Intuitionistic Subformula Linking

Pablo Donato, under the supervision of
Benjamin Werner and Pierre-Yves Strub
Typical team – LIX

August 30, 2020

The general context

Proof assistants are software systems allowing for the precise synthesis and verification of mathematical proofs. They are based on the idea that mathematical knowledge can be represented syntactically and unambiguously inside *proof formalisms*. These have been developed since the beginning of the 20th century, in the well-established field of mathematical logic known as *proof theory*.

State-of-the-art proof assistants such as Coq or Isabelle are usually based on very expressive logics, so that virtually any mathematical development can be expressed in them. While this generality hinders the potential for automated theorem-proving, formal proofs still tend to be too detailed and verbose to be written by hand, much like assembly code. This demands for a higher-level, more interactive approach to computer-assisted proof authoring.

The problem is usually tackled through *tactic languages*, which offer to users a set of primitive text commands to manipulate the proof state, as well as basic combinators to build more complex tactics. For example, given a proof `H1` of A and a proof `H2` of $A \Rightarrow B$ as hypotheses, one can type the command `apply H2 in H1` to get a proof of B.

The research problem

While basic logical reasoning is considered universal, if not intuitive, each proof assistant currently has its own syntax to perform it. This induces an unnecessary cognitive burden for newcomers, especially those unfamiliar with textual interfaces, but also for seasoned practicioners, who spend a non-negligible amount of time performing mundane tasks such as naming, destructuring and application of hypotheses. This calls for friendlier and more ergonomic interfaces, which fully exploit the capabilities of modern devices.

A first attempt in this direction was made in the 90's by the team of G. Kahn at Inria, where they coined the "proof by pointing" paradigm [1]. The idea was to synthesize complex tactics from the simple act of *pointing* at parts of expressions, typically with a mouse cursor. More recently [2], K. Chaudhuri proposed a variation on this idea termed "subformula linking", where instead of selecting expressions in isolation, one can *link* two of them together to make them interact. In both cases, the expressions considered were logical formulas, and the associated actions chains of inferences in first-order logic.

When I started my internship, the Typical team already had a prototype of graphical interface implementing some of the ideas of proof by pointing, within a more modern and

flexible web-based environment. In particular, experiments were being carried to explore the possible logical behaviors of *drag-and-drop* actions.

Your contribution

My first contribution was to design and implement a tactic `link` that would incorporate all drag-and-drop behaviors on formulas in a unified way. What I found was basically an extension of the intuitionistic system of proof by pointing, but taking two *dual* and *unifiable* subformulas as input instead of a single formula.

Still, its behaviour did not correspond to the one we envisioned for some important use cases. The *deep inference* methodology described in [2] came naturally as a solution, leading to my second contribution: a deep inference system for subformula linking in intuitionistic propositional logic, on top of which I built a variant `dlink` of the previous tactic.

Arguments supporting its validity

While I stated properties of correctness and sketched some proofs for both tactics, it is mainly through experimentation with the prototype that I convinced myself of their correctness and usefulness. Given the exploratory and applied nature of this work, it seemed more appropriate to keep experimenting until a stable specification of the tactics was reached, and only then make fully detailed proofs to detect and rectify bugs in the specification.

Another property of interest is how much can be proved with those tactics alone, that is doing only drag-and-drops between formulas. While completeness is lost with `link`, I conjecture based on the results in [2] that it holds for `dlink`.

In both cases, rules were carefully designed to minimize the loss of provability in generated subgoals. That is, transforming the proof state with a drag-and-drop should not commit the user to choices that can lead to a dead-end in the proof. This is the main advantage of *linking* over *pointing*, which is technically justified by the *focusing* discipline. The deep inference setting of `dlink` gives even more possibilities in that respect.

Summary and future work

My contribution is two-fold. I first designed and implemented a "drag-and-drop" tactic `link` based on a new approach to subformula linking, which is closer to the proof by pointing paradigm both in its logical basis and behavior. I then designed and implemented a variant `dlink` of this tactic based on an adaptation of the deep inference system of [2] from linear logic to intuitionistic logic, which is the core logic of many proof assistants.

It remains to prove formally the correctness of `dlink`, as well as that of `link` in the first-order case. Extending `dlink` to first-order logic and proving it complete is another challenging but rewarding goal, that would support it as a more powerful alternative to `link`. As pointed out in both [1] and [2], it would also be interesting to consider generalisations to more expressive logics (higher-order, inductive, with equality...), which might be easier to carry in the more standard *sequent calculus* underlying `link`.

Last but not least, such novel methods of interaction should be confronted directly to users of various proof assistants with various levels of expertise. This will give essential insights into their usability and usefulness, as well as new directions to consider in their design.

Introduction

While the use of formal rules in logical inference can be given roots dating back to Aristotle, the study of *proofs* as mathematical objects of their own really starts at the end of the 19th century, most notably with Frege's *Begriffsschrift* published in 1879 [3]. Then, by giving proof theory a central role in his program for proving the consistency of mathematics *by finite means*, Hilbert made it into a separate mathematical field, with its own goals and methodology. Despite Gödel's second incompleteness theorem putting an end to this program, logicians kept investigating how mathematical reasoning could be captured and studied through *proof calculi*. One of them, G. Gentzen, is now known as the father of modern (structural) proof theory, thanks to the two revolutionary formalisms he invented for representing proofs: *natural deduction* and *sequent calculus*.

Gentzen-style calculi

Natural deduction follows closely the way mathematicians reason in practice, by providing for each logical connective *introduction* and *elimination* rules. For example, the rules in figure 2 show how to give meaning to the connective $\vee$ of disjunction, not in terms of truth tables, but rather by giving canonical means to *prove* (introduction) and *use* (elimination) a disjunctive formula. More precisely, the rules $\vee I_i$ say that given a proof of the disjunct A_i (with $i \in \{1, 2\}$), one can build a proof of the disjunction $A_1 \vee A_2$; and the rule $\vee E$ is simply a formalization of the usual "reasoning by case": given a proof of $A_1 \vee A_2$, and two hypothetical proofs of C which respectively take A_1 and A_2 as assumptions, one gets a proof of C where the assumptions A_i can be discharged.

In contrast, sequent calculus offers only one way to *use* a proof: the cut rule. It takes the following form:

$$\frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\ \mathsf{cut}$$

Intuitively, it reads as follows: given a proof of A under assumptions Γ, as well as a proof of C under assumptions Δ and A, one can *substitute* the assumption of A by its actual proof, giving a proof of C under assumptions Γ and Δ. We see the emphasis here on *hypothetical* proofs, which is embodied in the judgment form $\Gamma \vdash A$, with Γ a multiset of formulas acting as hypotheses, and A the conclusion. These judgment forms, the eponymous *sequents*, are at the heart of most proof assistants, where they encode almost completely the *state* of an ongoing proof.

Apart from the cut rule, all rules of the sequent calculus are dedicated to the *construction* of complex proofs from smaller proofs. Indeed, each logical connective has so-called *right* and *left* introduction rules. While the right introduction rules are morally exactly the introduction rules of natural deduction, left introduction rules are a sort of reversal of elimination rules: since we cannot use directly the formula from a premise (a behavior reserved to the cut rule), we can only introduce it in the conclusion of the rule, but this time as an *hypothesis* of the sequent (that might be used later in a cut). This is exemplified in the rules for disjunction presented in figure 3. Thus the *construction/usage* duality of natural deduction is refined in sequent calculus as the duality between introduction rules and the cut rule, which is made distinct from the *left/right*, *hypothesis/conclusion* duality of sequents.

Of course a proof is a finite object, and thus there must be some kind of *axiom* to terminate the *trees* of inferences built using the previous rules. In purely logical Gentzen-style calculi[1], there is only one such axiom, called alternatively the *axiom* or *identity* rule. It has the following form in sequent calculus:

$$\frac{}{A \vdash A}\ \mathsf{id}$$

In fact, the id rule can be seen as the dual of the cut rule, where instead of *using* a proof of A to *eliminate* an assumption of A, we *construct* a proof of A by *introducing* an assumption of A. In a way, the two rules serve the same function: justifying an occurrence of A by another, which is so trivial that it seems more computational than logical. The *Haupstatz*, or *fundamental theorem*, is a result proved by Gentzen which asserts the redundancy of the cut rule: all provable formulas can be proved without it. Furthermore, the proof of this result is indeed computational: it is an algorithm that given a proof with cuts, rewrites it into a proof without cuts.

Then arises a question: can the essence of a proof be reduced to its instances of the identity rule? We will develop a little bit more on this in the conclusion, and see that the answer depends on the logic under consideration. However, this idea is really the heart of the *subformula linking* methodology studied in this report, thus outlining the potential for a fruitful interplay between theoretical considerations on the nature of proofs, and practical investigations on the ergonomics of computer-assisted proof authoring.

The Actema prototype

Developed by the Typical team since 2019, the Actema prototype (as in *Activ*e *ma*thematics) is a proof assistant whose originality lies in its user interface: proofs are entirely built through graphical actions performed by the user. A tab of the system corresponds to a goal or subgoal that remains to be proved, and under each tab are shown a variety of items (figure 1): 1. the current goal, or *conclusion* to be proved, in red; 2. a number of known facts, or *hypotheses*, in blue; 3. as well as various mathematical objects, or *expressions*, in green.

Then, an *action* from the user can imply one item, e.g. through a click or double-click, or two items, typically by performing a *drag-and-drop* movement. Those actions correspond to steps in the construction of a proof, that are usually performed by *tactics* in more traditional systems. For example, a double-click on a conclusion of the form $A \wedge B$ will generate two subgoals, of respective conclusions A and B. This would be achieved by invoking the tactic `split` in Coq, or `CONJ_TAC` in HOL.

Most tactics for performing basic logical tasks can be seen as the application of a rule from Gentzen-style calculi. For example, `split` corresponds in sequent calculus to the right introduction rule $\wedge R$ for conjunction (figure 3), where the goal is the conclusion of the rule, and the simpler generated subgoals are the premises.

Given the variety of logical constructs, and the fact that each proof assistant has its own logical framework, it can quickly become tricky to learn all rules, and then remember which tactic to use for which rule in which assistant, along with its specific syntax. Often though, the intuition is pretty clear: we want to *prove* or *use* a given formula, whatever rule is associated with it.

