Bachelor Thesis

Game Logic: A Proof Transformation from Gentzen to Hilbert

Author: S.J. van Schagen (s4070887)
Supervisor: Helle Hvid Hansen
Second supervisor: Tijs van der Storm

August 3, 2022

Abstract

Game logic is a modal logic, introduced by Parikh in 1985, that allows to reason about determined 2-player games. It introduces the modality $\langle\gamma\rangle\varphi$ which means that player 1, often called Angel, has a strategy in the game γ to ensure an outcome in which the formula φ holds.

Parikh also came up with a proof system for game logic, a Hilbert-style system Par, but could not yet prove its completeness. In 2019, Enqvist et al. proved completeness of game logic through a sequence of proof transformations. To do so, they introduced three proof systems, of which the last one is a cut-free sequent calculus named G. The last transformation step, G to Par, was done via an intermediate Hilbert-style system called Par_{Full}.

In 2021, the transformation from CloG to G was implemented by Worthington. In 2022, an automated theorem prover (ATP) for CloG proofs was implemented by Meerholz. An ATP for CloG proofs, together with a transformation tool for CloG to G and for G to Par, proofs can be found in the CloG system and transformed into G and Par. Thus, an ATP for CloG, G and Par proofs will be effective.

This thesis concerns the implementation of the transformation from G to Par_{Full}. Using a small number of propositional tautologies that are needed to derive the rules of G in Par_{Full}, the transformation from G to Par_{Full} is defined. The implementation of the tool was done in Rascal and can successfully transform G proofs into Par_{Full} proofs. Proofs in Par_{Full} are much longer than proofs in G, and they are also not always the most simplified proof. Extending the tool by transforming G proofs into Par proofs will complete the transformation step from G to Par.

Contents

Acknowledgments

First of all, I want to thank my supervisor Helle Hvid Hansen for providing me with this project, guiding me through the process and willing to meet with me every week.
I also want to thank Tijs van der Storm for being my second supervisor.
Finally, I thank Chris Worthington for providing me with the code of his CloG-to-G transformation tool and helping me in the beginning with understanding some of the concepts of game logic.

1 Introduction

Game logic is a monotone modal logic, introduced by Parikh in 1985 [16], that reasons about ensuring certain outcomes in determined 2-player games. It does so by introducing the modality $\langle\gamma\rangle\varphi$ which means that a player has the strategy in the game γ to ensure an outcome in which φ holds. Parikh also proposed a Hilbert-style proof system [16], which is called Par in [3]. Par was easily proven to be sound but Parikh could not yet prove its completeness. Completeness of a proof system means that a proof system is able to find a proof for every valid formula, that is if $\varphi_1, ..., \varphi_n \vDash \psi$ then $\varphi_1, ..., \varphi_n \vdash \psi$. Soundness of a proof system is when all provable formulas are in fact valid, that is if $\varphi_1, ..., \varphi_n \vdash \psi$ then $\varphi_1, ..., \varphi_n \vDash \psi$. The notations $\vdash$ and $\vDash$ refer to syntactic and semantic consequence, respectively.

Recently, in [3], completeness of game logic has been proven through a sequence of proof transformations, illustrated in (1). Starting from the complete proof system Clo [1], a proof system for modal μ-calculus [21],[12], three intermediate proof systems were introduced in order to transform a Clo proof into a Par proof. The first proof system is CloM, another proof system for modal μ-calculus. The second and third, CloG and G, are proof systems for game logic. Clo, CloM and CloG are all cyclic proof systems. Proofs are often tree structures where each branch results in an axiom (a leaf). Cyclic proofs can produce infinite trees that are still valid proofs [9].

The transformation from Clo to CloM was defined via a validity-preserving translation, thereby proving completeness of CloM. In a similar fashion, the transformations from CloM to CloG and from CloG to G also proved completeness of CloG and G. Finally, the transformation from G to Par went via an intermediate Hilbert system Par_{Full}. These proof systems were all proven to be sound and complete, which also implied that Par is complete [3]. Hence, proof transformations can also be used for proving the soundness and completeness of proof systems.

$$\text{Par} \leftarrow \text{G} \leftarrow \text{CloG} \leftarrow \text{CloM} \leftarrow \text{Clo} \tag{1}$$

In [25], the proof transformation from CloG to G has been implemented. At the time of writing of this thesis, an automated theorem prover (ATP) for CloG proofs is being worked on. Since there exists efficient proof search for CloG proofs, with an ATP for CloG proofs we now also have an efficient proof search strategy for G proofs. A next step would be to implement the transformation from G to Par to also create an efficient proof search strategy for Par proofs. With all three tools combined, proofs can first be obtained in the CloG system and then transformed into G and Par to also find proofs effectively in these systems. This thesis restricts itself to the proof transformation step from G to the intermediate Hilbert system Par_{Full}.

This has led to the following research question:

How can the proof transformation from the Gentzen system G to the Hilbert system Par_{Full} be implemented?

To answer this question, the following subquestions have been defined:

- What are the differences between a Gentzen system and a Hilbert system?
- What are the differences between G and Par_{Full}?
- How can the rules of G be derived in Par_{Full}?
- What are the requirements of the tool?
- How can this proof transformation be implemented?

With this tool, the differences between proofs in G and Par_{Full} can be examined as well. For example, it might be of interest to see if proofs in Par_{Full} are more intuitive to read than proofs in G. Additionally, this tool and its implementation could spark the interest of researchers of game logic specifically but also those in the scientific community who are concerned with proof theory of modal logics.

2 Game Logic history and principles

2.1 Background

Proof theory is a formalization of proofs by viewing them as mathematical objects, making it possible to analyze and prove logical facts about them. David Hilbert modernized this concept by introducing a formal deductive system consisting of a set of formuli called *axioms*, and a set of *inference rules* that take in premises in order to draw conclusions from them [10]. Hilbert came up with this system to provide a formalization of all mathematics. This proof system was characterized by a relative large number of axioms and small number of inference rules. Such proof systems were later referred to as *Hilbert systems* or *Hilbert calculi*. Shortly after, however, Kurt Gödel proved it to be impossible that all of mathematics could be formalized by a single logic system [7]. An improvement upon Hilbert calculi was Gerhard Gentzen's *natural deduction* and *sequent calculus*, which propose a more natural way of deduction that is closer to human reasoning [6].

Modal logic is an extension of Classical Propositional Logic obtained by introducing two unary operators, $\Box$ (box) and $\Diamond$ (diamond), that express statements about necessity and possibility, respectively. The basic principles of modal logic date back to Aristotle, but it did not receive much acknowledgment during the emergence of mathematical logic in the 19th century [20]. Saul Kripke modernized modal logic by introducing Kripke semantics in 1959 [13]. This concept was further generalized into neighbourhood semantics in [21],[15], which paved the way for Parikh to introduce game logic [16],[18],[3].

2.2 Syntax and Semantics of Game Logic

Game logic is a modal fixpoint logic to reason about determined 2-player games. Before discussing the syntax and semantics of game logic, it is important to discuss fixed points. Suppose a set S and a monotone mapping f from the power set of S to the power set of S, denoted by $f : \wp(S) \to \wp(S)$. If we have $f(M) = M$ for some $M \subseteq S$, then M is a *fixpoint* of f. Additionally, M is a *pre-fixpoint* of S if $f(M) \subseteq M$, and M is a *post-fixpoint* of S if $M \subseteq f(M)$. According to the Knaster-Tarski theorem [23], we have a least fixed point (lfp) and a greatest fixed point (gfp) for a set S in an ordering of $(\wp(S), \subseteq)$. The lfp and gfp of a monotone mapping f can be defined as follows:

$$lfp(f) = \bigcap\{M \subseteq S : f(M) = M\} = \bigcap\{M \subseteq S : f(M) \subseteq M\}$$
$$gfp(f) = \bigcup\{M \subseteq S : f(M) = M\} = \bigcup\{M \subseteq S : M \subseteq f(M)\}$$

$\mathrm{lfp}(f)$ and $\mathrm{gfp}(f)$ can be seen as the intersection and union of all sets where M is a fixpoint, respectively. Alternatively, $\mathrm{lfp}(f)$ is the intersection of all pre-fixpoints and $\mathrm{gfp}(f)$ is the union of all post-fixpoints. In other words, M is a least fixed point of S in a monotone mapping f, if for all fixpoints M' of S, it holds that $M \subseteq M'$. Similarly, M is a greatest fixed point of S in a monotone mapping f, if for all fixpoints M' of S, it holds that $M' \subseteq M$. $\mathrm{lfp}(f)$ and $\mathrm{gfp}(f)$ of a set S can be found by applying f repeatedly on the empty set and the set S, respectively.

Throughout this thesis, the terms *Angel* and *Demon* are used, as introduced by Pauly [17], as names for the player and its opponent, respectively. To express strategic ability, $\langle\gamma\rangle\varphi$ means that Angel has the strategy in the game γ to ensure an outcome in which φ holds. Given a set of atomic games Γ and a set of atomic propositions Φ, the syntax of game logic can be defined as follows.

Definition 2.1 (Grammar of $\mathcal{L}_{\mathrm{Par}}, \mathcal{L}_{\mathrm{NF}}$ and $\mathcal{L}_{\mathrm{Full}}$)**.**

$$\begin{array}{lcl}
\mathcal{L}_{\mathrm{Par}} \ni \varphi & ::= & p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\gamma\rangle\varphi, \gamma \in \mathcal{G}_{\mathrm{Par}} \\
\mathcal{G}_{\mathrm{Par}} \ni \gamma & ::= & g \mid \varphi? \mid \gamma;\gamma \mid \gamma \sqcup \gamma \mid \gamma^* \mid \gamma^d, \varphi \in \mathcal{L}_{\mathrm{Par}} \\
\mathcal{L}_{\mathrm{NF}} \ni \varphi & ::= & p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\gamma\rangle\varphi, \gamma \in \mathcal{G}_{\mathrm{NF}} \\
\mathcal{G}_{\mathrm{NF}} \ni \gamma & ::= & g \mid g^d \mid \gamma;\gamma \mid \gamma \sqcup \gamma \mid \gamma \sqcap \gamma \mid \gamma^* \mid \gamma^\times \mid \varphi? \mid \varphi!, \varphi \in \mathcal{L}_{\mathrm{NF}} \\
\mathcal{L}_{\mathrm{Full}} \ni \varphi & ::= & p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\gamma\rangle\varphi, \gamma \in \mathcal{G}_{\mathrm{Full}} \\
\mathcal{G}_{\mathrm{Full}} \ni \gamma & ::= & g \mid \gamma;\gamma \mid \gamma \sqcup \gamma \mid \gamma \sqcap \gamma \mid \gamma^* \mid \gamma^\times \mid \gamma^d \mid \varphi? \ , \varphi \in \mathcal{L}_{\mathrm{Full}}
\end{array}$$

where $p \in \Phi_0$ and $g \in \Gamma_0$.

In Parikh's language, referred to as $\mathcal{L}_{\text{Par}}$ in Definition 2.1, a formula φ can be an atomic proposition p where $p \in \Phi_0$. In other cases it can be the negation of a formula φ or a disjunction of two formulas. The strategic ability of Angel to ensure an outcome where φ holds when a game γ is played is denoted by $\langle\gamma\rangle\varphi$.
A game γ can be an atomic game g where $g \in \Gamma_0$. The *angelic test* $\varphi?$ refers to the game that if φ holds then play continues. If φ does not hold, then Angel loses the game immediately. The *sequential composition* operator denotes sequential games, so $\gamma;\delta$ means γ is played first and subsequently δ. *Angelic choice*, denoted by $\gamma \sqcup \delta$, describes the game where Angel decides if γ or δ is played. *Angelic iteration* γ^* refers to the game where Angel can decide whether γ is played zero times, one time, or multiple times. After every iteration of γ Angel can decide if γ is played again. The *dual operator* γ^d refers to the game where the roles of Angel and Demon are interchanged.

$\mathcal{L}_{\text{Par}}$ only allows angelic game constructors and free occurrence of dual and negation, while the normal form language $\mathcal{L}_{\text{NF}}$ also incorporates demonic game constructors, see Definition 2.2, and restricts the use of dual and negation to atoms. $\mathcal{L}_{\text{Full}}$ extends $\mathcal{L}_{\text{Par}}$ by allowing all connectives and game constructors, and freely placed duals and negations. Specific notation is used for demonic game constructors. These *demonic operations* are defined as follows:

Definition 2.2 (Demonic operators)**.**

$$\begin{aligned} \gamma \sqcap \delta &= (\gamma^d \sqcup \gamma^d)^d \\ \gamma^\times &= ((\gamma^d)^*)^d \\ \psi! &= ((\neg\psi)?)^d \end{aligned}$$

Since $(\gamma^d \sqcup \delta^d)^d$ reads as a *demonic choice*, we can interpret $\langle\gamma \sqcap \delta\rangle\varphi$ as Angel having the ability to ensure φ in both γ and δ. Similarly, $\langle\gamma^\times\rangle\varphi$ means that Angel can ensure that φ holds, regardless the number of times γ is played. In a *demonic test* $\psi!$, Angel instantly wins if ψ evaluates to true, meaning that $\langle\psi!\rangle\varphi$ is true if at least one of both formulas hold.

The angelic and demonic iteration formulas $\langle\gamma^*\rangle\varphi$ and $\langle\gamma^\times\rangle\varphi$ are *fixpoint formulas*. The least, greatest, and respectively all fixpoint formulas are defined as follows:

Definition 2.3 (Least, greatest and all fixpoint formulas within Game Logic)**.**

$$\begin{aligned} F^* &:= \{\langle\gamma^*\rangle\varphi \mid \gamma \in \mathcal{G}_{NF}, \varphi \in \mathcal{L}_{NF}\} \\ F^\times &:= \{\langle\gamma^\times\rangle\varphi \mid \gamma \in \mathcal{G}_{NF}, \varphi \in \mathcal{L}_{NF}\} \\ F &:= F^* \cup F^\times \end{aligned}$$

The semantics of game logic are defined via game models where a game model $\mathbb{S}$ is defined as

Definition 2.4 (Semantics of game logic)**.**

$$\mathbb{S} = (S, E, V)$$

where S is a set of states, E a set of effectivity functions $\{E_g \mid g \in \Gamma_0\}$ for each atomic game in Γ_0, and finally, V is a *valuation* function $V : \Phi_0 \to \mathcal{P}(S)$ that maps each atomic proposition p in Φ_0 to the set of all states where p is true. The meaning of a formula φ is defined as a set $[\![\varphi]\!]^{\mathbb{S}} \subseteq S$, such that if a state $s \in [\![\varphi]\!]^{\mathbb{S}}$, then φ is true in s. $[\![\varphi]\!]^{\mathbb{S}}$ is then inductively defined on φ as follows:

Definition 2.5 (Semantic valuation of a formula)**.**

$$\begin{array}{rcl}
[\![p]\!]^{\mathbb{S}} & := & V(p) \\
[\![\neg\varphi]\!]^{\mathbb{S}} & := & S \backslash [\![\varphi]\!]^{\mathbb{S}} \\
[\![\varphi \vee \psi]\!]^{\mathbb{S}} & := & [\![\varphi]\!]^{\mathbb{S}} \cup [\![\psi]\!]^{\mathbb{S}} \\
[\![\varphi \wedge \psi]\!]^{\mathbb{S}} & := & [\![\varphi]\!]^{\mathbb{S}} \cap [\![\psi]\!]^{\mathbb{S}} \\
[\![\langle\gamma\rangle\varphi]\!]^{\mathbb{S}} & := & E_\gamma([\![\varphi]\!]^{\mathbb{S}}) \\
E_g(X) & := & E(g)(X) \\
E_{\gamma^d}(X) & := & S \backslash E_\gamma(S \backslash X) \\
E_{\gamma;\delta}(X) & := & E_\gamma(E_\delta(X)) \\
E_{\gamma \sqcup \delta}(X) & := & E_\gamma(X) \cup (E_\delta(X) \\
E_{\gamma \sqcap \delta}(X) & := & E_\gamma(X) \cap (E_\delta(X) \\
E_{\gamma^*}(X) & := & lfpY.X \cup E_\gamma(Y) \\
E_{\gamma^\times}(X) & := & gfpY.X \cap E_\gamma(Y) \\
E_{\varphi?}(X) & := & [\![\varphi]\!]^{\mathbb{S}} \cap X \\
E_{\varphi!}(X) & := & [\![\varphi]\!]^{\mathbb{S}} \cup X
\end{array}$$

where $\cap$, $\cup$ and $\backslash$ are the usual set operation symbols for intersection, union and difference, respectively. The functions lfp and gfp are the least and greatest fixed point formulae, respectively.

Now follows an example of a game model with some formulas that are true or false at the different states of the model. Consider the following game model:

Example 2.1 (Example game model)**.**

$$\begin{array}{rcl}
S & = & \{s_1, s_2\} \\
\\
E_a(\varnothing) & = & \varnothing \\
E_a(\{s_1\}) & = & \varnothing \\
E_a(\{s_2\}) & = & \{s_2\} \\
E_a(\{s_1, s_2\}) & = & \{s_1, s_2\} \\
E_b(\varnothing) & = & \varnothing \\
E_b(\{s_1\}) & = & \{s_1, s_2\} \\
E_b(\{s_2\}) & = & \varnothing \\
E_b(\{s_1, s_2\}) & = & \{s_1, s_2\} \\
\\
V(p) & = & \{s_2\} \\
V(q) & = & \{s_1, s_2\}
\end{array}$$

where S is the set of all states containing states s_1 and s_2, E the set of effectivity functions for the games a and b, and V the valuation function mapping the atomic propositions p and q to truth values at all states in S.

Now consider the following formula φ_0:

Example 2.2 (Example formula φ_0)**.**

$$\varphi_0 = \langle b \sqcap (a; q?)\rangle \neg p \vee \langle (p!; a); b\rangle p$$

and its valuation:

$$
\begin{aligned}
[\![\varphi_0]\!]^{\mathbb{S}} &= [\![\langle b \sqcap (a;q?)\rangle \neg p \vee \langle (p!;a);b\rangle p]\!]^{\mathbb{S}} \\
&= [\![\langle b \sqcap (a;q?)\rangle \neg p]\!]^{\mathbb{S}} \cup [\![\langle (p!;a);b\rangle p]\!]^{\mathbb{S}} \\
&= E_{b\sqcap(a;q?)}([\![\neg p]\!]^{\mathbb{S}}) \cup E_{(p!;a);b}([\![p]\!]^{\mathbb{S}}) \\
&= (E_b(S \backslash [\![p]\!]^{\mathbb{S}}) \cap E_{a;q?}(S \backslash [\![p]\!]^{\mathbb{S}})) \cup (E_{p!;a}(E_b(\{s_2\}))) \\
&= (E_b(\{s_1\}) \cap E_a(E_{q?}(\{s_1\}))) \cup (E_{p!}(E_a(\varnothing)) \\
&= (\{s_1, s_2\} \cap E_a([\![q]\!]^{\mathbb{S}} \cap \{s_1\})) \cup ([\![p]\!]^{\mathbb{S}} \cup \varnothing) \\
&= (\{s_1, s_2\} \cap E_a(\{s_1, s_2\} \cap \{s_1\})) \cup (\{s_2\} \cup \varnothing) \\
&= (\{s_1, s_2\} \cap E_a(\{s_1\})) \cup \{s_2\} \\
&= (\{s_1, s_2\} \cap \varnothing) \cup \{s_2\} \\
&= \varnothing \cup \{s_2\} \\
&= \varnothing
\end{aligned}
$$

We have determined that $[\![\varphi_0]\!]^{\mathbb{S}}$ valuates to $\varnothing$, meaning that φ_0 is false in all states of S. Therefore, φ_0 is not satisfiable in S.

Now consider the formula φ_1 where the demonic choice in the left disjunct is replaced by an angelic choice:

Example 2.3 (Example formula φ_1)**.**

$$\varphi_1 = \langle b \sqcup (a;q?)\rangle \neg p \vee \langle (p!;a);b\rangle p$$

and its valuation:

$$
\begin{aligned}
[\![\varphi_1]\!]^{\mathbb{S}} &= [\![\langle b \sqcup (a;q?)\rangle \neg p \vee \langle (p!;a);b\rangle p]\!]^{\mathbb{S}} \\
&= [\![\langle b \sqcup (a;q?)\rangle \neg p]\!]^{\mathbb{S}} \cup [\![\langle (p!;a);b\rangle p]\!]^{\mathbb{S}} \\
&= E_{b\sqcup(a;q?)}([\![\neg p]\!]^{\mathbb{S}}) \cup E_{(p!;a);b}([\![p]\!]^{\mathbb{S}}) \\
&= (E_b(S \backslash [\![p]\!]^{\mathbb{S}}) \cup E_{a;q?}(S \backslash [\![p]\!]^{\mathbb{S}})) \cup (E_{p!;a}(E_b(\{s_2\}))) \\
&= (E_b(\{s_1\}) \cup E_a(E_{q?}(\{s_1\}))) \cup (E_{p!}(E_a(\varnothing)) \\
&= (\{s_1, s_2\} \cup E_a([\![q]\!]^{\mathbb{S}} \cap \{s_1\})) \cup ([\![p]\!]^{\mathbb{S}} \cup \varnothing) \\
&= (\{s_1, s_2\} \cup E_a(\{s_1, s_2\} \cap \{s_1\})) \cup (\{s_2\} \cup \varnothing) \\
&= (\{s_1, s_2\} \cup E_a(\{s_1\})) \cup \{s_2\} \\
&= (\{s_1, s_2\} \cup \varnothing) \cup \{s_2\} \\
&= \{s_1, s_2\} \cup \{s_2\} \\
&= \{s_1, s_2\}
\end{aligned}
$$

In this case, φ_1 valuates to $\{s_1, s_2\}$, meaning that φ_1 is true in all states of S.

2.3 Hilbert system and sequent calculus

2.3.1 Hilbert system

A Hilbert system is an axiomatic system for formal deduction and is characterized by having many axioms and few inference rules. A proof in a Hilbert system consists of a sequence of statements, where a statement is either an axiom or a statement based upon earlier statements by a rule of inference. Suppose we have a set of formulas Γ that a certain proof system has as axioms, then the notation $\Gamma \vdash \varphi$ means that φ is derivable from Γ. In other words, the proof system can deduce the formula φ from its axioms and inference rules. A Hilbert system has a finite set of axiom schemes, but by uniform substitution these axiom schemes can produce an infinite set of axiom instances.

Consider the following simple Hilbert system H consisting of two axiom schemes (A_1 and A_2) and modus ponens (MP) as the only inference rule:

Example 2.4 (Hilbert system H)**.**

$$\begin{aligned} \mathrm{A}_1: &\quad A \to (B \to A) \\ \mathrm{A}_2: &\quad (A \to (B \to C)) \to ((A \to B) \to (A \to C)) \end{aligned}$$

$$\frac{A, A \to B}{B}\ \mathrm{MP}$$

Proofs in a Hilbert system can become tedious and cumbersome, as can be illustrated by the proof of a simple statement as $A \to A$. In the justifications of a proof, an axiom is followed by a substitution of variables in the form of [substitution/variable], and a rule is given with line references. The proof of $A \to A$ can be derived as follows:

Example 2.5 (Proof of $A \to A$ in H)**.**

1.	$(A \to ((A \to A) \to A))$	$\mathrm{A}_1 : [A/A], [(A \to A)/B]$
2.	$((A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A)))$	$\mathrm{A}_2 : [A/A], [(A \to A)/B], [A/C]$
3.	$((A \to (A \to A)) \to (A \to A)))$	MP: $1, 2$
4.	$(A \to (A \to A))$	$\mathrm{A}_1 : [A/A], [A/B]$
5.	$A \to A$	MP: $3, 4$

2.3.2 Sequent calculus

Sequent calculus is a Gentzen-style proof calculus and is characterized by mostly relying on rules of inference instead of axioms. Therefore, a sequent calculus only has a few axioms, possibly zero, and many inference rules. A sequent is of the form $\Gamma \vdash \Delta$ where Γ is a set of formulas called the *antecedent*, and Δ is a set of formulas called the *consequent*. The formulas in the antecedent are conjunctively related while the consequent should be read disjunctively, such that a sequent can also be illustrated as $A_1 \wedge ... \wedge A_n \vdash B_1 \vee ... \vee B_m$. Both the antecedent and the consequent may contain zero or more formulas. If the antecedent is empty, then the consequent is a tautology. However, if the consequent is empty, then the antecedent is a contradiction. The sequent as described until now is a two-sided sequent with sets of formulas on both sides of the turnstile. A one-sided sequent is of the form $\vdash \Gamma$ where the formulas in Γ are to be read disjunctively.

