}{c}\mathcal{D}' + A \supset B^{\mu\eta}_{X,\mu\eta}\\ \Gamma, A^{\mu\eta}_{X,\mu\eta} \longrightarrow B^{\mu\eta}_{X,\mu\eta}, A \supset B^{\mu\eta}_{X,\mu\eta}, \Box_i(A \supset B)^{\mu}_{X}, \Delta\end{array}}{\Gamma \longrightarrow A \supset B^{\mu\eta}_{X,\mu\eta}, \Box_i(A \supset B)^{\mu}_{X}, \Delta} \rightarrow\supset}{\Gamma \longrightarrow \Box_i(A \supset B)^{\mu}_{X}, \Delta} \rightarrow \Box_i$$

For the converse construction, we assume an SCL derivation whose end-sequent—$\Gamma \longrightarrow \Delta$—takes a special form. Δ is the multiset union of Δ^{+} and Δ^{*} where Δ^{*} consists of all and only the occurrences of expressions of the form $A \supset B^{\mu\eta}_{X,\mu\eta}$ in Δ; moreover, for each $A \supset B^{\mu\eta}_{X,\mu\eta} \in \Delta^{*}$ there is a $A >_i B^{\mu}_{X} \in \Delta^{+}$ with η as determined for the $(\rightarrow \Box_i)$ or $(\rightarrow >_i)$ figure. (This assumption is met by DIALUP search problems, as they are in fact specified without any bare $\supset$-formulas in goals.) Under this assumption, we construct an SCLI proof of $\Gamma \longrightarrow \Delta^+$ inductively. The problematic case is an SCL derivation that ends:

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma, A^{\mu\eta}_{X,\mu\eta} \rightarrow B^{\mu\eta}_{X,\mu\eta}, A \supset B^{\mu\eta}_{X,\mu\eta}, \Delta\end{array}}{\Gamma \rightarrow A \supset B^{\mu\eta}_{X,\mu\eta}, \Delta} \rightarrow\supset$$

Since by assumption we have $A >_i B^{\mu}_{X} \in \Delta^+$, we derive an SCLI proof $\mathcal{D}'$ for $\mathcal{D}$ inductively then construct:

$$\dfrac{\begin{array}{c}\mathcal{D}'\\ \Gamma, A^{\mu\eta}_{X,\mu\eta} \rightarrow B^{\mu\eta}_{X,\mu\eta}, A >_i B^{\mu}_{X}, \Delta^+\end{array}}{\Gamma \rightarrow A >_i B^{\mu}_{X}, \Delta^+} \rightarrow\supset$$

Here (as always) the status of $B^{\mu\eta}_{X,\mu\eta}$ as a Δ^+ expression is guaranteed by the DIALUP fragment and the fact that Δ^+ is a multiset of tracked elements of $G(C \cup A_H)^{\Pi(\kappa_\Upsilon)}$. ■

4.2 *Uniform proofs and eager proofs*

[Miller, 1994] uses Definition 6 to characterize *abstract logic programming languages.*

Definition 6 *A cut-free sequent proof $\mathcal{D}$ is* uniform *if for for every subproof $\mathcal{D}'$ of $\mathcal{D}$ and for every non-atomic formula occurrence B in the right-hand side of the end-sequent of $\mathcal{D}'$ there is a proof $\mathcal{D}''$ that is equal to $\mathcal{D}'$ up to a permutation of inferences and is such that the base inference in $\mathcal{D}''$ introduces the top-level logical connective of B.*

Definition 7 *A logic with a sequent calculus proof system is an* abstract logic programming language *if restricting to uniform proofs does not lose completeness.*

It is easy to show that the sequent calculi SCL and SCLI are abstract logic programming languages in this sense. In fact, by this definition *every* SCL or SCLI derivation is uniform.

To anticipate our analysis of permutability in later sections, let us introduce the notion of an *eager* derivation in SCL or SCLI.

Definition 8 *Consider a derivation $\mathcal{D}$ containing a right inference R that applies to principal E. R is* delayed *exactly when there is a subderivation $\mathcal{D}'$ of $\mathcal{D}$ where: $\mathcal{D}'$ contains R; $\mathcal{D}'$ has a left inference L at the root; and the principal E of R is based in an occurrence of E in the end-sequent of $\mathcal{D}'$.*

Consider this schematic diagram of such a subderivation $\mathcal{D}'$:

$$\dfrac{\dfrac{\vdots}{\ldots E \ldots}\, R \atop \downarrow}{\ldots E \ldots}\, L$$

On an intuitive conception of a sequent proof as a record of proof search constructed from root upwards, R is delayed in that we have waited in $\mathcal{D}$ to apply R until after consulting the program by applying L, when we might have applied R earlier. Thus, we will also say in the circumstances of Definition 8 that R is delayed *with respect to L.*

Definition 9 *$\mathcal{D}$ is* eager *exactly when it contains no delayed applications of right rules.*

By transforming any derivation $\mathcal{D}$ into an eager derivation $\mathcal{D}'$ by permutations of inferences, we witness that $\mathcal{D}$ is uniform and provide a starting point for further analysis.

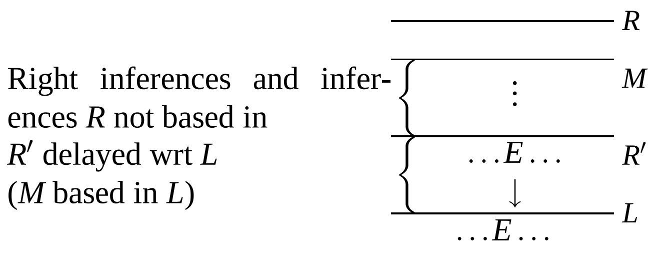
Theorem 1 *Any SCL(I) derivation $\mathcal{D}$ is equal to an eager derivation $\mathcal{D}'$ up to permutations of inferences.*

The **proof** follows [Kleene, 1951, Theorem 2]. A double induction transforms each derivation into an eager one; the inner induction rectifies the final rule of a derivation whose subderivations are eager by an interchange of inferences (and induction) [Kleene, 1951, Lemma 10]; the outer one rectifies a derivation by rectifying the furthest violation from the root (and induction).

The proof depends on a generalization of delayed inferences, which we can term *misplaced* inferences since we intend to eliminate them. We assume an overall derivation $\mathcal{D}$, and consider a right inference R that applies to principal E within some subderivation $\mathcal{D}'$ of $\mathcal{D}$.

Definition 10 *We say a right inference R is* right-based *on an inference R' in $\mathcal{D}$ if $R = R'$ or R is based on R' and every inference on which R is based above and including R' is a right inference. Then R is* misplaced *in $\mathcal{D}'$ exactly when there are inferences M and R' in $\mathcal{D}'$ such that, in $\mathcal{D}$, M is based on an inference L, R is right-based on R', and R' is delayed with respect to L.*

In this case we will also say R is misplaced *with respect to M*. We can abstract a key case of misplaced inferences by the following schematic derivation:



This schematic derivation shows informally how *misplaced inferences* help provide an inductive characterization of the inferences that stand in the way of obtaining an eager derivation. In an eager derivation, it will be impossible for R to appear above L. For R' cannot be delayed with respect to L, but once R' and L are interchanged, we will obtain a new delayed inference that R is based in, until finally we must interchange L and R. Of course, to do this, we must first interchange R with the *misplaced* inferences, such as M, which stand between R and L and cannot themselves be interchanged with L because they are based in L.

Observe that the relation R is misplaced with respect to M is asymmetrical. To see this, suppose R is misplaced with respect to M. By definition, R is right-based on R' which is delayed with respect to a left inference L on which M is based. Meanwhile, for M to be misplaced with respect to R, by definition, we must have M right-based on M' and R based in some left rule L_R. Any such M' would have to be based in L since no left inferences intervene between M and M'; M' must thus appear *inside* a schematic like that above. At the same time, since no left inferences intervene between R and R', R' would have to be based in any such L_R, which must thus appear *outside* such a schematic, closer to the root of the overall derivation. Accordingly, any such L_R must occur closer to the root of D than L; meanwhile the principal of M' is introduced further from the root than L. So we will not have M' delayed with respect to L_R.

Call R *badly misplaced* in D' if R is misplaced with respect to M and M occurs closer to the root than R. A subderivation D' with no badly misplaced inferences will be called *good*. An overall good derivation is also eager, since any delayed inference is badly misplaced.

We can now present the proof in full using a lemma.

Lemma 3 *Consider a subderivation D' of an overall derivation D, with the property that D' has good immediate subderivations and that D' ends in inference M. From D' we can construct a derivation with the same end-sequent that is good.*

Proof. The assumption that the immediate subderivations of D' are good is a very powerful one. For suppose that some inference is badly misplaced with respect to some other in D'. Then we can only have some rule R badly misplaced with respect to M—anything else would contradict that assumption.

In fact, we can show that some such R must be adjacent to M. Consider an inference S that intervenes between R and M: we will show that S must be badly misplaced with respect to M too. By the definition of misplaced, M is based on some left rule L in D, R is right-based on R', and R' is delayed with respect to L. Now consider the inferences that S is based on above L. If any of these is a left inference L', or S is itself a left inference, then R is also misplaced with respect to S—indeed, badly misplaced. This contradicts the assumption that the subderivations of D' are good. So none of these inferences can be a left inference, which means S is a right inference that is right-based on some inference S' above L. S' must be delayed with respect to L. Hence S is badly misplaced with respect to M.

Now we can proceed after [Kleene, 1951, Lemma 10]. Define the *grade* of D' as the number of badly misplaced inferences in D'. We show by induction on the grade that D' can be transformed to a good one.

The base case is a derivation of grade 0. This case has D' itself good. Thus, suppose the lemma holds for derivations of grade g, and consider D' of grade $g+1$. By the argument just given, one immediate subderivation—call it D''—must end with an inference R which is badly misplaced with respect to M. Such an R of course cannot be based in M, so we interchange inferences R and M. In the result, the subderivation(s) ending in M satisfy the condition of the lemma with grade g or less. By applying the induction hypothesis, we can replace these subderivations with good ones. By asymmetry, M is not now badly misplaced with respect to R, nor can any of the other inferences be badly misplaced with respect to R, since they were not so in the original derivation. It follows that the result is a good derivation. ■

Now, continuing the proof of Theorem 1, define the *reluctance* of D to be the number of rule applications R such that the subderivation D_R of D rooted in R is not good. We proceed by induction on reluctance. If reluctance is zero, D is itself good.

Now suppose the theorem holds for derivations of reluctance d, and consider D of reluctance $d+1$. Since D is finite, there must be a highest inference R such that some inference is badly misplaced with respect to R in the subderivation D_R rooted at R. This D_R satisfies the condition of Lemma 3. Therefore this D_R can be replaced with a corresponding eager derivation, giving a new derivation of smaller reluctance. The induction hypothesis then shows that the resulting derivation can be made eager. ■

4.3 Segment structure

Eager derivations do not make a satisfactory specification for a logic programming interpreter because they do not embody a particularly directed search strategy, in a number of respects. For one thing, eager derivations are free to work in parallel on different disjuncts of a goal using different program clauses; in logic programming we want *segments* in which a single program clause and a single goal is in force. Moreover, eager derivations can reuse work across separate case analyses; in logic programming we want *blocks* where particular cases are investigated separately. Finally, because of their classical formulation, eager derivations do not enforce or exploit any modularity in their underlying logic.

We will now remedy these faults of eager derivations. We begin with a trick that for now is purely formal—introducing an *articulated* SCLI. We represent assumptions as a pair $\Pi;\Gamma$ with Π encoding the global program and Γ encoding local clauses; eventually local clauses will be processed only in the current segment and then discarded. (Compare the similar notation and treatment from [Girard, 1993].) Similarly, we represent goals as a pair $\Delta;\Theta$, with Θ encoding the restart goals and Δ encoding the local goals; ultimately, we will also describe inference rules which will discard Δ between segments. With this representation, principal formulas of logical rules are local formulas, in Γ or Δ; so are the side formulas—with these exceptions: the $(\rightarrow \Box)$ and $(\rightarrow >)$ rules augment Π instead of Γ (when they add a new program clause) and Θ instead of Δ (when they add new restart goals).

New (decide) and (restart) rules keep this change general; they allow a global formula—a program clause or restart goal—to be selected and added to the local state.

$$\frac{\Pi, A^{\mu}_{X}; \Gamma, A^{\mu}_{X} \longrightarrow \Delta; \Theta}{\Pi, A^{\mu}_{X}; \Gamma \longrightarrow \Delta; \Theta}\text{(decide)} \qquad \frac{\Pi; \Gamma \longrightarrow \Delta, G^{\mu}_{X}; \Theta, G^{\mu}_{X}}{\Pi; \Gamma \longrightarrow \Delta; \Theta, G^{\mu}_{X}}\text{(restart)}$$

Lemma 4 (articulation) *Every SCLI deduction can be converted into an articulated SCLI deduction with an end-sequent of the form* $\Pi; \longrightarrow ; \Theta$ *in such a way that if the initial derivation is eager then so is the resulting derivation (and vice versa).*

The **proof** forward argues by straightforward structural induction that the derivation can be transformed assuming each formula in the end-sequent is allocated somehow either to Π or Γ. We preserve and extend this allocation in immediate subderivations, introducing instances of (decide) and (restart) as necessary when the principal expression is assigned occurrences in Π only; then argue by induction. Backward, another straightforward structural induction shows we return to SCLI by forgetting the distinction between Π and Γ, forgetting the (decide) and (restart) rules, and contracting copied formulas. ■

The next step is to introduce an inference figure $(\supset\rightarrow^{S})$ that imposes a *segment* structure on derivations, thus:

$$\frac{\Pi; \longrightarrow A^{\mu}_{X}, \Delta; \Theta \qquad \Pi; \Gamma, A \supset B^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Delta; \Theta}{\Pi; \Gamma, A \supset B^{\mu}_{X} \longrightarrow \Delta; \Theta}(\supset\rightarrow^{S})$$

The distinctive feature of the $(\supset\rightarrow^{S})$ figure is that the local results inferred from the program are discarded in the subderivation where the new goal is introduced. In an eager derivation, this will begin a new segment where first the new goal will be considered and then a new program clause will be selected to establish that goal.