[1] where by *purely logical*, we mean without any rules to reason within a particular mathematical theory such as arithmetic.

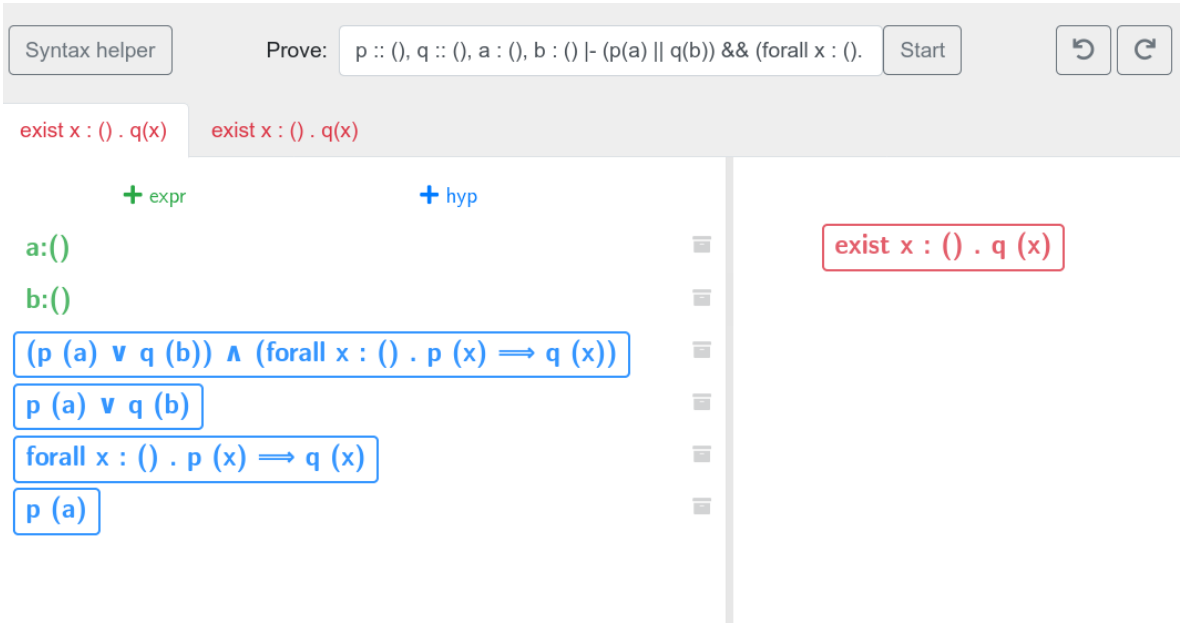

Figure 1: Interface of the Actema prototype

This is where the *pointing* mechanism comes in: it allows directly selecting the formula of interest in the current goal. And since a goal can be seen as a sequent, the system can straightforwardly determine which rule to apply, based on the *shape* and *position* of the formula. This is essentially what happens when performing a *double-click* on a formula A in Actema: it applies either the left introduction or the elimination rule associated with the main connective of A when A is an hypothesis, and the right introduction rule when A is the conclusion.

Now what about more elaborate tactics, like the `apply _ in _` tactic of Coq? This one takes two hypotheses of the form A and $A \Rightarrow B$ as inputs, and replaces A by B, which is simply an application of *modus ponens*. This is in fact a perfect candidate for a *drag-and-drop* action: just drag A and drop it on $A \Rightarrow B$, or the other way around! But as for double-click actions, is there a way to generalize this behavior to formulas of *any* shape/position? And can it have an elegant theoretical basis, like the rules of Gentzen-style calculi?

The answer to both questions is *yes*, and its content will be the main subject of the remaining pages. The key ingredient is to not only look at the surface shape of formulas in the form of their main connective, but to also introduce *depth* in the system, by allowing to select arbitrary *subformulas* of the current goal. While this idea is already present in the seminal *proof by pointing* paradigm of [1], it is taken to its fullest in the *subformula linking* methodology advocated in [2], which basically describes a generic behavior for drag-and-drop actions in first-order logic. Still, the setting in which it is presented, a *deep inference* system for *linear* logic, is quite far from what is available in current proof assistants, and in particular from the *intuitionistic* logic currently implemented in Actema. Our work can be seen as an exploration in how to embody the principles of subformula linking in a more conventional sequent-based, intuitionistic proof system.

Contributions

The paper is organized in two sections: in section 1, we recall the principles of the sequent calculus-based proof by pointing paradigm, which we then extend to define a generic proof procedure associated to drag-and-drop actions. We start by describing how to handle the linking of two propositional formulas, one at the top-level of the current goal, the other a subformula of arbitrary depth. We qualify this kind of linking of *simply deep*, since one of the two formulas is still at the top-level. We next lift that restriction by adding a light *focusing* discipline [4], giving rise to *doubly deep* linking. Finally, we extend the procedure to handle quantifiers, and thus full first-order logic. This involves a preliminary phase of *unification* and *dependency-checking* between the linked formulas and their quantified variables, much in the spirit of *expansion tree proofs* [5].

In section 2, we recall the principles of the *deep inference* approach to proof theory, which allows us to introduce an adaptation of the system for subformula linking of [2] to the intuitionistic setting. We discuss the correctness and completeness of this system, and use it to specify a procedure of *deep linking* that appears as a more ergonomic and versatile alternative to the previous one.

We conclude with some remarks on the implementation of the two procedures in the Actema prototype, as well as the variants and generalizations that could be explored as a continuation of this work. We also discuss existing occurrences of the idea of subformula linking in other proof-theoretical works, and how they relate to ours.

1 Subformula linking in sequent calculus

1.1 Proof by pointing

Proof by pointing (PbP hereafter) can be seen as an extension of the behavior of double-click actions presented in the introduction, where the selection can be *any* subformula A of a top-level formula A_0 occurring in the goal. Then instead of applying *one* rule based on the main connective of A_0, a full *sequence* of rules is applied, based on the connectives encountered along the path leading to A in the formula tree of A_0. A key point is to only use rules from sequent calculus: it allows to generate the sequence recursively, since all introduction rules have the property that at least one direct subformula of the principal formula appears in one of the premises. To illustrate, consider the tautological sequent $A \wedge (B \vee C) \vdash (A \wedge B) \vee C$, seen as the current goal. By selecting the occurrence of A in the conclusion, we instruct the PbP algorithm to first apply the $\vee R_1$ rule, since the selection occurs on the left of $\vee$, and then apply the $\wedge R$ rule. This gives the following *open* derivation, where the selected occurrence of A is underlined:

$$\dfrac{\dfrac{A \wedge (B \vee C) \vdash \underline{A} \quad A \wedge (B \vee C) \vdash B}{A \wedge (B \vee C) \vdash \underline{A} \wedge B}\ \wedge R}{A \wedge (B \vee C) \vdash (\underline{A} \wedge B) \vee C}\ \vee R_1$$

Thus, the user is left with the two subgoals $A \wedge (B \vee C) \vdash \underline{A}$ and $A \wedge (B \vee C) \vdash B$, with the focus on the first one since it contains the selected formula. Notice how the latter has been brought to the top-level: this is really what proof by pointing does from a user perspective, in addition to generating side goals.

Now, it would only be natural to select the other occurrence of A, so that we can close this branch of the proof with an instance of the id rule. In fact, the algorithm will automatically try to operate this closure, by matching the selection against the conclusion. Thus, a second click would generate the following derivation:

$$\dfrac{\dfrac{}{\underline{A}, B \vee C \vdash A}\,\mathsf{id}}{\underline{A} \wedge (B \vee C) \vdash A}\,\wedge L$$

Remark that we already knew beforehand that we wanted to close the goal with this occurrence of A, because of our implicit knowledge about the meaning of conjunction. The main advantage of PbP is to leave that knowledge implicit by hiding the underlying derivation, instead of having to explicitly call a tactic such as `destruct` in Coq. However, relying on the algorithm for closing goals with the id rule introduces undesirable situations, where bringing the selection to the top level induces wrong choices, even when the selected formula cannot be used to close the goal. Since every formula of interest is bound to be used in an instance of id, why not directly *link* the two occurrences?

1.2 Propositional linking

1.2.1 Simply deep interaction

Following on the previous remark, we will define here a very simple variant of PbP, which replaces the implicit, unguaranteed closure of goals by an *explicit* one. Thus it requires to add as input to the algorithm another occurrence of the selected formula, which to keep the "semantics" of PbP intact will be restricted to the top-level. Hence we call this variant *simply deep linking*.

To formalize and manipulate the notions of selection and subformula, we introduce the syntactic device of a *formula context*:

Definition 1 (Formula context)**.** *A* formula context*, or simply* context *when there is no ambiguity, is a formula with a* hole*, denoted by* $\Box$*. Contexts are defined by the following grammar:*

$$\begin{aligned}\chi, \xi, \zeta \;::=\; \Box \;&\mid \chi \wedge A \mid A \wedge \chi \\ &\mid \chi \vee A \mid A \vee \chi \\ &\mid \chi \Rightarrow A \mid A \Rightarrow \chi\end{aligned}$$

Definition 2 (Context filling)**.** *For any context* χ *and formula* A*, the* implicit filling *of* χ *with* A*, written* $\chi\{A\}$*, is defined as the formula obtained by substituting* A *to* $\Box$ *in* χ*.*

Dually, the grammar of formulas is extended with the explicit filling *operator* $\chi[A]$ *:*

$$A, B ::= \ldots \mid \chi[A]$$

This construction can be seen as an analogue to explicit substitution in λ*-calculus, in that it reifies the meta-operation of filling inside formulas.*

Finally, the composition $\chi\{\xi\}$ *of two contexts* χ *et* ξ *is the context obtained by substituting* ξ *to* $\Box$ *in* χ*.*

A subformula can now be seen as the pair of a formula A and its context χ, which can be direcly manipulated as the explicit filling $\chi[A]$. This will be used extensively to mark the linked subformulas in inference rules.