To illustrate how a sequent calculus and a proof in such a system looks like, consider the sequent calculus LK introduced by Gentzen in [6]. A subset of the axioms and rules of LK is given in Figure 2. The rules are either applied to the left part of a sequent (denoted by L in the rule name) or to the right part of a sequent (denoted by R in the rule name). To give some intuition into the logic of these rules, consider the rules in the left column of Figure 2. Firstly, LK has one axiom, I, which reflects the rule of identity. The rule $\wedge L_1$ states that if some Δ can be proven from some Γ and a formula A, then Δ can also be proven from Γ and $A \wedge B$ for any arbitrary B. Similarly, $\wedge L_2$ applies the same logic but then with B in the premise. The rules $\vee R_1$ and $\vee R_2$ apply the same the principle to the right side of the sequent. These rules, along with the rules in the middle column, are *logical rules* that perform some logical operation on the formulas in the sequent. LK also has a set of *structural rules*, see the right column in Figure 2 for a subset of these, that restructure the formulas in a sequent.

$$\frac{}{A \vdash A}\, I \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\, \neg L \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\, CL$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\, \wedge L_1 \qquad \frac{\Gamma, A \vdash \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \vee B \vdash \Delta, \Pi}\, \vee L \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\, CR$$

$$\frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\, \wedge L_2 \qquad \frac{\Gamma \vdash A, \Delta \qquad \Sigma \vdash B, \Pi}{\Gamma, \Sigma \vdash A \wedge B, \Delta, \Pi}\, \wedge R \qquad \frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}\, PL$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta}\, \vee R_1 \qquad \frac{\Gamma, A \vdash \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \to B \vdash \Delta, \Pi}\, \to L \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}\, PR$$

$$\frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}\, \vee R_2 \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\, \to R$$

Figure 2: Subset of LK rules.

With the axiom and rules in Figure 2, the sequent $\vdash ((A \to (B \vee C)) \to (((B \to \neg A) \wedge \neg C) \to \neg A))$ can be proven, see Figure 3. The proof has a tree structure with the sequent to be proven at the root of the tree. Every sequent is inferred from the next sequent in the tree according to an inference rule or axiom. Whenever a rule is applied that has two sequents in its premise, the proof splits into two branches. All branches eventually result in the I axiom.

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\, I}{\vdash \neg A, A}\, \neg R}{\vdash A, \neg A}\, PR \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\, I \quad \dfrac{}{C \vdash C}\, I}{B \vee C \vdash B, C}\, \vee L}{B \vee C \vdash C, B}\, PR}{B \vee C, \neg C \vdash B}\, \neg L \qquad \dfrac{}{\neg A \vdash \neg A}\, I}{(B \vee C), \neg C, (B \to \neg A) \vdash \neg A}\, \to L}{(B \vee C), \neg C, ((B \to \neg A) \wedge \neg C) \vdash \neg A}\, \wedge L_1}{(B \vee C), ((B \to \neg A) \wedge \neg C), \neg C \vdash \neg A}\, PL}{(B \vee C), ((B \to \neg A) \wedge \neg C), ((B \to \neg A) \wedge \neg C) \vdash \neg A}\, \wedge L_2}{((B \to \neg A) \wedge \neg C), (B \vee C) \vdash \neg A}\, CL}{((B \to \neg A) \wedge \neg C), (A \to (B \vee C)) \vdash \neg A, \neg A}\, PL}{((B \to \neg A) \wedge \neg C), (A \to (B \vee C)) \vdash \neg A}\, CR}{(A \to (B \vee C)), ((B \to \neg A) \wedge \neg C) \vdash \neg A}\, PL}{(A \to (B \vee C)) \vdash (((B \to \neg A) \wedge \neg C) \to \neg A)}\, \to R}{\vdash ((A \to (B \vee C)) \to (((B \to \neg A) \wedge \neg C) \to \neg A))}\, \to R$$

Figure 3: Example proof in LK system.

3 Related Work

A proof system for a logic is used to construct proofs for logical statements. Such proof systems consist of a *language* containing all formulas that the system can accept, a set of sentences assumed to be valid called *axioms*, and finally, a set of *inference rules* that derive conclusions from premises [24]. Multiple proof systems can exist for a logic, such that it might be convenient to transform from one proof system to the other. A proof transformation transforms proofs of one proof system into another, and they can play a role in the optimization and simplification of proof systems for multiple types of logic systems [14],[19]. Proof transformations can also provide proofs that are more natural to human reasoning, thus making them easier to understand and work with [14],[19].

One of the most used techniques in proof transformations is cut-elimination, which is based on Gentzen's cut-elimination theorem [6]. This theorem states that for any proof in a sequent calculus that uses the cut rule, there also exists a cut-free proof. In automated deduction, cut-elimination is normally not useful since proofs generated by ATPs are generally cut-free. On the other hand, introducing cuts in these cases seems useful as it may decrease the proof size significantly and fasten the proof search [2]. However, in the field of proof transformation, cut-elimination can be used to transform non-elementary proofs into elementary ones. For example, converting number-theoretic proofs into proofs by induction [22].

Another reason to use proof transformations is to generate proofs that are suitable for human understanding. Consider a student of mathematics or logic that wants to study proofs. A proof system in which proofs are difficult to understand for humans, transforming such proofs into a system that is more humanly readable might be useful. Many ATPs use resolution because the search space is homogeneous and the operations to determine are fairly straightforward. However, this homogeneous search space can usually only be achieved by normalizing the theorem, resulting in a normal form that is quite different from the original theorem. In contrast, natural deduction systems are more understandable for humans while also allowing interaction when constructing proofs [5]. In [5], a system χ was designed that allows proofs to be constructed by resolution and to be transformed afterwards into a proof by natural deduction.

4 Proof systems: G and Par

4.1 Proof system G

In this section, the axioms and inference rules of G are discussed. G is a cut-free sequent system where sequents are one-sided, such that a sequent is a finite set of $\mathcal{L}_{NF}$-formulas that should be read disjunctively. Such a sequent is of the form $A_1, ..., A_n$ where the formulas in A are separated by commas. Axioms and inference rules in G are given in the form where a line separates the *premise* from the *conclusion* and where the rule is placed to the right of the line. The structure of a rule in G can be found in Figure 4, where a rule can be seen as $A_1, A_2, ..., A_n \vdash B_1, B_2, ..., B_m$.

$$\frac{A_1, A_2, ..., A_n}{B_1, B_2, ..., B_m}\ \text{rule}$$

Figure 4: Structure of a G rule.

In most cases, the rule is applied to one formula of a sequent called the *active formula*. The rest of the formulas in the sequent are *side formulas*, denoted by Φ. A formula is proven in G if the formula and all of its subformulas are based on inference rules, and when all of the leaves are axioms.

4.1.1 Basic rules of G

The axiom and inference rules in Figure 5 correspond to those of monotone modal logic [8]. G has only one axiom, called Ax, which directly reflects the law of excluded middle, meaning that for every proposition φ it holds that either φ or its negation is true. The notation $\overline{\Phi}$ is the normal form of $\neg(\bigvee \Phi)$, where $\bigvee \Phi$ is a

disjunction of the formulas in Φ. The weak rule captures the idea of *weakening*, meaning that if φ is valid, then so is $\varphi \vee \psi$ for any ψ, such that a set of formulas Φ can be extended with any formula φ. Since sequents are read disjunctively, introducing $\varphi \vee \psi$ via the disjunction rule for any φ and ψ in a sequent is valid. For the introduction of $\wedge$, if there is a sequent with a set of formulas Φ and a formula φ, and another sequent with the same Φ and a formula ψ, then combining these two sequents into one results in a sequent with at least Φ, since Φ appears in both sequents. Additionally, in the one sequent we have φ that does not appear in the other, and conversely, the one sequent does not contain ψ while the other does. Combining φ and ψ results in a conjunction of the two formulas, resulting in a sequent $\Phi, \varphi \wedge \psi$.
Finally, the mod_m-rule states that for any φ and ψ in a hypothesis, a conclusion can be constructed where the angelic game g can be applied to φ and a dual game g^d to ψ.

$$\frac{}{\Phi, \overline{\Phi}}\,\text{Ax} \qquad \frac{\varphi, \psi}{\langle g\rangle\varphi, \langle g^d\rangle\psi}\,\text{mod}_m$$

$$\frac{\Phi}{\Phi, \varphi}\,\text{weak}$$

$$\frac{\Phi, \varphi, \psi}{\Phi, \varphi \vee \psi}\,\vee \qquad \frac{\Phi, \varphi \qquad \Phi, \psi}{\Phi, \varphi \wedge \psi}\,\wedge$$

Figure 5: Basic rules.

4.1.2 Game Operation Rules

In Figure 6, the rules are listed that capture the meaning of the game constructors as described in Definition 2.1. The * rule enables to infer $\langle\gamma^*\rangle\varphi$ from the unfolded formula $\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$. Semantically, this means that $\langle\gamma^*\rangle\varphi$ is a pre-fixpoint of the operation $\varphi \vee \langle\gamma\rangle X$. If Angel decides not to play γ, then φ should hold, otherwise γ is played and then $\langle\gamma^*\rangle\varphi$ should hold, starting the *angelic iteration* over again. The same principle applies to the $\times$ rule, except this is a demonic game where Angel has a strategy no matter what Demon decides to play. If Demon decides to play γ zero times, one time or multiple times, Angel needs to be able to ensure an outcome of φ regardless, hence the use of the conjunction between φ and $\langle\gamma\rangle\langle\gamma^\times\rangle\varphi$. In case of these dual situations, we speak of post-fixpoints. Angelic choice ($\sqcup$) describes the game where Angel decides whether to play γ or δ, such that a formula $\langle\gamma \sqcup \delta\rangle\varphi$ means that Angel has the strategic ability to ensure an outcome that satisfies φ in either γ or δ. Demonic choice ($\sqcap$), on the other hand, describes the game where Demon decides whether to play γ or δ, such that in a formula $\langle\gamma \sqcap \delta\rangle\varphi$ Angel has the strategic ability to ensure φ in both games. In case of an angelic test (?), Angel needs ψ and φ to be true in order not to lose, while in a demonic test (!) Demon loses if ψ is true, such that Angel needs at least ψ or φ to be true.

$$\frac{\Phi, \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi}{\Phi, \langle\gamma^*\rangle\varphi}\,* \qquad \frac{\Phi, \langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi}{\Phi, \langle\gamma \sqcup \delta\rangle\varphi}\,\sqcup \qquad \frac{\Phi, \psi \wedge \varphi}{\Phi, \langle\psi?\rangle\varphi}\,?$$

$$\frac{\Phi, \varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi}{\Phi, \langle\gamma^\times\rangle\varphi}\,\times \qquad \frac{\Phi, \langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi}{\Phi, \langle\gamma \sqcap \delta\rangle\varphi}\,\sqcap \qquad \frac{\Phi, \psi \vee \varphi}{\Phi, \langle\psi!\rangle\varphi}\,!$$

Figure 6: Game operation rules.

4.1.3 Deep inference rules

The deep inference rules of G in Figure 7 enable to operate on games and formulas that lie deep inside a formula. The notation $\psi(\varphi)$ describes some formula ψ, also called a context, that contains a uniquely occurring atomic proposition p, which is then substituted by φ. Similarly, the notation $\psi(\gamma)$ describes a context ψ that contains a uniquely occurring atomic game g, which is then substituted by γ.

$$\frac{\Phi, \psi(\gamma)}{\Phi, \psi(\chi!;\gamma)}\,\mathrm{Mon}_d^g \qquad \frac{\Phi, \psi(\varphi)}{\Phi, \psi(\langle\chi!\rangle\varphi)}\,\mathrm{Mon}_d^f \qquad \frac{\Phi, \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)}{\Phi, \psi(\langle\gamma;\delta\rangle\varphi)}\,;_d$$

Figure 7: Deep inference rules.

The Mon_d^f rule is then defined as follows: given $\psi(\varphi)$, φ can be substituted by $\langle\chi!\rangle\varphi$ within the context of ψ. For example, $\psi(\varphi)$ could be $\langle g \sqcap (\langle g^d; \varphi!\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)$, and applying the Mon_d^f rule to φ would then give $\langle g \sqcap (\langle g^d; (\langle\chi!\rangle\varphi!)\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)$. In Figure 8, this example can be found as it would look like in a G proof.

$$\frac{\langle g \sqcap (\langle g^d; \varphi!\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)}{\langle g \sqcap (\langle g^d; \langle\chi!\rangle\varphi!\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)}\,\mathrm{Mon}_d^f$$

Figure 8: Example of the Mon_d^f rule.

Similarly, Mon_d^g applies this principle to games, where a certain γ can be substituted for $\chi!;\gamma$. If we take the same ψ as before and now apply the Mon_d^g rule to g^d, we first substitute g^d for γ and then obtain the formula that can be seen in the conclusion in Figure 9.

$$\frac{\langle g \sqcap (\langle\gamma; \varphi!\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)}{\langle g \sqcap (\langle(\chi!;\gamma); \varphi!\rangle\neg q)\rangle p \wedge (\neg p \vee \langle h^\times\rangle q)}\,\mathrm{Mon}_d^g$$

Figure 9: Example of the Mon_d^g rule.

The deep composition rule $;_d$ enables to replace a formula of the form $\langle\gamma\rangle\langle\delta\rangle\varphi$ inside a context ψ for $\langle\gamma;\delta\rangle\varphi$. An example of this rule is given in Figure 10, where the rule is applied to $\langle g \sqcap h^d\rangle\langle g^\times\rangle\neg q$.

$$\frac{(\langle(p?;g)^*\rangle\ q) \to (\langle g \sqcap h^d\rangle\langle g^\times\rangle\neg q)}{(\langle(p?;g)^*\rangle\ q) \to (\langle(g \sqcap h^d); g^\times\rangle\neg q)}\,;_d$$

Figure 10: Example of the $;_d$ rule.

4.1.4 Strengthened induction rule

The strengthened induction rule ind_s detects the unfolding of greatest fixpoint formulae. It does so by applying demonic tests to the negation of the side formulas Φ and inserting them into the unfolded formula, as shown in Figure 11.

$$\frac{\Phi, \varphi \wedge \langle\gamma\rangle\langle(\overline{\Phi}!;\gamma)^\times\rangle\langle\overline{\Phi}!\rangle\varphi}{\Phi, \langle\gamma^\times\rangle\varphi}\,\mathrm{ind}_s$$

Figure 11: Strengthened induction rule.

4.1.5 G Proof example

With all the axioms and rules explained earlier in this section, proofs in G can now be constructed. In Figure 12, an example of a G proof is given, where at least one of each of the different rules is applied. In this proof, an equivalence relation between angelic and demonic iteration is used, that is that $a^\times$ is equal to a^{d^*}. The actual relation is that $a^\times$ is equal to $((a^d)^*)^d$, however, for ease the last dual operator is omitted in this proof example.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg p, p}\,\text{Ax}}{\neg p, \langle a^d\rangle\langle a^{d^*}\rangle\neg p, p}\,\text{weak}}{\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p, p}\,\vee}{\langle a^{d^*}\rangle\neg p, p}\,*}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p}\,\text{Ax}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p, p}\,\text{weak}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p \vee p}\,\vee}{\langle a^{d^*}\rangle\neg p, \langle\langle a^\times\rangle p!\rangle p}\,!}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p}\,\text{Ax}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\text{weak}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\vee}{\langle a^{d^*}\rangle\neg p, \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,!}{\langle a^{d^*}\rangle\neg p, \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,;_d}{\langle a^{d^*}\rangle\neg p, \langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\wedge}{\langle a^{d^*}\rangle\neg p, \langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\times}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg p, p}\,\text{Ax}}{\neg p, \langle a^d\rangle\langle a^{d^*}\rangle\neg p, p}\,\text{weak}}{\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p, p}\,\vee}{\langle a^{d^*}\rangle\neg p, p}\,*}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p}\,\text{Ax}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\text{weak}}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\vee}{\langle a^{d^*}\rangle\neg p, \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,!}{\langle a^{d^*}\rangle\neg p, \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,;_d}{\langle a^{d^*}\rangle\neg p, \langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\wedge}{\langle a^{d^*}\rangle\neg p, \langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\times}{\langle a^d\rangle\langle a^{d^*}\rangle\neg p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\text{mod}_m}{\neg p, \langle a^d\rangle\langle a^{d^*}\rangle\neg p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\text{weak}}{\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\vee}{\langle a^{d^*}\rangle\neg p, \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,*}{\langle a^{d^*}\rangle\neg p, p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p}\,\wedge}{\langle a^{d^*}\rangle\neg p, \langle a^\times\rangle p}\,\text{ind}_s$$

Figure 12: Example of a G proof.

4.2 Proof systems Par and Par$_{\text{Full}}$

When Parikh introduced game logic he also invented a Hilbert system for it. In [3] this system is referred to as Par. Another Hilbert system Par$_{\text{Full}}$ was constructed in [3] that functions as an intermediate system between G and Par. This proof system was introduced to facilitate the transformation from G to Par.

4.2.1 Axioms and rules of Par$_{\text{Full}}$

Most Hilbert systems consist of a few axioms and modus ponens. The axioms of Par are defined as "all propositional tautologies" and axioms for the game constructors in $\mathcal{G}_{Par}$, shown in Definition 4.1. Par also has two additional rules relating to game logic, namely Mon and Bar Induction. The Mon rule refers to monotonicity of neighbourhood functions [8], meaning that if some φ implies ψ, then if Angel can ensure an outcome φ when γ is played also implies that Angel can ensure ψ when γ is played [3]. The Bar Induction rule states that if $\langle\gamma\rangle\varphi$ implies φ, then $\langle\gamma\rangle$ can be replaced by $\langle\gamma^*\rangle$. Axiom 4 states that $\langle\gamma^*\rangle\varphi$ is a fixpoint of the operation $\varphi \vee \langle\gamma\rangle X$. The premise of BarInd is equivalent with $[\![\varphi]\!]$ being a pre-fixpoint of the monotone map $f(X) = [\![\varphi]\!] \cup E_\gamma(X)$. The BarInd rule then states that $lfp(f) \subseteq [\![\varphi]\!]$ can be concluded, in other words $[\![\langle\gamma^*\rangle\varphi]\!] \subseteq [\![\varphi]\!]$, thus saying that $\langle\gamma^*\rangle\varphi$ is a least pre-fixpoint, and therefore, the least fixpoint of the operation $\varphi \vee \langle\gamma\rangle X$.

Definition 4.1 (Axioms (left) and rules of Par (right))**.**

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}\ \text{MP}$$

1) All propositional tautologies
2) $\langle \gamma ; \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \langle \delta \rangle \varphi$
3) $\langle \gamma \sqcup \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \varphi \vee \langle \delta \rangle \varphi$
4) $\langle \gamma^* \rangle \varphi \leftrightarrow \varphi \vee \langle \gamma \rangle \langle \gamma^* \rangle \varphi$
5) $\langle \psi ? \rangle \varphi \leftrightarrow \psi \wedge \varphi$
6) $\langle \gamma^d \rangle \varphi \leftrightarrow \neg \langle \gamma \rangle \neg \varphi$

$$\frac{\varphi \to \psi}{\langle \gamma \rangle \varphi \to \langle \gamma \rangle \psi}\ \text{Mon}$$

$$\frac{\langle \gamma \rangle \varphi \to \varphi}{\langle \gamma^* \rangle \varphi \to \varphi}\ \text{BarInd}$$

Par_{Full} extends Par by adding the axioms 7-9 and the $\text{BarInd}^\times$ rule in Definition 4.2. Axiom 8 and $\text{BarInd}^\times$ both refer to the dual version of * and BarInd, such that Axiom 8 states that $\langle \gamma^\times \rangle \varphi$ is a fixpoint of the operation $\varphi \wedge \langle \gamma \rangle X$. The $\text{BarInd}^\times$ rule then states that $\langle \gamma^\times \rangle \varphi$ is the greatest post-fixpoint for $\varphi \wedge \langle \gamma \rangle X$, and therefore, its greatest fixpoint. The grammar of Par_{Full} extends Par by allowing all connectives in formulas, all game constructors in games, and freely placed duals and negations.

Definition 4.2 (Axioms (left) and rules of Par_{Full} (right))**.**

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}\ \text{MP}$$

1) All propositional tautologies
2) $\langle \gamma ; \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \langle \delta \rangle \varphi$
3) $\langle \gamma \sqcup \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \varphi \vee \langle \delta \rangle \varphi$
4) $\langle \gamma^* \rangle \varphi \leftrightarrow \varphi \vee \langle \gamma \rangle \langle \gamma^* \rangle \varphi$
5) $\langle \psi ? \rangle \varphi \leftrightarrow \psi \wedge \varphi$
6) $\langle \gamma^d \rangle \varphi \leftrightarrow \neg \langle \gamma \rangle \neg \varphi$
7) $\langle \gamma \sqcap \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \varphi \wedge \langle \delta \rangle \varphi$
8) $\langle \gamma^\times \rangle \varphi \leftrightarrow \varphi \wedge \langle \gamma \rangle \langle \gamma^\times \rangle \varphi$
9) $\langle \psi ! \rangle \varphi \leftrightarrow \psi \vee \varphi$

$$\frac{\varphi \to \psi}{\langle \gamma \rangle \varphi \to \langle \gamma \rangle \psi}\ \text{Mon}$$

$$\frac{\langle \gamma \rangle \varphi \to \varphi}{\langle \gamma^* \rangle \varphi \to \varphi}\ \text{BarInd}$$

$$\frac{\varphi \to \langle \gamma \rangle \varphi}{\varphi \to \langle \gamma^\times \rangle \varphi}\ \text{BarInd}^\times$$

4.2.2 Par_{Full} proof example

Proofs in Par_{Full} are a list of formulas where each line has a line number on the left side and a justification on the right side. Each justification states the axiom or rule upon which the formula is based. In case a formula follows from preceding lines, those lines are listed along with the justification. If a statement is a specification of some propositional tautology, the justification begins with "PL" followed by a generic formula describing the tautology. A table of all propositional tautologies that are used for the Par_{Full} derivations can be found in Appendix I. In order to decrease the proof size to some extent, a set of derivable rules for propositional logic is used that are often applied in a Par_{Full} proof. A full description of these derivable rules and their derivations can be found in Appendix II.
In Figure 13, an example of a Par_{Full} proof can be found. We can see that a Par_{Full} proof consists of a list of statements, where each statement has a line number on the left, the disjunction of formulas in the middle and the justification on the right. Each statement is either based on an axiom, rule of inference or a propositional tautology.