We will define two calculi using $(\supset\rightarrow^S)$. The first, SCLS, eliminates the $(\supset\rightarrow)$ inference of the articulated SCLI and instead has $(\supset\rightarrow^S)$. The second, SCLV, is a calculus like the articulated SCLI but also allows $(\supset\rightarrow^S)$; $(\supset\rightarrow)$ and $(\supset\rightarrow^S)$ can appear anywhere in an SCLV derivation. We introduce SCLV to facilitate the incremental transformation of articulated SCLI proofs into SCLS proofs. We show in this section that an SCLI proof with end-sequent $\Pi \longrightarrow \Theta$ corresponds to an SCLS proof with end-sequent $\Pi; \longrightarrow; \Theta$, and vice versa. In fact, to transform SCLS to articulated SCLI we have a simple structural induction which replaces $(\supset\rightarrow^S)$ with $(\supset\rightarrow)$ using the weakening lemma; the soundness of SCLS over SCLI then follows by Lemma 4. Thus, here we are primarily concerned with completeness of a new sequent inference figure.

Definition 11 (segment) *A* segment *of an SCLV derivation D is a maximal tree of contiguous inferences in which the left subtree of any $(\supset\rightarrow^S)$ inference is omitted.*

The use of $(\supset\rightarrow^S)$ in eager derivations ensures that the processing of each new goal refers directly to global program clauses. To formalize this idea, we introduce the notion of a *fresh* inference.

Definition 12 (fresh) *Let D be an SCLV derivation. An inference R in D is* fresh *exactly when R is a right inference and the path from R to the root never follows the left spur of any $(\supset\rightarrow)$ inference.*

Lemma 5 *Let D be an eager SCLV derivation with an end-sequent of the form*

$$\Pi; \rightarrow \Delta; \Theta$$

and consider a subderivation D' of D rooted in a fresh inference R. Then the end-sequent of D' also has the form

$$\Pi'; \rightarrow \Delta'; \Theta'$$

Proof. Suppose otherwise, and consider a maximal D' whose end-sequent contains a non-empty multiset of local clauses Γ. We can describe D' equivalently as the subderivation of D that is rooted in a lowest fresh inference R when the end-sequent of D contains some local clauses. R cannot be the first inference of D, so there must be an inference S in D immediately below R. If S is a left rule, then the fact that D is eager leads to a contradiction. R must be based in S, or else R will be delayed. This means S is an implication inference; but given that R is fresh, R must appear along the branch of $(\supset\rightarrow^S)$ without local clauses. Meanwhile, if S is a right rule, it follows from the formulation of the rules that if the end-sequent of D_R has nonempty local clauses then the end-sequent of D_L must also. This contradicts the assumption that R is first. ∎

Lemma 6 *An eager articulated SCLI derivation whose end-sequent is of the form*

$$\Pi; \rightarrow \Delta; \Theta$$

can be transformed to an eager SCLS derivation of the same end-sequent.

Proof. We assume an eager SCLV derivation D with such an end-sequent; we show that we can transform it into an eager SCLS derivation D' with the same end-sequent. The proof is by induction on the number of occurrences of $(\supset\rightarrow)$ inferences in D.

In the base case, there are no $(\supset\rightarrow)$ inferences and $\mathcal{D}'$ is just $\mathcal{D}$.

Suppose the claim holds for derivations where $(\supset\rightarrow)$ is used fewer than n times, and suppose $\mathcal{D}$ is a derivation in which $(\supset\rightarrow)$ is used n times. Choose an inference L of $(\supset\rightarrow)$ with no other $(\supset\rightarrow)$ inference closer to the root of $\mathcal{D}$; we must rewrite the left subderivation at L to match the $(\supset\rightarrow^S)$ inference figure. We distinguish a subderivation $\mathcal{D}'$ of $\mathcal{D}$ as a function of L and draw on the inferences in $\mathcal{D}'$ to replace this subderivation—in particular, we identify $\mathcal{D}'$ as the largest subderivation of $\mathcal{D}$ containing L but no right inferences or segment boundaries below L.

Using Lemma 5, we develop a schema of $\mathcal{D}'$ thus:

$$\cfrac{\begin{array}{cc} \mathcal{D}^A & \mathcal{D}^B \\ \Pi;\Gamma,A\supset B^{\mu}_{X} \rightarrow A^{\mu}_{X},\Delta;\Theta & \Pi;\Gamma,A\supset B^{\mu}_{X},B^{\mu}_{X} \rightarrow \Delta;\Theta \end{array}}{\cfrac{\mathcal{D}^L\left\{\begin{array}{c}\Pi;\Gamma,A\supset B^{\mu}_{X} \longrightarrow \Delta;\Theta\\ \vdots \\ \Pi;\longrightarrow \Delta;\Theta\end{array}\right.}{\text{(Segment boundary or right rule)}}}\;L$$

We suppose L applies to an expression $A\supset B^{\mu}_{X}$; the left subderivation of L, $\mathcal{D}^A$ adds the goal A; the right, $\mathcal{D}^B$, uses the assumption B. The subderivation of $\mathcal{D}'$ from the end-sequent of L abstracts the left inferences performed elsewhere in this segment (and any subgoals that these inferences trigger). We notate this tree of inferences $\mathcal{D}^L$. By Lemma 5, $\mathcal{D}'$ ends with a sequent of the form $\Pi;\longrightarrow\Delta;\Theta$. Because of the form of the intervening rules, we have the same succedent $\Delta;\Theta$ at L, as well as the same global clauses Π.

We use $\mathcal{D}^L$ to construct an eager SCLS derivation $\mathcal{A}$ corresponding to $\mathcal{D}^A$; we will substitute the result for the left subtree at L to revise L to fit the $(\supset\rightarrow^S)$ figure. In outline, the derivation we aim for is an eager SCLS version of:

$$\frac{\mathcal{D}^A}{\mathcal{D}^L + A^{\mu}_{X}}$$

The problem is that if $\mathcal{D}^A$ is rooted in a right inference to A, we will not obtain an eager derivation when we reassemble L. The SCLS derivation $\mathcal{A}$ we use is actually constructed by recursion on the structure of $\mathcal{D}^A$, applying this kind of transformation at appropriate junctures. At each stage, we call the subderivation of $\mathcal{D}^A$ we are considering $\mathcal{D}'^A$.

For the base case, this subderivation is an axiom, and we construct this subderivation as a result. If $\mathcal{D}'^A$ ends in a right rule, the construction proceeds inductively by constructing corresponding subderivations and recombining them by the same right rule. With a right inference here, the resulting derivation must be eager since the subderivations are eager.

If $\mathcal{D}'^A$ ends in a left inference, the construction is not inductive. We observe that $\mathcal{D}'^A$ has an end-sequent of the form

$$\Pi,\Pi';\longrightarrow\Delta,\Delta';\Theta,\Theta'$$

(The inventory of expressions can only be expanded, and that only in certain places, as we follow right inferences to reach $\mathcal{D}'^A$.) So we first weaken $\mathcal{D}^L$ by the needed additional expressions—Π' on the left and Δ' (locally) and Θ' (globally) on the right; then we identify the open leaf in $\mathcal{D}^L$ with

D'^A, obtaining a larger derivation D_I defined as:

$$\frac{D'^A}{\Pi' + D'^L + A^\mu_X + \Delta'; \Theta'}$$

Any delayed inference in D_I would in fact be delayed in D'^A, so this is an eager derivation. The result has, moreover, fewer than n $(\supset\rightarrow)$ inferences, since it omits at least L from D'. Then the induction hypothesis applies to give the needed SCLS derivation A.

Given the derivation A so constructed, we substitute A for D^A in D. The result D^* is an eager derivation; D^* contains an $(\supset\rightarrow^S)$ inference corresponding to L and therefore contains fewer than n uses of $(\supset\rightarrow)$. The induction hypothesis applies to transform D^* to the needed overall derivation. ■

4.4 Block structure

We now revise how we perform case analysis from assumptions. We introduce new rules where local work is discarded in the subderivation written on the right. Some *global* work may be discarded there also! (This helps clarify the structure of derivations.) The right subderivation may address either the (textually) first disjunct or the second disjunct, leading to the two inference figures below.

$$\frac{\Pi, \Pi'; \Gamma, A \vee B^\mu, A^\mu \longrightarrow \Delta; \Theta, \Theta' \qquad \Pi, B^\mu; \longrightarrow; \Theta}{\Pi, \Pi'; \Gamma, A \vee B^\mu \longrightarrow \Delta; \Theta, \Theta'} \vee\rightarrow^B_L$$

$$\frac{\Pi, \Pi'; \Gamma, A \vee B^\mu, B^\mu \longrightarrow \Delta; \Theta, \Theta' \qquad \Pi, A^\mu; \longrightarrow; \Theta}{\Pi, \Pi'; \Gamma, A \vee B^\mu \longrightarrow \Delta; \Theta, \Theta'} \vee\rightarrow^B_R$$

We call these inferences *blocking* $(\vee\rightarrow)$ inferences, or $(\vee\rightarrow^B)$ inferences. We will appeal to two calculi in which these inferences appear. The first, SCLU, permits both ordinary $(\vee\rightarrow)$ and $(\vee\rightarrow^B)$ inferences, without restriction. SCLU is convenient for describing transformations between proofs. The second, SCLB, permits $(\vee\rightarrow^B)$ inferences but not ordinary $(\vee\rightarrow)$ inferences. Obviously, we can use weakening to transform an SCLB or SCLU derivation into a SCLS derivation, so the blocking inference figures are sound. The completeness of SCLB is a consequence of Lemma 9, presented below in Section 4.4.3.

Blocks are more than just boundaries in the proof; they provide a locus for enforcing modularity. We will ensure that a disjunct contributes inferences to the new block where it is introduced. Thanks to this contribution, we can narrow down the choice of goals to restart in a modular way.

This result is made possible only by maintaining the right structure as we introduce $(\vee\rightarrow^B)$ inferences. Section 4.4.1 describes a tool that we can use to render explicit, with path prefixes, the connection between program clauses and any goals that they help establish. Section 4.4.2 transforms individual blocks using this tool to achieve a streamlined form, which already implicitly reflects the logic programming search strategy of focused search on particular goals and program clauses. Section 4.4.3 applies both results in stages to create proofs with an overall modular block structure.

4.4.1 Replacing Herbrand terms

To begin, it is convenient to observe that the use of indexed Herbrand terms allows us to rename Herbrand terms in a proof under certain conditions. These conditions are analyzed in terms of characterizations of the form of sequents in proofs; the key notions are *spanning* and *simplicity*.

Definition 13 (carrier) *The* carrier *of a nonempty Herbrand prefix* $\mu\eta$ *is* $B^{\mu\eta}_{X,\mu\eta}$ *if* η *is* $\eta^u_{A>_iB}(\mu,X)$ *and otherwise, when* η *is* $\eta^u_{\Box_iA}(\mu,X)$*, is* $A^{\mu\eta}_{X,\mu\eta}$.

Definition 14 (spanned) *Say one multiset of tracked prefixed formulas,* Π*, is* spanned *by another,* Θ*, if for every expression occurrence* A^{μ}_{X} *and every nonempty prefix* ν *of* μ *there is an occurrence of the carrier of* ν *in* Θ*. It is easy to see there is a minimal set* Θ *that spans* Π *and that such* Θ *spans itself. A sequent* $\Pi;\Gamma \longrightarrow \Delta;\Theta$ *is* spanned *if* Π *is spanned by* Θ*,* Γ *is spanned by* Θ*,* Δ *is spanned by* Θ *and* Θ *is spanned by* Θ*. A derivation or block is* spanned *if every sequent in it is spanned.*

Definition 15 (simple) *A multiset* Ψ *is* simple *if no expression occurs multiple times in* Ψ*; a sequent of the form* $\Pi;\Gamma \longrightarrow \Delta;\Theta$ *is* simple *if* Π *and* Θ *are simple. A derivation or block is* simple *iff every sequent in it is simple.*

Lemma 7 (Substitution) *Let* $\mathcal{D}$ *be an SCLU derivation with end-sequent*

$$\Pi; \longrightarrow ;\Theta$$

in which no Herbrand terms or Herbrand prefixes appear; consider a spanned simple subderivation $\mathcal{D}'$ *in which a modal Herbrand function* η^u_A *occurs in some sequent, but does not occur in the end-sequent. Let* η^v_A *be a Herbrand function that does not occur in* $\mathcal{D}$*. Then we can construct a proof* $\mathcal{D}^*$ *containing corresponding inferences in a corresponding order to* $\mathcal{D}$ *but in which Herbrand terms and Herbrand prefixes are adjusted so that* η^v_A *is used in place of* η^u_A *precisely in the subderivation corresponding to* $\mathcal{D}'$.

The **proof** follows the proofs of Lemma 24, Lemma 25 and Lemma 26 in [Stone, 1999a]. The result is an induction on the structure of derivations, in which certain technical details must be satisfied because the Herbrand calculus may require not only the replacement of η^u_A itself but also the replacement of Herbrand terms that depend indirectly on η^u_A. It is convenient to begin by replacing any first-order Herbrand term not introduced by a $(\exists \rightarrow)$ or $(\rightarrow \forall)$ inference by a distinguished constant c_0—starting with leaves of the derivation and working downward. This replacement is to ensure that each first-order and modal Herbrand term in $\mathcal{D}$ is determined from an expression in the end-sequent of $\mathcal{D}$ by a finite number of steps of inference. We continue with the systematic replacement of η^u_A and its dependents. In both cases, the form of $\mathcal{D}$ ensures that a finite substitution can systematically rename all these Herbrand terms as required. We use the fact that each sequent is simple and spanned to extend this substitution inductively upward. Because each sequent is spanned the substitution does not need to be extended at $(\Box \rightarrow)$ inferences; because each sequent is simple the substitution can be extended freshly at $(\rightarrow \Box)$ and $(\rightarrow >)$ inferences. Finally, the form of first-order Herbrand terms ensures that a finite extension of the substitution suffices for $(\rightarrow \exists)$ and $(\forall \rightarrow)$ inferences. ■.

4.4.2 Rectifying blocks

In any calculus which involves blocking $(\vee \rightarrow^B)$ inferences, we appeal to the following definitions in understanding how these inferences constrain the goal-directed search of a logic-programming interpreter. First, we have the constituents which the $(\vee \rightarrow^B)$ inferences allow us to find in derivations.

Definition 16 (block) *A* block *of a derivation is a maximal tree of contiguous inferences in which the right subtree of any* $(\vee \rightarrow^B)$ *inference in the block is omitted.*

Second, we will sometimes insist that global work be discarded symmetrically, using the notion of *balanced* sequents and derivations.

Definition 17 (balanced) *A pair of multisets of tracked, prefixed formulas* Π, Θ *is* balanced *if*

- *for any* $\eta = \eta^u_{B>_iC}(\mu, X)$, η *occurs in* Θ *exactly when the expression* $B^{\mu\eta}_{X,\mu\eta}$ *occurs in* Π *and exactly when the expression* $C^{\mu\eta}_{X,\mu\eta}$ *occurs in* Θ*; and*

- *for any* $\eta = \eta^u_{\Box A}(\mu, X)$, η *occurs in* Θ *exactly when the expression* $A^{\mu}_{X,\mu\eta}$ *occurs in* Θ.