Notation 1 (Sequent membership)**.** *Given a sequent $\mathcal{S} = \Gamma \vdash C$ and a formula A, we write $A \in \mathcal{S}$ as a shorthand for $A \in \Gamma \cup \{C\}$.*

Definition 3 (Goal)**.** *A* goal $\mathfrak{S}$ *is a sequent where each hypothesis (resp. conclusion) has a unique negative (resp. positive)* identifier $i \in \mathbb{Z} \setminus \{0\}$*. Since intuitionistic sequents only have one conclusion, we can systematically identify it with $+1$, and omit it in the notation. Thus goals have the form:*

$$\mathfrak{S} = i_1 : A_1, \dots, i_n : A_n \vdash C$$

We note $\lfloor \Gamma \rfloor$ the underlying multiset of formulas of $\Gamma = i_1 : A_1, \dots, i_n : A_n$, and $\lfloor \mathfrak{S} \rfloor = \lfloor \Gamma \rfloor \vdash C$ the underlying sequent of $\mathfrak{S}$.

Definition 4 (Occurrence)**.** *An* occurrence $\mathfrak{o}^A$ *of a formula A is the data of A together with an identifier i and a context χ, written $\mathfrak{o}^A = i : \chi \,/\, A$. We say that this occurrence belongs to a goal $\mathfrak{S}$, written $\mathfrak{o}^A \in \mathfrak{S}$, if $i : \chi\{A\} \in \mathfrak{S}$.*

Definition 5 (Suboccurrence)**.** *$\mathfrak{o}^B = j : \xi \,/\, B$ is a* suboccurrence *of $\mathfrak{o}^A = i : \chi \,/\, A$, written $\mathfrak{o}^A \leq \mathfrak{o}^B$, if $i = j$ and there exists some ζ such that $\xi = \chi\{\zeta\}$ and $A = \zeta\{B\}$. We say that they are* incomparable *if neither $\mathfrak{o}^A \leq \mathfrak{o}^B$ nor $\mathfrak{o}^B \leq \mathfrak{o}^A$.*

Definition 6 (Linkage)**.** *A* linkage $\mathfrak{L}$ *is the data of a goal $\mathfrak{S}$ together with two linked subformula occurrences $\mathfrak{o}^A, \mathfrak{o}^B \in \mathfrak{S}$, written $\mathfrak{L} = \mathfrak{o}^A \overset{\mathfrak{S}}{=\!=} \mathfrak{o}^B$.*

Linkages describe the input data that is fed to the linking procedure. For now, we only consider *simply deep* linkages, which correspond to the linking of a top-level formula with any subformula:

Definition 7 (Simply deep linkage)**.** *A linkage $i : \xi \,/\, A \overset{\mathfrak{S}}{=\!=} j : \chi \,/\, B$ is said to be* simply deep *if either $\xi = \Box$ or $\chi = \Box$.*

Recall that the behavior we want to associate with a simply deep linkage $i : \Box \,/\, A \overset{\mathfrak{S}}{=\!=} j : \chi \,/\, B$ is not only to bring B to the top-level of $\mathfrak{S}$, but also to apply an instance of the identity rule between A and B. Thus, we need a mean to ensure *beforehand* that B will either be the conclusion if A is an hypothesis, or an hypothesis if A is the conclusion. It turns out that this information can be deduced from the shape of the context χ of B, thanks to the notion of *polarity*:

Definition 8 (Polarized contexts)**.** Positive contexts π *and* negative contexts η *are defined by the following grammar:*

$$\begin{aligned} \pi &::= \Box \mid \pi \wedge A \mid A \wedge \pi \mid \pi \vee A \mid A \vee \pi \mid \eta \Rightarrow A \mid A \Rightarrow \pi \\ \eta &::= \pi \Rightarrow A \end{aligned}$$

That is, all connectives preserve polarity, except for implication $\Rightarrow$ which inverts the polarity of its antecedant.

Definition 9 (Occurrence polarity)**.** *An occurrence $i : \chi \,/\, A$ is said to be either* positive *(+) or* negative *(−) according to the following table:*

	$i > 0$	$i < 0$
$\chi = \pi$	$+$	$-$
$\chi = \eta$	$-$	$+$

Then, we know that B will end up in hypothesis if its polarity is $-$, and in conclusion if it is $+$. This is because polarized contexts mimick the exchange of side of formulas performed by introduction rules, which accordingly only occurs for the antecedant of an implication (rules $\Rightarrow L_1$ and $\Rightarrow R_1$ of figure 4). Now we can completely characterize linkages that correspond to a well-behaved linking:

Definition 10 (Well-formed linkage)**.** *A linkage $i : \xi \,/\, A \overset{\mathfrak{S}}{=\!=} j : \chi \,/\, B$ is said to be* well-formed *if $i \neq j$, A and B have opposite polarities, and $A = B$.*

The well-formedness condition was initially motivated by a feature of Actema's interface, where starting a drag-and-drop would search and highlight all subformulas of the goal that lead to a correct linking. Typically, dragging the conclusion can help the user find hypotheses that could be used to prove it, and dragging a hypothesis will indicate instantly all places where it could be used.

In the rest of this section, we consider all linkages to be simply deep and well-formed. It remains to specify a deterministic procedure that actually performs the linking. For this purpose, we reuse exactly the rules with selection propagation of PbP (figure 2 in [1]), but reformulated with our more precise (although heavier) syntax of formula contexts. This gives the system ISL_1^0 presented in figure 4, which stands for 0^{th} order (i.e. propositional) Intuitionistic Subformula Linking with 1 deep selection.

The procedure can then be seen as the bottom-up search for a proof of the current goal in ISL_1^0, with the subformula selection encoded as the explicit filling $\chi\,[B]$. The search gets stuck when an instance of the $\mathsf{id}L$ or $\mathsf{id}R$ rule is reached, resulting in an open derivation that we call an *interaction trace*:

Definition 11 (Interaction trace)**.** *The* interaction traces $\mathsf{Tr}(\mathfrak{L})$ *of a linkage $\mathfrak{L} = i : \square \,/\, A \overset{\Gamma \vdash C}{=\!=\!=} j : \chi \,/\, B$ are all the ISL_1^0 derivations that can result from performing a bottom-up proof-search on the sequent $\mathcal{S}$, where*

$$\mathcal{S} = \begin{cases} \lfloor \Gamma \rfloor \vdash \chi\,[B] & \text{if } j > 0 \\ \lfloor \Gamma \setminus j : \chi\,\{B\} \rfloor\,, \chi\,[B] \vdash C & \text{otherwise} \end{cases}$$

Notice how the top-level formula A does not play any role in this definition. Indeed, in the simply deep setting, it is only used to ensure that the linkage is well-formed.

Proposition 1 (Termination)**.** *Let $\mathfrak{L}$ be a linkage. Then $\mathsf{Tr}(\mathfrak{L}) \neq \varnothing$.*

Proof. It suffices to observe that each application of a rule in ISL_1^0 decreases strictly the size of the selected context, and no rule is applicable after discarding the empty context in an instance of the identity rules. □

Proposition 2 (Determinism)**.** *Let $\mathfrak{L}$ be a linkage, and $\mathcal{D}, \mathcal{D}' \in \mathsf{Tr}(\mathfrak{L})$. Then $\mathcal{D} = \mathcal{D}'$.*

Proof. This relies on the two following facts:

- There is exactly one left introduction rule and one right introduction rule associated with each form of context, and ISL_1^0 rules only deal with explicit fillings.
- Sequents in the derivation contain at most one selected subformula (one in the branch of the derivation decomposing χ, zero in open premises).

□

Thanks to these properties, we can define simply deep linking as a total function:

Definition 12 (Simply deep linking)**.** *The* linking *of a simply deep linkage $\mathfrak{L}$ is defined as the list of open premises of $\mathsf{Tr}(\mathfrak{L})$, ordered by decreasing height.*

Example 1. *Here is an example of linkage (on the left) and its associated interaction trace:*

$$\boxed{A} \vdash B \Rightarrow (\boxed{A} \wedge B) \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash \Box\,[A]}\,\mathsf{id}R \qquad A, B \vdash B}{A, B \vdash (\Box \wedge B)\,[A]}\,\wedge R_1}{A \vdash (B \Rightarrow (\Box \wedge B))\,[A]}\,\Rightarrow R_1$$

Thus, its linking is the subgoal $A, B \vdash B$.

Proposition 3 (Correctness)**.** *Every ISL_1^0 derivation $\mathcal{D}$ of $\mathcal{S}$ with open premises $\mathcal{S}_i$ can be mapped to a derivation $[\![\mathcal{D}]\!]$ of $[\![\mathcal{S}]\!]$ with open premises $\mathcal{S}_i$ in the standard sequent calculus LJ for intuitionistic logic.*

Proof. We do not detail the translation $[\![\cdot]\!]$ here, but since ISL_1^0 can be seen as a restricted variant of LJ, it suffices to:

- forget about explicit fillings by replacing them with implicit ones;
- add instances of weakening/contraction appropriately when switching between multiplicative and additive variants of rules.

□

1.2.2 Doubly deep interaction

We now consider the natural generalization of linking to arbitrary linkages $i : \xi / A \overset{\mathfrak{S}}{=\!=} j : \chi / B$ where A can be any subformula of $\mathfrak{S}$.

At first, it seems like we could just keep using the rules of ISL_1^0, the only change being in the addition of the selection $\xi\,[A]$ to the goal. The problem is that it would render the search procedure *non-deterministic*, since at any step arises the choice of which of the two selected contexts to decompose.

To recover determinism, we choose to impose a *focusing* discipline on the rules of ISL_1^0. First introduced by [4] in the context of linear logic programming, the concept of *focused proof* precisely aims at eliminating all choices during proof-search, while preserving completeness of the procedure. That is, every formula provable with a non-focused proof should be provable with a focused one. Focused systems have since been devised for classical logic [6] and even modal logics [7], but most importantly for us, intuitionistic logic [6].

The key idea of focusing is to delay as much as possible the application of rules that can lead to dead-ends in the proof. To illustrate, if we take the following linkage:

$$\boxed{A} \vee B \vdash \boxed{A} \vee B$$

we can choose to first deconstruct either the left disjunction ($\vee L_1$ rule), or the right one ($\vee R_1$). However, whereas the first choice leads to the provable subgoal $B \vdash A \vee B$, the second

one leaves us with the unprovable subgoal $B \vdash A$. It turns out that $\vee L_1$, as opposed to $\vee R_1$, is *invertible*, meaning that each premise of the rule can be deduced from its conclusion. Thus, decomposing a formula with an invertible rule guarantees that subgoals are provable, hence the interest in applying these rules as soon as possible. This can justify the common practice of starting proofs in Coq with sequences of tactics like `intros; split; intros` : this corresponds to the invertibility of the $\forall R$, $\Rightarrow R$ and $\wedge R$ rules!