1.	$\langle\neg p!\rangle\langle g^d\rangle q \vee \langle\neg p!\rangle\langle g^d\rangle\neg q$	PL: $A \vee \neg A$
2.	$(\langle\neg p!\rangle\langle g^d\rangle q \vee \langle\neg p!\rangle\langle g^d\rangle\neg q) \to (\langle\neg p!\rangle\langle g^d\rangle q \vee \langle\neg p!\rangle\langle g^d\rangle\neg q \vee p)$	PL: $A \to A \vee B$
3.	$\langle\neg p!\rangle\langle g^d\rangle q \vee \langle\neg p!\rangle\langle g^d\rangle\neg q \vee p$	MP: $1, 2$
4.	$p \vee \neg p$	PL: $A \vee \neg A$
5.	$(p \vee \neg p) \to (p \vee \neg p \vee \langle g^d\rangle\neg q)$	PL: $A \to A \vee B$
6.	$p \vee \neg p \vee \langle g^d\rangle\neg q$	MP: $4, 5$
7.	$(p \vee \neg p \vee \langle g^d\rangle\neg q) \to (p \vee (\neg p \vee \langle g^d\rangle\neg q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
8.	$(p \vee (\neg p \vee \langle g^d\rangle\neg q))$	MP: $6, 7$
9.	$(\neg(p) \to (\neg p \vee \langle g^d\rangle\neg q))$	$\text{PL}_{D;C}$: 8
10.	$(\langle\neg p!\rangle\langle g^d\rangle\neg q \leftrightarrow (\neg p \vee \langle g^d\rangle\neg q))$	Ax9 !
11.	$((\neg p \vee \langle g^d\rangle\neg q) \to \langle\neg p!\rangle\langle g^d\rangle\neg q)$	BE_R: 10
12.	$(\neg(p) \to \langle\neg p!\rangle\langle g^d\rangle\neg q)$	PL_{CUT}: $9, 11$

Figure 13: Example of a Par_{Full} proof

5 Proof transformation: G to Par_{Full}

The transformation from G to Par_{Full} is defined in [3], however, this section will provide a more detailed definition by deriving all the rules of G in the Par_{Full} system. First, all Par_{Full} derivations of the basic rules of G are given, followed by those of the game operation rules, and finally, those of the deep inference rules and the strengthened induction rule. A first thing to note is that a proof in G is a tree of sequents while a proof in Par_{Full} is a list of formulas. A sequent is defined as a set of formulas, visually separated by commas but to be read disjunctively. A sequent is normally divided into a set of side formulas, denoted by Φ, and an active formula or active formulas to which a rule is applied. The translation from a set of formulas to a disjunction of formulas, from $\{A_1, ..., A_n\}$ to $A_1 \vee A_2 \vee ...A_n$, is needed since sequents do not occur in Par_{Full}. Hence, the notation $\bigvee\Phi$ is used in the following Par_{Full} derivations to denote this disjunction of formulas. Another notable difference between G and Par_{Full} is how these proof systems deal with fixpoints. G uses the ind_s rule and unfolding rules for angelic and demonic iteration, while Par_{Full} uses the BarInd and $\text{BarInd}^\times$ rule and unfolding axioms for angelic and demonic iteration.

5.1 Par_{Full} derivations of basic rules of G

For a full list of all Par_{Full} derivations of the basic rules, see Appendix III. The only axiom in G is Ax and this axiom directly reflects the law of excluded middle. Therefore, this rule can be directly translated into a Par_{Full} derivation with only one propositional tautology, as shown in Figure 14.

$$\frac{}{\Phi, \overline{\Phi}}\ \text{Ax} \qquad\qquad \begin{array}{lll} 1. & \bigvee\Phi \vee \neg\bigvee\Phi & \text{PL: } A \vee \neg A \end{array}$$

Figure 14: Ax axiom (left), Par_{Full} derivation of Ax (right).

Given a set of side formulas, the weak rule can be derived by using the tautology $A \to A \vee B$, such that any arbitrary formula B can be introduced. $A \vee B$ can then be concluded by applying modus ponens, as depicted in Figure 15. The weak rule can only be applied when there are side formulas, otherwise there is nothing to be weakened.

$$\frac{\Phi}{\Phi, \varphi}\ \text{weak} \qquad\qquad \begin{array}{lll} 1. & \bigvee\Phi & \text{premise} \\ 2. & \bigvee\Phi \to \bigvee\Phi \vee \varphi & \text{PL: } A \to A \vee B \\ 3. & \bigvee\Phi \vee \varphi & \text{MP: 1,2} \end{array}$$

Figure 15: weak rule (left), Par_{Full} derivation of weak rule (right).

The $\vee$ rule can be applied with and without the presence of side formulas. The $\vee$ rule is applied to any two formulas φ and ψ in a sequent as shown in Figure 5, resulting in a sequent with the side formulas Φ and the disjunction of φ and ψ.

In $\mathrm{Par}_{\mathrm{Full}}$, a sequent is already translated into a disjunction of formulas, such that, in case there are side formulas, φ and ψ are put into a separate disjunct $\varphi \vee \psi$, as depicted in line 2 of Figure 16. If there are no side formulas besides φ and ψ, the $\vee$ rule can be disregarded entirely, since the sequent φ, ψ in G already transforms to $\varphi \vee \psi$ in $\mathrm{Par}_{\mathrm{Full}}$.

$$\frac{\Phi, \varphi, \psi}{\Phi, \varphi \vee \psi} \vee$$

1.	$\bigvee \Phi \vee \varphi \vee \psi$	premise
2.	$\bigvee \Phi \vee \varphi \vee \psi \rightarrow \bigvee \Phi \vee (\varphi \vee \psi)$	PL: $A \vee B \vee C \rightarrow A \vee (B \vee C)$
3.	$\bigvee \Phi \vee (\varphi \vee \psi)$	MP: 1,2

Figure 16: $\vee$ rule (left), $\mathrm{Par}_{\mathrm{Full}}$ derivation of $\vee$ rule (right).

The $\wedge$ rule is applied to two separate branches of a G proof. In the one branch is some formula φ which is conjuncted with some formula ψ in the other branch. In Figure 17, the $\wedge$ rule is derived when there are no side formulas. In this case, the $\wedge$ rule can be derived by using a propositional tautology to introduce a conjunction, followed by two applications of modus ponens.

$$\frac{\Phi, \varphi \qquad \Phi, \psi}{\Phi, \varphi \wedge \psi} \wedge$$

1.	φ	premise
2.	ψ	premise
3.	$\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$	PL: $A \rightarrow (B \rightarrow (A \wedge B))$
4.	$\psi \rightarrow (\varphi \wedge \psi)$	MP: $1, 3$
5.	$\varphi \wedge \psi$	MP: $2, 4$

Figure 17: $\wedge$ rule (left), $\mathrm{Par}_{\mathrm{Full}}$ derivation of $\wedge$ rule (without side formulas) (right).

Figure 18 shows the derivation in Par of the $\wedge$ rule in case there are side formulas. In the justifications for this derivation, the derivable rules $\mathrm{PL}_{D;C}$ and $\mathrm{PL}_{C;D}$ are introduced. $\mathrm{PL}_{D;C}$ and $\mathrm{PL}_{C;D}$ refer to the conversion of a disjunction into a conditional and vice versa, $A \vee B \leftrightarrow \neg A \rightarrow B$. In many cases, it is convenient to convert a disjunction $\bigvee \Phi \vee \varphi$ into a conditional where all the side formulas are placed on the left side and the "active formula" on the right side, resulting in $\neg \bigvee \Phi \rightarrow \varphi$. Applying a rule to the active formula φ results in $\varphi \rightarrow \varphi'$. Now the cut rule can be applied to these two formulas, leading to $\neg \bigvee \Phi \rightarrow \varphi'$. This principle is used in many of the coming Par derivations and is referred to as PL_{CUT} in the justification, followed by the lines of the two conditional statements to which the rule is applied.

1.	$\bigvee \Phi \vee \varphi$	premise
2.	$\bigvee \Phi \vee \psi$	premise
3.	$\neg \bigvee \Phi \rightarrow \varphi$	$\mathrm{PL}_{D;C}$: 1
4.	$\neg \bigvee \Phi \rightarrow \psi$	$\mathrm{PL}_{D;C}$: 2
5.	$(\neg \bigvee \Phi \rightarrow \varphi) \rightarrow ((\neg \bigvee \Phi \rightarrow \psi) \rightarrow (\neg \bigvee \Phi \rightarrow (\varphi \wedge \psi)))$	PL: $(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \wedge C))$
6.	$(\neg \bigvee \Phi \rightarrow \psi) \rightarrow (\neg \bigvee \Phi \rightarrow (\varphi \wedge \psi))$	MP: $3, 5$
7.	$\neg \bigvee \Phi \rightarrow (\varphi \wedge \psi)$	MP: 4,6
8.	$\bigvee \Phi \vee (\varphi \wedge \psi)$	$\mathrm{PL}_{C;D}$: 7

Figure 18: $\mathrm{Par}_{\mathrm{Full}}$ derivation of $\wedge$ rule (with side formulas).

To reiterate, the mod_m rule enables to place a modality of an atomic game g in front of a formula φ and a modality of a dual game g^d in front of some formula ψ in the sequent, such that $\varphi, \psi \vdash \langle g \rangle \varphi, \langle g^d \rangle \psi$. In $\mathrm{Par}_{\mathrm{Full}}$, this rule can be derived by using the axiom for dual games, see 19 for the full $\mathrm{Par}_{\mathrm{Full}}$ derivation. The antecedent of this sequent translates directly into the disjunct $\varphi \vee \psi$, which can then be rewritten as the conditional $\neg\psi \rightarrow \varphi$. Now, the Mon rule can be applied by adding $\langle g \rangle$ to both sides of the conditional, giving $\langle g \rangle \neg\psi \rightarrow \langle g \rangle \varphi$. In order to end up with an angelic game g on one side and a dual game g^d on the other side, the conditional can be rewritten as a disjunction, negating $\langle g \rangle \neg\psi$ into $\neg\langle g \rangle \neg\psi$. Then, the dual game axiom can be used to convert $\neg\langle g \rangle \neg\psi$ into $\langle g^d \rangle \psi$. Finally, by a few applications of modus ponens and some propositional tautologies, the conclusion $\langle g \rangle \varphi \vee \langle g^d \rangle \psi$ can be derived.

$$\frac{\varphi, \psi}{\langle g\rangle\varphi, \langle g^d\rangle\psi}\,\mathrm{mod}_m$$

1.	$\varphi \vee \psi$	premise
2.	$(\varphi \vee \psi) \to (\psi \vee \varphi)$	PL: $(A \vee B) \to (B \vee A)$
3.	$\psi \vee \varphi$	MP: 1, 2
4.	$\neg\psi \to \varphi$	$\mathrm{PL}_{D;C}$: 3
5.	$\langle g\rangle\neg\psi \to \langle g\rangle\varphi$	Mon: 4
6.	$\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi$	$\mathrm{PL}_{C;D}$: 5
7.	$\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi$	Ax6 dual
8.	$(\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi) \to ((\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi))$	PL: $(A \leftrightarrow B) \to ((B \vee C) \to (A \vee C))$
9.	$(\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi)$	MP: 7,8
10.	$\langle g^d\rangle\psi \vee \langle g\rangle\varphi$	MP: 6,9
11.	$(\langle g^d\rangle\psi \vee \langle g\rangle\varphi) \to (\langle g\rangle\varphi \vee \langle g^d\rangle\psi)$	PL: $(A \vee B) \to (B \vee A)$
12.	$\langle g\rangle\varphi \vee \langle g^d\rangle\psi$	MP: 10,11

Figure 19: mod_m rule (top), $\mathrm{Par}_{\mathrm{Full}}$ derivation of mod_m rule (bottom).

1.	$\psi \vee \varphi$	premise
2.	$\neg\psi \to \varphi$	$\mathrm{PL}_{D;C}$: 1
3.	$\langle g\rangle\neg\psi \to \langle g\rangle\varphi$	Mon: 2
4.	$\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi$	$\mathrm{PL}_{C;D}$: 3
5.	$\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi$	Ax6 dual
6.	$(\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi) \to ((\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi))$	PL: $(A \leftrightarrow B) \to ((B \vee C) \to (A \vee C))$
7.	$(\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi)$	MP: 5,6
8.	$\langle g^d\rangle\psi \vee \langle g\rangle\varphi$	MP: 4,7

Figure 20: $\mathrm{Par}_{\mathrm{Full}}$ derivation of mod_m rule (order of game and dual game switched).

5.2 $\mathrm{Par}_{\mathrm{Full}}$ derivations of game operation rules

The game operation rules of G directly reflect the semantic meaning of the game constructors. Therefore, each derivation in $\mathrm{Par}_{\mathrm{Full}}$ uses the corresponding axiom of $\mathrm{Par}_{\mathrm{Full}}$ to derive the rule at hand. Since these derivations are equal in their process, only the rules for angelic iteration and demonic test are explained in this section. A complete list of derivations can be found in Appendix IV. To reiterate, the rules for angelic iteration and demonic test are defined as follows:

$$\frac{\Phi, \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi}{\Phi, \langle\gamma^*\rangle\varphi}\,* \qquad\qquad \frac{\Phi, \psi \vee \varphi}{\Phi, \langle\psi!\rangle\varphi}\,!$$

The $\mathrm{Par}_{\mathrm{Full}}$ derivation for the * rule without side formulas can be found in Figure 21 on the right. In this case, the only formula in the premise is the unfolded formula $\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$. We can now use Ax4 of $\mathrm{Par}_{\mathrm{Full}}$ to infer the conclusion $\langle\gamma^*\rangle\varphi$. We introduce a derivable rule BE_R that performs biconditional elimination from right to left (BE_L performs this from left to right). Finally, the modus ponens rule can be applied to the premise and the conditional we just inferred in order to arrive at the conclusion. The derivation in case there are side formulas can be found in Figure 21 on the left. Here we can first use $PL_{D;C}$ to translate the disjunction of formulas to a conditional where we place the negation of the side formulas $\bigvee\Phi$ to the left and the active formula to the right. Then, we use Ax4 and BE_R again to obtain the conditional from line 4 in the figure. Now, we can apply the cut rule to the conditionals in line 2 and 4 to obtain $\neg\bigvee\Phi \to \langle\gamma^*\rangle\varphi$, which can then be translated back into a disjunction of formulas with $PL_{C;D}$.

1.	$\bigvee \Phi \vee (\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi)$	premise
2.	$\neg \bigvee \Phi \to (\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi)$	$\mathrm{PL}_{D;C}$: 1
3.	$\langle\gamma^*\rangle\varphi \leftrightarrow \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	Ax4 *
4.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi \to \langle\gamma^*\rangle\varphi$	BE_R: 3
5.	$\neg \bigvee \Phi \to \langle\gamma^*\rangle\varphi$	PL_{CUT}: 2,4
6.	$\bigvee \Phi \vee \langle\gamma^*\rangle\varphi$	$\mathrm{PL}_{C;D}$: 5

1.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	premise
2.	$\langle\gamma^*\rangle\varphi \leftrightarrow \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	Ax4 *
3.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi \to \langle\gamma^*\rangle\varphi$	$\mathrm{BE}_R : 2$
4.	$\langle\gamma^*\rangle\varphi$	MP: 1,3

Figure 21: $\mathrm{Par}_{\mathrm{Full}}$ derivation of * rule with side formulas (left) and without side formulas (right).

Similarly, the derivation of the ! rule without side formulas can be achieved by using the axiom of $\mathrm{Par}_{\mathrm{Full}}$ for demonic test followed by BE_R and modus ponens. The derivation of the ! rule with side formulas uses, just as with the * rule, the translation of a disjunction to a conditional, the cut rule, and a translation from a conditional back to a disjunction. In Figure 22, the derivation with side formulas can be found on the left, and without side formulas on the right.

1.	$\bigvee \Phi \vee (\psi \vee \varphi)$	premise
2.	$\neg \bigvee \Phi \to (\psi \vee \varphi)$	$\mathrm{PL}_{D;C} : 1$
3.	$\langle\psi!\rangle\varphi \leftrightarrow \psi \vee \varphi$	Ax9 !
4.	$\psi \vee \varphi \to \langle\psi!\rangle\varphi$	$\mathrm{BE}_R : 3$
5.	$\neg \bigvee \Phi \to \langle\psi!\rangle\varphi$	$\mathrm{PL}_{CUT} : 2, 4$
6.	$\bigvee \Phi \vee \langle\psi!\rangle\varphi$	$\mathrm{PL}_{C;D} : 5$

1.	$\psi \vee \varphi$	premise
2.	$\langle\psi!\rangle\varphi \leftrightarrow \psi \vee \varphi$	Ax9 !
3.	$\psi \vee \varphi \to \langle\psi!\rangle\varphi$	$\mathrm{BE}_R : 2$
4.	$\langle\psi!\rangle\varphi$	MP: 1,4

Figure 22: $\mathrm{Par}_{\mathrm{Full}}$ derivation of ! rule with side formulas (left) and without side formulas (right).

5.3 Deep inference rules derivations

The derivations for the deep inference rules require a different strategy than those of the basic and game operation rules. Since these rules are applied at a particular depth inside a context, we have to take apart that context recursively, connective by connective, until we arrive at the depth where the rule is applied. Once we arrive at that atomic level, we can apply the rule and then rebuild the context from inward to outward.

5.3.1 $\mathrm{Par}_{\mathrm{Full}}$ derivation of Mon_d^g

To reiterate, the Mon_d^g is defined as follows:

$$\frac{\Phi, \psi(\gamma)}{\Phi, \psi(\chi!;\gamma)}\,\mathrm{Mon}_d^g$$

In Figure 23, the derivations of the Mon_d^g rule with side formulas and without side formulas are given. The statement coloured in brown indicates the recursive step in the derivation. The derivation that follows in this step depends on the outermost game logic connective in ψ. If ψ is merely an atomic game g, then the recursive step is the derivation of the atomic case which can be found in Figure 24. Applying Mon_d^g at the atomic level is rather straightforward. First, the formula is weakened by introducing χ. Then, the axiom for demonic test is used to transform the disjunction into a modality of $\chi!$ followed by the original formula. The axiom for composition can then transform the two modalities into a composition, giving the desired conclusion. The last application of modus ponens in this and the following case derivations is left out, since this rule is applied at the top-level of a Mon_d^g application, as can be seen in Figure 23.

1.	$\bigvee \Phi \vee \psi(\gamma)$	premise
2.	$\neg \bigvee \Phi \to \psi(\gamma)$	$\mathrm{PL}_{D;C} : 1$
3.	$\psi(\gamma) \to \psi(\chi!;\gamma)$	Recursive step
4.	$\neg \bigvee \Phi \to \psi(\chi!;\gamma)$	PL_{CUT}: 2, 3
5.	$\bigvee \Phi \vee \psi(\chi!;\gamma)$	$\mathrm{PL}_{C;D}$: 4

1.	$\psi(\gamma)$	premise
2.	$\psi(\gamma) \to \psi(\chi!;\gamma)$	Recursive step
3.	$\psi(\chi!;\gamma)$	MP: 1, 2

Figure 23: $\mathrm{Par}_{\mathrm{Full}}$ derivation of Mon^{g}_{d} rule with side formulas (left) and without side formulas (right).

1.	$\langle g \rangle \varphi$	premise
2.	$\langle g \rangle \varphi \to \chi \vee \langle g \rangle \varphi$	PL: $A \to B \vee A$
3.	$\langle \chi! \rangle \langle g \rangle \varphi \leftrightarrow \chi \vee \langle g \rangle \varphi$	Ax9 !
4.	$\chi \vee \langle g \rangle \varphi \to \langle \chi! \rangle \langle g \rangle \varphi$	$\mathrm{BE}_R : 3$
5.	$\langle g \rangle \varphi \to \langle \chi! \rangle \langle g \rangle \varphi$	PL_{CUT}: 2, 4
6.	$\langle \chi!; g \rangle \varphi \leftrightarrow \langle \chi! \rangle \langle g \rangle \varphi$	Ax2 ;
7.	$\langle \chi! \rangle \langle g \rangle \varphi \to \langle \chi!; g \rangle \varphi$	$\mathrm{BE}_R : 6$
8.	$\langle g \rangle \varphi \to \langle \chi!; g \rangle \varphi$	PL_{CUT}: 5, 7

Figure 24: $\mathrm{Par}_{\mathrm{Full}}$ derivation of Mon^{g}_{d}-rule, atomic case.

However, the atomic game can be wrapped in any other game and formula and in such cases we need to take apart these formulas first. Figure 25 illustrates this in the case of an angelic test. In the first lines, before the recursive step, the angelic test is taken apart by using its axiom. Then we can apply the recursive step and use propositional tautologies and modus ponens to transform δ into $\chi!;\delta$ in the conjunction that came from using the axiom in line 3. Then we can construct back the original formula by using the axiom for angelic test again. Now, suppose that a demonic choice of some γ_1 and some γ_2 lies inside the angelic test, and that the rule is applied to γ_2. The recursive step consists of a derivation of a demonic choice where the rule is applied to the right game. The derivation of this case can be found in Figure 26. We can see that this derivation is identical in its structure to the derivation of the ! case. First, it takes apart the formula with the axiom for demonic choice. Then, the rule is applied in the recursive step and finally, the formula can be reconstructed again. This structure applies to all the game connectives of which the derivations can be found in Appendix V.

1.	$\langle \theta(\delta)? \rangle \varphi$	premise
2.	$\langle \theta(\delta)? \rangle \varphi \leftrightarrow \theta(\delta) \wedge \varphi$	Ax5 ?
3.	$\langle \theta(\delta)? \rangle \varphi \to \theta(\delta) \wedge \varphi$	$\mathrm{BE}_L : 2$
4.	$\theta(\delta) \to \theta(\chi!;\delta)$	Recursive step
5.	$(\theta(\delta) \to \theta(\chi!;\delta)) \to (\theta(\delta) \wedge \varphi \to \theta(\chi!;\delta) \wedge \varphi)$	PL: $(A \to B) \to (A \wedge C \to B \wedge C)$
6.	$\theta(\delta) \wedge \varphi \to \theta(\chi!;\delta) \wedge \varphi$	MP: 4, 5
7.	$\langle \theta(\delta)? \rangle \varphi \to \theta(\chi!;\delta) \wedge \varphi$	PL_{CUT}: 3, 6
8.	$\langle \theta(\chi!;\delta)? \rangle \varphi \leftrightarrow \theta(\chi!;\delta) \wedge \varphi$	Ax5 ?
9.	$\theta(\chi!;\delta) \wedge \varphi \to \langle \theta(\chi!;\delta)? \rangle \varphi$	BE_R: 8
10.	$\langle \theta(\delta)? \rangle \varphi \to \langle \theta(\chi!;\delta)? \rangle \varphi$	PL_{CUT}: 7, 9

Figure 25: $\mathrm{Par}_{\mathrm{Full}}$ derivation of Mon^{g}_{d}-rule, ? case.

1.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	$\text{BE}_L : 2$
4.	$\langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_2(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$(\langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_2(\chi!;\delta)\rangle\varphi) \rightarrow (\langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\chi!;\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \wedge A \rightarrow C \wedge B)$
6.	$\langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\chi!;\delta)\rangle\varphi$	MP: $4, 5$
7.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\chi!;\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\chi!;\delta)\rangle\varphi$	Ax7 $\sqcap$
9.	$\langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\chi!;\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta) \sqcap \gamma_2(\chi!;\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta) \sqcap \gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $7, 9$

Figure 26: Par_{Full} derivation of Mon^g_d-rule, $\sqcap$ case.

The derivations of propositional connectives require less proof lines than the game connectives. In Figure 27, the derivation for conjunction where the rule is applied to the left conjunct can be found. One application of a propositional tautology and modus ponens is enough to derive the conditional with on the left side the premise and on the right side the conclusion. The rest of the propositional cases can be found in Appendix V.

1.	$\gamma(\delta) \wedge p$	premise
2.	$\gamma(\delta) \rightarrow \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \rightarrow \gamma(\chi!;\delta)) \rightarrow ((\gamma(\delta) \wedge p) \rightarrow (\gamma(\chi!;\delta) \wedge p))$	PL: $(A \rightarrow B) \rightarrow ((A \wedge C) \rightarrow (B \wedge C))$
4.	$(\gamma(\delta) \wedge p) \rightarrow (\gamma(\chi!;\delta) \wedge p)$	MP: $2, 3$

Figure 27: Par_{Full} derivation of Mon^g_d rule, $\wedge$ case left.

5.3.2 Par_{Full} derivation of Mon^f_d

To reiterate, the Mon^f_d rule is defined as follows:

$$\frac{\Phi, \psi(\varphi)}{\Phi, \psi(\langle\chi!\rangle\varphi)}\ \text{Mon}^f_d$$

The case derivations of Mon^f_d have the same structure as those of Mon^g_d, except for the atomic case which is where the rule is applied. The main derivations are listed in Figure 28, on the left the derivation with side formulas and on the right the derivation without side formulas. The atomic case derivation, in Figure 29, is almost identical to that of Mon^g_d, except for the fact that Mon^f_d applies the rule to formulas. Therefore, the last part in the Mon^g_d derivation, where the formula is put in a game by sequential composition, is omitted.