A sequent $\Pi; \Gamma \longrightarrow \Delta; \Theta$ *is* balanced *if the pair* Π, Θ *is balanced. A block or derivation is* balanced *if every sequent in the block is balanced.*

Third, we refine the form of proofs which we are willing to count as goal-directed. Now it will often happen that, while each block of a derivation may be eager, the derivation as a whole will not be eager. As observed in [Nadathur and Loveland, 1995], derivations with blocks can nevertheless be seen as eager throughout by reconstructing the (restart) rule as backchaining against the negation of a subgoal. But we will simply consider *blockwise eager* derivations from now on.

Definition 18 (blockwise delayed) *R is* blockwise delayed *exactly when there is a tree of contiguous inferences* $\mathcal{D}'$ *within a single block of* $\mathcal{D}$ *where:* $\mathcal{D}'$ *contains R;* $\mathcal{D}'$ *has a left inference L at the root; and the principal E of R is based in an occurrence of E in the end-sequent of* $\mathcal{D}'$.

Definition 19 (blockwise eager) *D is* blockwise eager *exactly when it contains no blockwise delayed applications of right rules.*

We use the notion of an *isolated block* to obtain an even stronger characterization of a derivation in which work is discarded. In an isolated block, the only expressions preserved across a blocking inference are those that are in some sense intrinsic to the restart problem created by that inference.

Definition 20 (isolated block) *Let* $\mathcal{D}$ *be an SCLU derivation, and let* $\mathcal{B}$ *be a block of* $\mathcal{D}$*. Write the end-sequent of* $\mathcal{B}$ *as* $\Pi; \Gamma \longrightarrow \Delta; \Theta$ *and consider the right subproof of some* $(\vee \rightarrow^B)$ *inference L at the boundary of* $\mathcal{B}$ *has an end-sequent of the form* $\Pi', E; \longrightarrow; \Theta'$*. The* exported *expressions in* Π'*,* Π'_e*, consist of the occurrences of expressions F in* Π' *such that either is F based in an occurrence of F in* Π *or is based in an occurrence of F as the side expression of an inference in which E is also based.*

$\mathcal{B}$ *is* isolated *if the right subproof of each* $(\vee \rightarrow^B)$ *inference L at the boundary of* $\mathcal{B}$ *has an end-sequent of the form* $\Pi', E; \longrightarrow; \Theta'$ *meeting the following conditions: E is the side-expression of L;* Θ' *is the minimal multiset of expressions which spans* Π'_e, E*; and* Π' *is the smallest multiset including* Π'_e, E *for which* Π', Θ' *is balanced.* $\mathcal{D}$ *is* isolated *iff every block of* $\mathcal{D}$ *is isolated.*

Isolation allows us to keep close tabs on the uses of formulas within blocks, which is important for establishing modularity later. In particular, isolation provides a key notion in formalizing the obvious fact that an inference that makes no contribution to an SCLU derivation can be omitted.

Definition 21 (linked) *An expression E in a sequent in an SCLU derivation $\mathcal{D}$ is* linked *if the principal formula of an axiom in the same block of $\mathcal{D}$ as that sequent is based in E. An inference R is* linked *in $\mathcal{D}$ if some side expression of R is linked in each spur of R. A block is* cancelled *if it contains the root of $\mathcal{D}$, or if the side expression E of the $(\vee \rightarrow^{B})$ inference whose spur is the root of the block is linked. A derivation or block is* linked *iff all of the inferences in it are linked.*

Definition 22 (required) *Given a derivation $\mathcal{D}$ with end-sequent*

$$\Pi;\Gamma \longrightarrow \Delta;\Theta$$

we say that an expression occurrence E in Θ or Π is required *iff either it is linked or some block in $\mathcal{D}$ is adjacent to the root block and has an end-sequent*

$$\Pi';\longrightarrow;\Theta'$$

in which Π' or Θ' contains an expression occurrence based in E.

Lemma 8 (Rectification) *We are given a blockwise eager SCLU derivation $\mathcal{D}$ such that: every block in $\mathcal{D}$ is cancelled and isolated; every block in $\mathcal{D}$ other than the root is spanned, linked, balanced and simple; and the end-sequent of $\mathcal{D}$ is balanced. We transform $\mathcal{D}$ to an SCLU derivation $\mathcal{D}'$ in which every block is cancelled, linked, isolated, balanced and simple and every block other than the root is spanned. Every block in $\mathcal{D}'$ other than the root block is identical to a block of $\mathcal{D}$; and the inferences in the root block of $\mathcal{D}$ correspond to inferences in the same order in $\mathcal{D}$ (and so $\mathcal{D}'$ is blockwise eager). If the end-sequent of $\mathcal{D}$ is spanned then $\mathcal{D}'$ is spanned and isolated.*

Proof. We describe a transformation that establishes the following inductive property given $\mathcal{D}$. There are simple multisets $\Pi_M \subseteq \Pi$ and $\Theta_M \subseteq \Theta$, together with multisets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that: any Θ' that spans Π_M includes Θ_M; and for any simple Π' with $\Pi_M \subseteq \Pi' \subseteq \Pi$ and any simple Θ' with $\Theta' \subseteq \Theta$ such that Π' and Θ' are spanned by Θ' and the pair Π', Θ' is balanced, there is a $\mathcal{D}'$ in which every block is cancelled, linked, balanced, balanced and simple, with end-sequent:

$$\Pi';\Gamma' \longrightarrow \Delta';\Theta'$$

In this $\mathcal{D}'$, each expression in Γ' is linked; each expression in Δ' is linked; each Π_M expression that occurs in Π' is required and each Θ_M expression that occurs in Θ' is linked. Every block in $\mathcal{D}'$ other than the root block is identical to a block of $\mathcal{D}$; and the inferences in the root block of $\mathcal{D}$ correspond to inferences in the same order in $\mathcal{D}$. Finally, if Γ' and Δ' are spanned by Θ' then $\mathcal{D}'$ is spanned; if $\mathcal{D}$ is linked then $\mathcal{D}'$ contains all the axioms of $\mathcal{D}$.

At axioms, for $\mathcal{D}$ of

$$\Pi;\Gamma, A^{\mu}_{X} \longrightarrow A^{\mu}_{Y}, \Delta;\Theta$$

Π_M and Θ_M are empty, while $\Gamma' = A^{\mu}_{X}$ and $\Delta' = A^{\mu}_{X}$. Assume we are given simple Π' from Π and simple Θ' from Θ with Π' and Θ' spanned by Θ'. We construct $\mathcal{D}'$ of

$$\Pi';A^{\mu}_{X} \longrightarrow A^{\mu}_{Y};\Theta'$$

If A^{μ}_{X} is spanned by Θ', this axiom is spanned too; the remaining conditions are immediate.

At inferences, consider as a representative case $(\vee \rightarrow)$. D ends:

$$\dfrac{\overset{D_1}{\Pi;\Gamma,A\vee B^{\mu}_{X},A^{\mu}_{X} \longrightarrow \Delta;\Theta} \qquad \overset{D_2}{\Pi;\Gamma,A\vee B^{\mu}_{X},B^{\mu}_{X} \longrightarrow \Delta;\Theta}}{\Pi;\Gamma,A\vee B^{\mu}_{X} \longrightarrow \Delta;\Theta}$$

The blocks of D_1 and D_2 either contain the root or are blocks from D; the Herbrand prefixes in the end-sequents of D_1 and D_2 occur with the same distribution as in D. Therefore we can apply the induction hypothesis to get Π_{M1}, Θ_{M1}, Γ'_1 and Δ'_1 for D_1; we can apply it to get Π_{M2}, Θ_{M2}, Γ'_2 and Δ'_2 for D_2. To transform D itself, we perform case analysis on Γ'_1 and Γ'_2.

If Γ'_1 does not contain an occurrence of A^{μ}_{X}, then $\Pi_M = \Pi_{M1}$, $\Theta_M = \Theta_{M1}$, $\Gamma' = \Gamma'_1$ and $\Delta' = \Delta'_1$; D'_1 suffices to carry through the induction hypothesis.

Similarly, if Γ'_2 does not contain an occurrence of B^{μ}_{X}, then $\Pi_M = \Pi_{M2}$, $\Theta_M = \Theta_{M2}$, $\Gamma' = \Gamma'_2$ and $\Delta' = \Delta'_2$; D'_2 suffices to carry through the induction hypothesis.

Otherwise, we will set up $\Pi_M = \Pi_{M1} \cup \Pi_{M2}$ and $\Theta_M = \Theta_{M1} \cup \Theta_{M2}$ (as sets); by the inductive characterization of Π_{M1}, Π_{M2}, Θ_{M1} and Θ_{M2}, any Θ' that spans both Π_{M2} and Π_{M2} includes both Θ_{M1} and Θ_{M2}. We also set up Γ' as the multiset containing at least one occurrence of $A \vee B^{\mu}_{X}$ and as many expression occurrences of any expression as either are found in $\Gamma'_1 \backslash A^{\mu}_{X}$ or are found in $\Gamma'_2 \backslash B^{\mu}_{X}$; we set up Δ' as the multiset containing as many expression occurrences of any expression as are found in either Δ'_1 or Δ'_2.

To continue, we now consider simple Π' from Π and simple Θ' from Θ such that $\Pi_{M1} \subseteq \Pi'$, $\Pi_{M2} \subseteq \Pi'$, Π' and Θ' are spanned by Θ', and the pair Π', Θ' is balanced. We know that Θ' includes Θ_M. We can apply the inductive property to transform D_1 and D_2 into derivations with the inductive property:

$$\overset{D'_1}{\Pi';\Gamma'_1 \longrightarrow \Delta'_1;\Theta'} \qquad \overset{D'_2}{\Pi';\Gamma'_2 \longrightarrow \Delta'_2;\Theta'}$$

We weaken *the lowest block* of D'_1 on the left by the expressions in Γ^+ and not already in Γ' and on the right by the expressions in Δ^+ and not already in Δ', giving D^+_1. We similarly weaken the lowest block of D'_2 on the left by the expressions in Γ^+ and not already in Γ'_2 and on the right by the expressions in Δ^+ and not already in Δ'_2, giving D^+_2. Only the lowest blocks are affected by the weakening transformations, so other blocks remain cancelled, linked, spanned, isolated and simple; the lowest block in each case remains cancelled. The lowest blocks also remain linked since no inferences are added; and they remain simple (and balanced) because no weakening occurs in the global areas. Construct D' as

$$\dfrac{\overset{D^+_1}{\Pi';\Gamma^+,A^{\mu}_{X} \longrightarrow \Delta^+;\Theta'} \qquad \overset{D^+_2}{\Pi';\Gamma^+,B^{\mu}_{X} \longrightarrow \Delta^+;\Theta'}}{\Pi';\Gamma^+ \longrightarrow \Delta^+;\Theta'}$$

The end-sequent is simple and balanced so the root block is simple and balanced; the inference is linked since A^{μ}_{X} and B^{μ}_{X} are linked in the subderivations, so the root block is linked. The root block remains cancelled as always.

Any Π_M expression is required here because it is required either in D^+_1 in virtue of its membership in Π_{M1} or in D^+_2 in virtue of its membership in Π_{M2}; likewise any Θ_M expression is linked here because it is linked either in D_1^+ in virtue of its membership in Θ_{M1} or in D_2^+ in virtue of its membership in Θ_{M2}. Thus, except for the spanning conditional, we have shown everything we need of this D'.

Finally, then, if Γ' and Δ' is spanned by Θ', Δ_1' and Δ_2' are spanned by Θ' and Γ_1' and Γ_2' are spanned by Θ' in the resulting (spanned) subderivations D_1' and D_2'. This shows that the end-sequent of D' is also spanned, so D' itself is spanned.

This reasoning is representative of the construction required also for $(\wedge \rightarrow)$, $(\exists \rightarrow)$, $(\forall \rightarrow)$, $(\rightarrow \wedge)$, $(\rightarrow \vee)$, $(\rightarrow \exists)$, $(\rightarrow \forall)$, (decide) and (restart). It applies also for $(\supset\rightarrow^S)$, with the obvious caveat that we do not weaken the left subderivation to match local left expressions, since the form of the $(\supset\rightarrow^S)$ inference requires there to be none.

Next we have $(\vee \rightarrow^B)$; we consider the representative case of $(\vee \rightarrow_L^B)$. D ends:

$$\frac{\overset{D_1}{\Pi_0, \Pi; \Gamma, A \vee B_X^\mu, A_X^\mu \longrightarrow \Delta; \Theta_0, \Theta} \qquad \overset{D_2}{\Pi_0, B_X^\mu; \longrightarrow \Theta_0}}{\Pi_0, \Pi; \Gamma, A \vee B_X^\mu \longrightarrow \Delta; \Theta_0, \Theta}$$

We treat this specially to respect the block boundary before D_2. In particular, we apply the induction hypothesis to D_1 (as we may since its end-sequent has the same distribution of Herbrand prefixes as does that of D), to get Π_{M1}, Θ_{M1}, Γ_1' and Δ_1'. If A_X^μ does not occur in Γ_1', we let $\Pi_M = \Pi_{M1}$, $\Theta_M = \Theta_{M1}$, $\Gamma' = \Gamma_1'$ and $\Delta' = \Delta_1'$; any derivation D_1' constructed from appropriate Π' and Θ' suffices to carry through the induction hypothesis.

Otherwise, we get $\Pi_M = \Pi_{M1} \cup \Pi_{e0}$ (as a set), $\Theta_M = \Theta_{M1}$; any Θ' that spans Π_M also spans Π_{M1} and so includes Θ_M. $\Delta' = \Delta_1'$ and Γ' contains Γ_1' with the occurrence of A_X^μ removed, together with an occurrence of $A \vee B_X^\mu$ if Γ_1' does not already contain such an expression.

Assume simple Π' with $\Pi_M \subseteq \Pi' \subseteq \Pi$ and simple Θ' with $\Theta' \subseteq \Theta$ with Π' and Θ' spanned by Θ' and the pair Π', Θ' balanced. As before, we must have Θ_M included in Θ'. We therefore obtain D_1' by the inductive property; we then weaken D_1' locally within the lowest block by $A \vee B_X^\mu$ on the left if necessary, to obtain a good derivation D_1^*.