A finer analysis of the *permutations* of inferences in derivations allows to obtain the following result, which is a generalization of the previous example:

Theorem 1 (Inference permutation)**.** *Every successive (bottom-up) applications of two inference rules can be permuted systematically without loss of provability, except in the case of an invertible rule followed by a non-invertible one.*

The prioritizing of invertible rules is only one way to exploit this result: in its original formulation [4], focusing goes even further by imposing that non-invertible formulas be recursively decomposed until an invertible subformula is reached, which is possible thanks to the permutability of non-invertible rules with themselves.

In our setting of doubly deep linking, focusing simply consists in decomposing the invertible context when the other one is not invertible. If the two contexts have the same invertibility, the choice can be made arbitrarily thanks to the previous result. Thus, our focusing discipline is less constraining than the standard one. Still, in contrast to a full search procedure, it completely eliminates *don't-know* non-determinism, leaving only *don't-care* non-determinism. This is because linking is not concerned with providing a full proof of the goal, but only an open derivation which turns it into weaker subgoals, where all choices are implicit in the contexts of the selected subformulas. Furthermore, it does not prevent subgoals from being each time the same or at least equivalent, and thus linking from being in some sense deterministic.

We start by introducing two sub-categories of contexts, corresponding to those having non-invertible right (resp. left) introduction rules:

Definition 13 (Non-invertible contexts)**.** Right *(resp.* left*)* non-invertible contexts ρ *(resp.* λ*) are defined by the following grammar:*

$$\begin{aligned} \rho &::= \Box \mid \chi \vee A \mid A \vee \chi \\ \lambda &::= \Box \mid \chi \Rightarrow A \mid A \Rightarrow \chi \end{aligned}$$

This simple addition is enough to define the rules for doubly deep linking, which are gathered in the ISL_2^0 system presented in figure 5. To make the presentation lighter, we chose a *linear* version of the rules in the style of [1] (section 4.1). This means that selected assumptions are *consumed*, instead of being duplicated in all premises (which is implemented in ISL_1^0 with implicit fillings). While the question of linearity is important from a usability standpoint, it is orthogonal to focusing in the propositional case since it has no impact on invertibility, and will be discussed more thoroughly in section 2.2.

Focused calculi usually involve specialized forms of sequents to encode the start and end of a selection (e.g. with the decide and release rules of [4]), but we prefer here to rely on the simpler device of explicit fillings. In addition to the fact that we will reuse them in the deep inference formulation of linking to be presented in section 2, it also allows us to apply the id rule as soon as both contexts have been decomposed, instead of performing full η-expansion to reach atoms.

Invertible rules (colored blue in figure 5) stay unchanged from ISL_1^0, modulo linearity. Indeed, they can be applied on a selected context independently of the other, since we want to apply them as soon as possible and in arbitrary order.

The real change happens for non-invertible rules (colored red in figure 5): we want to apply them *iff* no invertible rule is applicable, that is when the two contexts are non-invertible. However for left rules, we need to distinguish between the "hypothesis *vs* hypothesis" or *forward* case (rules whose name ends with a ','), and the "hypothesis *vs* conclusion" or *backward* case (rules whose name ends with a '$\vdash$'). Then arises an interesting phenomenon, in that the natural rule $\Rightarrow L_1 \vdash$ for implication is *incorrect* for our purpose: indeed, the linked formulas are split among the two premises, which necessarily prevents their final interaction in the id rule. Thus, we colored this rule in gray to indicate that it must not be included in the system.

Even though ISL_2^0 contains rules absent from ISL_1^0, it is morally a subsystem of the latter since we reduced the number of derivations with the focusing discipline. Therefore, aside from the slight non-determinism discussed earlier, previously shown properties of termination and correctness should still be true. We postpone to future work a formal proof of these, based on a translation from ISL_2^0 to ISL_1^0.

Then we can adapt the definition of linking to the doubly deep setting:

Definition 14 (Doubly deep interaction trace)**.** *The* interaction traces $\mathsf{Tr}(\mathfrak{L})$ *of a linkage* $\mathfrak{L} = i : \xi \,/\, A \overset{\Gamma \vdash C}{=\!=} j : \chi \,/\, B$ *are all the* ISL_2^0 *derivations that can result from performing a bottom-up proof-search on the sequent* $\mathcal{S}$*, where*

$$\mathcal{S} = \begin{cases} \lfloor \Gamma \setminus i : \xi \{A\} \rfloor, \xi [A] \vdash \chi [B] & \text{if } j > 0 \\ \lfloor \Gamma \setminus j : \chi \{B\} \rfloor, \chi [B] \vdash \xi [A] & \text{if } i > 0 \\ \lfloor \Gamma \setminus i : \xi \{A\}, j : \chi \{B\} \rfloor, \xi [A], \chi [B] \vdash C & \text{otherwise} \end{cases}$$

Definition 15 (Doubly deep linking)**.** *The* linking *of a linkage* $\mathfrak{L}$ *is defined as the list of open premises of any derivation in* $\mathsf{Tr}(\mathfrak{L})$*, ordered by decreasing height.*

1.3 First-order linking

In this section, we are concerned with the extension of linking to *first-order logic*. To preserve determinism and termination of the procedure, we add a preliminary phase of *unification* between the linked subformulas, which generates a substitution used to instantiate some of the quantifiers in their contexts.

To simplify the presentation, we treat formulas up to α-equivalence by following the Barendregt's convention, which states that: 1. No variable is both free and bound. 2. Bound variables all have different names. In practice though, this convention is not respected when working with proof assistants. We will discuss briefly in the conclusion how this has been handled in the implementation of the `link` tactic of Actema.

1.3.1 Unification

In section 1.2.1, we introduced a *well-formedness* condition to ensure that any linking terminates on an instance of the identity rule, which relied crucially on the *equality* of the linked subformulas. When going first-order, propositional atoms X are extended to *predicates* $X(t_1, \ldots, t_n)$: while we could content ourselves with basic syntactic equality, this ignores many potential usecases of subformula linking. Indeed, consider the following linkage:

$$\boxed{X(t)} \vdash \exists x. \boxed{X(x)}$$

Even if $X(t) \neq X(x)$, the intuitive meaning is very clear: we want to *instantiate* the existentially quantified variable x with the term t. Dually, one often encounters in practice goals where the following linkage would be useful:

$$\boxed{X(t)}, \forall x. \boxed{X(x)} \Rightarrow A \vdash C$$

This would both instantiate x with t and apply the implication, generating the subgoal:

$$X(t), \forall x. X(x) \Rightarrow A, A\{x := t\} \vdash C$$

In both cases, the information on which witness to give to the quantifier was obtained by a direct matching of the argument of the two occurrences of X. More generally, there will be as many equations as the summed arities of all matching occurrences of predicates in the linked formulas, which must all be solved with the same instantiations of quantified variables. Thus it defines a *first-order unification problem* whose solution is the set of instantiations, or equivalently a *substitution* σ. For more information on first-order unification, we refer the reader to the standard algorithm of Martelli and Montanari [8], which is the one actually used in our implementation.

We still need to determine which quantified variables should be regarded as *unification variables* to be instantiated during unification, and which should not. It turns out that this information can also be deduced from the shape of the contexts of selected formulas. But first we need to extend context-based notions to support quantifiers:

Definition 16 (First-order formula contexts)**.** *The grammar of formula contexts is extended in the following way:*

$$\chi, \xi, \zeta \ ::= \ \ldots \mid \forall x. \chi \mid \exists x. \chi$$

Definition 17 (First-order polarized contexts)**.** *The grammar of polarized contexts is extended in the following way:*

$$\pi \ ::= \ \ldots \mid \forall x. \chi \mid \exists x. \chi$$

Definition 18 (Quantifier occurrence)**.** *The* quantifier occurrence *of a variable x in an occurrence $\mathfrak{o}^A$ is defined by:*

$$\mathfrak{o}^A_x = \begin{cases} \mathfrak{o}^B & \text{if there exists } \mathfrak{o}^B \leq \mathfrak{o}^A \text{ s.t. either } B = \exists x.C \text{ or } B = \forall x.C \\ \varnothing & \text{otherwise} \end{cases}$$

Definition 19 (Unification variables)**.** *The set of* unification variables *of an occurrence $\mathfrak{o}^A$ is defined by:*

$$\mathcal{V}(\mathfrak{o}^A) = \left\{ x \;\middle|\; \mathfrak{o}^A_x = \mathfrak{o}^B \text{ and } B = \begin{cases} \exists x.C & \text{if } \mathfrak{o}^B \text{ is positive} \\ \forall x.C & \text{otherwise} \end{cases} \right\}$$

In other words, it is the set of variables quantified by a positive (resp. negative) existential (resp. universal) occurrence.

Definition 20 (Unification problem)**.** *Let $A \doteq B$ be the set of equations between first-order terms defined recursively by:*

$$\begin{aligned}
X(t_1,\dots,t_n) \doteq X(u_1,\dots,u_n) &= \bigcup_{i=1}^{n} \{t_i \doteq u_i\} \\
\top \doteq \top &= \varnothing \\
\bot \doteq \bot &= \varnothing \\
A \wedge B \doteq C \wedge D &= A \doteq C \ \cup \ B \doteq D \\
A \vee B \doteq C \vee D &= A \doteq C \ \cup \ B \doteq D \\
A \Rightarrow B \doteq C \Rightarrow D &= A \doteq C \ \cup \ B \doteq D \\
\forall x.A \doteq \forall y.B &= A \doteq B\,\{y := x\} \\
\exists x.A \doteq \exists y.B &= A \doteq B\,\{y := x\}
\end{aligned}$$

and undefined otherwise. The unification problem $\mathcal{P}(\mathfrak{L})$ *of a linkage $\mathfrak{L} = \mathfrak{o}^A \overset{\mathfrak{S}}{=\!=} \mathfrak{o}^B$ is defined as $A \doteq B$, where unification variables are restricted to $\mathcal{V}(\mathfrak{o}^A) \cup \mathcal{V}(\mathfrak{o}^B)$.*