1.	$\bigvee\Phi \vee \psi$	premise
2.	$\neg\bigvee\Phi \rightarrow \psi$	$\text{PL}_{D;C}$: 1
3.	$\psi \rightarrow \langle\chi!\rangle\psi$	Recursive step
4.	$\neg\bigvee\Phi \rightarrow \langle\chi!\rangle\psi$	PL_{CUT}: $2, 3$
5.	$\bigvee\Phi \vee \langle\chi!\rangle\psi$	$\text{PL}_{C;D}$: 4

1.	ψ	premise
2.	$\psi \rightarrow \langle\chi!\rangle\psi$	Recursive step
3.	$\langle\chi!\rangle\psi$	MP: $1, 2$

Figure 28: Par_{Full} derivation of Mon^f_d rule with side formulas (left) and without side formulas (right).

1.	ψ	premise
2.	$\psi \rightarrow \chi \vee \psi$	PL: $A \rightarrow B \vee A$
3.	$\langle\chi!\rangle\psi \leftrightarrow \chi \vee \psi$	Ax9 !
4.	$\chi \vee \psi \rightarrow \langle\chi!\rangle\psi$	BE_R: 3
5.	$\psi \rightarrow \langle\chi!\rangle\psi$	PL_{CUT}: $2, 4$

Figure 29: Par_{Full} derivation of Mon^f_d rule, atomic case.

Despite the fact that the case derivations of all deep inference rules are equal, for illustrative purposes the case derivation of a right application of ; is given in Figure 30. At first, the formula is taken apart by using the axiom for sequential composition. Then, the recursive step is applied to the right game, after which the left part can be put before the right game by applying the Mon rule. We can then use the axiom for sequential composition again to reconstruct the formula. The case of *, in Figure 31, differs slightly from other cases. The concluding context has to be used when we take the formula apart, since an unfolded fixed point formula of * contains two consecutive modalities that both contain the same game γ. Otherwise, the recursive step in line 6 would not have that concluding formula in the second modality of the unfolded fixpoint formula.

1.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	Ax2 ;
3.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	$\text{BE}_L : 2$
4.	$\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$\langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Mon: 6
6.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: 3, 5
7.	$\langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Ax2 ;
8.	$\langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	BE_R: 7
9.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: 6, 8

Figure 30: Par_{Full} derivation of Mon_d^f rule, ; case right.

1.	$\langle\psi(\delta)^*\rangle\varphi$	premise
2.	$\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \leftrightarrow \varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Ax4
3.	$\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	$\text{BE}_R : 2$
4.	$(\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi) \rightarrow$ $(\langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi)$	PL: $(A \vee B \rightarrow C) \rightarrow (B \rightarrow C)$
5.	$\langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	MP: 3,4
6.	$\langle\psi(\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Recursive step
7.	$\langle\psi(\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	PL_{CUT}: 6, 5
8.	$\langle\psi(\delta)^*\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	BarInd: 7
9.	$(\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi) \rightarrow (\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi)$	PL: $(A \vee B \rightarrow C) \rightarrow (A \rightarrow C)$
10.	$\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	MP: 3,9
11.	$\langle\psi(\delta)^*\rangle\varphi \rightarrow \langle\psi(\delta)^*\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Mon: 10
12.	$\langle\psi(\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	$\text{PL}_{CUT} : 11, 8$

Figure 31: Par_{Full} derivation of Mon_d^f rule, * case.

5.3.3 Par_{Full} derivation of $;_d$

To reiterate, the $;_d$ rule is defined as follows:

$$\frac{\Phi, \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)}{\Phi, \psi(\langle\gamma;\delta\rangle\varphi)}\ ;_d$$

Since all deep inference rules are equal in their case derivations, only the main derivations and the atomic case derivation of the $;_d$ rule are given. In Figure 32, the main derivations are given. The atomic case derivation, given in Figure 33, simply uses the axiom for sequential composition.

1.	$\bigvee \Phi \vee \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	premise
2.	$\neg \bigvee \Phi \to \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	$\mathrm{PL}_{D;C}$: 1
3.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
4.	$\neg \bigvee \Phi \to \psi(\langle\gamma;\delta\rangle\varphi)$	PL_{CUT}: 2, 3
5.	$\bigvee \Phi \vee \psi(\langle\gamma;\delta\rangle\varphi)$	$\mathrm{PL}_{C;D}$: 4

1.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	premise
2.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
3.	$\psi(\langle\gamma;\delta\rangle\varphi)$	MP: 1, 2

Figure 32: $\mathrm{Par}_{\mathrm{Full}}$ derivation of $;_d$ rule with side formulas (left) and without side formulas (right).

1.	$\langle\gamma\rangle\langle\delta\rangle\varphi$	premise
2.	$\langle\gamma;\delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\langle\delta\rangle\varphi$	Ax2 ;
3.	$\langle\gamma\rangle\langle\delta\rangle\varphi \to \langle\gamma;\delta\rangle\varphi$	BE_R: 2
4.	$\langle\gamma;\delta\rangle\varphi$	MP: 1, 3

Figure 33: $\mathrm{Par}_{\mathrm{Full}}$ derivation of $;_d$ rule, atomic case.

5.3.4 Example of a deep inference rule application

To give an example of how a deep inference rule is applied, Figure 34 gives the recursive derivation of Mon_d^g on a complex formula with the atomic game wrapped in a sequential composition and angelic choice. First, the angelic choice is taken apart in the red lines (2-3). Then, we arrive at the sequential composition part which is taken apart in the blue lines (4-5). At the atomic case the Mon_d^g is applied to the atomic game c, given by the brown lines (6-12). After applying the rule, the sequential composition part is reconstructed (13-17) followed by the angelic choice (18-24).

1.	$\langle(b;c) \sqcup d\rangle p$	premise
2.	$\langle(b;c) \sqcup d\rangle p \leftrightarrow (\langle b;c\rangle p \vee \langle d\rangle p)$	Ax3 $\sqcup$
3.	$\langle(b;c) \sqcup d\rangle p \to (\langle b;c\rangle p \vee \langle d\rangle p)$	BE_L: 2
4.	$\langle b;c\rangle p \leftrightarrow \langle b\rangle\langle c\rangle p$	Ax2 ;
5.	$\langle b;c\rangle p \to \langle b\rangle\langle c\rangle p$	BE_L: 4
6.	$\langle c\rangle p \to (\chi \vee \langle c\rangle p)$	PL: $A \to B \vee A$
7.	$\langle\chi!\rangle\langle c\rangle p \leftrightarrow (\chi \vee \langle c\rangle p)$	Ax9 !
8.	$(\chi \vee \langle c\rangle p) \to \langle\chi!\rangle\langle c\rangle p$	BE_R: 7
9.	$\langle c\rangle p \to \langle\chi!\rangle\langle c\rangle p$	PL_{CUT}: 6, 8
10.	$\langle\chi!;c\rangle p \leftrightarrow \langle\chi!\rangle\langle c\rangle p$	Ax2 ;
11.	$\langle\chi!\rangle\langle c\rangle p \to \langle\chi!;c\rangle p$	BE_R: 10
12.	$\langle c\rangle p \to \langle\chi!;c\rangle p$	PL_{CUT}: 9, 11
13.	$\langle b\rangle\langle c\rangle p \to \langle b\rangle\langle\chi!;c\rangle p$	Mon: 12
14.	$\langle b;c\rangle p \to \langle b\rangle\langle\chi!;c\rangle p$	PL_{CUT}: 5, 13
15.	$\langle b;(\chi!;c)\rangle p \leftrightarrow \langle b\rangle\langle\chi!;c\rangle p$	Ax2 ;
16.	$\langle b\rangle\langle\chi!;c\rangle p \to \langle b;(\chi!;c)\rangle p$	BE_R: 15
17.	$\langle b;c\rangle p \to \langle b;(\chi!;c)\rangle p$	PL_{CUT}: 14, 16
18.	$(\langle b;c\rangle p \to \langle b;(\chi!;c)\rangle p) \to$ $((\langle b;c\rangle p \vee \langle d\rangle p) \to (\langle b;(\chi!;c)\rangle p \vee \langle d\rangle p))$	PL: $(A \to B) \to (A \vee C \to B \vee C)$
19.	$(\langle b;c\rangle p \vee \langle d\rangle p) \to (\langle b;(\chi!;c)\rangle p \vee \langle d\rangle p)$	MP: 17, 18
20.	$\langle(b;c) \sqcup d\rangle p \to (\langle b;(\chi!;c)\rangle p \vee \langle d\rangle p)$	PL_{CUT}: 3, 19
21.	$\langle(b;(\chi!;c)) \sqcup d\rangle p \leftrightarrow (\langle b;(\chi!;c)\rangle p \vee \langle d\rangle p)$	Ax3 $\sqcup$
22.	$\langle b;(\chi!;c)\rangle p \vee \langle d\rangle p) \to \langle(b;(\chi!;c)) \sqcup d\rangle p$	BE_R: 21
23.	$\langle(b;c) \sqcup d\rangle p \to \langle(b;(\chi!;c)) \sqcup d\rangle p$	PL_{CUT}: 20, 22
24.	$\langle(b;(\chi!;c)) \sqcup d\rangle p$	MP: 1, 23

Figure 34: Example of Mon_d^g rule application on a complex formula.

5.4 PAR_{FULL} DERIVATION OF IND_s RULE

To reiterate, the ind_s rule is defined as follows:

$$\frac{\Phi, \varphi \wedge \langle\gamma\rangle\langle(\overline{\Phi}!;\gamma)^\times\rangle\langle\overline{\Phi}!\rangle\varphi}{\Phi, \langle\gamma^\times\rangle\varphi}\ \text{ind}_s$$

In [3], the ind_s rule was proven to be admissible in Par_{Full} through a number of claims that proves that if $\text{Par}_{\text{Full}} \vdash \bigvee\Gamma \vee (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle)\langle\overline{\Gamma}!\rangle\varphi)$, then it also holds that $\text{Par}_{\text{Full}} \vdash \bigvee\Gamma \vee \langle\gamma^\times\rangle\varphi$.

Suppose that we have:

$$\text{Par}_{\text{Full}} \vdash \bigvee\Gamma \vee (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle)\langle\overline{\Gamma}!\rangle\varphi)$$

Now, we can rewrite this disjunction as an implication which looks like:

$$(*)\ \text{Par}_{\text{Full}} \vdash \overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle)\langle\overline{\Gamma}!\rangle\varphi)$$

The first claim to prove here is referred to as Claim 2 in [4, p. 21]. This claim states that in the right conjunct in the conclusion of the implication the last modality $\langle\overline{\Gamma}!\rangle$ can be omitted, leaving us with:

CLAIM 2: $\text{Par}_{\text{Full}} \vdash \overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle)\varphi)$

This claim can be proven by first inferring $\overline{\Gamma} \to \varphi$ from (*). If we combine this with the demonic test axiom, $\text{Par}_{\text{Full}} \vdash \langle\overline{\Gamma}!\rangle\varphi \leftrightarrow \overline{\Gamma} \vee \varphi$, we can get $\text{Par}_{\text{Full}} \vdash \langle\overline{\Gamma}!\rangle\varphi \to \varphi$. By applying monotonicity to this formula twice, we can obtain $\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$. Together with this formula and (*), we have proven that Claim 2 indeed holds.

CLAIM 3: $\text{Par}_{\text{Full}} \vdash \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$

Claim 3 can be proven by first unfolding the fixpoint of $\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$ and then take the right conjunct, resulting in:

$$\text{Par}_{\text{Full}} \vdash \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

The consequent of this implication can be rewritten as $\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$ using the axiom for composition and demonic test on the first modality. If we apply Claim 2 to the left disjunct, we obtain:

$$\text{Par}_{\text{Full}} \vdash \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

Now, we can apply the Bar Induction rule for demonic iteration in order to get to Claim 3:

$$\text{Par}_{\text{Full}} \vdash \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

CLAIM 4: $\text{Par}_{\text{Full}} \vdash \overline{\Gamma} \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$

We can weaken the formula $\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$ to get:

$$\text{Par}_{\text{Full}} \vdash \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

By using the axiom for demonic test and composition, we can rewrite the consequent of this implication in order to get:

$$\text{Par}_{\text{Full}} \vdash \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

Now, we can use this implication to transform Claim 2 into:

$$\text{Par}_{\text{Full}} \vdash \overline{\Gamma} \to (\varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$$

The right conjunct in the consequent of this implication is the unfolding of $\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$. We can omit the left conjunct, so we can rewrite the formula as:

$$\mathrm{Par}_{\mathrm{Full}} \vdash \overline{\Gamma} \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

CLAIM 5: $\mathrm{Par}_{\mathrm{Full}} \vdash \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi$

Claim 5 can be derived by unfolding $\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$ and omitting the right conjunct, leaving the consequent with only φ.
Now the conclusion can be derived. If we perform the cut rule on Claim 4 and Claim 3, we get:

$$\mathrm{Par}_{\mathrm{Full}} \vdash \overline{\Gamma} \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$$

If we add $\langle\gamma^\times\rangle$ to Claim 5 by applying the Mon rule, we get:

$$\mathrm{Par}_{\mathrm{Full}} \vdash \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\varphi$$

Now we can apply the cut rule on the previous two formulas in order to get:

$$\mathrm{Par}_{\mathrm{Full}} \vdash \overline{\Gamma} \to \langle\gamma^\times\rangle\varphi$$

We can rewrite this conjunction back to a disjunction, leaving us with the desired conclusion:

$$\mathrm{Par}_{\mathrm{Full}} \vdash \bigvee\Gamma \vee \langle\gamma^\times\rangle\varphi$$

Now that we have the intuition on how the derive the ind_s rule in $\mathrm{Par}_{\mathrm{Full}}$, we can construct its full derivation, which can be found in Figure 35. The derivation has been divided into parts that derive each of the aforementioned claims. A horizontal line is placed between each part of the derivation.

0.	$\Gamma \vee (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	premise
1.	$\overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	$\mathrm{PL}_{D;C} : 0$
2.	$(\overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)) \to (\overline{\Gamma} \to \varphi)$	PL: $(A \to B \wedge C) \to (A \to B)$
3.	$\overline{\Gamma} \to \varphi$	MP: 1,2
4.	$\langle\overline{\Gamma}!\rangle\varphi \leftrightarrow \overline{\Gamma} \vee \varphi$	Ax9 !
5.	$\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi$	BE_L: 4
6.	$(\overline{\Gamma} \to \varphi) \to ((\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi) \to (\langle\overline{\Gamma}!\rangle\varphi \to \varphi))$	PL: $(A \to B) \to ((C \to A \vee B) \to (C \to B))$
7.	$(\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi) \to (\langle\overline{\Gamma}!\rangle\varphi \to \varphi)$	MP: $3, 6$
8.	$\langle\overline{\Gamma}!\rangle\varphi \to \varphi$	MP: $5, 7$
9.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Mon: 8
10.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Mon: 9
11.	$(\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi) \to (\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
12.	$\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi$	MP: 1,11
13.	$\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 12, 10$
14.	$(\overline{\Gamma} \to \varphi) \to ((\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
15.	$(\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: $3, 14$
16.	$\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: $13, 15$
		CLAIM 2
17.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax8 ×
18.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 17$
19.	$(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to$ $(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
20.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: 18,19
21.	$\langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax2 ;
22.	$\langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 21$
23.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax9 !

24.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 23$
25.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $20, 22$
26.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 25, 24$
27.	$(\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to ((\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to$ $((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B \wedge C) \to$ $((A \vee C) \to ((B \wedge C) \vee C))$
28.	$(\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to ((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: $16, 27$
29.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to ((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	PL_{CUT}: $26, 28$
30.	$((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL: $((A \wedge B) \vee B) \to B$
31.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $29, 30$
32.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BarInd}^\times : 31$
		CLAIM 3
33.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL: $A \to B \vee A$
34.	$\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 23$
35.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT} :$33, 34$
36.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 21$
37.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $35, 36$
38.	$\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $13, 37$
39.	$(\overline{\Gamma} \to \varphi) \to ((\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
40.	$(\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: $3, 39$
41.	$\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: $38, 40$
42.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax8 $\times$
43.	$\varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 42$
44.	$\overline{\Gamma} \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $41, 43$
		CLAIM 4
45.	$(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi)$	PL: $(A \to B \wedge C) \to (A \to B)$
46.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi$	MP: 18,45
		CLAIM 5
47.	$\overline{\Gamma} \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $44, 31$
48.	$\langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\varphi$	Mon: 46
49.	$\overline{\Gamma} \to \langle\gamma^\times\rangle\varphi$	PL_{CUT}: $47, 48$

Figure 35: $\mathrm{Par}_{\mathrm{Full}}$ derivation of ind_s rule.

6 Tool requirements

To ensure a successful completion of the transformation tool within the time frame of the project, a distinction between must-have, should-have, could-have and non-functional requirements is made. Such a subdivision of tasks ensures that the essential functionalities are present and working, possibly leaving time to implement less important features.

6.1 Must-have

The most important features are considered *must-have* requirements, such that without these the tool would not be able to serve its primary purpose. Therefore, these requirements cover the basics of the transformation itself.

The tool must have:

R-1.1 an abstract representation of the language $\mathcal{L}_{\text{NF}}$.

R-1.2 a concrete and abstract representation of a G proof tree.

R-1.3 the ability to parse a G proof tree into a concrete syntax tree, and transform it to an abstract syntax tree.

R-1.4 an abstract representation of a Par_{Full} proof.

R-1.5 the ability to transform any valid G proof (tree) into a valid Par_{Full} proof (list).

6.2 Should-have

To increase the usability and practicality of the tool, the following *should-have* requirements are defined.

The tool should have:

R-2.1 the ability to transform a Par_{Full} proof into a LaTeX document.

R-2.2 a main function which is able to take in a G proof file and output a LaTeX file of a Par_{Full} proof.

6.3 Could-have

In order to make the tool more comprehensive, maintainable and extendable, a number of *could-have* requirements are defined.

The tool could have:

R-3.1 an input checker that checks if the G proof is a valid proof.

R-3.2 an option to generate a LaTeX document that indicates which lines of the Par_{Full} proof are derived from a rule in the G proof. Currently, the tool keeps track of this, but it is not used as of yet.

6.4 Non-functional requirements

To add additional features that do not enhance the functionality of the tool, a number of *non-functional* requirements are defined.

The tool could:

R-4.1 be tested thoroughly, with unit tests for every function. This would improve the maintainability if the tool would be extended in the future.

R-4.2 have documentation for better understanding of its workings.

7 Implementation

In this section, the implementation of the tool is discussed. First, a description of the technology that was used for developing the tool is given. Second, the structure of the code is outlined by listing and explaining the different modules. Then, the syntax implementations of G, Par_{Full} and game logic are given, followed by a description of the transformations for every rule of G. Afterwards, the transformation of a Par_{Full} proof into a LaTeX document is given, and finally, the testing of the tool.

7.1 Language and Editor

The tool was written in the metaprogramming language *Rascal* [11]. Rascal is a domain-specific language that provides a programming environment where source code analysis and manipulation are integrated. It takes inspiration from many logic programming systems, making it highly suitable for the purpose of this project. Rascal provides functionality for easy definition and use of concrete and abstract syntax trees, which is used for defining the syntax of G, Par_{Full} and game logic. Additionally, Rascal's pattern matching allows for easy case distinction of game logic formulas and tree traversal.
The editor Eclipse was used for writing the source code and running the tool. Eclipse has a version called *IDE for RCP and RAP Developers* that has integrated Rascal support and is able to automatically build Rascal projects, check errors while developing, highlight syntax and browse through modules.

7.2 Code Structure

Inside the main directory of the Rascal Project, there is a *src* directory that contains the code modules of the tool. The directories *input* and *output* were added inside the main directory for storing the G proof input files and the LaTeX output files, respectively. The *src* directory contains the following modules:

- `GSyntax.rsc`: Contains the concrete syntax definition of G.
- `GLAST.rsc`: Contains the abstract syntax definitions of G, Par_{Full} and game logic.
- `CST2AST.rsc`: Contains the functions to transform a concrete syntax tree of a G proof into an abstract syntax tree.
- `GtoPar.rsc`: Contains the transformation functions for the basic rules and game operation rules.
- `DeepInferenceRules.rsc`: Contains the transformation functions for the deep inference rules.
- `StrongInductionRule.rsc`: Contains the transformation function of the strengthened induction rule.
- `LaTeXOutput.rsc`: Contains the transformation functions for transforming a Par_{Full} proof into a LaTeX document.
- `GtoParTool.rsc`: Contains the functions to perform all of the above operations, with an additional *main* function and a function to activate IDE support such as syntax highlighting.

7.3 Concrete and Abstract Syntax

7.3.1 Concrete Syntax of G

The input syntax for a G proof is defined by using Rascal's integrated functionality for syntax definition and parsing. Rascal then generates a scannerless parser based upon this definition, which then produces a 'parse tree'. Such a parse tree can then be processed further into a concrete syntax tree.
An input file of a G proof starts with the symbol "G" followed by curly brackets that contain the actual proof. This proof consists of a list of statements: the sequent; the rule upon which the sequent is based; and a semi-colon that marks the end of the line. A sequent in a statement is a list of formulas separated by white space and enclosed by square brackets. The first statement is the root of the proof tree, and therefore, it is the conclusion of the proof. The rule of a statement connects the next statement, which is the premise of this

rule, to the current statement, which is the conclusion of the rule. Most rules are *unary references*, meaning that the rule translates one sequent into another. Branching in the proof tree, however, occurs whenever the conjunction rule is applied, since the conclusion of this rule comes from the merging of two separate sequents. This is the only *binary reference* in G. These two branches are denoted by two consecutive pairs of curly brackets, where the first pair is a left branch and the right pair is a right branch. Every branch finally results in the application of Ax, which is then followed by the *leaf* keyword. An example of an input file of a G proof is given in Figure 36.

```
G {
    [<(a^d)*>~p <a^x>p] inds;
    [<(a^d)*>~p (p & <a><(<a^x>p!;a)^x><<a^x>p!>p)] and;
    {
        [<(a^d)*>~p p] iter;
        [(~p | <a^d><(a^d)*>~p) p] or;
        [~p <a^d><(a^d)*>~p p] weak;
        [~p p] Ax;
        leaf;
    }
    {
        [<(a^d)*>~p <a><(<a^x>p!;a)^x><<a^x>p!>p] iter;
        [(~p | <a^d><(a^d)*>~p) <a><(<a^x>p!;a)^x><<a^x>p!>p] or;
        [~p <a^d><(a^d)*>~p <a><(<a^x>p!;a)^x><<a^x>p!>p] weak;
        [<a^d><(a^d)*>~p <a><(<a^x>p!;a)^x><<a^x>p!>p] modm;
        [<(a^d)*>~p <(<a^x>p!;a)^x><<a^x>p!>p] dIter;
        [<(a^d)*>~p (<<a^x>p!>p & <(<a^x>p!;a)><(<a^x>p!;a)^x><<a^x>p!>p)] and;
        {
            [<(a^d)*>~p <<a^x>p!>p] dTest;
            [<(a^d)*>~p (<a^x>p | p)] or;
            [<(a^d)*>~p <a^x>p p] weak;
            [<(a^d)*>~p <a^x>p] Ax;
            leaf;
        }
        {
            [<(a^d)*>~p <(<a^x>p!;a)><(<a^x>p!;a)^x><<a^x>p!>p] concatd;
            [<(a^d)*>~p <<a^x>p!><a><(<a^x>p!;a)^x><<a^x>p!>p] dTest;
            [<(a^d)*>~p (<a^x>p | <a><(<a^x>p!;a)^x><<a^x>p!>p)] or;
            [<(a^d)*>~p <a^x>p <a><(<a^x>p!;a)^x><<a^x>p!>p] weak;
            [<(a^d)*>~p <a^x>p] Ax;
            leaf;
        }
    }
}
```

Figure 36: Example of a G input file

In `GSyntax.rsc` in Appendix X, the code for the concrete syntax definition of G can be found. It starts by defining a *SGProof*, which merely states that a G proof is of the form "G SGPart* ". A *SGPart* is a statement, which can be a unary reference, a binary reference (conjunction rule with two branches) or a leaf. The asterisk in the definition of SGProof can be used in Rascal to denote that an expression can be repeated zero or more times. A statement of either a unary or binary reference consists of game logic formulas denoted by *SGExpr**. *SGGame* represents games and they occur in SGExpr in case of a modality operator. These formulas and games are defined in terms of SGExpr, SGGame and ASCII characters such as "*" and "?". For example, the demonic test is defined as a SGExpr followed by a "!" character.