The needed D' is now constructed as:

$$\frac{\overset{D_1^*}{\Pi'; \Gamma', A_X^\mu \longrightarrow \Delta'; \Theta'} \qquad \overset{D_2}{\Pi_0, B_X^\mu; \longrightarrow \Theta_0}}{\Pi'; \Gamma' \longrightarrow \Delta'; \Theta'}$$

We first argue that the construction instantiates the $(\vee \rightarrow_L^B)$ inference rule. Every Herbrand prefix in Π_{0e} and B_X^μ occurs in Π' or Γ', so Π_{0e} and B_X^μ are spanned by Θ'. But because the root block in D is isolated, Π_{0e} and B_X^μ are spanned minimally by Θ_0. Thus $\Theta_0 \subseteq \Theta'$. $\Pi_{0e} \subseteq \Pi_M$ by construction; by isolation Π_0 is the smallest set such that the pair of Π_0, Θ_0 is balanced. But since Π', Θ' is balanced, $\Pi_0 \subseteq \Pi'$.

Now we show that D' so constructed has the needed properties. The end-sequent is simple and balanced so the root block is simple and balanced. The inference is linked: A_X^μ is linked in D_1' by the induction hypothesis; B_X^μ is linked in D_2 because D_2 begins a new block which by assumption is cancelled. The root block remains cancelled as always. Any Π_M expression is required here because either a corresponding expression Π_{0e} in the new block at the left subderivation is based on it, or because it is required in D_1'. Every Θ_M is linked because it is linked in D_1^*.

Finally, if Γ' and Δ' are spanned by Θ', then Δ'_1 and Γ'_1 are spanned by Θ'_1. The new subderivation D'_1 is therefore spanned by the inductive property; this ensures that the overall derivation is spanned.

Next consider $(\Box \rightarrow)$. D ends:

$$\frac{\begin{array}{c} D_1 \\ \Pi; \Gamma, \Box_i A^{\mu}_{X}, A^{\mu\nu}_{X,\mu\nu} \longrightarrow \Delta; \Theta \end{array}}{\Pi; \Gamma, \Box_i A^{\mu}_{X} \longrightarrow \Delta; \Theta}$$

As always, we apply the induction hypothesis to D_1 (as we may since the Herbrand prefixes on Π and Θ formulas remain the same) to obtain Π_{M1}, Θ_{M1}, Γ'_1 and Δ'_1. If $A^{\mu\nu}_{X,\mu\nu}$ does not occur in Γ'_1, we let $\Pi_M = \Pi_{M1}$, $\Theta_M = \Theta_{M1}$, $\Gamma' = \Gamma'_1$ and $\Delta' = \Delta'_1$; any subderivation D'_1 obtained by the inductive property suffices to witness the inductive property for D.

Otherwise we obtain Γ' by extending Γ'_1 by the principal expression $\Box_i A^{\mu}_{X}$ if necessary and eliminating the side expression $A^{\mu\nu}_{X,\mu\nu}$; $\Pi_M = \Pi_{M1}$, $\Theta_M = \Theta_{M1}$ and $\Delta' = \Delta'_1$. (Since these are common to the subderivation, any Π' that spans Π_M includes Θ_M.) Now we consider Π' with $\Pi_M \subseteq \Pi' \subseteq \Pi$ and Θ' with $\Theta' \subseteq \Theta$, Π' and Θ' spanned by Θ' and the pair Π', Θ' balanced. As always, we have $\Theta_M \subseteq \Theta'$. We obtain D'_1 using Π' and Θ', and weaken the lowest block by local formulas; calling the result D^+_1, we can produce D' by the following construction:

$$\frac{\begin{array}{c} D^+_1 \\ \Pi'; \Gamma', A^{\mu\nu}_{X,\mu\nu} \longrightarrow \Delta'; \Theta' \end{array}}{\Pi'; \Gamma' \longrightarrow \Delta'; \Theta'}$$

Everything is largely as before. The key new reasoning comes when we assume that Γ' and Δ' are spanned by Θ'. We must argue that $\Gamma', A^{\mu\nu}_{X,\mu\nu}$ is in fact spanned by Θ'. Since $A^{\mu\nu}_{X,\mu\nu}$ is linked in D^+_1, there must be an axiom in this block which is based in $A^{\mu\nu}_{X,\mu\nu}$; indeed, since the expression occurs as a local antecedent, this axiom must occur within the segment. This axiom must pair expressions prefixed by a path μ' where $\mu\nu$ is a prefix of μ'. But because D' remains blockwise eager, no inferences apply to Δ' or Θ' formulas within the segment (nor can they in this fragment augment the Δ' or Θ' formulas within the segment); therefore some Δ' expression is associated with Herbrand prefix μ'. But since Δ' is spanned by Θ', we have that every prefix of μ' is associated with some Θ' expression; so every prefix of $\mu\nu$ is associated with some Θ' expression. Thus D^+_1 is spanned and in turn D' is spanned.

We have one last representative class of inferences in D: $(\rightarrow \Box)$ and $(\rightarrow >)$. We illustrate with the case where D ends in $(\rightarrow >)$:

$$\frac{\begin{array}{c} D_1 \\ \Pi, A^{\mu\eta}_{X,\mu\eta}; \Gamma \longrightarrow \Delta, A >_i B^{\mu}_{X}; \Theta, B^{\mu\eta}_{X,\mu\eta} \end{array}}{\Pi; \Gamma \longrightarrow \Delta, A >_i B^{\mu}_{X}; \Theta}$$

We begin by applying the induction hypothesis to D_1 (as we can, given the symmetric extension of Π and Θ by labeled expressions). We obtain Θ_{M1}, Π_{M1}, Γ'_1 and Δ'_1; we consider alternative cases in response to Θ and Θ_{M1}. First we suppose $B^{\mu\eta}_{X,\mu\eta} \notin \Theta$. It follows by our assumption about D that $A^{\mu\eta}_{X,\mu\eta} \notin \Pi$ either, nor does η occur in Θ. For this case, we start by defining an overall Π_M

and Θ_M: Θ_M is Θ_{M1} with any occurrence of $B^{\mu\eta}_{X,\mu\eta}$ eliminated; Π_M is Π_{M1} with any occurrence of $A^{\mu\eta}_{X,\mu\eta}$ eliminated. Π_M contains no occurrences of $\mu\eta$, since Π does not; thus given the inductive property of Θ_{M1} and Π_{M1}, any Θ' that spans Π_M spans Θ_M. We define Γ' and Δ' so that $\Gamma' = \Gamma'_1$ and Δ' contains Δ'_1 together with an occurrence of $A >_i B^{\mu}_{X}$, provided Δ'_1 does not already contain one and $B^{\mu\eta}_{X,\mu\eta} \in \Theta_{M1}$ or $A^{\mu\eta}_{X,\mu\eta} \in \Pi_{M1}$. So, assume we are given simple Π' with $\Pi_M \subseteq \Pi' \subseteq \Pi$ and simple Θ' with $\Theta' \subseteq \Theta$ (and so $\Theta_M \subseteq \Theta'$) such that Π' and Θ' are spanned by Θ' and the pair Π', Θ' is balanced.

We consider whether $B^{\mu\eta}_{X,\mu\eta} \in \Theta_{M1}$ or $A^{\mu\eta}_{X,\mu\eta} \in \Pi_{M1}$. If neither, we apply the induction hypothesis to D_1 for the case that Θ'_1 is Θ' and Π'_1 is Π'. The resulting derivation D'_1 serves as D'.

Otherwise, $B^{\mu\eta}_{X,\mu\eta} \in \Theta_{M1}$ or $A^{\mu\eta}_{X,\mu\eta} \in \Pi_{M1}$; we apply the inductive property of D_1 for the case that Θ'_1 is $\Theta', B^{\mu\eta}_{X,\mu\eta}$ and Π'_1 is $\Pi', A^{\mu\eta}_{X,\mu\eta}$ (clearly Π'_1 and Θ'_1 are spanned by Θ'_1 assuming Π' and Θ' are spanned by Θ'; the pair Π'_1, Θ'_1 is also balanced given its symmetric extension). If $B^{\mu\eta}_{X,\mu\eta} \in \Theta_{M1}$, by the inductive property it is linked. If $A^{\mu\eta}_{X,\mu\eta} \in \Pi_{M1}$, it is required, but we shall show that it is in fact linked. By the definition of being required, the other possibility is that there is a block adjacent to the root block of D'_1 with end-sequent

$$\Pi'', E; \longrightarrow \Theta''$$

in which the $(\vee \rightarrow^B)$ inference R that bounds the block is based in E and Π'', E or Θ'' contains an expression occurrence based in $A^{\mu\eta}_{X,\mu\eta}$. But since the original block is isolated in the original D, it is E that must be based in $A^{\mu\eta}_{X,\mu\eta}$. But then R is based in $A^{\mu\eta}_{X,\mu\eta}$ and R is linked: in particular its side expression in the left spur) must be linked; so $A^{\mu\eta}_{X,\mu\eta}$ is linked too.

Thus we can weaken D'_1 in its lowest block if necessary by $A >_i B^{\mu}_{X}$ as a local right formula (in Γ), producing D^+_1; D^+_1 remains good by the same argument as the earlier cases. Thus we can construct D' as:

$$\cfrac{\begin{array}{c} D^+_1 \\ \Pi', A^{\mu\eta}_{X,\mu\eta}; \Gamma' \longrightarrow \Delta', A >_i B^{\mu}_{X}; \Theta', B^{\mu\eta}_{X,\mu\eta} \end{array}}{\Pi'; \Gamma' \longrightarrow \Delta'; \Theta'}$$

The end-sequent here is simple and balanced, so the whole root block is simple and balanced. The new inference is linked (in virtue of the linked occurrence of one side expression—$A^{\mu\eta}_{X,\mu\eta}$ or $B^{\mu\eta}_{X,\mu\eta}$) so the whole root block is linked. The root block is of course cancelled. Each element of Π_M is required because it is an element of Π_{M1} and required in the immediate subderivation; each element of Θ_M is linked, because it is an element of Θ_{M1} and therefore linked in the immediate subderivation.

To conclude the case, suppose the end-sequent of D is spanned and that Γ' and Δ' are spanned by Θ'; it follows that same property applies to D_1 so the subderivation is spanned. Then the end-sequent must also be spanned.

The alternative case has $B^{\mu\eta}_{X,\mu\eta} \in \Theta$. By assumption, it also has $A^{\mu\eta}_{X,\mu\eta} \in \Pi$. We therefore define an overall Π_M and Θ_M directly as Π_{M1} and Θ_{M1}, respectively; similarly $\Gamma' = \Gamma'_1$ and $\Delta' = \Delta'_1$. To construct the needed D' for appropriate Π' and Θ', we simply apply the induction hypothesis to D_1 for the case that Θ'_1 is Θ' and Π'_1 is Π'. The resulting derivation D'_1 suffices.

Having completed the induction, we argue that we can obtain an overall D' that is isolated, assuming the original D is not only isolated but spanned. Apply the inductive result to D for the case

$\Pi' = \Pi$ and $\Theta' = \Theta$; since $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ we obtain a spanned derivation D' ending

$$\Pi; \Gamma' \longrightarrow \Delta'; \Theta$$

Consider the end-sequent of any block other than the root in D'; it is

$$\Pi_0, E; \longrightarrow; \Theta_0$$

where a corresponding block occurs in D. I argue by contradiction that for any $F \in \Pi_0$ either $F \in \Pi$ or F is based in an occurrence of F as the side expression of an inference in D' in which E is also based. (This will show that D' is isolated.) So consider an exceptional F. Since D is isolated, if $F \notin \Pi$, F is based in an occurrence of F as the side expression of an inference in D in which E is also based; this inference introduces some path symbol η which occurs in the label of F and E. In D', E can not be based in such an inference; otherwise F would also be based in that inference, since D' is simple. (We have assumed that F is not based in such an inference.) But in this case the expression in the end-sequent of D' on which E is based must contain η. Because the end-sequent of D' is spanned the form of Π and Θ is constrained in D, F must occur in Π. This is absurd. ■

We conclude Section 4.4.2 by observing some facts about this construction. First, let D' be a derivation obtained by the construction of Lemma 8, and suppose D' is weakened (in a spanned and balanced way) to D'' by adding occurrences of global expressions that either already occur in the end-sequent of D' or never occur as global expressions in D'. Then a straightforward induction shows that D' is obtained again from D'' by the construction of Lemma 8.

Second, observe that if D' is a derivation obtained by the construction of Lemma 8, and D'' is obtained from D'' by the renaming of Herbrand prefixes (as in Lemma 7), then straightforward induction shows that D'' is obtained again from D'' by the construction of Lemma 8.

Third, let D' be a derivation for which the construction of Lemma 8 yields itself. Let ν be a prefix and let the $\Pi; \Theta$ be the smallest balanced pair where Θ contains all the carriers of prefixes of ν introduced in D'. Suppose each expression in Π and Θ has the property that at most one inference of D' has an occurrence of that expression as a side expression. Consider a derivation D'' obtained from D' by weakening globally by Π (on the left) and by Θ (on the right). Let D^* be the result of applying the construction of Lemma 8 to D''. Then D^* contains any subderivation of D' whose end-sequent contains Π and Θ as global formulas. Again this is a straightforward induction; the base case considers a subderivation of D' whose end-sequent contains Π and Θ as global formulas; in this case we apply the first observation. Unary inferences extend the claim immediately. At binary inferences, one subderivation must be unchanged, by the first observation: since Π and Θ are introduced on a unique path, each Π and Θ formula never occurs or already occurs in the end-sequent in that subderivation. Thus the other subderivation necessarily appears in the derivation obtained by the construction of Lemma 8.

4.4.3 Block conversion

We now have the background required to perform the conversion to block structure.

Lemma 9 *We are given a blockwise eager SCLS derivation D whose end-sequent is spanned and balanced and takes the form:*

$$\Pi; \longrightarrow; \Theta$$

We transform D into a blockwise eager SCLB derivation in which every block is cancelled, linked, isolated, simple, balanced and spanned.

Proof. Our induction hypothesis is stronger than the lemma. We assume a blockwise eager SCLU derivation D with end-sequent of the form

$$\Pi; \longrightarrow ; \Theta$$

in which every block is cancelled, linked, isolated, simple, balanced and spanned, such that that the subproof rooted at any $(\vee \rightarrow)$ inference in D is an SCLS derivation. And we identify a distinguished expression occurrence E in the end-sequent of D which is linked. By Lemma 8, it is straightforward to obtain such a derivation from the SCLS derivation (containing only a single block) that we have assumed. We transform D into a blockwise eager SCLB derivation in which every block is cancelled, linked, isolated, simple, balanced and spanned and in which E is also linked; we perform induction on the number of $(\vee \rightarrow)$ inferences in D.

In the base case there are no $(\vee \rightarrow)$ inferences, so D itself is an SCLB derivation.

In the inductive case, we assume D with n $(\vee \rightarrow)$ inferences, and assume the hypothesis true for derivations with fewer. We find an application L of $(\vee \rightarrow)$ with no other closer to the root of D. We will transform D to eliminate L.