There is one last necessary ingredient to formulate our refined well-formedness condition for first-order linkages. Consider the following linkage:

$$\forall a.\exists b.\boxed{X(b,a)} \vdash \exists x.\forall y.\boxed{X(x,y)}$$

Let us try to grasp its deductive meaning. The first step of reasoning would be to instantiate either x with b or a with y, following the substitution given by unification: but neither b nor y are available yet, since they can only be obtained as functions of the witnesses given in place of a and x[2]! Thus we detect here a form of *vicious circle*, which indicates erroneous reasoning. To forbid such linkages, we introduce a *dependency relation* on the quantified variables of an occurrence, which is akin to the one of expansion-tree proofs [5]:

Definition 21 (Dependency relation)**.** *Let $\mathfrak{o}^A$ be an occurrence, and let $<^0_{\mathfrak{o}^A}$ be the relation defined on the set of its quantified variables by $x <^0_{\mathfrak{o}^A} y$ if $\mathfrak{o}^A_x < \mathfrak{o}^A_y$ with $x \in \mathcal{V}(\mathfrak{o}^A)$ and $y \notin \mathcal{V}(\mathfrak{o}^A)$. The* dependency relation $<_{\mathfrak{o}^A}$ *of $\mathfrak{o}^A$ is defined as the transitive closure of $<^0_{\mathfrak{o}^A}$.*

The dependency relation $<_{\mathfrak{L}}$ *of a linkage $\mathfrak{L} = \mathfrak{o}^A \overset{\mathfrak{S}}{=\!=} \mathfrak{o}^B$ is defined as $<_{\mathfrak{o}^A} \cup <_{\mathfrak{o}^B}$.*

The dependency relation of a linkage can be seen as the juxtaposition of two directed graphs, corresponding to the dependency relations of the two linked occurrences. Then, if unification succeeds with a substitution σ, the informal reasoning sketched above is captured formally by *applying* σ to both graphs, which will often have the effect of connecting them together, sometimes creating *cycles*.

Definition 22 (Dependency substitution)**.** *The application of a substitution σ to a dependency relation $<_{\mathfrak{L}}$ is the relation $<^\sigma_{\mathfrak{L}}$ defined by $x <^\sigma_{\mathfrak{L}} y$ if either $x <_{\mathfrak{L}} y$, or there exists z such that $z <_{\mathfrak{L}} y$ and x occurs in $\sigma(z)$.*

Definition 23 (Well-formed first-order linkage)**.** *A linkage $\mathfrak{L} = i : \xi \,/\, A \overset{\mathfrak{S}}{=\!=} j : \chi \,/\, B$ is said to be* well-formed *if $i \neq j$, A and B have opposite polarities, $\mathcal{P}(\mathfrak{L})$ has a solution σ, and $<^\sigma_{\mathfrak{L}}$ is acyclic.*

[2] we rely here on a functional interpretation of hypotheses of the form $\forall x.\exists y.R(x,y)$, which can be found at work in the procedure of Skolemization dating back to 1920, or even before in some axioms of first-order theories like the Axiom of Choice in ZFC.

1.3.2 Interaction

When a linkage $i : \xi \,/\, A \overset{\mathfrak{S}}{=\!=} j : \chi \,/\, B$ is well-formed, it will now produce a substitution σ instantiating those quantifiers in ξ and χ which bind variables that have been unified in A and B. It only remains to actually perform this instantiation during linking, as specified by the $\exists R$ and $\forall L$ rules of figure 6. These rules rely on sequents of the form $\Gamma \vdash_\sigma C$, so that witnesses can be fetched in the substitution σ. We call ISL_2^1 the system containing the quantifier rules of figure 6, as well as the rules of ISL_2^0 upgraded with the new decorated sequents.

The only new non-invertible rule is $\exists R$, thus we extend the grammar of non-invertible context as follows:

$$\rho \ ::= \ \ \ldots \mid \exists x.\chi$$

One might expect dually to have $\forall L$ as non-invertible, but we explicitly chose the non-linear, invertible variant of the rule to minimize the loss of provability in generated subgoals.

One important property of the rules $\exists R$ and $\forall L$ is that they are *incomplete*, in the sense that they can only be applied if the quantified variable has been instantiated during unification. Otherwise, the search procedure simply get stuck before reaching the identity rule, and thus linking is *undefined*. We could either forbid the faulty linkages by strengthening the well-formedness condition, or find a way to preserve uninstantiated quantifiers. Unfortunately, the rules of sequent calculus can only deal with formulas at the top-level, making this second alternative impossible. A third one would be to introduce a mechanism of *existential variables*, as is done in some proof assistants like Coq. But they add a *global* flavor to the system, by requiring that delayed instantiations be synchronized in all branches of a proof, which also cannot be expressed straightforwardly in sequent calculus.

As for correctness of first-order linking, it is less obvious than in the propositional case because of the intervention of unification and dependency-checking in the well-formedness condition. In fact, this condition might be compared to the *correctness criterions* of compact representations of proofs such as the already mentioned expansion-tree proofs, but also Hughes' combinatorial proofs or Girard's proof-nets [9]. Then, the correctness theorem of linking would be analoguous to the *sequentialization theorem* of proof-nets, where the possible sequentializations are the interaction traces.

2 Subformula linking in deep inference

The linking procedure described in section 1 already captures a lot of usecases for drag-and-drop actions, and arguably improves over PbP in at least two respects: the well-formedness condition and the focusing discipline protect the user from many wrong choices in the proof; and unification provides a powerful mechanism to instantiate quantifiers, that often prevents the user from having to "invent" witnesses. But this power comes at the price of *completeness*: most goals become unprovable with linking alone, because the well-formedness condition forbids the two linked formulas from occurring in the same top-level formula ($i \neq j$). This is especially problematic with goals of the form $\vdash C$, where for example

$$\vdash \boxed{A} \Rightarrow \boxed{A}$$

is not well-formed. One approach, which is the one currently implemented in Actema, is to let linking coexist with other complementary actions, e.g. $\vdash A \Rightarrow A$ can be turned into

$A \vdash A$ by double-clicking on $A \Rightarrow A$. But there are other issues beside completeness: linking is intrinsically *destructive*, in the sense that in order to bring the two selected formulas to the top-level for the id rule, both of their contexts need to be completely decomposed by introduction rules. This was already a problem for uninstantiated quantifiers in section 1.3.2, but also at the propositional level in the example illustrating PbP in section 1.1. Indeed, the generated subgoal $A \wedge (B \vee C) \vdash B$ was *not* provable, because the selection forced too soon the choice of $A \wedge B$ over C with the $\vee R_1$ rule, when the intuitive result would rather be to obtain the provable subgoal $A \wedge (B \vee C) \vdash B \vee C$. To preserve contexts, we need to avoid altogether the process of bringing the selected formulas to the top-level, which requires the ability to apply the id rule *inside* a context. Since this is against the principles of standard Gentzen calculi, we turn to a less known framework for proof formalisms called *deep inference*.

2.1 Deep inference

The ability to apply rules deep inside a context is precisely the common feature of all formalisms following the deep inference methodology. The first of these formalisms, the *calculus of structures* (CoS hereafter), was introduced in 1999 by Alessio Guglielmi [10], and is still the most used and studied. The original technique of subformula linking devised by Chaudhuri [2] was defined in the CoS, and so is our adaptation of his work to intuitionistic logic.

In the CoS, there is no distinction between formulas and sequents, meaning that the judgments manipulated by inference rules are plain formulas. Also, the *tree*-shaped derivations of Gentzen calculi are collapsed into plain *lists* of inferences. Add to this the contextual closure of rules, and you get something closer to a *rewrite* system, where all rules have the form:

$$\frac{\chi\{A\}}{\chi\{B\}}$$

We will most of the time omit the context χ when defining rules, except when we require that it has a particular shape. In the context of intuitionistic logic, a *proof* is then a derivation starting from the truth constant $\top$, which stands for the absence of prior knowledge, or equivalently the empty set of premises of an axiom:

Definition 24 (CoS proof)**.** *A* derivation ϕ *of* B *from* A *in a CoS system* S*, written* $\phi : A \xrightarrow{\mathsf{S}} B$*, is a sequence of rules with the bottom-most rule having conclusion* B *and the top-most rule having premise* A*. We write* $A \xrightarrow{\mathsf{S}} B$ *to assert that there exists a* ϕ *such that* $\phi : A \xrightarrow{\mathsf{S}} B$*.* A proof *of* A *in* S *is a derivation* $\phi : \top \xrightarrow{\mathsf{S}} A$*.*

With these ingredients, it is easy to *encode* rules from sequent calculus. For example, the $\vee L$ rule of figure 3 can be encoded as:

$$\frac{\pi\{(A_1 \Rightarrow C) \wedge (A_2 \Rightarrow C)\}}{\pi\{(A_1 \vee A_2) \Rightarrow C\}}$$

There are three things to notice in this encoding: 1. the '$\vdash$' of sequents is replaced by the implication $\Rightarrow$; 2. the set of premisses is explictly written as a conjunction; and 3. the context of hypotheses Γ is generalized into the positive context π. Indeed, if $\Gamma = A_1, \ldots, A_n$, it can be seen as the special case $\pi = (\bigwedge_i A_i) \Rightarrow \square$. Many rules to be introduced are better understood as deep variants of sequent calculus rules encoded this way.

2.2 Propositional linking

As in section 1.2.2, we will define linking as the result of pseudo-deterministic proof-search in a specially tailored calculus. However, we consider a larger set of rules that is able to represent not only partial derivations performing one linking (the so-called interaction traces), but complete proofs based on a sequence of linkings. This requires crucially the ability to create a linkage by looking deep inside formulas, which was not possible in the shallow setting of sequent calculus.

The complete system for intuitionistic propositional logic is shown in figure 7. We call it DISL^0, the additional D standing for Deep. It is comprised of 4 parts, and each contiguous, maximally long sequence of rules that belong to the same part in a derivation is called a *phase*.

Backward A backward phase is generally initiated with the lnp rule, which creates a linkage between two *incomparable* occurrences (definition 5) ancestrally connected by a positive occurrence of implication. This is a generalization of linkages between an hypothesis and conclusion, where instead of turning the linkage

$$\Gamma, \chi\left\{\boxed{A}\right\} \vdash \xi\left\{\boxed{B}\right\}$$

into the sequent $\Gamma, \chi[A] \vdash \xi[B]$ by relying on the meta-level definition of interaction traces, we turn the formula reading of the sequent $\bigwedge \Gamma \Rightarrow (\chi\{A\} \Rightarrow \xi\{B\})$ into the formula $\bigwedge \Gamma \Rightarrow (\chi[A] \vdash \xi[B])$ by instantiating the object-level rule lnp with $\pi = \bigwedge \Gamma \Rightarrow \Box$. Thus in this setting, the symbol '$\vdash$' plays the role of what we call an *interaction connective*, following the terminology of Chaudhuri [2]. It has the same semantic reading as implication, and its sole purpose is to guide proof-search by making all other backward rules act as "introduction rules" for this connective.