7.3.2 Abstract syntax

In Rascal, abstract syntax definitions are defined using algebraic data types (ADT) that can be analyzed by pattern matching. Upon declaration, these ADTs can be initialized as variables through their constructors. On the lowest level, there are atomic games and atomic propositions, which are both represented by either a string or a string with an integer as a subscript. At one lever higher there are game logic formulas and games, which are mutually recursively defined. A formula can either be an atomic proposition or one of the propositional connectives which consist of other formulas or games. For example, recall the modality formula $\langle\gamma\rangle\varphi$ which consists of a game γ and a formula φ. Games can either be an atomic game or one of the game connectives that consist of other games or formulas.
The abstract syntax tree of a G proof is defined recursively, where the initial proof, the root, is like the *main*

proof and each node in the tree is a *subproof*. A node in the tree can either be a *leaf*, a *unary inference* or a *binary inference*. As noted before, a leaf marks the end of a branch and therefore, it does not contain any parameters. A unary inference contains a sequent as a list of game logic formulas, a rule, and the subproof. The rules of G are merely names and do not need to contain any other information. The subproof contains the sequent that is the premise of the current sequent, thus the recursion lies in these subproofs. Binary inferences come from applications of the conjunction rule, so instead of one subproof they contain two subproofs, the left and the right branch. Since the conjunction rule is the only binary operator, it is unnecessary to have a rule as a parameter.
A proof in Par_{Full} is not a tree but a list of statements, thus the abstract syntax of Par_{Full} is implemented as a list of *proof lines*. A proof line can be one of four alternatives, where each alternative represents some type of statement. Primarily, a statement is a list of game logic formulas (disjunction of formulas) and a justification. A justification is similarly implemented as the rules in G, yet some of these justifications contain integers in case they need to refer to earlier lines in the proof. The other three alternatives can be used for improving the readability of the LaTeX proofs. One has an additional colour parameter, which is a string that can be used to colour statements in a proof with the `xcolor` package. The other two can be used to mark the begin and end of a derivation of a G rule, such that in the LaTeX proof we can see which sections of the Par_{Full} proof correspond to the rules in the G proof.

7.4 G to Par_{Full} Transformation

The transformations of the rules of G into derivations of Par_{Full} are recursively performed on a G proof tree by pattern matching on the rule and the game logic formulas. Every node in the proof tree is either a leaf, a unary inference or a binary inference. Whenever Ax is encountered, the derivation of that branch comes to an end and the Par_{Full} proof list at that point is terminated. The leaf node is used to terminate a recursive pattern in derivations of the deep inference rules, which is explained later. A unary inference is identified by pattern matching on the rule such that the correct transformation function is called. The conclusion and premise of the rule are determined by the sequent of the current node and the sequent of the next node in the proof tree, respectively. The active formulas in the sequents can be extracted by set difference using Rascal's integrated support for set operations. The active formula in the conclusion, and similarly for the premise, can be extracted by taking the difference between the current sequent and the next sequent. The presence of side formulas can be determined by set difference between a sequent and its active formula. If this difference results in an empty set, then there are no side formulas in that sequent. Every transformation function returns a list of Par_{Full} statements, where the first line is a recursive call on the subproof of a node. This recursion continues until the Ax axiom has been encountered in every branch of the tree, thus resulting in a complete Par_{Full} proof.

7.4.1 Basic rules

For a description of the basic rules of G and their transformations, we refer back to Figure 5 and Section 5.1, respectively.
The Ax axiom marks the end of a branch in the proof, and in Par_{Full} the concluding sequent of this axiom is directly translated into a disjunction of formulas. Therefore, the Ax transformation function merely returns a Par_{Full} list containing only one statement, which is the disjunction of formulas followed by the law of excluded middle as a propositional tautology for the justification.
The weak rule consists of simply extending a sequent with another formula. Therefore, the premise and conclusion of this rule always have side formulas. This weakening formula can be extracted by set difference between the current and next sequent. Then we return a recursive call upon the subproof with the additional proof lines that are needed for the weak rule.
The disjunction rule separates two formulas from a sequent into a disjunction. If there are no side formulas such that these two formulas are the only formulas in a sequent, then this rule requires no derivation in Par_{Full}. After all, a sequent is directly translated into a disjunction of formulas.
In the transformation of the conjunction rule two recursive calls and the derivation in Par_{Full} is returned. These recursive calls are executed on the left and right branch of the subproof, respectively. The derivation is based on Figure 17 when there are no side formulas, and the derivation is based on Figure 18 when there are side formulas.

For the mod_m rule, the game and dual game that are added in the conclusion as well as the formulas φ and ψ can be extracted by pattern matching on the game logic formulas. The derivation is then implemented exactly as in Figure 19.

7.4.2 Game operation rules

For a description of the game operation rules and their transformations, we refer back to Figure 6 and Section 5.2, respectively. Since each derivation of the game operation rules has an identical structure, all these derivations can be handled by a single function. The only difference between the derivations are the axioms that are used and these directly relate to the game operation rule. So, each game operation transformation function calls the same generic function that returns the same derivation but where the axioms are based on the rule at hand.

7.4.3 Deep inference rules

To reiterate, the deep inference rules and their transformations can be found in Figure 7 and Section 5.3, respectively. Just as with the game operation rules, the deep inference rules follow equal patterns in their derivations. They only differ in the case of the atomic level where the rule is applied. Therefore, a generic transformation function can handle the derivations of Mon_d^g, Mon_d^f and $;_d$. This generic function calls the transformation function that handles the derivation from the active formula of the premise to the active formula of the conclusion. The transformation function that is called is based upon case distinction by pattern matching on the game logic formula of the premise and conclusion. In each case, the recursive step is achieved by calling the main transformation function. The parameter for this function is a unary inference with as sequent the inner part of the current conclusion and as rule the current deep inference rule. The subproof is another unary inference with as sequent the inner part of the current premise, a special rule that indicates it is the premise of a recursive step, and as subproof a leaf such that the recursion stops immediately. This way, the inner part of a case is derived. In cases where there are a left and right side (composition, angelic choice, conjunction), this main function is called on both sides. If in such a function call the premise and conclusion are identical, it means that the rule is not applied in this particular game or formula. In this case, the function returns an empty list.

7.4.4 Strengthened induction rule

The strengthened induction rule and its derivation can be found in Figure 11 and Section 5.4, respectively. The implementation of this derivation is straightforward as it is merely a long list of Par_{Full} lines. Since the derivation requires many subformulas of the premise, pattern matching on this formula is used to create variables that match these subformulas. This greatly improves the readability of the derivation code.

7.5 LaTeX Output

G proofs and Par_{Full} proofs can both be transformed into a LaTeX document by using Rascal's string templates for code generation.
The module `LaTeXOutput.rsc` takes an abstract syntax tree of a G proof and outputs the generated code into a Tex file in the output folder. The function that is called first is the template for the LaTeX document, providing the document class and packages of the document. For the proof trees, a package called `prooftrees` is used. Within the prooftree environment, the function is called that transforms the G proof into LaTeX code. This transformation is performed recursively on the G proof, in a similar way as with the transformation from G to Par_{Full}. The sequents of G can be transformed using reduction statements.
The Par_{Full} proofs is initiated by a for loop that runs over all lines in the proof that calls the transformation function. This function differentiates between the alternatives by pattern matching. Again, the disjunction of formulas is transformed into code by reduction.

7.6 Main function

The module `GtoParTool.rsc` contains all the functions to transform an input file of a G proof into a Par_{Full} proof in LaTeX. The main function takes in a string indicating the input file of a G proof. Then it calls a

function that parses the file, generates a concrete syntax tree and transforms it into an abstract syntax tree. Then a function is called that transforms the abstract syntax tree into a Par_{Full} proof. Finally, the Par_{Full} proof is converted into a Tex file and is put in the output folder.

7.7 Testing

The testing of the tool was initially done by constructing proofs that function as unit tests for every G rule. For example, the unit test for the mod_m rule was the following proof:

$$\dfrac{\dfrac{}{\langle a^* \rangle p, \langle a^* \rangle \neg p}\,\text{Ax}}{\langle b^d \rangle \langle a^* \rangle p, \langle b \rangle \langle a^* \rangle \neg p}\,\text{mod}_m$$

which then results in the following Par_{Full} proof:

1.	$\langle a^* \rangle p \vee \langle a^* \rangle \neg p$	PL: $A \vee \neg A$
2.	$\neg \langle a^* \rangle p \rightarrow \langle a^* \rangle \neg p$	$\text{PL}_{D;C}$: 1
3.	$\langle b \rangle \neg \langle a^* \rangle p \rightarrow \langle b \rangle \langle a^* \rangle \neg p$	Mon: 2
4.	$\neg \langle b \rangle \neg \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p$	$\text{PL}_{C;D}$: 3
5.	$\langle b^d \rangle \langle a^* \rangle p \leftrightarrow \neg \langle b \rangle \neg \langle a^* \rangle p$	Ax6 *dual*
6.	$(\langle b^d \rangle \langle a^* \rangle p \leftrightarrow \neg \langle b \rangle \neg \langle a^* \rangle p) \rightarrow ((\neg \langle b \rangle \neg \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p) \rightarrow (\langle b^d \rangle \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p))$	PL: $(A \leftrightarrow B) \rightarrow (B \vee C \rightarrow A \vee C)$
7.	$(\neg \langle b \rangle \neg \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p) \rightarrow (\langle b^d \rangle \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p)$	MP: 5, 6
8.	$\langle b^d \rangle \langle a^* \rangle p \vee \langle b \rangle \langle a^* \rangle \neg p$	MP: 4, 7

Gradually, proofs with increasing complexity were used to test the tool and were checked manually manually for correctness. An example of such a test proof can be found in Appendix IX.

8 Results

One of the main differences between proofs in G and proofs in Par_{Full} is the structure and size. In Table 1, the number of lines in Par_{Full} are given for the basic rules, game operation rules and strengthened induction rule. In the left column the G rule is stated, and in the right columns the number of lines in Par_{Full} in the absence of side formulas and in the presence of side formulas, respectively. Each of these rules map to a fixed number of lines in Par_{Full}. Normally, if the premise of a rule has side formulas, the number of lines increases a little since the disjunction of formulas needs to be converted to a conditional in order to perform the cut rule, and afterwards the conditional needs to be converted back into a disjunction. The strengthened induction rule is an exception in this regard, since this rule depends on the presence of side formulas. If the premise of an ind_s application has no side formulas, the formula $p \vee \neg p$ is introduced as a side formula, such that the number of lines in Par_{Full} increases by three.
In the absence of side formulas, the $\vee$ rule can be omitted entirely from a Par_{Full} proof. Since the $\vee$ rule is often applied in a G proof, this can potentially reduce a Par_{Full} proof depending on whether there are side formulas when applied.

G rule	**# of Par lines**	
	w/o Φ	**with Φ**
Ax	1	N/A
weak	N/A	2
or	0	2
and	3	6
mod_m	7/11	N/A
$*/\times/\sqcup/\sqcap/!/?$	3	5
ind_s	50	53

Table 1: Lengths of Par_{Full} derivations (right columns) of basic rules, game operation rules and strengthened induction rule (left column).

The derivations of the deep inference rules do not result in a fixed number of lines in Par_{Full}. Instead, the number of lines increases linearly with respect to the depth of the rule application, see Table 2. For each of the deep inference rules, the Par_{Full} proof takes up at least one or three lines, depending on the presence of side formulas, plus additional lines for the atomic case. Then, for each case where the game or formula is wrapped in, the Par_{Full} proof increases by a certain amount of lines. If we have the formula

$$\langle g \rangle \varphi \wedge \langle ((g^d \sqcap (p?; ((h \sqcup (p \to q)!)^*)))^\times \rangle \psi$$

and the Mon_d^f rule is applied to the formula q, then we see that q is wrapped in a number of games and formulas. From the deepest level to the shallowest level, these are: $[\to, !, \sqcup, *, ;, \sqcap, \times, \wedge]$. Suppose there are no side formulas, then the number of lines in Par_{Full} would add up to: $2 + 8 + 8 + 10 + 7 + 8 + 10 + 2 = 55$ lines.

Deep inf rule	**# of Par lines**	
	w/o Φ	**with Φ**
Deep inf	1	3
Mon_d^g atomic	7	N/A
Mon_d^f atomic	4	N/A
$;_d$ atomic	3	N/A
case $!/?/\sqcup/\sqcap$	8	N/A
case ;	6/7	N/A
case $*/\times$	10	N/A
case $\wedge, \vee, \to, \leftrightarrow$	2	N/A

Table 2: Lengths of Par_{Full} derivations (right columns) of deep inference rules (left column).

Despite that every G proof results in a larger Par_{Full} proof, in some cases the transformed Par_{Full} proof is

not the most simplified proof. Whenever in a G proof multiple branches close with identical axioms, they appear as many times in the Par_{Full} proof. Other formulas can also appear multiple times in a Par_{Full} proof, if they occur in multiple derivations of G rules. Although this is likely, it is difficult to anticipate how often this will occur.

Consider the G proof in Figure 37. In G, it is normal to have duplicate sequents in separate branches. However, when the proof is transformed from G to Par_{Full}, these branches are derived separately and therefore, the Par_{Full} proof contains duplicate statements. In the Par_{Full} derivation in Figure 38, there are two equal groups of statements coloured red and blue. The shortest version of this proof can be found in Figure 39, where the duplicate statements are omitted.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p, \neg p}\,\text{Ax}}{p, \langle g\rangle q, \neg p}\,\text{weak}}{p \vee \langle g\rangle q, \neg p}\,\vee}{\langle p!\rangle\langle g\rangle q, \neg p}\,!}{p \wedge q, \langle p!\rangle\langle g\rangle q, \neg p}\,\text{weak}}{(p \wedge q) \vee \langle p!\rangle\langle g\rangle q, \neg p}\,\vee}{\langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q, \neg p}\,! \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{p, \neg p}\,\text{Ax}}{p, \langle g\rangle q, \neg p}\,\text{weak}}{p \vee \langle g\rangle q, \neg p}\,\vee}{\langle p!\rangle\langle g\rangle q, \neg p}\,!}{\neg p, \langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q}\,\wedge$$

Figure 37: Example of a G proof with duplicate sequents.

1.	$p \vee \neg p$	PL: $A \vee \neg A$
2.	$(p \vee \neg p) \to (p \vee \neg p \vee \langle g\rangle q)$	PL: $A \to A \vee B$
3.	$p \vee \neg p \vee \langle g\rangle q$	MP: $1, 2$
4.	$(p \vee \langle g\rangle q \vee \neg p) \to (\neg p \vee (p \vee \langle g\rangle q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
5.	$p(\neg p \vee (p \vee \langle g\rangle q))$	MP: $3, 4$
6.	$(\neg(\neg p) \to (p \vee \langle g\rangle q))$	$\text{PL}_{D;C}$: 5
7.	$(\langle p!\rangle\langle g\rangle q \leftrightarrow (p \vee \langle g\rangle q))$	Ax9 !
8.	$((p \vee \langle g\rangle q) \to \langle p!\rangle\langle g\rangle q)$	BE_R: 7
9.	$(\neg(\neg p) \to \langle p!\rangle\langle g\rangle q)$	PL_{CUT}: $6, 8$
10.	$\langle p!\rangle\langle g\rangle q \vee \neg p$	$\text{PL}_{C;D}$: 9
11.	$(\langle p!\rangle\langle g\rangle q \vee \neg p) \to (\langle p!\rangle\langle g\rangle q \vee \neg p \vee (p \wedge q))$	PL: $A \to A \vee B$
12.	$\langle p!\rangle\langle g\rangle q \vee \neg p \vee (p \wedge q)$	MP: $10, 11$
13.	$((p \wedge q) \vee \langle p!\rangle\langle g\rangle q \vee \neg p) \to (\neg p \vee ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
14.	$(\neg p \vee ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	MP: $12, 13$
15.	$(\neg(\neg p) \to ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	$\text{PL}_{D;C}$: 14
16.	$(\langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \leftrightarrow ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	Ax9 !
17.	$(((p \wedge q) \vee \langle p!\rangle\langle g\rangle q) \to \langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	BE_R: 16
18.	$(\neg(\neg p) \to \langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	PL_{CUT}: $15, 17$
19.	$\langle (p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \vee \neg p$	$\text{PL}_{C;D}$: 18
20.	$p \vee \neg p$	PL: $A \vee \neg A$
21.	$(p \vee \neg p) \to (p \vee \neg p \vee \langle g\rangle q)$	PL: $A \to A \vee B$
22.	$p \vee \neg p \vee \langle g\rangle q$	MP: $20, 21$
23.	$(p \vee \langle g\rangle q \vee \neg p) \to (\neg p \vee (p \vee \langle g\rangle q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
24.	$(\neg p \vee (p \vee \langle g\rangle q))$	MP: $22, 23$
25.	$(\neg(\neg p) \to (p \vee \langle g\rangle q))$	$\text{PL}_{D;C}$: 24
26.	$(\langle p!\rangle\langle g\rangle q \leftrightarrow (p \vee \langle g\rangle q))$	Ax9 !
27.	$((p \vee \langle g\rangle q) \to \langle p!\rangle\langle g\rangle q)$	BE_R: 26
28.	$(\neg(\neg p) \to \langle p!\rangle\langle g\rangle q)$	PL_{CUT}: $25, 27$
29.	$\langle p!\rangle\langle g\rangle q \vee \neg p$	$\text{PL}_{C;D}$: 28

30.	$(\neg(\neg p) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	$\text{PL}_{D;C}$: 19
31.	$(\neg(\neg p) \to \langle p!\rangle\langle g\rangle q)$	$\text{PL}_{D;C}$: 29
32.	$((\neg(\neg p) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q) \to$ $((\neg(\neg p) \to \langle p!\rangle\langle g\rangle q) \to (\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
33.	$((\neg(\neg p) \to \langle p!\rangle\langle g\rangle q) \to (\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q)))$	MP: 30, 32
34.	$(\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))$	MP: 31, 33
35.	$(\neg p \vee (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))$	$\text{PL}_{C;D}$: 34

Figure 38: Par_{Full} proof with duplicate formulas.

1.	$p \vee \neg p$	PL: $A \vee \neg A$
2.	$(p \vee \neg p) \to (p \vee \neg p \vee \langle g\rangle q)$	PL: $A \to A \vee B$
3.	$p \vee \neg p \vee \langle g\rangle q$	MP: 1, 2
4.	$(p \vee \langle g\rangle q \vee \neg p) \to (\neg p \vee (p \vee \langle g\rangle q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
5.	$p(\neg p \vee (p \vee \langle g\rangle q))$	MP: 3, 4
6.	$(\neg(\neg p) \to (p \vee \langle g\rangle q))$	$\text{PL}_{D;C}$: 5
7.	$(\langle p!\rangle\langle g\rangle q \leftrightarrow (p \vee \langle g\rangle q))$	Ax9 !
8.	$((p \vee \langle g\rangle q) \to \langle p!\rangle\langle g\rangle q)$	BE_R: 7
9.	$(\neg(\neg p) \to \langle p!\rangle\langle g\rangle q)$	PL_{CUT}: 6, 8
10.	$\langle p!\rangle\langle g\rangle q \vee \neg p$	$\text{PL}_{C;D}$: 9
11.	$(\langle p!\rangle\langle g\rangle q \vee \neg p) \to (\langle p!\rangle\langle g\rangle q \vee \neg p \vee (p \wedge q))$	PL: $A \to A \vee B$
12.	$\langle p!\rangle\langle g\rangle q \vee \neg p \vee (p \wedge q)$	MP: 10, 11
13.	$((p \wedge q) \vee \langle p!\rangle\langle g\rangle q \vee \neg p) \to (\neg p \vee ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
14.	$(\neg p \vee ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	MP: 12, 13
15.	$(\neg(\neg p) \to ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	$\text{PL}_{D;C}$: 14
16.	$(\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \leftrightarrow ((p \wedge q) \vee \langle p!\rangle\langle g\rangle q))$	Ax9 !
17.	$(((p \wedge q) \vee \langle p!\rangle\langle g\rangle q) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	BE_R: 16
18.	$(\neg(\neg p) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	PL_{CUT}: 15, 17
19.	$\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \vee \neg p$	$\text{PL}_{C;D}$: 18
20.	$(\neg(\neg p) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q)$	$\text{PL}_{D;C}$: 19
21.	$(\neg(\neg p) \to \langle p!\rangle\langle g\rangle q)$	$\text{PL}_{D;C}$: 10
22.	$((\neg(\neg p) \to \langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q) \to$ $((\neg(\neg p) \to \langle p!\rangle\langle g\rangle q) \to (\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
23.	$((\neg(\neg p) \to \langle p!\rangle\langle g\rangle q) \to (\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q)))$	MP: 20, 22
24.	$(\neg(\neg p) \to (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))$	MP: 21, 23
25.	$(\neg p \vee (\langle(p \wedge q)!\rangle\langle p!\rangle\langle g\rangle q \wedge \langle p!\rangle\langle g\rangle q))$	$\text{PL}_{C;D}$: 24

Figure 39: Simplified proof of Par_{Full} proof with duplicate formulas.

Line 15 in the Par_{Full} proof in Figure 39 also illustrates that proofs in Hilbert systems are more difficult to read than proofs in sequent systems. Instances of propositional tautologies tend to get very large, especially when the formulas in a proof are complex, making it harder to understand the logic of such a statement.

9 Conclusion

9.1 Evaluation

The research question of this thesis was "How can the proof transformation from the Gentzen system G to the Hilbert system Par_{Full} be implemented?". A number of requirements were defined to describe the functionality the tool needs and other non-functional aspects. Must-have, should-have, could-have and non-functional requirements were defined to differentiate between the level of importance. All the must-have requirements, R-1.1 to R-1.5, have been successfully implemented. The tool has an abstract representation of $\mathcal{L}_{\text{NF}}$ (R-1.1) and Par_{Full} (R-1.4), and a concrete and abstract representation of G (R-1.2). The code in [25] for the abstract syntax of $\mathcal{L}_{\text{NF}}$ and CloG has been used to implement an abstract syntax representation of $\mathcal{L}_{\text{NF}}$ and G. Additionally, the tool can parse a G proof tree into a concrete syntax tree and transform it to an abstract syntax tree (R-1.3), and transform a G proof into a Par_{Full} proof (R-1.5). All should-have requirements, R-2.1 to R-2.2, have also been implemented, such that a Par_{Full} proof can be converted into a LaTeX document (R-2.1), and the tool has a main function that can take a string of an input file and output the Par_{Full} proof in LaTeX (R-2.2). The could-have requirements have not been implemented. R-3.1, which suggests a validity checker for input proofs, was too time-consuming for the scope of this thesis, and a proper solution for R-3.2 (indicator of G rule derivations in Par_{Full} proof in LaTeX) could not yet be found. The non-functional requirements, R-4.1 and R-4.2 also have not been implemented. Currently, the testing of the tool is limited to inputting unit test proofs and complex proofs to check if the output Par_{Full} proofs are correct, and unit tests for every function are absent (R-4.1). Documentation of the tool has not been written so that gaining understanding of the tool is limited to this thesis and code comments (R-4.2).

9.2 Future work

Initially, the purpose of this project was to implement the transformation step from G to Par. Due to time constraints and the G-to-Par transformation being too nontrivial, it was decided that first the transformation step from G to Par_{Full} should be implemented. Therefore, in a future project the transformation step from Par_{Full} to Par can be implemented. This can be done by deriving the axioms 7-9 and the $\text{BarInd}^{\times}$ rule from Figure 4.2 in Par.

The tool could be improved in several ways. First, all functions could be unit tested using Rascal's integrated unit test framework. This could help with maintaining and checking the functionality of the tool if the tool were to be extended. Second, as was mentioned in Section 8, G proofs with duplicate sequents in separate branches result in Par_{Full} proofs with duplicate statements. This can be prevented by checking for every statement in a Par_{Full} proof if that statement already occurs earlier in the proof, and removing these duplicate statements from the Par_{Full} proof. Implementing this could enforce a change of implementation, since justification references would have to be altered.