Let D' denote the smallest subderivation of D containing the full block of D in which L occurs. Explicitly, D' may be D itself; otherwise, D' is rooted at the right subderivation of the highest $(\vee \rightarrow^B)$ inference below L—an inference we will refer to as H. In either case, our assumptions allow us to identify a distinguished linked expression F in the end-sequent of D': either the assumed E from D, or the side expression of the inference H (assumed cancelled). Suppose $A \vee B^{\nu}_{Y}$ is the principal of L. We can apply Lemma 7 to rename $A \vee B^{\nu}_{Y}$ to $A \vee B^{\mu}_{X}$ in such a way that each symbol in μ that is introduced in D' is introduced by a unique inference there. Now we can infer the following schema for D':

$$\left[\dfrac{\begin{array}{c} D^A \\ \Pi_0, F, \Pi; \Gamma, A \vee B^{\mu}_{X}, A^{\mu}_{X} \longrightarrow \Delta; \Theta_0, \Theta \end{array} \qquad \begin{array}{c} D^B \\ \Pi_0, F, \Pi; \Gamma, A \vee B^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Delta; \Theta_0, \Theta \end{array}}{\Pi_0, F, \Pi; \Gamma, A \vee B^{\mu}_{X} \longrightarrow \Delta; \Theta_0, \Theta}\ L\right]$$
$$\begin{array}{c} D^L \\ \Pi_0, F; \longrightarrow ; \Theta_0 \end{array}$$

That is, the subderivation of D' below L isD^L; the right subderivation above L (in which B is assumed) is D^B; the left is D^A.

We will use the inferences from D^L to construct alternative smaller derivations in place of D^A and D^B. By Θ', indicate the minimal set of formulas required in addition to Θ_0 to span A^{μ}_{X}; by Π' indicate the minimal set of formulas required in addition to Π_0, F and A^{μ}_{X} to ensure that the pair given by $\Pi_0, \Pi', F, A^{\mu}_{X}$ and Θ_0, Θ' is balanced. (This is well-defined because the sequent $\Pi_0, F \longrightarrow \Theta_0$ is already spanned and balanced.) Now we can construct two new subderivations D'^A and D'^B given respectively as follows:

$$\left[\begin{array}{c} \Pi'+A^{\mu}_X+D^A+\Theta' \\ \dfrac{\Pi_0,F,\Pi,\Pi',A^{\mu}_X;\Gamma,A\vee B^{\mu}_X,A^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta,\Theta'}{\Pi_0,F,\Pi,\Pi',A^{\mu}_X;\Gamma,A\vee B^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta,\Theta'}\text{ decide} \end{array}\right]$$
$$\begin{array}{c} \Pi'+A^{\mu}_X+D^L+\Theta' \\ \Pi_0,F,\Pi',A^{\mu}_X;\longrightarrow;\Theta_0,\Theta' \end{array}$$

$$\left[\begin{array}{c} [\Pi'+B^{\mu}_X+D^B+\Theta'] \\ \dfrac{\Pi_0,F,\Pi,\Pi',B^{\mu}_X;\Gamma,B\vee B^{\mu}_X,B^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta,\Theta'}{\Pi_0,F,\Pi,\Pi',B^{\mu}_X;\Gamma,B\vee B^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta,\Theta'}\text{ decide} \end{array}\right]$$
$$\begin{array}{c} \Pi'+B^{\mu}_X+D^L+\Theta' \\ \Pi_0,F,\Pi',B^{\mu}_X;\longrightarrow;\Theta_0,\Theta' \end{array}$$

That is, we weaken D^A and D^B by global versions of the side expression of inference L throughout their *lowest blocks*; we apply a (decide) inference to obtain a new subderivation to substitute for the subderivation rooted at L in D^L. We weaken by sufficient additional formulas globally in the *lowest blocks* to ensure that the end-sequents of these derivations are balanced and spanned.

Since we have changed only the lowest block here, we can now apply Lemma 8 to obtain corresponding derivations D^A_I and D^B_I in which every block is cancelled, linked, isolated, simple, balanced and spanned. In light of our first observation about the construction of Lemma 8, we can see that the inferences of D^A are preserved up to the new (decide) inference. And in light of our third observation about the construction of Lemma 8, given the unique inferences introducing Θ_0 and Π_0, this (decide) inference must be preserved in D^A_I. Thus A^{μ}_X is linked in D^A_I and for analogous reasons B^{μ}_X is linked in D^B_I. These derivations satisfy the induction hypothesis as deductions with fewer than n $(\vee\longrightarrow)$ inferences; we can apply the induction hypothesis with A^{μ}_X and B^{μ}_X as the distinguished linked formulas to preserve. This results in SCLB derivations $\mathcal{A}$ and $\mathcal{B}$ with the same end-sequents as D'^A and D'^B, in which every block is cancelled, linked, isolated, simple and spanned, and in which respectively A^{μ}_X and B^{μ}_X are linked.

We need only one of $\mathcal{A}$ and $\mathcal{B}$ to reconstruct D' using blocking inferences. For example, we obtain a proof using $(\vee\longrightarrow^B_L)$ by using $\mathcal{B}$ in place of D^B as schematized below:

$$\left[\dfrac{\begin{array}{c} D^A \\ \Pi_0,F,\Pi;\Gamma,A\vee B^{\mu}_X,A^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta \end{array} \qquad \begin{array}{c} \mathcal{B} \\ \Pi_0,F,\Pi',B^{\mu}_X \longrightarrow \Theta_0,\Theta' \end{array}}{\Pi_0,F,\Pi;\Gamma,A\vee B^{\mu}_X \longrightarrow \Delta;\Theta_0,\Theta}\vee\longrightarrow^B_L\right]$$
$$\begin{array}{c} D^L \\ \Pi_0,F;\longrightarrow;\Theta_0 \end{array}$$

In a complementary way, we obtain a proof using $(\vee \rightarrow_R^B)$ by using A in place of D^A as schematized below:

$$\left[\frac{\begin{array}{c}D^B\\ \Pi_0, F, \Pi; \Gamma, A \vee B_X^\mu, B_X^\mu \longrightarrow \Delta; \Theta_0, \Theta\end{array} \qquad \begin{array}{c}A\\ \Pi_0, F, \Pi', A_X^\mu \longrightarrow \Theta_0, \Theta'\end{array}}{\Pi_0, F, \Pi; \Gamma, A \vee B_X^\mu \longrightarrow \Delta; \Theta_0, \Theta} \vee \rightarrow_L^B\right]$$
$$\begin{array}{c}D^L\\ \Pi_0, F; \longrightarrow; \Theta_0\end{array}$$

Note that the root block is isolated in both cases, because we have added only as many formulas to Π' and Θ' as are necessary to obtain a balanced, spanned sequent; the remaining expressions originate in the end-sequent of the previous block, which we know was isolated. Thus, in both cases, we have blockwise eager derivations in which every block is cancelled, isolated, simple, balanced and spanned, in which fewer than n $(\vee \rightarrow)$ inferences are used, and in which only the root block may fail to be linked. We thus need to apply the construction of Lemma 8 again to ensure that the root block is linked. It is possible for the distinguished occurrence of F not to be linked in one of the resulting derivations, but not both. To see this, consider applying the construction of Lemma 8 to D' itself, as a test: the result will be D' since D' is linked. Starting from D^A and D^B and axioms elsewhere, each inference in D' corresponds to an inference in the alternative derivations schematized above. We can argue by straightforward induction that no formula is linked in the reconstructed D' unless it is also linked in the one of the corresponding reconstructed alternative derivations. And F is linked in D'.

Call the derivation in which F is linked D''; we substitute D'' for D' in D. Since F remains linked in D'', when we do so, we obtain a blockwise eager SCLU derivation with an appropriate end-sequent, with fewer original $(\vee \rightarrow)$ inferences, and in which every block remains cancelled, linked, isolated, simple, balanced and spanned, and in which $(\vee \rightarrow)$ inferences lie at the root of SCLS derivations. Applying the induction hypothesis to the result gives the required SCLB derivation. ■

4.5 Modularity

The final step in the justification of the calculus is to enforce modularity. This again is accomplished by adapting the rules of the sequent calculus. We rewrite inference figures so that every sequent in the proof has at most one formula in the left and right local areas, and further if a right rule applies the left local area is empty. The new inferences are presented in Definition 23 and 24 as the sequent calculus SCLP. Definition 23 shows the rules for decomposing program statements; Definition 24 shows the rules for decomposing goals.

Definition 23 (Logic programming calculus—programs) *The following inference figures describe the logic programming sequent calculus SCLP as it applies to program clauses.*

1. *axiom—A atomic:*

$$\Gamma; A^\nu \longrightarrow A^\nu; \Delta$$

2. *decision (program consultation)—again A atomic:*

$$\frac{\Gamma, P^{\mu}_{X}; P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma, P^{\mu}_{X}; \longrightarrow A^{\nu}_{Y}; \Delta}\ \textit{decide}$$

3. *conjunctive:*

$$\frac{\Gamma; P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma; P \wedge Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \wedge \rightarrow_{L}$$

$$\frac{\Gamma; Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma; P \wedge Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \wedge \rightarrow_{R}$$

4. *disjunctive:*

$$\frac{\Gamma; P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta \qquad \Gamma, Q^{\mu}_{X}; \longrightarrow ; \Delta}{\Gamma; P \vee Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \vee \rightarrow_{L}$$

$$\frac{\Gamma; Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta \qquad \Gamma, P^{\mu}_{X}; \longrightarrow ; \Delta}{\Gamma; P \vee Q^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \vee \rightarrow_{R}$$

5. *implication:*

$$\frac{\Gamma; \longrightarrow Q^{\mu}_{X}; \Delta \qquad \Gamma; P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma; Q \supset P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \supset\rightarrow$$

6. *necessity—subject to the side condition that there is a typing derivation* $S, \Xi_{\nu} \triangleright \mu/\mu\nu : i$*:*

$$\frac{\Gamma; P^{\mu\nu}_{X,\mu\nu} \longrightarrow A^{\nu'}_{Y}; \Delta}{\Gamma, \Box_{i} P^{\mu}_{X} \longrightarrow A^{\nu'}_{Y}; \Delta}\ \Box_{i} \rightarrow$$

7. *existential—subject to the side condition that h is* $h_{\exists xP}(\mu, X)$*:*

$$\frac{\Gamma; P[h(X)/x])^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma; \exists x.P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \exists \rightarrow$$

8. *universal—subject to the side condition that there is a typing derivation* $S, \Xi_{t,\mu} \triangleright t : \mu$*:*

$$\frac{\Gamma; P[t/x]^{\mu}_{X,t} \longrightarrow A^{\nu}_{Y}; \Delta}{\Gamma; \forall x.P^{\mu}_{X} \longrightarrow A^{\nu}_{Y}; \Delta}\ \forall \rightarrow$$

Definition 24 (Logic programming calculus—goals) *The following inference figures describe the logic programming sequent calculus SCLP as it applies to goals.*

1. *restart:*

$$\frac{\Gamma; \longrightarrow G^{\nu}_{X}; G^{\nu}_{X}, \Delta}{\Gamma; \longrightarrow ; G^{\nu}_{X}, \Delta}\ \textit{restart}$$

2. *conjunctive goals:*

$$\frac{\Gamma; \longrightarrow F^{\mu}_{X}; \Delta \qquad \Gamma; \longrightarrow G^{\mu}_{X}; \Delta}{\Gamma; \longrightarrow F \wedge G^{\mu}_{X}; \Delta} \rightarrow \wedge$$

3. *disjunctive goals:*

$$\frac{\Gamma; \longrightarrow F^{\mu}_{X}; \Delta}{\Gamma; \longrightarrow F \vee G^{\mu}_{X}; \Delta} \rightarrow \vee_L$$

$$\frac{\Gamma; \longrightarrow G^{\mu}_{X}; \Delta}{\Gamma; \longrightarrow F \vee G^{\mu}_{X}; \Delta} \rightarrow \vee_R$$

4. *necessary goals—where η is $\eta^{u}_{A}(\mu, X)$ for A^{μ}_{X} the principal of the rule and for some u for which η^{u}_{A} does not occur in Δ or Γ:*

$$\frac{\Gamma, F^{\mu\eta}_{X,\mu\eta}; \longrightarrow G^{\mu\eta}_{X,\mu\eta}; G^{\mu\eta}_{X,\mu\eta}, \Delta}{\Gamma; \longrightarrow F >_i G^{\mu}_{X}; \Delta} \rightarrow \Box_i \supset$$

$$\frac{\Gamma; \longrightarrow G^{\mu\eta}_{X,\eta}; G^{\mu\eta}_{X,\mu\eta}, \Delta}{\Gamma; \longrightarrow \Box_i G^{\mu}_{X}; \Delta} \rightarrow \Box_i$$

5. *universal goals—subject to the side condition that h is $h_{\forall x G}(\mu, X)$:*

$$\frac{\Gamma; \longrightarrow G[h/x]^{\mu}_{X,h}; \Delta}{\Gamma; \longrightarrow \forall x. G^{\mu}_{X}; \Delta} \rightarrow \forall$$

6. *existential goals—subject to the side condition that there is a typing derivation $S, \Xi_{t,\mu} \triangleright t : \mu$:*

$$\frac{\Gamma; \longrightarrow G[t/x]^{\mu}_{X,t}; \Delta}{\Gamma; \longrightarrow \exists x. G^{\mu}_{X}; \Delta} \rightarrow \exists$$

SCLP proofs can be rewritten to SCLB rules by a weakening transformation. Conversely, rewriting SCLB proofs to SCLP proofs is accomplished by induction on the structure of proofs. The transformation is possible because multiple formulas in sequents are needed only for passing ambiguities and work done across branches in the search; this is ruled out by the use of $(\vee \rightarrow_L)$, $(\vee \rightarrow_R)$ and $(\supset \rightarrow_L)$.

Lemma 10 *Given a blockwise eager SCLB derivation D, with end-sequent*

$$\Pi; \longrightarrow; \Theta$$

in which every block is linked, simple and spanned, we can construct a corresponding SCLP derivation of the same end-sequent.

Proof. We construct by induction on the structure of the linked, simple, spanned, blockwise eager SCLB derivation D with end-sequent

$$\Pi; \Gamma \longrightarrow \Delta; \Theta$$

an SCLP derivation D' of which four additional properties hold:

- the end-sequent of D' takes the form

$$\Pi; \Gamma' \longrightarrow \Delta'; \Theta$$

  with $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$;

- D' contains in each segment or block all and only the axioms of the corresponding segment or block of D;

- whenever D' contains a sequent of the form

$$\Pi^*; \Gamma^* \rightarrow F; \Theta^*$$

  F is the only right formula on which an axiom in that block is based; and

- whenever D' contains a sequent of the form

$$\Pi^*; F \rightarrow \Delta^*; \Theta^*$$

  then F is the only left formula on which an axiom in that segment is based.