The most straightforward way for a backward phase to end is when the two linked formulas are brought together on each side of the $\vdash$ connective. If they are equal, the lnpid rule, which plays the role of the id rule of sequent calculus, is applied. If not, the unlnp rule releases the linkage by turning $\vdash$ back into $\Rightarrow$. In a way, this rule makes linking closer to the original semantics of PbP, since it leaves room for linkings that do not end on the identity rule.

The remaining backward rules serve as a mean to bring the linked formulas closer, not by *bubbling* them to the top-level, but by instead *descending* the interaction connective $\vdash$ into their contexts, much like the commutative cases of substitution or cut-elimination. Rules whose naming pattern is $\mathsf{lnpr}*$ (resp. $\mathsf{lnpl}*$) are deep variants of the left (resp. right) focused rules of ISL^0_2 shown in the 2nd (resp. 3rd) column of figure 5. In particular, the right rules $\mathsf{lnprd}i$ for disjunction preserve the context C, thus solving the problem of loss of provability of the $\vee R_i$ rules mentioned earlier. More importantly, the $\mathsf{lnpri1}$ rule switches $\vdash$ for a new interaction connective '$*$', ending the backward phase and starting a *forward* phase. This is because it inverts the polarity of the linked antecedant χ, exactly like the $\Rightarrow R_1$ rule.

Remark 1. *In our presentation of these "commutation" rules (also called* switch *rules in standard CoS systems), we omit writing the linked subformulas since they only need to be inspected in the* lnpid *and* unlnp *rules. For example, the rigorous formulation of* $\mathsf{lnplc1}$ *is:*

$$\frac{C \Rightarrow (\chi[A] \vdash \xi[B])}{(\chi \wedge C)[A] \vdash \xi[B]}\ \mathsf{lnplc1}$$

Forward The forward phase can in some sense be understood as a *left adjoint* to the backward phase, following the standard adjunction $- \wedge A \dashv A \Rightarrow -$ of cartesian closed categories. The lnn rule creates a linkage between two incomparable occurrences ancestrally connected by a negative occurrence of conjunction. This is a generalization of linkages between two hypotheses, where the sequent $\Gamma, \chi[A], \xi[B] \vdash C$ is replaced by the formula $(\bigwedge \Gamma \wedge (\chi[A] * \xi[B])) \Rightarrow C$ obtained by applying lnn with $\eta = (\bigwedge \Gamma \wedge \Box) \Rightarrow C$, '$*$' having the same semantic reading as conjunction.

Contrarily to lnpid, the lnnid rule is quite accessory, and intuitively reads as "linking two occurrences of the hypothesis A merges them together". We also need the lnncomm rule to express the commutativity of $*$. Finally, the lnni1 rule is like the inverse of lnpri1, as it ends the forward phase and initiates a backward phase.

Definition 25 (Interaction phase)**.** *We call* interaction phase *the alternation of backward and forward phases produced by one linkage in a derivation, where the phase switch occurs each time a polarity is inverted.*

Invertibility While commutation rules for the backward phase share similarities with the introduction rules of sequent calculus, their emphasis on context preservation changes their invertibility. Furthermore, there is no sequent calculus equivalent of forward commutation rules. Thus we need to refine the notion of invertible context:

Definition 26 (Deeply non-invertible context)**.** *Right (ρ), left (λ) and forward (ϕ)* deeply non-invertible contexts *are defined by the following grammar:*

$$\begin{aligned}
\rho &::= \Box \mid \chi \wedge A \mid A \wedge \chi \mid \chi \vee A \mid A \vee \chi \\
\lambda &::= \Box \mid \chi \Rightarrow A \mid A \Rightarrow \chi \\
\phi &::= \Box \mid \chi \vee A \mid A \vee \chi \mid \chi \Rightarrow A \mid A \Rightarrow \chi
\end{aligned}$$

Units When a backward phase ends successfully on lnpid, the linked occurrences are rewritten into $\top$. Thus after many linkings, the goal will contain a lot of occurrences of $\top$ that obscure what really remains to be proved, especially since the contexts of linked formulas have been preserved. Fortunately, these contexts can be simplified using algebraic unit laws, which for intuitionistic logic correspond to provable identities in Heyting algebras. The neul and neur (resp. absl and absr) rules describe neutral (resp. absorbing) elements with respect to each connective. This includes falsity $\bot$, which is essential to define negation as $\neg A \triangleq A \Rightarrow \bot$. Last but not least, the efq rule implements the familiar *ex falso quodlibet* principle, without which we would only be in minimal logic.

Resources A remarkable property of the DISL^0 rules considered up until now is that they are linear, like the rules of ISL^0_2. A nice consequence of linearity is that linkings do not complexify arbitrarily the goal by introducing or duplicating formulas, which is essential from a usability standpoint to keep track of what is happening. However in terms of provability, it makes many formulas unprovable, the canonical example being the diagonal map $A \Rightarrow (A \wedge A)$. Instead of systematically duplicating each part of the selected contexts as in ISL^0_1, we add here a *contraction* rule conn, which corresponds to the usual left contraction rule of sequent calculus. This means that (negative occurrences of) formulas can be manually duplicated in the goal whenever needed, which can easily be associated to some input mechanism in an

interface. Another alternative discussed in [2] is to systematically duplicate the whole selected contexts *once* when creating a linkage, with the following variants of lnp and lnn:

$$\frac{\pi\{(\chi\{A\} \wedge \xi\{B\}) \Rightarrow (\chi[A] \vdash \xi[B])\}}{\pi\{\chi\{A\} \Rightarrow \xi\{B\}\}}\ \mathsf{lnp} \qquad \frac{\eta\{(\chi\{A\} \wedge \xi\{B\}) \wedge (\chi[A] * \xi[B])\}}{\eta\{\chi\{A\} \wedge \xi\{B\}\}}\ \mathsf{lnn}$$

While it prevents the user from worrying about duplication issues, it has the dual overhead of possibly surcharging the goal with useless hypotheses. This is linked to the old question of *relevance* in logic, and there does not seem to be any generic way to determine in advance which hypotheses will need duplication. Thus we also add the *left weakening* rule wkn to manually discard useless hypotheses.

Correctness/Completeness Relying only on the intuitionistic reading of $\wedge$, $\vee$ and $\Rightarrow$, it is quite easy to convince oneself that in all rules of DISL^0, the premiss entails the conclusion. A more formal proof of correctness would translate each rule as a derivation in a standard calculus, e.g. lnpli2 would be translated in the natural deduction system NJ as:

$$\dfrac{\dfrac{\dfrac{C \wedge (\chi \Rightarrow \xi)}{C}\wedge E_1 \qquad [C \Rightarrow \chi]^1}{\chi}\Rightarrow E \qquad \dfrac{C \wedge (\chi \Rightarrow \xi)}{\chi \Rightarrow \xi}\wedge E_2}{\xi}\Rightarrow E}{(C \Rightarrow \chi) \Rightarrow \xi}\Rightarrow I^1$$

Conversely, completeness could be obtained by simulating rules of a standard system with DISL^0 derivations. However, the linking and focusing disciplines make the proof much harder to obtain as a direct translation from a Gentzen calculus. We still conjecture that completeness holds, and the proof will probably rely on a sequence of intermediate calculi as is done in [2].

Deep linking in sequents We can now leverage the rules of DISL^0 to specify an alternative behavior of linkages on sequents, that we call *deep linking*:

Definition 27 (Occurrence intersection)**.** *The* intersection $\mathfrak{o}^A \sqcap \mathfrak{o}^B$ *of two occurrences* $\mathfrak{o}^A$ *and* $\mathfrak{o}^B$ *is their greatest lower bound in the suboccurrence ordering if they have one, and is undefined otherwise.*

Definition 28 (Deep well-formedness)**.** *A linkage* $\mathfrak{o}^A \overset{\mathfrak{S}}{=\!=} \mathfrak{o}^B$ *is said to be* deeply well-formed *if* $\mathfrak{o}^A = i : \chi \,/\, A$ *and* $\mathfrak{o}^B = j : \xi \,/\, B$ *are incomparable, and if* $i = j$ *then* $\mathfrak{o}^A \sqcap \mathfrak{o}^B = i : \zeta \,/\, D \circ E$ *with* $\langle \zeta, \circ \rangle \in \{\langle \pi, \Rightarrow \rangle, \langle \eta, \wedge \rangle\}$*,* $D, E \in \{\chi\{A\}, \xi\{B\}\}$ *and* $D \neq E$*.*

Definition 29 (Deep linking)**.** *Let* $\mathfrak{L} = \mathfrak{o}^A \overset{\Gamma \vdash C}{=\!=} \mathfrak{o}^B$ *be a deeply well-formed linkage with* $\mathfrak{o}^A = i : \chi \,/\, A$ *and* $\mathfrak{o}^B = j : \xi \,/\, B$*. The* deep linking *of* $\mathfrak{L}$ *is given by the following table:*

	$i < 0$	$i > 0$
$i \neq j < 0$	$\Gamma, [\![\chi[A] * \xi[B]]\!] \vdash C$	$\Gamma \vdash [\![\xi[B] \vdash \chi[A]]\!]$
$i \neq j > 0$	$\Gamma \vdash [\![\chi[A] \vdash \xi[B]]\!]$	—
$i = j$	$\Gamma, \zeta\{[\![D \bullet E]\!]\} \vdash C$	$\Gamma \vdash \zeta\{[\![D \bullet E]\!]\}$
	with $\mathfrak{o}^A \sqcap \mathfrak{o}^B = i : \zeta \,/\, D \circ E$ *and* $\langle \circ, \bullet \rangle \in \{\langle \Rightarrow, \vdash \rangle, \langle \wedge, * \rangle\}$	

where for any A, $[\![A]\!]$ denotes the formula obtained by proof-search on A in the system $\mathsf{DISL}^0 \setminus \{\mathsf{lnp}, \mathsf{lnn}, \mathsf{conn}, \mathsf{wkn}\}$.