In this current implementation of the transformation step from G to Par_{Full}, a small number of propositional tautologies were chosen to derive all the G rules in Par_{Full}. In more classical Hilbert systems only a few propositional tautologies are used, from which other instances can be derived by performing uniform substitution. If this were to be used in the G to Par_{Full} transformation, it could display a finer difference between Gentzen systems and Hilbert systems. However, it will also further increase the size and decrease the readability of Par_{Full} proofs.

References

[1] B. Afshari and G. Leigh. Cut-free completeness for modal mu-calculus. *LICS 2017*, pages 1–12, 2017.

[2] M. Baaz and A. Leitsch. Complexity of resolution proofs and function introduction. *Ann. Pure Appl. Log.*, (57):181–215, 1992.

[3] S. Enqvist, H. Hansen, C. H., Kupke, J. Marti, and Y Venema. Completeness for game logic. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. LICS, 2019.

[4] Sebastian Enqvist, Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. Completeness for game logic. *CoRR*, abs/1904.07691, 2019.

[5] P. Felty, A. Using extended tactics to do proof transformations. Technical report, University of Pennsylvania Department of Computer and Information Science Technical Report No. MS- CIS-86-89, 1986.

[6] G. Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39(1):176–210, 1935.

[7] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik,*, 38:173–198, 1931.

[8] H. Hansen, H. Monotone modal logic. Master's thesis, University of Amsterdam, 2003.

[9] E. Hazard and D. Kuperberg. Cyclic proofs for transfinite expressions. In *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Computer Science Logic, 2022.

[10] D. Hilbert and P. Bernays. Logische grundlagen der mathematik, 1923. Lecture notes by Paul Bernays, with handwritten notes by Hilbert. Hilbert-Nachlaß, Niedersächsische Staats- und Universitätsbibliothek.

[11] P. Klint, T. Storm, van der, and J. J Vinju. Rascal: a domain specific language for source code analysis and manipulation. In *In Proceedings of the Ninth IEEE International Working Conference on Source Code Analysis and Manipulation, September 20-21, 2009, Edmonton, Canada*, pages 168–177. IEEE Computer Society, 2009.

[12] D. Kozen. Results on the propositional -calculus. *ICALP*, 140:348—-359, 1982.

[13] S. A. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.

[14] D. Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, October 1983.

[15] R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

[16] R. Parikh. The logic of games and its applications. *Annals of Discrete Mathematics*, 24, 1985.

[17] M. Pauly. Game logic for game theorists. Technical report, Centrum voor Wiskunde en Informatica, 2000.

[18] M. Pauly and R. Parikh. Game logic: An overview. *Studia Logica: An International Journal for Symbolic Logic*, 75(2):165–182, 2003.

[19] F. Pfenning. *Proof Transformations in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, January 1987.

[20] N. Rescher. Russell and modal logic. *George Allen and Unwin*, page 146, 1979.

[21] Bakker J. Scott, D. A theory of programs. 1969.

[22] G. Takeuti. *Two Applications of Logic to Mathematics*. Princeton University Press, 1978.

[23] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[24] Anita Wasilewska. *Logics for Computer Science: Classical and Non-classical*. Springer, 1 edition, 2018.

[25] C. Worthington. Proof transformations for game logic. Bachelor's Thesis: University of Groningen, 2021.

I Table of Propositional Tautologies Used in Par_{Full} Derivations

$A \to (B \to (A \wedge B))$
$A \vee B \to B \vee A$
$A \vee B \to (\neg A \to B)$
$A \vee B \to (\neg B \to A)$
$(\neg A \to B) \to A \vee B$
$(A \to B \wedge C) \to (A \to B)$
$(A \to B \wedge C) \to (A \to C)$
$(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
$(A \leftrightarrow B) \to (B \vee C \to A \vee C)$
$A \vee \neg A$
$A \to A \vee B$
$A \to (B \to A \wedge B)$
$(A \to B) \to (A \to B \vee C)$
$(A \to B) \to (A \to C \vee B)$
$(A \vee B \to C) \to (B \to C)$
$(A \vee B \to C) \to (A \to C)$
$((A \to B \vee D) \wedge (C \to B \vee D)) \to (A \vee C \to B \vee D)$
$A \to B \vee A$
$(A \to B) \to (A \vee C \to B \vee C)$
$(A \to B) \to (C \vee A \to C \vee B)$
$(A \to B) \to (A \wedge C \to B \wedge C)$
$(A \to B) \to (C \wedge A \to C \wedge B)$
$(A \vee B \vee C) \to (A \vee (B \vee C))$
$((A \wedge B) \wedge (A \to C)) \to (C \wedge B)$
$((A \wedge B) \wedge (B \to C)) \to (A \wedge C)$
$((A \vee B) \wedge (A \to C)) \to (C \vee B)$
$((A \vee B) \wedge (B \to C)) \to (A \vee C)$
$((A \to B) \wedge (A \to C)) \to (B \to C)$
$((A \to B) \wedge (C \to A)) \to (C \to B)$
$((A \to B) \wedge (A \leftrightarrow C)) \to (B \leftrightarrow C)$
$((A \to B) \wedge (C \leftrightarrow A)) \to (C \leftrightarrow B)$
$A \to ((A \to B) \to B)$
$(A \to B) \to ((C \to A \vee B) \to (C \to B)))$
$(A \to B \wedge C) \to ((A \vee C) \to ((B \wedge C) \vee C))$
$((A \wedge B) \vee B) \to B$

II Derivable Rules for Propositional Logic

Derivable rule: Biconditional elimination left (BE_L).

1.	$A \leftrightarrow B$	premise
2.	$(A \leftrightarrow B) \rightarrow (A \rightarrow B)$	PL
3.	$A \rightarrow B$	MP: 1,2

Derivable rule: Biconditional elimination right (BE_R).

1.	$A \leftrightarrow B$	premise
2.	$(A \leftrightarrow B) \rightarrow (B \rightarrow A)$	PL
3.	$B \rightarrow A$	MP: 1,2

Derivable rule: disjunction to conjunction ($\text{PL}_{D;C}$).

1.	$A \vee B$	premise
2.	$(A \vee B) \rightarrow (\neg A \rightarrow B)$	PL
3.	$\neg A \rightarrow B$	MP: 1,2

Derivable rule: conjunction to disjunction ($\text{PL}_{C;D}$).

1.	$A \rightarrow B$	premise
2.	$(A \rightarrow B) \rightarrow (\neg A \vee B)$	PL
3.	$\neg A \vee B$	MP: 1,2

Derivable rule: cut rule (PL_{CUT}).

1.	$A \rightarrow B$	premise
2.	$B \rightarrow C$	premise
3.	$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$	PL
4.	$(B \rightarrow C) \rightarrow (A \rightarrow C)$	MP: $1, 3$
5.	$A \rightarrow C$	MP: $2, 4$

III PAR_{FULL} DERIVATIONS OF BASIC RULES

Par_{Full} derivation of Ax.

1.	$\bigvee \Phi \vee \neg \bigvee \Phi$	PL

Par_{Full} derivation of weak rule.

1.	$\bigvee \Phi$	premise
2.	$\bigvee \Phi \to \bigvee \Phi \vee \varphi$	PL: $A \to A \vee B$
3.	$\bigvee \Phi \vee \varphi$	MP: 1,2

Par_{Full} derivation of $\vee$ rule.

1.	$\bigvee \Phi \vee \varphi \vee \psi$	premise
2.	$\bigvee \Phi \vee \varphi \vee \psi \to \bigvee \Phi \vee (\varphi \vee \psi)$	PL: $A \vee B \vee C \to A \vee (B \vee C)$
3.	$\bigvee \Phi \vee (\varphi \vee \psi)$	MP: 1,2

Par_{Full} derivation of $\wedge$ rule (without side formulas).

1.	φ	premise
2.	ψ	premise
3.	$\varphi \to (\psi \to (\varphi \wedge \psi))$	PL: $A \to (B \to (A \wedge B))$
4.	$\psi \to (\varphi \wedge \psi)$	MP: $1, 3$
5.	$\varphi \wedge \psi$	MP: $2, 4$

Par_{Full} derivation of $\wedge$ rule (with side formulas).

1.	$\bigvee \Phi \vee \varphi$	premise
2.	$\bigvee \Phi \vee \psi$	premise
3.	$\neg \bigvee \Phi \to \varphi$	$\text{PL}_{D;C}$: 1
4.	$\neg \bigvee \Phi \to \psi$	$\text{PL}_{D;C}$: 2
5.	$(\neg \bigvee \Phi \to \varphi) \to ((\neg \bigvee \Phi \to \psi) \to (\neg \bigvee \Phi \to (\varphi \wedge \psi)))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
6.	$(\neg \bigvee \Phi \to \psi) \to (\neg \bigvee \Phi \to (\varphi \wedge \psi))$	MP: $3, 5$
7.	$\neg \bigvee \Phi \to (\varphi \wedge \psi)$	MP: 4,6
8.	$\bigvee \Phi \vee (\varphi \wedge \psi)$	$\text{PL}_{C;D}$: 7

$\mathrm{Par}_{\mathrm{Full}}$ derivation of mod_m rule.

1.	$\varphi \vee \psi$	premise
2.	$(\varphi \vee \psi) \to (\psi \vee \varphi)$	PL: $(A \vee B) \to (B \vee A)$
3.	$\psi \vee \varphi$	MP: $1, 2$
4.	$\neg\psi \to \varphi$	$\mathrm{PL}_{D;C}$: 3
5.	$\langle g\rangle\neg\psi \to \langle g\rangle\varphi$	Mon: 4
6.	$\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi$	$\mathrm{PL}_{C;D}$: 5
7.	$\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi$	Ax6 dual
8.	$(\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi) \to ((\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi))$	PL: $(A \leftrightarrow B) \to ((B \vee C) \to (A \vee C))$
9.	$(\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi)$	MP: 7,8
10.	$\langle g^d\rangle\psi \vee \langle g\rangle\varphi$	MP: 6,9
11.	$(\langle g^d\rangle\psi \vee \langle g\rangle\varphi) \to (\langle g\rangle\varphi \vee \langle g^d\rangle\psi)$	PL: $(A \vee B) \to (B \vee A)$
12.	$\langle g\rangle\varphi \vee \langle g^d\rangle\psi$	MP: 10,11

$\mathrm{Par}_{\mathrm{Full}}$ derivation of mod_m rule (order of game and dual game switched).

1.	$\psi \vee \varphi$	premise
2.	$\neg\psi \to \varphi$	$\mathrm{PL}_{D;C}$: 1
3.	$\langle g\rangle\neg\psi \to \langle g\rangle\varphi$	Mon: 2
4.	$\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi$	$\mathrm{PL}_{C;D}$: 3
5.	$\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi$	Ax6 dual
6.	$(\langle g^d\rangle\psi \leftrightarrow \neg\langle g\rangle\neg\psi) \to ((\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi))$	PL: $(A \leftrightarrow B) \to ((B \vee C) \to (A \vee C))$
7.	$(\neg\langle g\rangle\neg\psi \vee \langle g\rangle\varphi) \to (\langle g^d\rangle\psi \vee \langle g\rangle\varphi)$	MP: 5,6
8.	$\langle g^d\rangle\psi \vee \langle g\rangle\varphi$	MP: 4,7

IV PAR_{FULL} DERIVATIONS OF GAME OPERATION RULES

Par_{Full} derivation of *-rule (with side formulas).

1.	$\bigvee\Phi \vee (\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi)$	premise
2.	$\neg\bigvee\Phi \to (\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi)$	$\text{PL}_{D;C}$: 1
3.	$\langle\gamma^*\rangle\varphi \leftrightarrow \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	Ax4 *
4.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi \to \langle\gamma^*\rangle\varphi$	BE_R: 3
5.	$\neg\bigvee\Phi \to \langle\gamma^*\rangle\varphi$	PL_{CUT}: 2,4
6.	$\bigvee\Phi \vee \langle\gamma^*\rangle\varphi$	$\text{PL}_{C;D}$: 5

Par_{Full} derivation of *-rule.

1.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	premise
2.	$\langle\gamma^*\rangle\varphi \leftrightarrow \varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi$	Ax4 *
3.	$\varphi \vee \langle\gamma\rangle\langle\gamma^*\rangle\varphi \to \langle\gamma^*\rangle\varphi$	$\text{BE}_R : 2$
4.	$\langle\gamma^*\rangle\varphi$	MP: 1,3

Par_{Full} derivation of $\times$-rule (with side formulas).

1.	$\bigvee\Phi \vee (\varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi)$	premise
2.	$\neg\bigvee\Phi \to (\varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi)$	$\text{PL}_{D;C}$: 1
3.	$\langle\gamma^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi$	Ax8 $\times$
4.	$\varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi \to \langle\gamma^\times\rangle\varphi$	BE_R: 3
5.	$\neg\bigvee\Phi \to \langle\gamma^\times\rangle\varphi$	PL_{CUT}: 2,4
6.	$\bigvee\Phi \vee \langle\gamma^\times\rangle\varphi$	$\text{PL}_{C;D}$: 5

Par_{Full} derivation of $\times$-rule.

1.	$\varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi$	premise
2.	$\langle\gamma^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi$	Ax8 $\times$
3.	$\varphi \wedge \langle\gamma\rangle\langle\gamma^\times\rangle\varphi \to \langle\gamma^\times\rangle\varphi$	$\text{BE}_R : 2$
4.	$\langle\gamma^\times\rangle\varphi$	MP: 1,4

Par_{Full} derivation of $\sqcup$-rule (with side formulas).

1.	$\bigvee\Phi \vee (\langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi)$	premise
2.	$\neg\bigvee\Phi \to (\langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi)$	$\text{PL}_{D;C} : 1$
3.	$\langle\gamma \sqcup \delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi$	Ax3 $\sqcup$
4.	$\langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi \to \langle\gamma \sqcup \delta\rangle\varphi$	$\text{BE}_R : 3$
5.	$\neg\bigvee\Phi \to \langle\gamma \sqcup \delta\rangle\varphi$	$\text{PL}_{CUT} : 2, 4$
6.	$\bigvee\Phi \vee \langle\gamma \sqcup \delta\rangle\varphi$	$\text{PL}_{C;D} : 5$

Par_{Full} derivation of $\sqcup$-rule.

1.	$\langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi$	premise
2.	$\langle\gamma \sqcup \delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi \to \langle\gamma \sqcup \delta\rangle\varphi$	$\text{BE}_R : 2$
4.	$\langle\gamma \sqcup \delta\rangle\varphi$	MP: 1,3

Par_{Full} derivation of $\sqcap$-rule (with side formulas).

1.	$\bigvee\Phi \vee (\langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi)$	premise
2.	$\neg\bigvee\Phi \to (\langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi)$	$\text{PL}_{D;C} : 1$
3.	$\langle\gamma \sqcap \delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\varphi \vee \langle\delta\rangle\varphi$	Ax7 $\sqcap$
4.	$\langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi \to \langle\gamma \sqcap \delta\rangle\varphi$	$\text{BE}_R : 3$
5.	$\neg\bigvee\Phi \to \langle\gamma \sqcap \delta\rangle\varphi$	$\text{PL}_{CUT} : 2, 4$
6.	$\bigvee\Phi \vee \langle\gamma \sqcap \delta\rangle\varphi$	$\text{PL}_{C;D} : 5$

Par_{Full} derivation of $\sqcap$-rule.

1.	$\langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi$	premise
2.	$\langle\gamma \sqcap \delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\gamma\rangle\varphi \wedge \langle\delta\rangle\varphi \to \langle\gamma \sqcap \delta\rangle\varphi$	$\text{BE}_R : 2$
4.	$\langle\gamma \sqcap \delta\rangle\varphi$	MP: 1,3

$\mathrm{Par}_{\mathrm{Full}}$ derivation of ?-rule (with side formulas).

1.	$\bigvee \Phi \vee (\psi \wedge \varphi)$	premise
2.	$\neg \bigvee \Phi \to (\psi \wedge \varphi)$	$\mathrm{PL}_{D;C} : 1$
3.	$\langle \psi ? \rangle \varphi \leftrightarrow \psi \wedge \varphi$	Ax5 ?
4.	$\psi \wedge \varphi \to \langle \psi ? \rangle \varphi$	$\mathrm{BE}_R : 3$
5.	$\neg \bigvee \Phi \to \langle \psi ? \rangle \varphi$	$\mathrm{PL}_{CUT} : 2, 4$
6.	$\bigvee \Phi \vee \langle \psi ? \rangle \varphi$	$\mathrm{PL}_{C;D} : 5$

$\mathrm{Par}_{\mathrm{Full}}$ derivation of ?-rule

1.	$\psi \wedge \varphi$	premise
2.	$\langle \psi ? \rangle \varphi \leftrightarrow \psi \wedge \varphi$	Ax5 ?
3.	$\psi \wedge \varphi \to \langle \psi ? \rangle \varphi$	$\mathrm{BE}_R : 2$
4.	$\langle \psi ? \rangle \varphi$	MP: 1,4

$\mathrm{Par}_{\mathrm{Full}}$ derivation of !-rule (with side formulas).

1.	$\bigvee \Phi \vee (\psi \vee \varphi)$	premise
2.	$\neg \bigvee \Phi \to (\psi \vee \varphi)$	$\mathrm{PL}_{D;C} : 1$
3.	$\langle \psi ! \rangle \varphi \leftrightarrow \psi \vee \varphi$	Ax9 !
4.	$\psi \vee \varphi \to \langle \psi ! \rangle \varphi$	$\mathrm{BE}_R : 3$
5.	$\neg \bigvee \Phi \to \langle \psi ! \rangle \varphi$	$\mathrm{PL}_{CUT} : 2, 4$
6.	$\bigvee \Phi \vee \langle \psi ! \rangle \varphi$	$\mathrm{PL}_{C;D} : 5$

$\mathrm{Par}_{\mathrm{Full}}$ derivation of !-rule

1.	$\psi \vee \varphi$	premise
2.	$\langle \psi ! \rangle \varphi \leftrightarrow \psi \vee \varphi$	Ax9 !
3.	$\psi \vee \varphi \to \langle \psi ! \rangle \varphi$	$\mathrm{BE}_R : 2$
4.	$\langle \psi ! \rangle \varphi$	MP: 1,4

V PAR$_{\text{FULL}}$ DERIVATION OF MON^g_d

Par$_{\text{Full}}$ derivation of Mon^g_d rule with side formulas (left) and without side formulas (right).

1.	$\bigvee \Phi \vee \psi(\gamma)$	premise
2.	$\neg \bigvee \Phi \to \psi(\gamma)$	PL$_{D;C}$: 1
3.	$\psi(\gamma) \to \psi(\chi!;\gamma)$	Recursive step
4.	$\neg \bigvee \Phi \to \psi(\chi!;\gamma)$	PL$_{CUT}$: 2, 3
5.	$\bigvee \Phi \vee \psi(\chi!;\gamma)$	PL$_{C;D}$: 4

1.	$\psi(\gamma)$	premise
2.	$\psi(\gamma) \to \psi(\chi!;\gamma)$	Recursive step
3.	$\psi(\chi!;\gamma)$	MP: 1, 2

Par$_{\text{Full}}$ derivation of Mon^g_d-rule, atomic case.

1.	$\langle g\rangle\varphi$	premise
2.	$\langle g\rangle\varphi \to \chi \vee \langle g\rangle\varphi$	PL: $A \to B \vee A$
3.	$\langle \chi!\rangle\langle g\rangle\varphi \leftrightarrow \chi \vee \langle g\rangle\varphi$	Ax9 !
4.	$\chi \vee \langle g\rangle\varphi \to \langle \chi!\rangle\langle g\rangle\varphi$	BE$_R$: 3
5.	$\langle g\rangle\varphi \to \langle \chi!\rangle\langle g\rangle\varphi$	PL$_{CUT}$: 2, 4
6.	$\langle \chi!; g\rangle\varphi \leftrightarrow \langle \chi!\rangle\langle g\rangle\varphi$	Ax2 ;
7.	$\langle \chi!\rangle\langle g\rangle\varphi \to \langle \chi!; g\rangle\varphi$	BE$_R$: 6
8.	$\langle g\rangle\varphi \to \langle \chi!; g\rangle\varphi$	PL$_{CUT}$: 5, 7

Mon^g_d-rule, ! case.

1.	$\langle \theta(\delta)!\rangle\varphi$	premise
2.	$\langle \theta(\delta)!\rangle\varphi \leftrightarrow \theta(\delta) \vee \varphi$	Ax9 !
3.	$\langle \theta(\delta)!\rangle\varphi \to \theta(\delta) \vee \varphi$	BE$_L$: 2
4.	$\theta(\delta) \to \theta(\chi!;\delta)$	Recursive step
5.	$(\theta(\delta) \to \theta(\chi!;\delta)) \to (\theta(\delta) \vee \varphi \to \theta(\chi!;\delta) \vee \varphi)$	PL: $(A \to B) \to (A \vee C \to B \vee C)$
6.	$\theta(\delta) \vee \varphi \to \theta(\chi!;\delta) \vee \varphi$	MP: 4, 5
7.	$\langle \theta(\delta)!\rangle\varphi \to \theta(\chi!;\delta) \vee \varphi$	PL$_{CUT}$: 3, 6
8.	$\langle \theta(\chi!;\delta)!\rangle\varphi \leftrightarrow \theta(\chi!;\delta) \vee \varphi$	Ax9 !
9.	$\theta(\chi!;\delta) \vee \varphi \to \langle \theta(\chi!;\delta)!\rangle\varphi$	BE$_R$: 8
10.	$\langle \theta(\delta)!\rangle\varphi \to \langle \theta(\chi!;\delta)!\rangle\varphi$	PL$_{CUT}$: 7, 9

Mon^g_d-rule, ? case.

1.	$\langle \theta(\delta)?\rangle\varphi$	premise
2.	$\langle \theta(\delta)?\rangle\varphi \leftrightarrow \theta(\delta) \wedge \varphi$	Ax5 ?
3.	$\langle \theta(\delta)?\rangle\varphi \to \theta(\delta) \wedge \varphi$	BE$_L$: 2
4.	$\theta(\delta) \to \theta(\chi!;\delta)$	Recursive step
5.	$(\theta(\delta) \to \theta(\chi!;\delta)) \to (\theta(\delta) \wedge \varphi \to \theta(\chi!;\delta) \wedge \varphi)$	PL: $(A \to B) \to (A \wedge C \to B \wedge C)$
6.	$\theta(\delta) \wedge \varphi \to \theta(\chi!;\delta) \wedge \varphi$	MP: 4, 5
7.	$\langle \theta(\delta)?\rangle\varphi \to \theta(\chi!;\delta) \wedge \varphi$	PL$_{CUT}$: 3, 6
8.	$\langle \theta(\chi!;\delta)?\rangle\varphi \leftrightarrow \theta(\chi!;\delta) \wedge \varphi$	Ax5 ?
9.	$\theta(\chi!;\delta) \wedge \varphi \to \langle \theta(\chi!;\delta)?\rangle\varphi$	BE$_R$: 8
10.	$\langle \theta(\delta)?\rangle\varphi \to \langle \theta(\chi!;\delta)?\rangle\varphi$	PL$_{CUT}$: 7, 9

Mon^g_d-rule, $\sqcup$ case left.

1.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	BE$_L$: 2
4.	$\langle\gamma_1(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$(\langle\gamma_1(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi) \rightarrow (\langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \vee C \rightarrow B \vee C)$
6.	$\langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	MP: 4, 5
7.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	PL$_{CUT}$: 3, 6
8.	$\langle\gamma_1(\chi!;\delta) \sqcup \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\gamma_1(\chi!;\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta) \sqcup \gamma_2(\delta)\rangle\varphi$	BE$_R$: 8
10.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta) \sqcup \gamma_2(\delta)\rangle\varphi$	PL$_{CUT}$: 7, 9

Mon^g_d-rule, $\sqcup$ case right.