In the base case, D is

$$\Pi; \Gamma, A_X^\mu \longrightarrow B_X^\nu, \Delta; \Theta$$

and D' is

$$\Pi; A_X^\mu \longrightarrow B^\nu; \Theta$$

Supposing the claim true for proofs of height h, consider a proof D with height h+1. We consider cases for the different rules with which D could end.

The treatment of $(\rightarrow \wedge)$ is representative of the case analysis for the right rules other than $(\rightarrow >)$. D ends

$$\dfrac{\Pi; \longrightarrow A_X^\mu, A \wedge B_X^\mu, \Delta; \Theta \qquad \Pi; \longrightarrow B_X^\mu, A \wedge B_X^\mu, \Delta; \Theta}{\Pi; \longrightarrow A \wedge B_X^\mu, \Delta; \Theta} \rightarrow \wedge$$

(It is a consequence of Lemma 5 that in the initial derivation there is an empty local area.) We simply apply the induction hypotheses to the immediate subderivations. If the resulting derivations end with (restart), consider the immediate subderivation of the results, otherwise consider the results themselves. These derivations end

$$\Pi; \longrightarrow C; \Theta$$
$$\Pi; \longrightarrow D; \Theta$$

We must have $C = A$; we know from the structure of D that A is linked, and A could not be linked in D unless $C = A$ since D' shows that all of the axioms in D derive from C. For the same reason $D = B$. So we can combine the resulting proofs by an $(\rightarrow \wedge)$ inference to give the needed D'.

The case of $(\rightarrow >)$ proceeds similarly, but relies on an additional observation. D ends

$$\dfrac{\begin{array}{c} D_1 \\ \Pi, A_{X,\mu\eta}^{\mu\eta}; \longrightarrow \Delta, A >_i B_X^\mu; B_{X,\mu\eta}^{\mu\eta}, \Theta \end{array}}{\Pi; \longrightarrow \Delta, A >_i B_X^\mu; \Theta} \rightarrow >$$

We apply the induction hypothesis to D_1 and eliminate any final (restart) inference. This gives us a derivation D_1' of

$$\Pi, A^{\mu\eta}_{X,\mu\eta}; \longrightarrow E; B^{\mu\eta}_{X,\mu\eta}, \Theta$$

If we know that the B-side expression of this inference is linked in this block, then we can conclude, as before, that E is an occurrence of the expression $B^{\mu\eta}_{X,\mu\eta}$. We show this as follows. We know from the structure of D only that *one* of the A-expression and the B-expression must be linked. However, it is straightforward to show that no left expression $A^{\mu\eta}_{X,\mu\eta}$ is linked in an SCLP derivation with a local goal C^{ν}_{Y} unless $\mu\eta$ is a prefix of ν. (The argument is a straightforward variant for example of [Stone, 1999b, Lemma 2].) Since D is simple and spanned, η must be new; $B^{\mu\eta}_{X,\mu\eta}$ is the only expression whose associated path term has $\mu\eta$ as a prefix.

Thus, we construct D' using an SCLP inference as

$$\dfrac{\overset{D_1'}{\Pi, A^{\mu\eta}_{X,\mu\eta}; \longrightarrow B^{\mu\eta}_{X,\mu\eta}; B^{\mu\eta}_{X,\mu\eta}, \Theta}}{\Pi; \longrightarrow A >_i B^{\mu}_{X}; \Theta} \rightarrow >$$

Now suppose D ends in a left rule other than $(\supset\rightarrow^S)$ or $(\vee \rightarrow^B)$. We take $(\wedge \rightarrow)$ as a representative case; then D is:

$$\dfrac{\overset{D_1}{\Pi; \Gamma, A \wedge B^{\mu}_{X}, A^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Delta; \Theta}}{\Pi; \Gamma, A \wedge B^{\mu}_{X} \longrightarrow \Delta; \Theta} \wedge \rightarrow$$

Apply the induction hypothesis to D_1. If the result ends in a (decide) inference, let D_1' be the immediate subderivation of the result; otherwise let D_1' be the result itself. D_1' is an SCLP derivation with an end-sequent of the form:

$$\Pi; E \longrightarrow F; \Theta$$

E must be a side expression of the inference in question, here $(\wedge \rightarrow)$; otherwise the corresponding inference could not have been linked in D. One of the inference figures $(\wedge \rightarrow_L)$ and $(\wedge \rightarrow_R)$ must apply depending on which side expression E is. For concrete illustration, we suppose E is A^{μ}_{X}; then we construct D' as:

$$\dfrac{\overset{D_1'}{\Pi; A^{\mu}_{X} \longrightarrow F; \Theta}}{\Pi; A \wedge B^{\mu}_{X} \longrightarrow F; \Theta} \wedge \rightarrow_L$$

Next, we suppose D ends in $(\supset\rightarrow^S)$, as follows:

$$\dfrac{\overset{D_1}{\Pi; \longrightarrow A^{\mu}_{X}, \Delta; \Theta} \qquad \overset{D_2}{\Pi; \Gamma, A \supset B^{\mu}_{X}, B^{\mu}_{X} \longrightarrow \Delta; \Theta}}{\Pi; \Gamma, A \supset B^{\mu}_{X} \longrightarrow \Delta; \Theta} \supset\rightarrow^S$$

We begin by applying the induction hypothesis to the subderivation D_1. After stripping off any (restart), we obtain an SCLP derivation D_1 with end-sequent

$$\Pi; \longrightarrow C; \Theta$$

By the usual linking argument, the expression C must be identical to A^{μ}_{X}. We then apply the induction hypothesis also to the right subderivation. Again, after stripping off any (decide), we get an SCLP derivation D_2 with end-sequent

$$\Pi;D \longrightarrow E;\Theta$$

By the usual linking argument, D must in fact be identical to B^{μ}_{X}. Thus we obtain the needed D' by combining the two derivations by the SCLP $(\supset\rightarrow)$ rule:

$$\dfrac{\begin{array}{c}D_1'\\ \Pi;\longrightarrow A^{\mu}_{X};\Theta\end{array}\qquad\qquad \begin{array}{c}D_2'\\ \Pi;B^{\mu}_{X}\longrightarrow E;\Theta\end{array}}{\Pi;A\supset B^{\mu}_{X}\longrightarrow E;\Theta}\supset\rightarrow$$

Finally, for $(\vee\rightarrow^{B})$, we consider the representative case of D as schematized below:

$$\dfrac{\begin{array}{c}D_1\\ \Pi;\Gamma,A^{\mu}_{X}\longrightarrow \Delta;\Theta\end{array}\qquad\qquad \begin{array}{c}D_2\\ \Pi',B^{\mu}_{X};\longrightarrow;\Theta'\end{array}}{\Pi;\Gamma,A\vee B^{\mu}_{X}\longrightarrow \Delta;\Theta}\vee\rightarrow^{B}_{L}$$

We begin by applying the induction hypothesis to D_1, the subderivation in the current block; if necessary, we strip off any initial (decide) inference, obtaining D_1' with an end-sequent that by linking takes the form:

$$\Pi;A^{\mu}_{X}\longrightarrow E;\Theta$$

Next, we apply the induction hypothesis to the other subderivation. Since both local areas are empty in the input subderivation, they remain empty in the result subderivation: this gives D_2' with end-sequent:

$$\Pi',B^{\mu}_{X};\longrightarrow;\Theta'$$

The two subderivations can be recombined by the SCLP $(\vee\rightarrow_{L})$ inference to obtain the needed D':

$$\dfrac{\begin{array}{c}D_1'\\ \Pi;A^{\mu}_{X}\longrightarrow E;\Theta\end{array}\qquad\qquad \begin{array}{c}D_2'\\ \Pi',B^{\mu}_{X};\longrightarrow;\Theta'\end{array}}{\Pi;A\vee B^{\mu}_{X};\longrightarrow E;\Theta}\vee\rightarrow_{L}$$

■

The discussion of the previous subsections represents an outline of the proof of the following theorem.

Theorem 2 *Let Γ and Δ be multisets of tracked prefixed expression in which each formula is tracked by the empty set and prefixed by the empty prefix. There is a proof of $\Gamma\longrightarrow\Delta$ in SCL exactly when there is a proof of $\Gamma;\longrightarrow;\Delta$ in SCLP in which every block is cancelled.*

Proof. As observed already in Section 4.1, there is an SCL proof of $\Gamma\longrightarrow\Delta$ exactly when there is an SCLI proof of $\Gamma\longrightarrow\Delta$. By Theorem 1 of Section 4.2, there is an SCLI proof of $\Gamma\longrightarrow\Delta$ exactly when there is an *eager* SCLI proof of $\Gamma\longrightarrow\Delta$. By Lemma 4, there is an eager SCLI proof of $\Gamma\longrightarrow\Delta$ exactly when there is an eager articulated SCLI proof of $\Gamma;\longrightarrow;\Delta$. And by Lemma 6, there is an eager articulated SCLI proof of $\Gamma; \longrightarrow; \Delta$ exactly when there is an eager SCLS proof of $\Gamma; \longrightarrow; \Delta$.

Continuing through the argument, we know from its simple form that the sequent $\Gamma; \longrightarrow; \Delta$ is spanned and balanced. By Lemma 9 of Section 4.4.3, then, there is an eager SCLS proof of $\Gamma; \longrightarrow; \Delta$ exactly when there is a blockwise eager SCLB derivation of $\Gamma; \longrightarrow; \Delta$ in which every block is cancelled, linked, isolated, simple, balanced and spanned. And by Lemma 10 which we have just proved, there is a a blockwise eager SCLB derivation of $\Gamma; \longrightarrow; \Delta$ in which every block is cancelled, linked, isolated, simple, balanced and spanned exactly when there is an SCLP derivation of $\Gamma; \longrightarrow; \Delta$ in which every inference is linked. And if every inference is linked, every block is cancelled. ■

5 Sequent Calculus and Operational Behavior

The action the interpreter specified by SCLP can be summarized as follows. A distinguished formula on the right in sequents represents the current goal at any state in proof search; if possible, the interpreter first breaks this goal down into its components. In particular, as outlined in Sections 2 and 3, modular goals like [COFFEE]*get-coffee* and [TICKET]*get-ticket* are processed by considering transitions to fresh possible worlds where only the information from that module is available.

Once an atomic goal is derived, the program is consulted by applying (decide); the chosen clause is decomposed and matched against the current goal by applicable logical rules. In particular, at $(\vee \rightarrow)$, the second case analysis allows the current goal to be chosen flexibly by the (restart) rule. The (restart) rule is modular in that it limits the work that is reanalyzed to the scope of the ambiguity just introduced; this conforms to the description in Sections 2 and 3.

The operational rules presented in Section 3.4 go beyond the skeleton implicit in the sequent calculus SCLP by describing certain concrete data structures for managing search. We justify these data structures briefly here.

5.1 Cancellations and modularity

First, the operational rules of Section 3.4 are specialized to deliver only cancelled blocks.

The specialization takes effect by the clause of rule (15), where we require that the bookkeeping information κ_O obtained as a result from this block should have the value *true* for $c^?$. Observe that this variable $c^?$ is true exactly when there is a cancellation in the block. For the initial value for $c^?$ is *false*, as set by rule (13e). Each rule except rule (12) simply passes the values for $c^?$ along unchanged through its subproofs. And rule (12) sets $c^?$ exactly when the key premise for the block is accessed; we have seen from the proof of Lemma 10 that this ensures a cancellation for this premise within the segment. Of course, Lemma 10 ensures that an SCLP proof system that delivers only cancelled blocks is sound and complete.

Given that the operational rules deliver only cancelled blocks, we can enforce modularity, using the following observation. Using these rules, whenever an SCLP derivation D' contains a sequent of the form $\Pi; \rightarrow G^\nu; \Theta$ then G^ν will be the only right formula on which an axiom in that block is based. The observation we also appeal to in treating the $(\rightarrow>)$ case of Lemma 10—the variant of [Stone, 1999b, Lemma 2]—shows that if a left formula P^μ from Π is the ancestor of a principal formula of an axiom in such a block, μ must equal a prefix of ν. Now we have ensured that each time a $(\vee \rightarrow)$ rule applies, the key *disjunct* P^μ has such a cancellation. Thus, in search, we can restrict the subsequent (restart) rule to goals G^ν with μ a prefix of ν. This accounts for the added constraint of this form in rule (15). ■

5.2 Pruning $(\vee \to_R)$ search paths

The second feature reflected in the operational rules of Section 3.4 but not in SCLP is the pruning of search paths for $(\vee \to_R)$ inferences motivated by informal argument in Section 3.3. We can now formalize the argument and show that the operational rules of Section 3.4 implement this argument.

Definition 25 (free restart goals) *Let D be a cancelled SCLP derivation, and let B be a block in D other than the root. By cancellation, B includes some axioms based in the side expression K of the $(\vee \to)$ inference at the root boundary of B. An expression F is* free *in B if every sequent in B at which an inference applies to an occurrence of K also contains a global goal occurrence of F.*

Now suppose D is a cancelled SCLP derivation in which there is a $(\vee \to_R)$ inference thus:

$$\dfrac{\begin{array}{c}D_1\\ \Pi; B^{\mu}_{X} \to G; \Theta\end{array} \qquad \dfrac{\begin{array}{c}D_2\\ \Pi, A^{\mu}_{X}; \to H; \Theta\end{array}}{\Pi, A^{\mu}_{X}; \to; \Theta}\,\text{restart}}{\Pi; A \vee B^{\mu}_{X} \to G; \Theta}\,\vee \to_R$$

We will recreate the construction of (10) under certain conditions. To describe the construction, let B be the block of D in which this $(\vee \to_R)$ inference occurs; let D' be smallest subderivation of D containing B. We situate this $(\vee \to_R)$ inference, L, within D' as in:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}D_1\\ \Pi; B^{\mu}_{X} \to G; \Theta\end{array} \qquad \dfrac{\begin{array}{c}[D_A]\\ D_U\\ \Pi, A^{\mu}_{X}; \to H; \Theta\end{array}}{\Pi, A^{\mu}_{X}; \to; \Theta}\,\text{restart}}{\Pi; A \vee B^{\mu}_{X} \to G; \Theta}\,L}{D_\vee}}{D_G}}{D_0}$$

Namely, we define $D_\vee$ as the maximal subderivation of D' *from* the end-sequent of L that contains no right inferences below L. We define D_G as the maximal subderivation of D' *from* the end-sequent of $D_\vee$ all of whose sequents have an occurrence of H on which the restart occurrence of H is based. We let D_0 be the subderivation of D' *from* the end-sequent of D_G. Finally, we use the notation $[D_A]$ to abstract each of the subderivations of D_2 which begin with a (decide) inference whose principal expression is an occurrence of A^{μ}_{X} based in inference L. We use D_U to name the remaining subtree of D_2.