In other words, deep linking performs an interaction phase followed by a unit elimination phase. Indeed, proof-search will stop when no more unit rule is applicable since we removed the lnp and lnn rules that could create other linkages, as well as the conn and wkn rules. The definition ensures that selected hypotheses are implicitly duplicated.

Conclusion

Implementation The backend of the Actema prototype is written in the functional programming language OCaml, which is a standard choice in the area of proof assistants[3]. PbP-based first-order linking and deep propositional linking are implemented as pure recursive functions named `link` and `dlink` respectively, that both take a linkage and return the list of generated subgoals (which is always a singleton in the case of `dlink`). Therefore they can be seen as instances of *tactics* in more traditional proof systems, even though they are never invoked textually. The *don't-care* non-determinism of proof-search is resolved arbitrarily through the semantics of pattern matching, which will apply the inference rule that corresponds to the first declared clause.

In the first-order case, a lot of effort has been put in handling naming conflicts. In particular, while we use generic unification and dependency-checking algorithms that work on a single substitution, the interaction phase works with one substitution for each selected context, in order to preserve as much as possible user-defined names. Deep linking is naturally expressed as mutually recursive functions `backward` and `forward`, composed with a unit elimination function `elim_units` that applies simplification rules recursively until a fixpoint is reached. We also implemented a function `search_match` that builds a list of all well-formed linkages that can be performed in the current goal. Hence it is in charge of performing unification and dependency-checking, and is also used to provide information to the highlighting mechanism discussed in section 1.2.1.

Overall, we tried to keep the implementation simple, and thus ignored issues of algorithmic efficiency. Two web demos are available at [12] and [13], that associate respectively the `link` and `dlink` tactics to drag-and-drop actions. In the case of `dlink`, the interface does not support yet the selection of arbitrary subformulas, and is thus restricted to the well-formed linkages of definition 10.

Generalizations As pointed out in both [1] and [2], there are a number of ways to extend the notion of subformula linking. The most obvious one is to consider different logics, such as classical logic and modal logics. For classical logic, it can be done within the PbP-based approach by just allowing multiple conclusions on the right of sequents and by using rules of the LK system, an idea which is surprisingly not mentioned in [1]. More interesting would be extensions to expressive logics that support higher-order and inductive features such as HOL or the Calculus of Constructions. While there are some experimental sequent systems in the litterature for such logics [14], there are none in the setting of deep inference. It is also known that unification becomes undecidable starting from third-order logic [15]. An easier

[3]Indeed, OCaml is a descendent of the ML language, whose design was motivated by the need of a proof language for one of the very first interactive theorem provers, LCF [11].

extension would be to support theories with *equality*, for example by rewriting equalities through drag-and-drop actions.

Related works The idea of reducing a proof to a collection of links between its dual occurrences of formulas is not new, and can be traced back to the *matings* of Andrews [16] in the context of automated deduction. Matings are *sets* of links covering all *atomic* occurrences, and proofs are matings satisfying certain conditions. Our work differs in that we are interested in *interactive* deduction, and thus consider links as a mechanism of inference rather than a syntactic criterion to discriminate proofs. Then a proof is better understood as a *list* of links, and the atomicity constraint is relaxed to gain expressivity, since the creation of links is offloaded to the user instead of the search procedure.

Another line of work, starting with the *proof-nets* of Girard [17], is concerned with the more fundamental problem of *proof identity*, which requires a canonical notion of proof object [9]. In the case of unit-free multiplicative linear logic, the absence of any form of duplication/sharing/removal mechanism allows to completely characterize a proof-net by the set of its *axiom* links, the difference with matings being that correctness of a proof structure can be checked in *polynomial* instead of exponential time. This is because adding additives or exponentials, which can encode intuitionistic and classical logic, requires additional structure to represent uses of weakening and contraction. The *combinatorial proofs* of Hughes [18] [19] are examples of polynomially-checkable proof objects exhibiting such structure, and have recently been extended to handle first-order classical quantifiers [20] (intuitionistic quantifiers are still an open problem). This compartmentalization of axiom links and structural rules resembles the distinction between interaction phases and manual applications of conn and wkn in DISL^0, which is itself inspired by the *decomposition theorem* of deep inference formalisms.

Another approach to canonicity is to consider a generalization of focused proofs called *maximally multi-focused proofs*, which have been proven isomorphic to compact representations of proofs such as proof-nets [21] and expansion tree proofs [22]. We conjecture that our system ISL^1_2 could be reformulated as a multi-focused proof system, where foci correspond to the linked formulas, and thus their number is restricted to exactly 2.

Hence, it appears that subformula linking exhibits properties of both approaches to canonicity, but at the level of *partial proofs*: well-formed linkages make for *compact-parallel-spatial* representations of inferences, whose operational meaning is given by their *detailed-sequential-temporal* interaction traces.

References

[1] Yves Bertot, Gilles Kahn, and Laurent Théry. Proof by pointing. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software*, volume 789, pages 141–160. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[2] Kaustuv Chaudhuri. Subformula linking as an interaction method. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, volume 7998, pages 386–401. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[3] Jan von Plato. The development of proof theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2018 edition, 2018.

[4] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. 2(3):297–347.

[5] Dale A. Miller. A compact representation of proofs. *Studia Logica*, 46:347–370, 1987.

[6] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. 410(46):4747–4768.

[7] Dale Miller and Marco Volpe. Focused labeled proof systems for modal logic. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 9450, pages 266–280. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[8] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 4(2):258282, April 1982.

[9] Lutz StraSSburger. The problem of proof identity, and why computer scientists should care about hilbert's 24th problem. 377(2140):20180038.

[10] Alessio Guglielmi. A calculus of order and interaction. 1999.

[11] Tobias Nipkow. Logic and computationinteractive proof with cambridge lcf: By l.c. paulson. cambridge university press, cambridge, 1987, price č27.50, isbn 0 521 34632 0. *Science of Computer Programming*, 11:178–180, 1988.

[12] Typical team. *Actema prototype with PbP-based first-order linking*, 2020. `https://prover-interface-icjs29hlj.vercel.app`.

[13] Typical team. *Actema prototype with deep propositional linking*, 2020. `https://prover-interface-2iow9pf77.vercel.app`.

[14] Étienne Miquey, Xavier Montillet, and Guillaume Munch-Maccagnoni. Dependent type theory in polarised sequent calculus (abstract). page 4.

[15] Gerard P. Huet. The undecidability of unification in third order logic. 22(3):257 – 267.

[16] Andrews. Refutations by matings. *IEEE Transactions on Computers*, C-25(8):801–807, 1976.

[17] Jean-Yves Girard. Linear logic. 50(1):1 – 101.

[18] Dominic Hughes. Proofs without syntax. *Annals of Mathematics*, 164(3):10651076, Nov 2006.

[19] Willem B. Heijltjes, Dominic J. D. Hughes, and Lutz StraBburger. Intuitionistic proofs without syntax. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE.

[20] Dominic J. D. Hughes. First-order proofs without syntax, 2019.

[21] Kaustuv Chaudhuri, Dale Miller, and Alexis Saurin. Canonical sequent proofs via multi-focusing. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and Luke Ong, editors, *Fifth Ifip International Conference On Theoretical Computer Science Tcs 2008*, volume 273, pages 383–396. Springer US. ISSN: 1571-5736 Series Title: IFIP International Federation for Information Processing.

[22] Kaustuv Chaudhuri, Stefan Hetzl, and Dale Miller. A multi-focused proof system isomorphic to expansion proofs. 26(2):577–603.

A Proof calculi

$$\frac{A_1}{A_1 \vee A_2}\ \vee I_1 \qquad \frac{A_2}{A_1 \vee A_2}\ \vee I_2 \qquad \frac{A_1 \vee A_2 \quad \begin{array}{c}[A_1]\\ \vdots \\ C\end{array} \quad \begin{array}{c}[A_2]\\ \vdots \\ C\end{array}}{C}\ \vee E$$

Figure 2: Natural deduction rules for disjunction $\vee$

$$\frac{\Gamma, A_1 \vdash C}{\Gamma, A_1 \wedge A_2 \vdash C}\ \wedge L_1 \qquad \frac{\Gamma, A_2 \vdash C}{\Gamma, A_1 \wedge A_2 \vdash C}\ \wedge L_2 \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge R$$

$$\frac{\Gamma, A_1 \vdash C \quad \Gamma, A_2 \vdash C}{\Gamma, A_1 \vee A_2 \vdash C}\ \vee L \qquad \frac{\Gamma \vdash A_1}{\Gamma \vdash A_1 \vee A_2}\ \vee R_1 \qquad \frac{\Gamma \vdash A_2}{\Gamma \vdash A_1 \vee A_2}\ \vee R_2$$

Figure 3: Sequent calculus rules for disjunction $\vee$ and conjunction $\wedge$

$$\frac{}{\Gamma, \Box\,[A] \vdash A}\ \mathsf{id}L \qquad \frac{}{\Gamma, A \vdash \Box\,[A]}\ \mathsf{id}R$$

$$\frac{\Gamma, (\chi \wedge B)\,\{A\}, \chi\,[A], B \vdash C}{\Gamma, (\chi \wedge B)\,[A] \vdash C}\ \wedge L_1 \qquad \frac{\Gamma \vdash \chi\,[A] \quad \Gamma \vdash B}{\Gamma \vdash (\chi \wedge B)\,[A]}\ \wedge R_1$$

$$\frac{\Gamma, (B \wedge \chi)\,\{A\}, B, \chi\,[A] \vdash C}{\Gamma, (B \wedge \chi)\,[A] \vdash C}\ \wedge L_2 \qquad \frac{\Gamma \vdash B \quad \Gamma \vdash \chi\,[A]}{\Gamma \vdash (B \wedge \chi)\,[A]}\ \wedge R_2$$

$$\frac{\Gamma, (\chi \vee B)\,\{A\}, \chi\,[A] \vdash C \quad \Gamma, (\chi \vee B)\,\{A\}, B \vdash C}{\Gamma, (\chi \vee B)\,[A] \vdash C}\ \vee L_1 \qquad \frac{\Gamma \vdash \chi\,[A]}{\Gamma \vdash (\chi \vee B)\,[A]}\ \vee R_1$$

$$\frac{\Gamma, (B \vee \chi)\,\{A\}, B \vdash C \quad \Gamma, (B \vee \chi)\,\{A\}, \chi\,[A] \vdash C}{\Gamma, (B \vee \chi)\,[A] \vdash C}\ \vee L_2 \qquad \frac{\Gamma \vdash \chi\,[A]}{\Gamma \vdash (B \vee \chi)\,[A]}\ \vee R_2$$

$$\frac{\Gamma, (\chi \Rightarrow B)\,\{A\} \vdash \chi\,[A] \quad \Gamma, (\chi \Rightarrow B)\,\{A\}, B \vdash C}{\Gamma, (\chi \Rightarrow B)\,[A] \vdash C}\ \Rightarrow L_1 \qquad \frac{\Gamma, \chi\,[A] \vdash B}{\Gamma \vdash (\chi \Rightarrow B)\,[A]}\ \Rightarrow R_1$$

$$\frac{\Gamma, (B \Rightarrow \chi)\,\{A\} \vdash B \quad \Gamma, (B \Rightarrow \chi)\,\{A\}, \chi\,[A] \vdash C}{\Gamma, (B \Rightarrow \chi)\,[A] \vdash C}\ \Rightarrow L_2 \qquad \frac{\Gamma, B \vdash \chi\,[A]}{\Gamma \vdash (B \Rightarrow \chi)\,[A]}\ \Rightarrow R_2$$