1.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi$	BE$_L$: 2
4.	$\langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_2(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$(\langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_2(\chi!;\delta)\rangle\varphi) \rightarrow (\langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\chi!;\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \vee A \rightarrow C \vee B)$
6.	$\langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\chi!;\delta)\rangle\varphi$	MP: 4, 5
7.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\chi!;\delta)\rangle\varphi$	PL$_{CUT}$: 3, 6
8.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\chi!;\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\chi!;\delta)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\gamma_1(\delta)\rangle\varphi \vee \langle\gamma_2(\chi!;\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta) \sqcup \gamma_2(\chi!;\delta)\rangle\varphi$	BE$_R$: 8
10.	$\langle\gamma_1(\delta) \sqcup \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta) \sqcup \gamma_2(\chi!;\delta)\rangle\varphi$	PL$_{CUT}$: 7, 9

Mon^g_d-rule, $\sqcap$ case left.

1.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	BE$_L$: 2
4.	$\langle\gamma_1(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$(\langle\gamma_1(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi) \rightarrow (\langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \wedge C \rightarrow B \wedge C)$
6.	$\langle\gamma_1(\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	MP: 4, 5
7.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	PL$_{CUT}$: 3, 6
8.	$\langle\gamma_1(\chi!;\delta) \sqcap \gamma_2(\delta)\rangle\varphi \leftrightarrow \langle\gamma_1(\chi!;\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
9.	$\langle\gamma_1(\chi!;\delta)\rangle\varphi \wedge \langle\gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta) \sqcap \gamma_2(\delta)\rangle\varphi$	BE$_R$: 8
10.	$\langle\gamma_1(\delta) \sqcap \gamma_2(\delta)\rangle\varphi \rightarrow \langle\gamma_1(\chi!;\delta) \sqcap \gamma_2(\delta)\rangle\varphi$	PL$_{CUT}$: 7, 9

Mon^g_d-rule, $\sqcap$ case right.

1.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\delta)\rangle\varphi$	$\mathrm{BE}_L: 2$
4.	$\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_2(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$(\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_2(\chi!;\delta)\rangle\varphi)\rightarrow(\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\chi!;\delta)\rangle\varphi)$	PL: $(A\rightarrow B)\rightarrow(C\wedge A\rightarrow C\wedge B)$
6.	$\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\chi!;\delta)\rangle\varphi$	MP: $4,5$
7.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $3,6$
8.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\chi!;\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\chi!;\delta)\rangle\varphi$	Ax7 $\sqcap$
9.	$\langle\gamma_1(\delta)\rangle\varphi\wedge\langle\gamma_2(\chi!;\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\sqcap\gamma_2(\chi!;\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\gamma_1(\delta)\sqcap\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\sqcap\gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $7,9$

Mon^g_d-rule, ; case left.

1.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	Ax2 ;
3.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	$\mathrm{BE}_L: 2$
4.	$\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\chi!;\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	Recursive step
5.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\chi!;\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	PL_{CUT}: $3,4$
6.	$\langle\gamma_1(\chi!;\delta);\gamma_2(\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\chi!;\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	Ax2 ;
7.	$\langle\gamma_1(\chi!;\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\chi!;\delta);\gamma_2(\delta)\rangle\varphi$	BE_R: 6
8.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\chi!;\delta);\gamma_2(\delta)\rangle\varphi$	PL_{CUT}: $5,7$

Mon^g_d-rule, ; case right.

1.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi$	premise
2.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	Ax2 ;
3.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi$	$\mathrm{BE}_L: 2$
4.	$\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_2(\chi!;\delta)\rangle\varphi$	Recursive step
5.	$\langle\gamma_1(\delta)\rangle\langle\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\chi!;\delta)\rangle\varphi$	Mon: 4
6.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $3,5$
7.	$\langle\gamma_1(\delta);\gamma_2(\chi!;\delta)\rangle\varphi\leftrightarrow\langle\gamma_1(\delta)\rangle\langle\gamma_2(\chi!;\delta)\rangle\varphi$	Ax2 ;
8.	$\langle\gamma_1(\delta)\rangle\langle\gamma_2(\chi!;\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta);\gamma_2(\chi!;\delta)\rangle\varphi$	BE_R: 7
9.	$\langle\gamma_1(\delta);\gamma_2(\delta)\rangle\varphi\rightarrow\langle\gamma_1(\delta);\gamma_2(\chi!;\delta)\rangle\varphi$	PL_{CUT}: $6,8$

Mon^g_d-rule, * case.

1.	$\langle\gamma(\delta)^*\rangle\varphi$	premise
2.	$\langle\gamma(\chi!;\delta)^*\rangle\varphi \leftrightarrow \varphi \vee \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi$	Ax4
3.	$\varphi \vee \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	$\mathrm{BE}_R : 2$
4.	$(\varphi \vee \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi) \to (\langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi)$	PL: $(A \vee B \to C) \to (B \to C)$
5.	$\langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	MP: 3,4
6.	$\langle\gamma(\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi$	Recursive step
7.	$\langle\gamma(\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	PL_{CUT}: 6, 5
8.	$\langle\gamma(\delta)^*\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	BarInd: 7
9.	$(\varphi \vee \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi) \to (\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi)$	PL: $(A \vee B \to C) \to (A \to C)$
10.	$\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	MP: 3,9
11.	$\langle\gamma(\delta)^*\rangle\varphi \to \langle\gamma(\delta)^*\rangle\langle\gamma(\chi!;\delta)^*\rangle\varphi$	Mon: 10
12.	$\langle\gamma(\delta)^*\rangle\varphi \to \langle\gamma(\chi!;\delta)^*\rangle\varphi$	$\mathrm{PL}_{CUT} : 11, 8$

Mon^g_d-rule, $\times$ case.

1.	$\langle\gamma(\delta)^\times\rangle\varphi$	premise
2.	$\langle\gamma(\delta)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi$	Ax8
3.	$\langle\gamma(\delta)^\times\rangle\varphi \to \varphi \wedge \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$(\langle\gamma(\delta)^\times\rangle\varphi \to \varphi \wedge \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi) \to (\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
5.	$\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi$	MP: 3,4
6.	$\langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi$	Recursive step
7.	$\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\chi!;\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 5, 6$
8.	$\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\chi!;\delta)^\times\rangle\langle\gamma(\delta)^\times\rangle\varphi$	$\mathrm{BarInd}^\times : 7$
9.	$(\langle\gamma(\delta)^\times\rangle\varphi \to \varphi \wedge \langle\gamma(\delta)\rangle\langle\gamma(\delta)^\times\rangle\varphi) \to (\langle\gamma(\delta)^\times\rangle\varphi \to \varphi)$	PL: $(A \to B \wedge C) \to (A \to B)$
10.	$\langle\gamma(\delta)^\times\rangle\varphi \to \varphi$	MP: 3,9
11.	$\langle\gamma(\chi!;\delta)^\times\rangle\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\chi!;\delta)^\times\rangle\varphi$	Mon: 10
12.	$\langle\gamma(\delta)^\times\rangle\varphi \to \langle\gamma(\chi!;\delta)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 8, 11$

Mon^g_d rule, $\wedge$ case left.

1.	$\gamma(\delta) \wedge p$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((\gamma(\delta) \wedge p) \to (\gamma(\chi!;\delta) \wedge p))$	PL: $(A \to B) \to ((A \wedge C) \to (B \wedge C))$
4.	$(\gamma(\delta) \wedge p) \to (\gamma(\chi!;\delta) \wedge p)$	MP: 2, 3

Mon^g_d rule, $\wedge$ case right.

1.	$p \wedge \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((p \wedge \gamma(\delta)) \to (p \wedge \gamma(\chi!;\delta)))$	PL: $(A \to B) \to ((C \wedge A) \to (C \wedge B))$
4.	$(p \wedge \gamma(\delta)) \to (p \wedge \gamma(\chi!;\delta))$	MP: 2, 3

Mon^g_d rule, $\vee$ case left.

1.	$\gamma(\delta) \vee p$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((\gamma(\delta) \vee p) \to (\gamma(\chi!;\delta) \vee p))$	PL: $(A \to B) \to ((A \vee C) \to (B \vee C))$
4.	$(\gamma(\delta) \vee p) \to (\gamma(\chi!;\delta) \vee p)$	MP: 2, 3

Mon_d^g rule, $\vee$ case right.

1.	$p \vee \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((p \vee (\gamma(\delta)) \to (p \vee \gamma(\chi!;\delta)))$	PL: $(A \to B) \to ((C \vee A) \to (C \vee B))$
4.	$(p \vee (\gamma(\delta)) \to (p \vee \gamma(\chi!;\delta))$	MP: $2, 3$

Mon_d^g rule, $\to$ case left.

1.	$\gamma(\delta) \to p$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((\gamma(\delta) \to p) \to (\gamma(\chi!;\delta) \to p))$	PL: $(A \to B) \to ((A \to C) \to (B \to C))$
4.	$(\gamma(\delta) \to p) \to (\gamma(\chi!;\delta) \to p)$	MP: $2, 3$

Mon_d^g rule, $\to$ case right.

1.	$p \to \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((p \to \gamma(\delta)) \to (p \to \gamma(\chi!;\delta)))$	PL: $(A \to B) \to ((C \to A) \to (C \to B))$
4.	$(p \to \gamma(\delta)) \to (p \to \gamma(\chi!;\delta))$	MP: $2, 3$

Mon_d^g rule, $\leftrightarrow$ case left.

1.	$\gamma(\delta) \leftrightarrow p$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((\gamma(\delta) \leftrightarrow p) \to (\gamma(\chi!;\delta) \leftrightarrow p))$	PL: $(A \to B) \to ((A \leftrightarrow C) \to (B \leftrightarrow C))$
4.	$(\gamma(\delta) \leftrightarrow p) \to (\gamma(\chi!;\delta) \leftrightarrow p)$	MP: $2, 3$

Mon_d^g rule, $\leftrightarrow$ case right.

1.	$p \leftrightarrow \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\chi!;\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\chi!;\delta)) \to ((p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\chi!;\delta)))$	PL: $(A \to B) \to ((C \leftrightarrow A) \to (C \leftrightarrow B))$
4.	$(p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\chi!;\delta))$	MP: $2, 3$

VI PAR$_{\text{FULL}}$ DERIVATION OF MON^f_d

MON^f_d rule, with side formulas.

1.	$\bigvee \Phi \vee \psi$	premise
2.	$\neg \bigvee \Phi \to \psi$	PL$_{D;C}$: 1
3.	$\psi \to \langle \chi! \rangle \psi$	Recursive step
4.	$\neg \bigvee \Phi \to \langle \chi! \rangle \psi$	PL$_{CUT}$: 2, 3
5.	$\bigvee \Phi \vee \langle \chi! \rangle \psi$	PL$_{C;D}$: 4

MON^f_d rule, without side formulas.

1.	ψ	premise
2.	$\psi \to \langle \chi! \rangle \psi$	Recursive step
3.	$\langle \chi! \rangle \psi$	MP: 1, 2

MON^f_d rule, atomic case.

1.	ψ	premise
2.	$\psi \to \chi \vee \psi$	PL: $A \to B \vee A$
3.	$\langle \chi! \rangle \psi \leftrightarrow \chi \vee \psi$	Ax9 !
4.	$\chi \vee \psi \to \langle \chi! \rangle \psi$	BE$_R$: 3
5.	$\psi \to \langle \chi! \rangle \psi$	PL$_{CUT}$: 2, 4

MON^f_d rule, ? case.

1.	$\langle \psi? \rangle \varphi$	premise
2.	$\langle \psi? \rangle \varphi \leftrightarrow \psi \wedge \varphi$	Ax5 ?
3.	$\langle \psi? \rangle \varphi \to \psi \wedge \varphi$	BE$_L$: 2
4.	$\psi \to \langle \chi! \rangle \psi$	Recursive step
5.	$(\psi \to \langle \chi! \rangle \psi) \to (\psi \wedge \varphi \to \langle \chi! \rangle \psi \wedge \varphi)$	PL: $(A \to B) \to (A \wedge C \to B \wedge C)$
6.	$\psi \wedge \varphi \to \langle \chi! \rangle \psi \wedge \varphi$	MP: 4, 5
7.	$\langle \psi? \rangle \varphi \to \langle \chi! \rangle \psi \wedge \varphi$	PL$_{CUT}$: 3, 6
8.	$\langle \langle \chi! \rangle \psi? \rangle \varphi \leftrightarrow \langle \chi! \rangle \psi \wedge \varphi$	Ax5 ?
9.	$\langle \chi! \rangle \psi \wedge \varphi \to \langle \langle \chi! \rangle \psi? \rangle \varphi$	BE$_R$: 8
10.	$\langle \psi? \rangle \varphi \to \langle \langle \chi! \rangle \psi? \rangle \varphi$	PL$_{CUT}$: 7, 9

MON^f_d rule, ! case.

1.	$\langle \psi! \rangle \varphi$	premise
2.	$\langle \psi! \rangle \varphi \leftrightarrow \psi \vee \varphi$	Ax9 !
3.	$\langle \psi! \rangle \varphi \to \psi \vee \varphi$	BE$_L$: 2
4.	$\psi \to \langle \chi! \rangle \psi$	Recursive step
5.	$(\psi \to \langle \chi! \rangle \psi) \to (\psi \vee \varphi \to \langle \chi! \rangle \psi \vee \varphi)$	PL: $(A \to B) \to (A \vee C \to B \vee C)$
6.	$\psi \vee \varphi \to \langle \chi! \rangle \psi \vee \varphi$	MP: 4, 5
7.	$\langle \psi! \rangle \varphi \to \langle \chi! \rangle \psi \vee \varphi$	PL$_{CUT}$: 3, 6
8.	$\langle \langle \chi! \rangle \psi! \rangle \varphi \leftrightarrow \langle \chi! \rangle \psi \vee \varphi$	Ax9 !
9.	$\langle \chi! \rangle \psi \vee \varphi \to \langle \langle \chi! \rangle \psi! \rangle \varphi$	BE$_R$: 8
10.	$\langle \psi! \rangle \varphi \to \langle \langle \chi! \rangle \psi! \rangle \varphi$	PL$_{CUT}$: 7, 9

Mon^f_d rule, $\sqcup$ case left.

1.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$(\langle\psi_1(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi) \rightarrow (\langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \vee C \rightarrow B \vee C)$
6.	$\langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\langle\chi!\rangle\delta) \sqcup \psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta) \sqcup \psi_2(\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta) \sqcup \psi_2(\delta)\rangle\varphi$	PL_{CUT}: $7, 9$

Mon^f_d rule, $\sqcup$ case right.

1.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$(\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi) \rightarrow (\langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \vee A \rightarrow C \vee B)$
6.	$\langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\delta) \sqcup \psi_2(\langle\chi!\rangle\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\psi_1(\delta)\rangle\varphi \vee \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta) \sqcup \psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\delta) \sqcup \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta) \sqcup \psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $7, 9$

Mon^f_d rule, $\sqcap$ case left.

1.	$\langle\psi_1(\delta) \sqcap \psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta) \sqcap \psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\psi_1(\delta) \sqcap \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$(\langle\psi_1(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi) \rightarrow (\langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \wedge C \rightarrow B \wedge C)$
6.	$\langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\delta) \sqcap \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\langle\chi!\rangle\delta) \sqcap \psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
9.	$\langle\psi_1(\langle\chi!\rangle\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta) \sqcap \psi_2(\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\delta) \sqcap \psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta) \sqcap \psi_2(\delta)\rangle\varphi$	PL_{CUT}: $7, 9$

Mon^f_d rule, $\sqcap$ case right.

1.	$\langle\psi_1(\delta)\sqcap\psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta)\sqcap\psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	Ax7 $\sqcap$
3.	$\langle\psi_1(\delta)\sqcap\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$(\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi) \rightarrow (\langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \wedge A \rightarrow C \wedge B)$
6.	$\langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\delta)\sqcap\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\delta)\sqcap\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Ax7 $\sqcap$
9.	$\langle\psi_1(\delta)\rangle\varphi \wedge \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\sqcap\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\delta)\sqcap\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\sqcap\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $7, 9$

Mon^f_d rule, ; case left.

1.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	Ax2 ;
3.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	Recursive step
5.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	PL_{CUT}: $3, 4$
6.	$\langle\psi_1(\langle\chi!\rangle\delta);\psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\chi!\rangle\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	Ax2 ;
7.	$\langle\psi_1(\langle\chi!\rangle\delta)\rangle\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta);\psi_2(\delta)\rangle\varphi$	BE_R: 6
8.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\langle\chi!\rangle\delta);\psi_2(\delta)\rangle\varphi$	PL_{CUT}: $5, 7$

Mon^f_d rule, ; case right.

1.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi$	premise
2.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	Ax2 ;
3.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Recursive step
5.	$\langle\psi_1(\delta)\rangle\langle\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Mon: 6
6.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $3, 5$
7.	$\langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \leftrightarrow \langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	Ax2 ;
8.	$\langle\psi_1(\delta)\rangle\langle\psi_2(\langle\chi!\rangle\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	BE_R: 7
9.	$\langle\psi_1(\delta);\psi_2(\delta)\rangle\varphi \rightarrow \langle\psi_1(\delta);\psi_2(\langle\chi!\rangle\delta)\rangle\varphi$	PL_{CUT}: $6, 8$

Mon^f_d rule, * case.

1.	$\langle\psi(\delta)^*\rangle\varphi$	premise
2.	$\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \leftrightarrow \varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Ax4
3.	$\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	$\mathrm{BE}_R : 2$
4.	$(\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi) \rightarrow (\langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi)$	PL: $(A \vee B \rightarrow C) \rightarrow (B \rightarrow$
5.	$\langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	MP: 3,4
6.	$\langle\psi(\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Recursive step
7.	$\langle\psi(\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	PL_{CUT}: 6, 5
8.	$\langle\psi(\delta)^*\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	BarInd: 7
9.	$(\varphi \vee \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi) \rightarrow (\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi)$	PL: $(A \vee B \rightarrow C) \rightarrow (A \rightarrow$
10.	$\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	MP: 3,9
11.	$\langle\psi(\delta)^*\rangle\varphi \rightarrow \langle\psi(\delta)^*\rangle\langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	Mon: 10
12.	$\langle\psi(\delta)^*\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^*\rangle\varphi$	$\mathrm{PL}_{CUT} : 11, 8$

Mon^f_d rule, $\times$ case.

1.	$\langle\psi(\delta)^\times\rangle\varphi$	premise
2.	$\langle\psi(\delta)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi$	Ax8
3.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \varphi \wedge \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$(\langle\psi(\delta)^\times\rangle\varphi \rightarrow \varphi \wedge \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi) \rightarrow (\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi)$	PL: $(A \rightarrow B \wedge C) \rightarrow (A \rightarrow C)$
5.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi$	MP: 3,4
6.	$\langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi$	Recursive step
7.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 5, 6$
8.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^\times\rangle\langle\psi(\delta)^\times\rangle\varphi$	$\mathrm{BarInd}^\times : 7$
9.	$(\langle\psi(\delta)^\times\rangle\varphi \rightarrow \varphi \wedge \langle\psi(\delta)\rangle\langle\psi(\delta)^\times\rangle\varphi) \rightarrow (\langle\psi(\delta)^\times\rangle\varphi \rightarrow \varphi)$	PL: $(A \rightarrow B \wedge C) \rightarrow (A \rightarrow B)$
10.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \varphi$	MP: 3,9
11.	$\langle\psi(\langle\chi!\rangle\delta)^\times\rangle\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^\times\rangle\varphi$	Mon: 10
12.	$\langle\psi(\delta)^\times\rangle\varphi \rightarrow \langle\psi(\langle\chi!\rangle\delta)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 8, 11$

Mon^f_d rule, $\wedge$ case left.

1.	$\gamma(\delta) \wedge p$	premise
2.	$\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)) \rightarrow ((\gamma(\delta) \wedge p) \rightarrow (\gamma(\langle\chi!\rangle\delta) \wedge p))$	PL: $(A \rightarrow B) \rightarrow ((A \wedge C) \rightarrow (B \wedge C))$
4.	$(\gamma(\delta) \wedge p) \rightarrow (\gamma(\langle\chi!\rangle\delta) \wedge p)$	MP: 2, 3

Mon^f_d rule, $\wedge$ case right.

1.	$p \wedge \gamma(\delta)$	premise
2.	$\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)) \rightarrow ((p \wedge \gamma(\delta)) \rightarrow (p \wedge \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \rightarrow B) \rightarrow ((C \wedge A) \rightarrow (C \wedge B))$
4.	$(p \wedge \gamma(\delta)) \rightarrow (p \wedge \gamma(\langle\chi!\rangle\delta))$	MP: 2, 3

Mon^f_d rule, $\vee$ case left.

1.	$\gamma(\delta) \vee p$	premise
2.	$\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \rightarrow \gamma(\langle\chi!\rangle\delta)) \rightarrow ((\gamma(\delta) \vee p) \rightarrow (\gamma(\langle\chi!\rangle\delta) \vee p))$	PL: $(A \rightarrow B) \rightarrow ((A \vee C) \rightarrow (B \vee C))$
4.	$(\gamma(\delta) \vee p) \rightarrow (\gamma(\langle\chi!\rangle\delta) \vee p)$	MP: 2, 3

Mon_d^f rule, $\vee$ case right.

1.	$p \vee \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \vee (\gamma(\delta)) \to (p \vee \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \vee A) \to (C \vee B))$
4.	$(p \vee (\gamma(\delta)) \to (p \vee \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

Mon_d^f rule, $\to$ case left.

1.	$\gamma(\delta) \to p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \to p) \to (\gamma(\langle\chi!\rangle\delta) \to p))$	PL: $(A \to B) \to ((A \to C) \to (B \to C))$
4.	$(\gamma(\delta) \to p) \to (\gamma(\langle\chi!\rangle\delta) \to p)$	MP: $2, 3$

Mon_d^f rule, $\to$ case right.

1.	$p \to \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \to \gamma(\delta)) \to (p \to \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \to A) \to (C \to B))$
4.	$(p \to \gamma(\delta)) \to (p \to \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

Mon_d^f rule, $\leftrightarrow$ case left.

1.	$\gamma(\delta) \leftrightarrow p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \leftrightarrow p) \to (\gamma(\langle\chi!\rangle\delta) \leftrightarrow p))$	PL: $(A \to B) \to ((A \leftrightarrow C) \to (B \leftrightarrow C))$
4.	$(\gamma(\delta) \leftrightarrow p) \to (\gamma(\langle\chi!\rangle\delta) \leftrightarrow p)$	MP: $2, 3$

Mon_d^f rule, $\leftrightarrow$ case right.

1.	$p \leftrightarrow \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \leftrightarrow A) \to (C \leftrightarrow B))$
4.	$(p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

VII PAR$_{\text{FULL}}$ DERIVATION OF $;_d$

$;_d$ rule with side formulas.

1.	$\bigvee\Phi \vee \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	premise
2.	$\neg\bigvee\Phi \to \psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	PL$_{D;C}$: 1
3.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
4.	$\neg\bigvee\Phi \to \psi(\langle\gamma;\delta\rangle\varphi)$	PL$_{CUT}$: 2, 3
5.	$\bigvee\Phi \vee \psi(\langle\gamma;\delta\rangle\varphi)$	PL$_{C;D}$: 4

$;_d$ rule without side formulas.

1.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)$	premise
2.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
3.	$\psi(\langle\gamma;\delta\rangle\varphi)$	MP: 1, 2

$;_d$ rule, atomic case.

1.	$\langle\gamma\rangle\langle\delta\rangle\varphi$	premise
2.	$\langle\gamma;\delta\rangle\varphi \leftrightarrow \langle\gamma\rangle\langle\delta\rangle\varphi$	Ax2 ;
3.	$\langle\gamma\rangle\langle\delta\rangle\varphi \to \langle\gamma;\delta\rangle\varphi$	BE$_R$: 2
4.	$\langle\gamma;\delta\rangle\varphi$	MP: 1, 3

$;_d$ rule, ? case.