Since we have identified D_U so that no inference applies to an occurrences of A^{μ}_{X} based in L in D_U, we can construct a subtree like D_U except omitting all occurrences of A^{μ}_{X} based in L. Call this D_V. Of course, this changes the form of all open leaves in D_V. Consider any such open leaf

$$\Pi'; \longrightarrow H'; \Theta'$$

We construct a derivation of this sequent inductively from $D_\vee$. With the exception of $\vee \to$ inferences and the open leaf of $D_\vee$ (where L occurs in D), this derivation is constructed by taking subderivations inductively and recombining inductively obtained derivations by a corresponding inference. At $(\vee \rightarrow)$ inferences, we may have to adapt the (restart) subderivation, if the restart is to a goal not in Θ'. We do that by restarting to H, copying the necessary reasoning from D_G, and then supplying the original subderivation (minus its restart). At the site of L, we use a derivation of the form schematized below—supplying the original reasoning for B, and, for A, copying the necessary reasoning from D_G, and—after an appropriate (decide) inference—supplying the reasoning for A from D:

$$\dfrac{\begin{array}{c} D_B \\ \Pi';B^{\mu}_{X} \longrightarrow H';\Theta' \end{array} \qquad \begin{array}{c} \dfrac{\begin{array}{c} D_A \\ \Pi';A^{\mu}_{X} \longrightarrow G \end{array}}{D_G} \\ \Pi',A^{\mu}_{X} \longrightarrow H;\Theta' \end{array}}{\Pi';A \vee B \longrightarrow H'}$$

Now, if we replace all the open leaves in D_V by derivations so constructed, and in turn substitute the result at the open leaf of D_0, we obtain a derivation of the same end-sequent as D_0, in which the occurrence L of $(\vee \rightarrow_R)$ has been turned into an occurrence of $(\vee \rightarrow_L)$.

Call the transformed proof D^*. Under what conditions is D^* cancelled? Clearly, if L occurs in the root block of D, D^* remains cancelled. Otherwise, there is some key premise K which must be cancelled in the block where L occurs. Suppose K is linked in D_0: then H is not free and moreover D^* remains cancelled because these D_0 inferences are not moved into a different block in the construction of D^*. Suppose K is linked in D_2: then these inferences *are* moved in the construction of D^*, *into* the block where K must be cancelled. So D^* remains cancelled. Finally, suppose K is linked in $D_\vee$ (outside D_1). These inferences also remain in the same block in the construction of D^*: D^* remains cancelled.

Thus, it is complete to use $(\vee \rightarrow_R)$ only in a block other than the root with a key premise K, where the restart goal of the new block is free, where K is suppressible in the new block—corresponding to inferences D_2—and where K is suppressible throughout the proofs of subgoals introduced in reaching the disjunction—corresponding to inferences $D_\vee$. We can now immediately account for the manipulation of $d^?$ to identify any $D_\vee$ subgoals and to enforce the necessary suppression there in rule (13c); we can immediately account for the testing of suppression at rule (12). Moreover, assuming E correctly accumulates the free restart goals in a block, we can use the condition on κ_O setting F in rule (15), the propagation of F throughout the block, and the constraint $G^\mu \in F$ for delayed restarts, to show that the use of $(\vee \rightarrow_R)$ is limited to new blocks with free restarts.

There are two cases for the incremental propagation of values of E throughout the proof. In the case that $c^?$ is *false*, E is assigned all the possible restart goals that have been introduced. This is because with $c^?$ false, E is initialized to the initial restart goals G in the block and augmented whenever a new restart goal is introduced by rules (11d), (11e) or (11f). Thus, in the case of the first cancellation, in rule (12), E is set to those goals that are present at every sequent where an inference applies to K. Thereafter, E is updated only by the rule (12), so that E continues to hold all those goals that are present at every sequent where an inference applies to K. It follows straightforwardly that at the end of a block, which must contain some cancellation, E contains exactly the free restart goals of the block. ∎

5.3 Constraints and unification

The only feature of the operational rules that remains to be accounted for is the use of *logic variables* and *constraints* in place of ordinary instantiation of terms. The regime adopted in Section 3.4 represents one standard formalization of the inevitable need to *lift* logical calculi allow instantiations to be solved for rather than searched over.

We can justify our particular constraints and constraint propagation technique as follows. We start by following [Voronkov, 1996] in describing a range of parameterized atomic constraints on substitutions. For each inference, an appropriate constraint can be selected and appropriate parameters provided to it, so as to ensure that when a substitution that satisfies the constraint is applied to a proof, to replace logic variables with ordinary terms, the side condition on that inference is met. These provide the vocabulary of constraints used in rules (11)–(15). Proofs themselves are now conceived in two steps. In the first step, we build a lifted tree of inferences paired with an appropriate composite constraint. The constraints are accumulated from separate subtrees and composed together to provide the constraints for a larger tree. In the second step, we find a substitution that solves the constraints; we can obtain an ordinary proof from the lifted proof using this substitution. For our modal system, in particular, we can naturally adapt the construction of [Stone, 1999a, Section 4], which discusses the modal language and moreover shows how to formulate the constraints as a conjunction of atomic constraints in the natural case where each inference in the proof introduces distinct logic variables.

The accumulation of these constraints incrementally follows [Lincoln and Shankar, 1994]. Having argued that the constraints associated with a lifted deduction amount to a conjunction—in effect, a multiset—of atomic constraints, it amounts to the same thing whether the constraints are accumulated statically, by union, as in Voronkov's presentation, or dynamically, by accumulating them throughout a proof, as in the Lincoln and Shankar presentation. The latter system offers the advantage of representing the constraints in place in a partial proof more directly, allowing the more natural statement of the unifiability constraint of rule (14). ■

6 A Worked Example

We illustrated some potential applications for DIALUP in section 2. Now that we have justified the DIALUP interpreter in detail, we return to the first application, discourse planning, to consider the action of that interpreter more precisely. We will consider the same discourse as before:

(16) I have account 42. What is my balance?

However, we will adopt an account of conversational action that is more sophisticated than the earlier one in three important respects.

First, we now explicitly consider contributions to dialogue that span multiple utterances; we will be interested in how *both* sentences of (16) combine together to achieve the patron *P*'s communicative goals.

Second, we will represent the communicative goals themselves more finely and with less indirection. Earlier we saw how the patron could conclude that, assuming certain information was to be provided, the teller would be in possession of a specific piece of information that the patron needed. In fact, what the patron wants is something more: the patron wants the teller to *give* that information. Otherwise, we cannot explain why the patron actually goes on to ask the question.

Third, we will represent the communicative actions more faithfully. We represent utterances explicitly. Moreover, in describing how making an utterance adds to the conversational record, we take into account both the knowledge that a speaker must have to contribute the utterance to the discourse honestly, and the information the conversational record must already provide to satisfy any presuppositions that the utterance carries.

Despite this richer representation, the outline—and the moral—of this example follows the earlier example of Section 2.1. We represent and refer to the states of knowledge of the participants in the conversation using modal operators. We construct a program that describes both the initial state of the conversation and the available communicative actions; *existential assertions* figure will prominently in both descriptions. Then, we give DIALUP a query that assesses the *ability* of the speaker to achieve a communicative goal with some sequence of utterances. This query uses *nested implications* to describe successive updates to the conversational record.

It is best to begin with the formalization of ability; it figures not only in the assessment of the effect of multiple utterances but also in the formulation of the communicative goals themselves. Consider how an agent A makes a decision of a single best communicative action to take next. A must choose a *single* specific utterance u on the basis of the known effects that u will have in this context. To assess what effects are known to A, we can consider A's information only, assume that A has uttered u normally and sincerely—*uttered*(A,u)—and query whether this adds some desired fact G to the conversational record. This means we pose the query in (17).

(17) $\exists u.[\textsc{a}](\mathit{uttered}(A,u) \supset [\textsc{cr}]G)$

This query is, not surprisingly, a variant of the query (4) used to assess the effects of communicative action in Section 2.1. (A simplification: we have used a hypothesis $\mathit{uttered}(A,u)$ to sidestep the temporal reasoning required in general for planning. As a contingent assertion, any use of $\mathit{uttered}(A,u)$ must be made in the state of information available to A's deliberation at this step.)

Our first use of this query will be to characterize the communicative goal that the patron P means (16) to accomplish. P wants the teller to be able to provide an answer. Let us use the formula $\mathit{has\text{-}answer}(P)$ to represent the condition that P has needed information. We can now use (17) to characterize situations where the teller S *can* get this condition on the conversational record:

(18) $\exists u.[\textsc{t}](\mathit{uttered}(T,u) \supset [\textsc{cr}]\mathit{has\text{-}answer}(P))$

Be aware of two assumptions that (18) encodes. First, (18) implicitly appeals to our expectations about cooperative conversation. In a cooperative conversation, P can expect that T will give a response to the question simply because T recognizes T *can* give a response to the question. Second, (18) exploits the formula *has-answer* in order to avoid more structured reasoning about information needs. P and T's exchange depends on P signaling that P wants to know the balance; otherwise T cannot know to provide it. Hence in (18), the formula $[\textsc{cr}]\mathit{has\text{-}answer}(P)$ indicates that the conversational record contains not only P's information, but also P's information need itself.

We now return to the problem P faces of sequencing together some number of utterances which together will establish the result expressed in (18). We imagine that P adopts an iterative deepening planning strategy, considering first one-sentence contributions to conversation, then two-sentence contributions, and so forth. In each case, P attempts to construct the plan by posing a corresponding query. Following analyses of ability such as [Davis, 1994, Stone, 1998a], at each step in carrying out a plan, the agent must only select a single concrete next action for its known effects. The effect the agent must verify at that step is that the agent can continue making appropriate choices as needed in carrying out the plan until a goal is reached.

In this context, a two-sentence contribution will suffice. For such a plan, the agent makes a single choice now, considers the consequences, and anticipates a single further choice. The corresponding formula is (19).

(19) $\exists u.[\text{A}](\mathit{uttered}(A,u) \supset \exists u'.[\text{A}](\mathit{uttered}(A,u') \supset [\text{CR}]G))$

(Overall, then, at *each step* in the plan, an additional level of nested knowledge is introduced that includes the additional information that is available after that action is taken; the new information may then be used to select what the subsequent action should be.)

We can combine (19) with communicative goal represented in (18) to describe what the patron *P* must establish to justify the contribution in (16):

(20)
$$\begin{aligned}&\exists u.[\text{P}](\mathit{uttered}(P,u) \supset\\ &\quad\exists u'.[\text{P}](\mathit{uttered}(P,u') \supset\\ &\qquad[\text{CR}]\exists b \exists u''.[\text{T}](\mathit{uttered}(T,u'') \supset [\text{CR}]\mathit{has\text{-}answer}(P))))\end{aligned}$$

That is: *P* selects utterances u and u' in sequence because once they are added to the conversational record, the conversational record will also guarantee that the teller can—and, by reasoning cooperatively from *P*'s wants, will—provide a reply by which *P* learns the balance value. What we would then like to do is prove this query in DIALUP. This would not only check that *P*'s contribution to discourse is sensible, it could use unification to arrive at that contribution. By assuming the occurrences of events rather than unifying, we could get an abductive discourse planner (cf. [Thomason and Hobbs, 1997]).

What do we need to complete this proof? Obviously we need to represent the update to the conversation made by utterances. We now propose such a representation; it adopts some idealizations of cooperative conversation, in order to describe utterances as strongly and precisely as possible. First, we suppose that speakers only consider formulating utterances that they know to be true. Second, we suppose that utterances carry presuppositions: for an utterance to be interpreted appropriately, the common ground must provide a justification for these presuppositions. Provided that an utterance is uttered normally, truly, and with its presuppositions met in this way—and provided only this—the content of the utterance becomes part of the conversational record. Of course, this knowledge of how to use and interpret utterances must itself be common knowledge among a speech community.

(21) formalizes this picture; it assumes an utterance *u* with assertion *a* and presupposition *p*, to be uttered by agent *A*.

(21) $[\text{CR}]([\text{A}]a \wedge [\text{CR}]p \wedge \mathit{uttered}(A,u) \supset [\text{CR}]a)$

Note that we can easily accommodate the linguistically important *anaphoric* approach to presupposition by treating the formulas p and a as open formulas [van der Sandt, 1992, Stone and Webber, 1998]. The free variables of p must be instantiated by finding suitable values (or discourse referents) from the context; that assertion a is then made about those values. The contribution of the presupposition to interpretation on this account is in specifying how to retrieve from the context the referents we want to talk about; we will see examples of this below.

$$\begin{tikzcd}
 & \text{BANK} \arrow[rr] & & \text{CR} \arrow[dr] \arrow[ddr] & \\
 & & \text{IP} \arrow[rr] & & \text{P} \\
\text{TELLER} \arrow[rr] & & \text{IT} \arrow[rr] & & \text{T}
\end{tikzcd}$$

Figure 4: Modalities and inclusion relationships for the example.

From the point of view of logic programming search, this rule is somewhat problematic: if $[\text{CR}]a$ provides a way to establish $[\text{A}]a$, this rule may be applied recursively and so lead to an infinite regress in proof search. Intuitively, it is clear that no agent should search for ways of convincing itself of something by telling it to itself. We can use this insight for this example to reformulate (21) to refer to some distinguished subset of the agent's information—information that they bring to the conversation initially and would consider communicating. (An alternative, which would take us too far afield here, is to incorporate a more sophisticated account of time.) Notationally, we distinguish $[\text{A}]$ (the agent's increasing knowledge) from $[\text{IA}]$ (the agent's initial stock of information to contribute). With this new expressive power, we will refine the template of (21) as (22):

(22) $[\text{CR}]([\text{IA}]a \wedge [\text{CR}]p \wedge \mathit{uttered}(A,u) \supset [\text{CR}]a)$

Our example now illustrates the specification of modular problem-solving described in Section 2.2 as well as the specification of agents motivated in Section 2.1! Figure 4 diagrams all the modalities that we will use in this example.

Let us apply this formalization to our discourse starting with the initial utterance. It has the presupposition that account 42 is mutually identifiable: that there is an a satisfying $\varphi_0(a)$ as defined below:

(23) $\varphi_0(a) \equiv (\mathit{account}(a) \wedge \mathit{number}(a,42) \wedge \mathit{identifiable}(a))$

It carries the assertion that a belongs to the patron P. Thus, we can describe the preconditions and effects of the first utterance using the definition:

(24) $[\text{CR}]\forall a \exists u [\text{CR}]([\text{IP}]\mathit{belongs}(a,P) \wedge [\text{CR}]\varphi_0(a) \wedge \mathit{uttered}(P,u) \supset [\text{CR}]\mathit{belongs}(a,P))$

(In Prolog, we would use an explicit structured term rather than an existential quantifier to specify the utterance u. In our case Herbrand terms are the only compound terms the logical development has described; the double nesting of modalities is therefore required to account for increasing domains.)