Figure 4: Rules of ISL_1^0

$$\frac{}{\Gamma, \Box\,[A] \vdash \Box\,[A]}\ \mathsf{id}$$

$$\frac{\Gamma, \chi\,[A], B \vdash C}{\Gamma, (\chi \wedge B)\,[A] \vdash C}\ \wedge L_1 \qquad \frac{\Gamma \vdash \chi\,[A] \quad \Gamma \vdash B}{\Gamma \vdash (\chi \wedge B)\,[A]}\ \wedge R_1$$

$$\frac{\Gamma, B, \chi\,[A] \vdash C}{\Gamma, (B \wedge \chi)\,[A] \vdash C}\ \wedge L_2 \qquad \frac{\Gamma \vdash B \quad \Gamma \vdash \chi\,[A]}{\Gamma \vdash (B \wedge \chi)\,[A]}\ \wedge R_2$$

$$\frac{\Gamma, \chi\,[A] \vdash C \quad \Gamma, B \vdash C}{\Gamma, (\chi \vee B)\,[A] \vdash C}\ \vee L_1 \qquad \frac{\Gamma, \lambda\,[C] \vdash \chi\,[A]}{\Gamma, \lambda\,[C] \vdash (\chi \vee B)\,[A]}\ \vee R_1$$

$$\frac{\Gamma, B \vdash C \quad \Gamma, \chi\,[A] \vdash C}{\Gamma, (B \vee \chi)\,[A] \vdash C}\ \vee L_2 \qquad \frac{\Gamma, \lambda\,[C] \vdash \chi\,[A]}{\Gamma, \lambda\,[C] \vdash (B \vee \chi)\,[A]}\ \vee R_2$$

$$\frac{\Gamma, \lambda\,[D] \vdash \chi\,[A] \quad \Gamma, B \vdash C}{\Gamma, \lambda\,[D], (\chi \Rightarrow B)\,[A] \vdash C}\ \Rightarrow L_1, \qquad \frac{\Gamma \vdash \chi\,[A] \quad \Gamma, B \vdash \rho\,[C]}{\Gamma, (\chi \Rightarrow B)\,[A] \vdash \rho\,[C]}\ \Rightarrow L_1 \vdash \qquad \frac{\Gamma, \chi\,[A] \vdash B}{\Gamma \vdash (\chi \Rightarrow B)\,[A]}\ \Rightarrow R_1$$

$$\frac{\Gamma \vdash B \quad \Gamma, \lambda\,[D], \chi\,[A] \vdash C}{\Gamma, \lambda\,[D], (B \Rightarrow \chi)\,[A] \vdash C}\ \Rightarrow L_2, \qquad \frac{\Gamma \vdash B \quad \Gamma, \chi\,[A] \vdash \rho\,[C]}{\Gamma, (B \Rightarrow \chi)\,[A] \vdash \rho\,[C]}\ \Rightarrow L_2 \vdash \qquad \frac{\Gamma, B \vdash \chi\,[B]}{\Gamma \vdash (B \Rightarrow \chi)\,[A]}\ \Rightarrow R_2$$

Figure 5: Rules of ISL_2^0

The $\Rightarrow L_1 \vdash$ rule is not included in the system, it is only shown for explanatory purposes.

$$\frac{\Gamma, \chi\,[A] \vdash_\sigma C}{\Gamma, (\exists x.\chi)\,[A] \vdash_\sigma C}\ \exists L \qquad (x \in \mathsf{dom}(\sigma))\ \frac{\Gamma, \lambda\,[B] \vdash_\sigma \chi\,[A]\,\{\sigma\}}{\Gamma, \lambda\,[B] \vdash_\sigma (\exists x.\chi)\,[A]}\ \exists R$$

$$(x \in \mathsf{dom}(\sigma))\ \frac{\Gamma, (\forall x.\chi)\,\{A\}, \chi\,[A]\,\{\sigma\} \vdash_\sigma C}{\Gamma, (\forall x.\chi)\,[A] \vdash_\sigma C}\ \forall L \qquad \frac{\Gamma \vdash_\sigma \chi\,[A]}{\Gamma \vdash_\sigma (\forall x.\chi)\,[A]}\ \forall R$$

Figure 6: Quantifier rules of ISL_2^1

BACKWARD

$$\frac{\pi\{\chi[A] \vdash \xi[B]\}}{\pi\{\chi\{A\} \Rightarrow \xi\{B\}\}}\ \mathsf{lnp}$$

$$\frac{\top}{\Box[A] \vdash \Box[A]}\ \mathsf{lnpid} \qquad (A \neq B)\ \frac{A \Rightarrow B}{\Box[A] \vdash \Box[B]}\ \mathsf{unlnp}$$

$$\frac{C \Rightarrow (\chi \vdash \xi)}{(\chi \wedge C) \vdash \xi}\ \mathsf{lnplc1}\quad \frac{C \Rightarrow (\chi \vdash \xi)}{(C \wedge \chi) \vdash \xi}\ \mathsf{lnplc2}\quad \frac{(\lambda \vdash \chi) \wedge C}{\lambda \vdash (\chi \wedge C)}\ \mathsf{lnprc1}\quad \frac{C \wedge (\lambda \vdash \chi)}{\lambda \vdash (C \wedge \chi)}\ \mathsf{lnprc2}$$

$$\frac{(\chi \vdash \xi) \wedge (C \Rightarrow \xi)}{(\chi \vee C) \vdash \xi}\ \mathsf{lnpld1}\quad \frac{(C \Rightarrow \xi) \wedge (\chi \vdash \xi)}{(C \vee \chi) \vdash \xi}\ \mathsf{lnpld2}\quad \frac{(\lambda \vdash \chi) \vee C}{\lambda \vdash (\chi \vee C)}\ \mathsf{lnprd1}\quad \frac{C \vee (\lambda \vdash \chi)}{\lambda \vdash (C \vee \chi)}\ \mathsf{lnprd2}$$

$$\frac{C \wedge (\chi \vdash \rho)}{(C \Rightarrow \chi) \vdash \rho}\ \mathsf{lnpli2}\quad \frac{(\chi * \xi) \Rightarrow C}{\chi \vdash (\xi \Rightarrow C)}\ \mathsf{lnpri1}\quad \frac{C \Rightarrow (\chi \vdash \xi)}{\chi \vdash (C \Rightarrow \xi)}\ \mathsf{lnpri2}$$

FORWARD

$$\frac{\eta\{\chi[A] * \xi[B]\}}{\eta\{\chi\{A\} \wedge \xi\{B\}\}}\ \mathsf{lnn}$$

$$\frac{A}{\Box[A] * \Box[A]}\ \mathsf{lnnid} \qquad (A \neq B)\ \frac{A \wedge B}{\Box[A] * \Box[B]}\ \mathsf{unlnn}$$

$$\frac{(\chi * \xi) \wedge C}{\chi * (\xi \wedge C)}\ \mathsf{lnnc1}\quad \frac{C \wedge (\chi * \xi)}{\chi * (C \wedge \xi)}\ \mathsf{lnnc2}$$

$$\frac{(\phi * \chi) \vee C}{\phi * (\chi \vee C)}\ \mathsf{lnnd1}\quad \frac{C \vee (\phi * \chi)}{\phi * (C \vee \chi)}\ \mathsf{lnnd2}$$

$$\frac{(\phi \vdash \chi) \Rightarrow C}{\phi * (\chi \Rightarrow C)}\ \mathsf{lnni1}\quad \frac{C \Rightarrow (\phi * \chi)}{\phi * (C \Rightarrow \chi)}\ \mathsf{lnni2}$$

$$\frac{\xi * \chi}{\chi * \xi}\ \mathsf{lnncomm}$$

Figure 7: Rules of DISL^0

UNITS

$$\langle \circ, \dagger \rangle \in \{\langle \wedge, \top \rangle, \langle \vee, \bot \rangle, \langle \Rightarrow, \top \rangle\} \; \frac{A}{\dagger \circ A}\,\mathsf{neul} \qquad \langle \circ, \dagger \rangle \in \{\langle \wedge, \top \rangle, \langle \vee, \bot \rangle\} \; \frac{A}{A \circ \dagger}\,\mathsf{neur}$$

$$\langle \circ, \dagger \rangle \in \{\langle \wedge, \bot \rangle, \langle \vee, \top \rangle\} \; \frac{\dagger}{\dagger \circ A}\,\mathsf{absl} \qquad \langle \circ, \dagger \rangle \in \{\langle \wedge, \bot \rangle, \langle \vee, \top \rangle, \langle \Rightarrow, \top \rangle\} \; \frac{\dagger}{A \circ \dagger}\,\mathsf{absr} \qquad \frac{\top}{\bot \Rightarrow A}\;\mathsf{efq}$$

RESOURCES

$$\frac{\eta\{A \wedge A\}}{\eta\{A\}}\;\mathsf{conn} \qquad\qquad \frac{\eta\{\top\}}{\eta\{A\}}\;\mathsf{wkn}$$

Figure 7: Rules of DISL^0