Once the first utterance succeeds, the second requires that the balance b in P's account a be mutually identifiable, formalized as $\varphi_1(a,b)$:

(25) $\varphi_1(a,b) \equiv (\mathit{account}(a) \wedge \mathit{belongs}(a,P) \wedge \mathit{balance}(a,b) \wedge \mathit{identifiable}(b))$

We treat this utterance as contributing the information that P's information needs are to know what the value of b is; accordingly, we can describe the preconditions and effects of the second utterance in the following rules:

(26) $$[\textsc{cr}]\forall a \forall b \exists u' [\textsc{cr}]([\textsc{ip}](\exists v[\textsc{p}]value(b,v) \supset has\text{-}answer(P)) \wedge \\ [\textsc{cr}]\varphi_1(a,b) \wedge uttered(P,u') \supset \\ [\textsc{cr}](\exists v[\textsc{p}]value(b,v) \supset has\text{-}answer(P)))$$

Finally, we need a specification of the hypothetical third utterance—the teller's reply, which might go along the lines: your balance is $600. We can assign this the same presupposition as (25), and regard it as informing what the balance of the account is. That leads to the rule:

(27) $$[\textsc{cr}]\forall a \forall b \forall v \exists u'' [\textsc{cr}]([\textsc{it}]value(b,v) \wedge [\textsc{cr}]\varphi_1(a,b) \wedge uttered(T,u'') \supset \\ [\textsc{cr}]value(b,v))$$

As it happens, we cannot simply prove (20) from (24), (26) and (27). To establish this query, we need first of all some general facts about banking. In fact, the whole point of formulating the query is to allow us to draw on a variety of shared knowledge about banks and banking in streamlining the discourse. An important part of what makes these facts general is that they characterize the information available to different participants abstractly, by making essential use of existential assertions.

To facilitate reuse of these facts about banking, we can specify them using a special modality $[\textsc{bank}]$ with $[\textsc{bank}]p \supset [\textsc{cr}]p$; we can add another modality $[\textsc{teller}]$ for what any teller knows. Since T is the teller now, this is subject to $[\textsc{teller}]p \supset [\textsc{it}]p$. People familiar with banks know that accounts are named by codes like 42:

(28) $$[\textsc{bank}]\ \forall c.(code(c) \supset \\ \exists a\ [\textsc{bank}]\ (account(a) \wedge number(a,c) \wedge identifiable(a))) \\ [\textsc{bank}]\ code(42)$$

They also know that there is a balance for any account, that it is identifiable if the account is, that the teller knows its value:

(29) $$[\textsc{bank}]\ \forall a\ \exists b\ [\textsc{bank}]\ (account(a) \supset balance(a,b)) \\ [\textsc{bank}]\ \forall ab\ (balance(a,b) \wedge identifiable(a) \supset identifiable(b)) \\ [\textsc{bank}]\ \forall ab(balance(a,b) \supset \exists v.\ [\textsc{teller}]value(b,v))$$

We need not only general facts about banking, but we need the patron P to bring to the discourse context the information that the two utterances in the discourse carry.

(30) $$[\textsc{ip}]\forall a(account(a) \wedge number(a,42) \supset belongs(a,P)) \\ [\textsc{ip}]\forall a \forall b(account(a) \wedge number(a,42) \wedge balance(a,b) \wedge \\ \exists c[\textsc{p}]value(b,c) \supset has\text{-}answer(P))$$

By adding all these facts, it becomes possible to prove the query. In fact, DIALUP reports for this example that there is a single proof—the expected one where the three utterances described in (24), (26) and (27) are used as witnesses for the three corresponding existential quantifiers in (20).

Exhausting the search space for this example involves what may seem a surprising amount of work. DIALUP attempts about one hundred possible applications of the (decide) rule for backward chaining in the course of tabulating all ways to prove the query from the specification. In part this reflects the size of the proof; in a logic programming derivation, for each communicative goal that the utterance helps establish, the interpreter must prove separately that the speaker knows the assertion of the utterance and that the conversational record supports the presupposition of the utterance. But the number also includes a quota of applications of program clauses that are discarded immediately because of unification mismatches between the first-order terms or modal prefix associated with the program clause and the current goal. It is clear therefore that it will be impossible to spell out each step DIALUP takes in exploring the proof search space for this query.

We can, however, summarize the notable features of this exploration. DIALUP begins by unwrapping the query, introducing assumptions for the three communicative actions that we are solving for and considering the goal *has-answer*(P). Although two clauses are headed by the *has-answer* predicate corresponding to the resulting goal, the modal prefix of that goal (requiring shared information at a certain stage) is only compatible with the communication clause (26). Thus, DIALUP immediately turns to considering how *P*'s want could be communicated. The condition that *P* know that want to start is established straightforwardly. (But observe again that only one of the *has-answer* clauses applies; as we now require initial knowledge on the part of *P*, the communicative clause is discarded.) Next we turn to establishing the presupposition as shared. Notably, for the *belongs* presupposition, we must match a communication clause (again *P*'s knowledge is not relevant to the shared goal); DIALUP thus recognizes the dependence of *P*'s question on the declarative sentence that precedes it. DIALUP establishes that *P* knows the assertion of that first utterance, that account 42 belongs to *P*, by the appropriate clause; the presupposition of the first utterance likewise follows. Now there is an ambiguity: should we treat utterance one *uttered* by the first or the second assumption we have made? DIALUP works through both alternatives—this redundancy is the attraction of an abductive approach to discourse planning, in which a single unambiguous assumption of utterance can be carried through. Having established the effects of utterance one and then the presupposition of utterance two, we go to unify utterance two with a *uttered* assumption. At this stage, because of the dependence of utterance two on utterance one and the modal prefixes in which that dependence is encoded, the only possibility is that where utterance one is linked with the first assumption and utterance two is linked with the second. DIALUP backtracks until this possibility is recognized.

The communication clause associated with *P*'s question requires a further subgoal to be established before we can establish *has-answer*: we must show that *P* knows the balance. The *value* fact in question can be proved only assuming some communication. DIALUP use the teller's initial knowledge to establish the possibility of making the assertion; it uses the patron's first, background utterance to establish the presupposition; the utterance is identified with the teller's assumed utterance. With this the proof, and indeed the proof search, is concluded. ■

7 Conclusion

To execute modal specifications requires leveraging both the flexibility of efficient classical theorem-proving and the distinctive modularity of modal logic. This is a significant problem because the two are at odds. On the one hand, flexible search strategies impose no constraints on the relationships among inference and, by thus ignoring modularity, leave open hopelessly wild possibilities for search. On the other hand, brute-force modular systems may place such strong constraints on the order in which search must proceed that it becomes impossible to guide that search in a predictable, goal-directed way. In this paper, we have explored one strategy for balancing the flexibility of classical goal-directed search with the modularity of modal logic. This strategy culminates in the development of a modular logic programming sequent calculus SCLP and and associated interpreter, DIALUP.

It may be delicate to construct these systems, but the SCLP presentation, and the associated DIALUP operational rules that we justified here, work according to a simple intuition. A modular goal can be thought of as an assignment of a problem to a new independent agent that has access to precisely the information in the corresponding modular statements. This intuition provides a powerful handle on the close connection between modularity and ambiguity, proof size and search control in deductions. It supports applications that can be mocked up simply, as in Section 2, but that can also be fleshed out substantially and usefully, as suggested in Section 6.

References

[Andreoli, 1992] Andreoli, J.-M. (1992). Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347.

[Auffray and Enjalbert, 1992] Auffray, Y. and Enjalbert, P. (1992). Modal theorem proving: an equational viewpoint. *Journal of Logic and Computation*, 2(3):247–295.

[Baldoni et al., 1993] Baldoni, M., Giordano, L., and Martelli, A. (1993). A multimodal logic to define modules in logic programming. In *ILPS*, pages 473–487.

[Baldoni et al., 1996] Baldoni, M., Giordano, L., and Martelli, A. (1996). A framework for modal logic programming. In Maher, M., editor, *JICSLP 96*, pages 52–66. MIT Press.

[Basin et al., 1998] Basin, D., Matthews, S., and Viganò, L. (1998). Labelled modal logics: Quantifiers. *Journal of Logic, Language and Information*, 7(3):237–263.

[Cassell et al., 1994] Cassell, J., Stone, M., Douville, B., Prevost, S., Achorn, B., Steedman, M., Badler, N., and Pelachaud, C. (1994). Modeling the interaction between speech and gesture. In *Proceedings of the Cognitive Science Society*.

[Clark and Marshall, 1981] Clark, H. H. and Marshall, C. R. (1981). Definite reference and mutual knowledge. In Joshi, A. K., Webber, B. L., and Sag, I., editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, Cambridge.

[Davis, 1994] Davis, E. (1994). Knowledge preconditions for plans. *Journal of Logic and Computation*, 4(5):721–766.

[Debart et al., 1992] Debart, F., Enjalbert, P., and Lescot, M. (1992). Multimodal logic programming using equational and order-sorted logic. *Theoretical Computer Science*, 105:141–166.

[Fariñas del Cerro, 1986] Fariñas del Cerro, L. (1986). MOLOG: A system that extends PROLOG with modal logic. *New Generation Computing*, 4:35–50.

[Fitting, 1972] Fitting, M. (1972). Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2).

[Fitting, 1983] Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics*, volume 169 of *Synthese Library*. D. Reidel, Dordrecht.

[Frisch and Scherl, 1991] Frisch, A. M. and Scherl, R. B. (1991). A general framework for modal deduction. In *Proceedings of KR*, pages 196–207. Morgan Kaufmann.

[Giordano and Martelli, 1994] Giordano, L. and Martelli, A. (1994). Structuring logic programs: A modal approach. *Journal of Logic Programming*, 21:59–94.

[Girard, 1993] Girard, J.-Y. (1993). On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217.

[Green and Carberry, 1994] Green, N. and Carberry, S. (1994). A hybrid reasoning model for indirect answers. In *Proceedings of ACL*, pages 58–65.

[Hintikka, 1971] Hintikka, J. (1971). Semantics for propositional attitudes. In Linsky, editor, *Reference and Modality*, pages 145–167. Oxford.

[Hodas and Miller, 1994] Hodas, J. S. and Miller, D. (1994). Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365.

[Houghton, 1986] Houghton, G. (1986). *The Production of Language in Dialogue: A Computational Model*. PhD thesis, University of Sussex.

[Kanger, 1957] Kanger, S. (1957). *Provability in Logic*, volume 1 of *Stockholm Studies in Philosophy*. Almqvist and Wiksell, Stockholm.

[Kleene, 1951] Kleene, S. C. (1951). Permutation of inferences in Gentzen's calculi LK and LJ. In *Two papers on the predicate calculus*, pages 1–26. American Mathematical Society, Providence, RI.

[Kripke, 1963] Kripke, S. A. (1963). Semantical analysis of modal logic. I. Normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96.

[Lincoln and Shankar, 1994] Lincoln, P. D. and Shankar, N. (1994). Proof search in first-order linear logic and other cut-free sequent calculi. In *LICS*, pages 282–291.

[Loveland, 1991] Loveland, D. W. (1991). Near-horn Prolog and beyond. *Journal of Automated Reasoning*, 7:1–26.

[Miller, 1989] Miller, D. (1989). A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1–2):79–108.

[Miller, 1994] Miller, D. (1994). A multiple-conclusion meta-logic. In Abramsky, S., editor, *Proceedings of the International Symposium on Logics in Computer Science*, pages 272–281.

[Miller et al., 1991] Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. (1991). Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157.

[Moore and Paris, 1993] Moore, J. D. and Paris, C. L. (1993). Planning text for advisory dialogues: capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695.

[Nadathur, 1998] Nadathur, G. (1998). Uniform provability in classical logic. *Journal of Logic and Computation*, 8(2):209–229.

[Nadathur and Loveland, 1995] Nadathur, G. and Loveland, D. W. (1995). Uniform proofs and disjunctive logic programming. In *LICS*, pages 148–155.

[Nonnengart, 1993] Nonnengart, A. (1993). First-order modal logic theorem proving and functional simulation. In *Proceedings of IJCAI*, pages 80–87.

[Ohlbach, 1991] Ohlbach, H. J. (1991). Semantics-based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746.

[Onishi and Matsumoto, 1957] Onishi, M. and Matsumoto, K. (1957). Gentzen method in modal calculi I. *Osaka Mathematical Journal*, 9:113–130.

[Power, 1977] Power, R. (1977). The organisation of purposeful dialogues. *Linguistics*, 17:107–152.

[Prior, 1967] Prior, A. N. (1967). *Past, Present and Future*. Clarendon Press, Oxford.

[Schild, 1991] Schild, K. (1991). A correspondence theory for terminological logics: preliminary report. In *IJCAI*, pages 466–471.

[Stone, 1998a] Stone, M. (1998a). Abductive planning with sensing. In *AAAI*, pages 631–636, Madison, WI.

[Stone, 1998b] Stone, M. (1998b). *Modality in Dialogue: Planning, Pragmatics and Computation*. PhD thesis, University of Pennsylvania.

[Stone, 1999a] Stone, M. (1999a). First-order multi-modal deduction. Technical Report RUCCS Report 55, Rutgers University.

[Stone, 1999b] Stone, M. (1999b). Representing scope in intuitionistic deductions. *Theoretical Computer Science*, 211(1–2):129–188.

[Stone, 2000] Stone, M. (2000). Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246.

[Stone and Webber, 1998] Stone, M. and Webber, B. (1998). Textual economy through close coupling of syntax and semantics. In *Proceedings of INLG*, pages 178–187.

[Thomason, 1990] Thomason, R. H. (1990). Accommodation, meaning and implicature. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*, pages 325–363. MIT Press, Cambridge, MA.

[Thomason and Hobbs, 1997] Thomason, R. H. and Hobbs, J. R. (1997). Interrelating interpretation and generation in an abductive framework. In *AAAI Fall Symposium on Communicative Action*.

[van der Sandt, 1992] van der Sandt, R. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics*, 9(2):333–377.

[Voronkov, 1996] Voronkov, A. (1996). Proof-search in intuitionistic logic based on constraint satisfaction. In *TABLEAUX 96*, volume 1071 of *LNAI*, pages 312–329, Berlin. Springer.

[Wallen, 1990] Wallen, L. A. (1990). *Automated Proof Search in Non-Classical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics*. MIT Press, Cambridge.