1.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)?\rangle\phi$	premise
2.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)?\rangle\phi \leftrightarrow \psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \wedge \phi$	Ax5 ?
3.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)?\rangle\phi \to \psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \wedge \phi$	BE$_L$: 2
4.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
5.	$(\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)) \to (\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \wedge \phi \to \psi(\langle\gamma;\delta\rangle\varphi) \wedge \phi)$	PL: $(A \to B) \to (A \wedge C \to B \wedge C)$
6.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \wedge \phi \to \psi(\langle\gamma;\delta\rangle\varphi \wedge \phi$	MP: 4, 5
7.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)?\rangle\phi \to \psi(\langle\gamma;\delta\rangle\varphi) \wedge \phi$	PL$_{CUT}$: 3, 6
8.	$\langle\psi(\langle\gamma;\delta\rangle\varphi)?\rangle\phi \leftrightarrow \psi(\langle\gamma;\delta\rangle\varphi) \wedge \phi$	Ax5 ?
9.	$\psi(\langle\gamma;\delta\rangle\varphi) \wedge \phi \to \langle\psi(\langle\gamma;\delta\rangle\varphi)?\rangle\phi$	BE$_R$: 8
10.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)?\rangle\phi \to \langle\psi(\langle\gamma;\delta\rangle\varphi)?\rangle\phi$	PL$_{CUT}$: 7, 9

$;_d$ rule, ! case.

1.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)!\rangle\phi$	premise
2.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)!\rangle\phi \leftrightarrow \psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \vee \phi$	Ax9 !
3.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)!\rangle\phi \to \psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \vee \phi$	BE$_L$: 2
4.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)$	Recursive step
5.	$(\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \to \psi(\langle\gamma;\delta\rangle\varphi)) \to (\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \vee \phi \to \psi(\langle\gamma;\delta\rangle\varphi) \vee \phi)$	PL: $(A \to B) \to (A \vee C \to B \vee C)$
6.	$\psi(\langle\gamma\rangle\langle\delta\rangle\varphi) \vee \phi \to \psi(\langle\gamma;\delta\rangle\varphi) \vee \phi$	MP: 4, 5
7.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)!\rangle\phi \to \psi(\langle\gamma;\delta\rangle\varphi) \vee \phi$	PL$_{CUT}$: 3, 6
8.	$\langle\psi(\langle\gamma;\delta\rangle\varphi)!\rangle\phi \leftrightarrow \psi(\langle\gamma;\delta\rangle\varphi) \vee \phi$	Ax9 !
9.	$\psi(\langle\gamma;\delta\rangle\varphi) \vee \phi \to \langle\psi(\langle\gamma;\delta\rangle\varphi)!\rangle\phi$	BE$_R$: 8
10.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\varphi)!\rangle\phi \to \langle\psi(\langle\gamma;\delta\rangle\varphi)!\rangle\phi$	PL$_{CUT}$: 7, 9

;$_d$ rule, $\sqcup$ case left.

1.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Recursive step
5.	$(\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi) \rightarrow$ $(\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \vee C \rightarrow B \vee C)$
6.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \vee \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcup \psi_2(\phi)\rangle\varphi$	PL_{CUT}: $7, 9$

;$_d$ rule, $\sqcup$ case right.

1.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	Ax3 $\sqcup$
3.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Recursive step
5.	$(\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi) \rightarrow$ $(\langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \vee \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \vee A \rightarrow C \vee B)$
6.	$\langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \vee \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \vee \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Ax3 $\sqcup$
9.	$\langle\psi_1(\phi)\rangle\varphi \vee \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: $7, 9$

;$_d$ rule, $\sqcap$ case left.

1.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi$	Ax3 $\sqcap$
3.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Recursive step
5.	$(\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi) \rightarrow$ $(\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (A \wedge C \rightarrow B \wedge C)$
6.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi$	MP: $4, 5$
7.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi$	PL_{CUT}: $3, 6$
8.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi$	Ax3 $\sqcap$
9.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\varphi \wedge \langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi) \sqcap \psi_2(\phi)\rangle\varphi$	PL_{CUT}: $7, 9$

;$_d$ rule, $\sqcap$ case right.

1.	$\langle\psi_1(\phi) \sqcup \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	Ax3 $\sqcap$
3.	$\langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Recursive step
5.	$(\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi) \rightarrow$ $(\langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \wedge \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi)$	PL: $(A \rightarrow B) \rightarrow (C \wedge A \rightarrow C \wedge B)$
6.	$\langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \wedge \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	MP: 4, 5
7.	$\langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\varphi) \wedge \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: 3, 6
8.	$\langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Ax3 $\sqcap$
9.	$\langle\psi_1(\phi)\rangle\varphi \wedge \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	BE_R: 8
10.	$\langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi) \sqcap \psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: 7, 9

;$_d$ rule, ; case left.

1.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi);\psi_2(\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi);\psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi$	Ax2 ;
3.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi);\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi$	Recursive step
5.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi);\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi$	PL_{CUT}: 3, 4
6.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi);\psi_2(\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi$	Ax2 ;
7.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi);\psi_2(\phi)\rangle\varphi$	BE_R: 6
8.	$\langle\psi_1(\langle\gamma\rangle\langle\delta\rangle\phi);\psi_2(\phi)\rangle\varphi \rightarrow \langle\psi_1(\langle\gamma;\delta\rangle\phi);\psi_2(\phi)\rangle\varphi$	PL_{CUT}: 5, 7
9.	$\langle\psi_1(\langle\gamma;\delta\rangle\phi);\psi_2(\phi)\rangle\varphi$	MP: 1, 8

;$_d$ rule, ; case right.

1.	$\langle\psi_1(\phi);\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	premise
2.	$\langle\psi_1(\phi);\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	Ax2 ;
3.	$\langle\psi_1(\phi);\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Recursive step
5.	$\langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Mon: 6
6.	$\langle\psi_1(\phi);\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: 3, 5
7.	$\langle\psi_1(\phi);\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \leftrightarrow \langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	Ax2 ;
8.	$\langle\psi_1(\phi)\rangle\langle\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi);\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	BE_R: 7
9.	$\langle\psi_1(\phi);\psi_2(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\varphi \rightarrow \langle\psi_1(\phi);\psi_2(\langle\gamma;\delta\rangle\phi)\rangle\varphi$	PL_{CUT}: 6, 8

$;_d$ rule, * case.

1.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^*\rangle\varphi$	premise
2.	$\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \leftrightarrow \varphi \vee \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	Ax4
3.	$\varphi \vee \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	$\mathrm{BE}_R : 2$
4.	$(\varphi \vee \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi) \to$ $(\langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi)$	PL: $(A \vee B \to C) \to (B \to C)$
5.	$\langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	MP: 3,4
6.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	Recursive step
7.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	PL_{CUT}: 6, 5
8.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^*\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	BarInd: 7
9.	$(\varphi \vee \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi) \to (\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi)$	PL: $(A \vee B \to C) \to (A \to C)$
10.	$\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	MP: 3,9
11.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^*\rangle\langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	Mon: 10
12.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^*\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^*\rangle\varphi$	$\mathrm{PL}_{CUT} : 11, 8$

$;_d$ rule, $\times$ case.

1.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	premise
2.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	Ax8
3.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \varphi \wedge \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	$\mathrm{BE}_L : 2$
4.	$(\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \varphi \wedge \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi) \to$ $(\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
5.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	MP: 3,4
6.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	Recursive step
7.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 5, 6$
8.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^\times\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi$	$\mathrm{BarInd}^\times : 7$
9.	$(\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \varphi \wedge \langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi) \to (\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \varphi)$	PL: $(A \to B \wedge C) \to (A \to B)$
10.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \varphi$	MP: 3,9
11.	$\langle\psi(\langle\gamma;\delta\rangle\phi)^\times\rangle\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^\times\rangle\varphi$	Mon: 10
12.	$\langle\psi(\langle\gamma\rangle\langle\delta\rangle\phi)^\times\rangle\varphi \to \langle\psi(\langle\gamma;\delta\rangle\phi)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 8, 11$

$;_d$ rule, $\wedge$ case left.

1.	$\gamma(\delta) \wedge p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \wedge p) \to (\gamma(\langle\chi!\rangle\delta) \wedge p))$	PL: $(A \to B) \to ((A \wedge C) \to (B \wedge C))$
4.	$(\gamma(\delta) \wedge p) \to (\gamma(\langle\chi!\rangle\delta) \wedge p)$	MP: 2, 3

$;_d$ rule, $\wedge$ case right.

1.	$p \wedge \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \wedge \gamma(\delta)) \to (p \wedge \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \wedge A) \to (C \wedge B))$
4.	$(p \wedge \gamma(\delta)) \to (p \wedge \gamma(\langle\chi!\rangle\delta))$	MP: 2, 3

$;_d$ rule, $\vee$ case left.

1.	$\gamma(\delta) \vee p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \vee p) \to (\gamma(\langle\chi!\rangle\delta) \vee p))$	PL: $(A \to B) \to ((A \vee C) \to (B \vee C))$
4.	$(\gamma(\delta) \vee p) \to (\gamma(\langle\chi!\rangle\delta) \vee p)$	MP: $2, 3$

$;_d$ rule, $\vee$ case right.

1.	$p \vee \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \vee (\gamma(\delta)) \to (p \vee \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \vee A) \to (C \vee B))$
4.	$(p \vee (\gamma(\delta)) \to (p \vee \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

$;_d$ rule, $\to$ case left.

1.	$\gamma(\delta) \to p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \to p) \to (\gamma(\langle\chi!\rangle\delta) \to p))$	PL: $(A \to B) \to ((A \to C) \to (B \to C))$
4.	$(\gamma(\delta) \to p) \to (\gamma(\langle\chi!\rangle\delta) \to p)$	MP: $2, 3$

$;_d$ rule, $\to$ case right.

1.	$p \to \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \to \gamma(\delta)) \to (p \to \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \to A) \to (C \to B))$
4.	$(p \to \gamma(\delta)) \to (p \to \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

$;_d$ rule, $\leftrightarrow$ case left.

1.	$\gamma(\delta) \leftrightarrow p$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((\gamma(\delta) \leftrightarrow p) \to (\gamma(\langle\chi!\rangle\delta) \leftrightarrow p))$	PL: $(A \to B) \to ((A \leftrightarrow C) \to (B \leftrightarrow C))$
4.	$(\gamma(\delta) \leftrightarrow p) \to (\gamma(\langle\chi!\rangle\delta) \leftrightarrow p)$	MP: $2, 3$

$;_d$ rule, $\leftrightarrow$ case right.

1.	$p \leftrightarrow \gamma(\delta)$	premise
2.	$\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)$	Recursive step
3.	$(\gamma(\delta) \to \gamma(\langle\chi!\rangle\delta)) \to ((p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\langle\chi!\rangle\delta)))$	PL: $(A \to B) \to ((C \leftrightarrow A) \to (C \leftrightarrow B))$
4.	$(p \leftrightarrow \gamma(\delta)) \to (p \leftrightarrow \gamma(\langle\chi!\rangle\delta))$	MP: $2, 3$

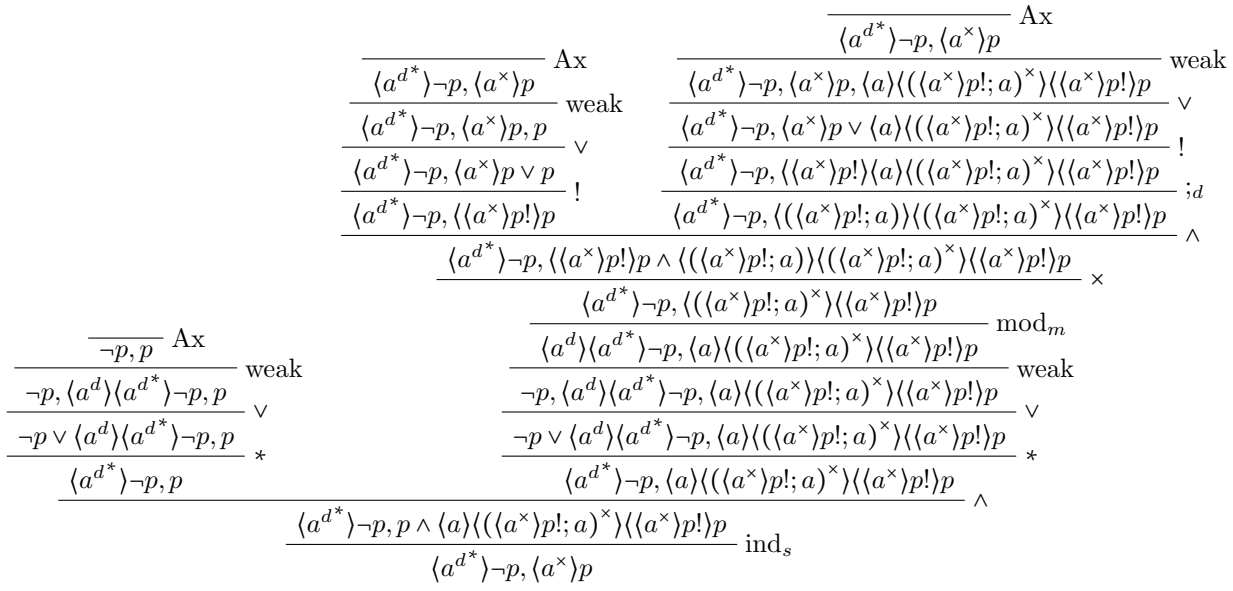
VIII $\mathrm{PAR}_{\mathrm{FULL}}$ DERIVATION OF STRENGTHENED INDUCTION RULE

$\mathrm{Par}_{\mathrm{Full}}$ derivation of ind_s rule.

0.	$\Gamma \vee (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	premise
1.	$\overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	$\mathrm{PL}_{D;C} : 0$
2.	$(\overline{\Gamma} \to (\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)) \to (\overline{\Gamma} \to \varphi)$	PL: $(A \to B \wedge C) \to (A \to B)$
3.	$\overline{\Gamma} \to \varphi$	MP: 1,2
4.	$\langle\overline{\Gamma}!\rangle\varphi \leftrightarrow \overline{\Gamma} \vee \varphi$	Ax9 !
5.	$\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi$	BE_L: 4
6.	$(\overline{\Gamma} \to \varphi) \to ((\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi) \to (\langle\overline{\Gamma}!\rangle\varphi \to \varphi))$	PL: $(A \to B) \to ((C \to A \vee B) \to (C \to B))$
7.	$(\langle\overline{\Gamma}!\rangle\varphi \to \overline{\Gamma} \vee \varphi) \to (\langle\overline{\Gamma}!\rangle\varphi \to \varphi)$	MP: 3, 6
8.	$\langle\overline{\Gamma}!\rangle\varphi \to \varphi$	MP: 5, 7
9.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi \to \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Mon: 8
10.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Mon: 9
11.	$(\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi) \to (\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
12.	$\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\langle\overline{\Gamma}!\rangle\varphi$	MP: 1,11
13.	$\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 12, 10$
14.	$(\overline{\Gamma} \to \varphi) \to ((\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
15.	$(\overline{\Gamma} \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: 3, 14
16.	$\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: 13, 15
		CLAIM 2
17.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax8 ×
18.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 17$
19.	$(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to$ $(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	PL: $(A \to B \wedge C) \to (A \to C)$
20.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: 18,19
21.	$\langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax2 ;
22.	$\langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 21$
23.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax9 !
24.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_L : 23$
25.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: 20, 22
26.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{PL}_{CUT} : 25, 24$
27.	$(\overline{\Gamma} \to \varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to ((\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to$ $((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B \wedge C) \to$ $((A \vee C) \to ((B \wedge C) \vee C))$
28.	$(\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to ((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: 16, 27
29.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to ((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	PL_{CUT}: 26, 28
30.	$((\varphi \wedge \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL: $((A \wedge B) \vee B) \to B$
31.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: 29, 30
32.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BarInd}^\times : 31$
		CLAIM 3
33.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL: $A \to B \vee A$
34.	$\overline{\Gamma} \vee \langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 23$
35.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT} :33, 34
36.	$\langle\overline{\Gamma}!\rangle\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 21$
37.	$\langle\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: 35, 36
38.	$\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: 13, 37
39.	$(\overline{\Gamma} \to \varphi) \to ((\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
40.	$(\overline{\Gamma} \to \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \to (\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi)$	MP: 3, 39
41.	$\overline{\Gamma} \to \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	MP: 38, 40

42.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \leftrightarrow \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	Ax8 $\times$
43.	$\varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \rightarrow \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	$\mathrm{BE}_R : 42$
44.	$\overline{\Gamma} \rightarrow \langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $41, 43$
		CLAIM 4
45.	$(\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \rightarrow \varphi \wedge \langle\overline{\Gamma}!;\gamma\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi) \rightarrow (\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \rightarrow \varphi)$	PL: $(A \rightarrow B \wedge C) \rightarrow (A \rightarrow B)$
46.	$\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \rightarrow \varphi$	MP: 18,45
		CLAIM 5
47.	$\overline{\Gamma} \rightarrow \langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi$	PL_{CUT}: $44, 31$
48.	$\langle\gamma^\times\rangle\langle(\overline{\Gamma}!;\gamma)^\times\rangle\varphi \rightarrow \langle\gamma^\times\rangle\varphi$	Mon: 46
49.	$\overline{\Gamma} \rightarrow \langle\gamma^\times\rangle\varphi$	PL_{CUT}: $47, 48$

IX Complex test



1.	$\neg p \vee p$	PL: $A \vee \neg A$
2.	$(\neg p \vee p) \to (\neg p \vee p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p)$	PL: $A \to A \vee B$
3.	$\neg p \vee p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p$	MP: $1, 2$
4.	$(\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee p) \to (p \vee (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
5.	$(p \vee (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	MP: $3, 4$
6.	$(\neg(p) \to (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	$\mathrm{PL}_{D;C}$: 5
7.	$(\langle a^{d^*}\rangle\neg p \leftrightarrow (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	Ax4 $*$
8.	$((\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p) \to \langle a^{d^*}\rangle\neg p)$	BE_R: 7
9.	$(\neg(p) \to \langle a^{d^*}\rangle\neg p)$	PL_{CUT}: $6, 8$
10.	$\langle a^{d^*}\rangle\neg p \vee p$	$\mathrm{PL}_{C;D}$: 9
11.	$\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p$	PL: $A \vee \neg A$
12.	$(\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p) \to (\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee p)$	PL: $A \to A \vee B$
13.	$\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee p$	MP: $11, 12$
14.	$(\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee p) \to (\langle a^{d^*}\rangle\neg p \vee (\langle a^\times\rangle p \vee p))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
15.	$(\langle a^{d^*}\rangle\neg p \vee (\langle a^\times\rangle p \vee p))$	MP: $13, 14$
16.	$(\neg(\langle a^{d^*}\rangle\neg p) \to (\langle a^\times\rangle p \vee p))$	$\mathrm{PL}_{D;C}$: 15
17.	$(\langle\langle a^\times\rangle p!\rangle p \leftrightarrow (\langle a^\times\rangle p \vee p))$	Ax9 !
18.	$((\langle a^\times\rangle p \vee p) \to \langle\langle a^\times\rangle p!\rangle p)$	BE_R: 17
19.	$(\neg(\langle a^{d^*}\rangle\neg p) \to \langle\langle a^\times\rangle p!\rangle p)$	PL_{CUT}: $16, 18$
20.	$\langle a^{d^*}\rangle\neg p \vee \langle\langle a^\times\rangle p!\rangle p$	$\mathrm{PL}_{C;D}$: 19
21.	$\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p$	PL: $A \vee \neg A$
22.	$(\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p) \to (\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	PL: $A \to A \vee B$
23.	$\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p$	MP: $21, 22$
24.	$(\langle a^{d^*}\rangle\neg p \vee \langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \to$ $(\langle a^{d^*}\rangle\neg p \vee (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	PL: $(A \vee B \vee C) \to (A \vee (B \vee C))$
25.	$(\langle a^{d^*}\rangle\neg p \vee (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	MP: $23, 24$
26.	$(\neg(\langle a^{d^*}\rangle\neg p) \to (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	$\mathrm{PL}_{D;C}$: 25
27.	$(\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \leftrightarrow$ $(\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	Ax9 !

28.	$((\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	BE_R: 27
29.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	PL_{CUT}: 26, 28
30.	$\langle a^{d^*}\rangle\neg p \vee \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p$	$\mathrm{PL}_{C;D}$: 29
31.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{D;C}$: 30
32.	$(\langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \leftrightarrow$ $\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	Ax2 ;
33.	$(\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \rightarrow$ $\langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	BE_R: 32
34.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	PL_{CUT}: 31, 33
35.	$(\langle a^{d^*}\rangle\neg p \vee \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{C;D}$: 34
36.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{D;C}$: 20
37.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{D;C}$: 35
38.	$((\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle\langle a^\times\rangle p!\rangle p) \rightarrow ((\neg(\langle a^{d^*}\rangle\neg p) \rightarrow$ $\langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow (\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))))$	PL: $(A \rightarrow B) \rightarrow$ $((A \rightarrow C) \rightarrow (A \rightarrow B \wedge C))$
39.	$((\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow (\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)))$	MP: 36, 38
40.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow (\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	MP: 37, 39
41.	$(\langle a^{d^*}\rangle\neg p \vee (\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	$\mathrm{PL}_{C;D}$: 40
42.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow (\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	$\mathrm{PL}_{D;C}$: 41
43.	$(\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \leftrightarrow$ $(\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	Ax8 ×
44.	$((\langle\langle a^\times\rangle p!\rangle p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	BE_R: 43
45.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow \langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	PL_{CUT}: 42, 44
46.	$\langle a^{d^*}\rangle\neg p \vee \langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p$	$\mathrm{PL}_{C;D}$: 45
47.	$(\neg\langle a^{d^*}\rangle\neg p \rightarrow \langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{D;C}$: 46
48.	$(\langle a\rangle\neg\langle a^{d^*}\rangle\neg p \rightarrow \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	Mon: 47
49.	$(\neg\langle a\rangle\neg\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{C;D}$: 48
50.	$(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \leftrightarrow \neg\langle a\rangle\neg\langle a^{d^*}\rangle\neg p)$	Ax6 *dual*
51.	$((\langle a^d\rangle\langle a^{d^*}\rangle\neg p \leftrightarrow \neg\langle a\rangle\neg\langle a^{d^*}\rangle\neg p) \rightarrow$ $((\neg\langle a\rangle\neg\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)))$	PL: $(A \leftrightarrow B) \rightarrow (B \vee C \rightarrow A \vee C)$
52.	$((\neg\langle a\rangle\neg\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	MP: 50, 51
53.	$(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	MP: 49, 52
54.	$(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \vee \neg p)$	PL: $A \rightarrow A \vee B$
55.	$\langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \vee \neg p$	MP: 53, 54
56.	$(\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow$ $(\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \vee (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	PL: $(A \vee B \vee C) \rightarrow (A \vee (B \vee C))$
57.	$(\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \vee (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	MP: 55, 56
58.	$(\neg(\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	$\mathrm{PL}_{D;C}$: 57
59.	$(\langle a^{d^*}\rangle\neg p \leftrightarrow (\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p))$	Ax4 $*$
60.	$((\neg p \vee \langle a^d\rangle\langle a^{d^*}\rangle\neg p) \rightarrow \langle a^{d^*}\rangle\neg p)$	BE_R: 59
61.	$(\neg(\langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \rightarrow \langle a^{d^*}\rangle\neg p)$	PL_{CUT}: 58, 60
62.	$\langle a^{d^*}\rangle\neg p \vee \langle a\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p$	$\mathrm{PL}_{C;D}$: 61
63.	$(\neg(\langle a^{d^*}\rangle\neg p) \rightarrow p)$	$\mathrm{PL}_{D;C}$: 10

64.	$(\neg(\langle a^{d^*}\rangle\neg p) \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	$\mathrm{PL}_{D;C}$: 62
65.	$((\neg(\langle a^{d^*}\rangle\neg p) \to p) \to ((\neg(\langle a^{d^*}\rangle\neg p) \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \to (\neg(\langle a^{d^*}\rangle\neg p) \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
66.	$((\neg(\langle a^{d^*}\rangle\neg p) \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p) \to (\neg(\langle a^{d^*}\rangle\neg p) \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)))$	MP: 63, 65
67.	$(\neg(\langle a^{d^*}\rangle\neg p) \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	MP: 64, 66
68.	$(\langle a^{d^*}\rangle\neg p \vee (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	$\mathrm{PL}_{C;D}$: 67
69.	$(\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	$\mathrm{PL}_{D;C}$: 68
70.	$((\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)) \to (\langle a^\times\rangle p \to p))$	PL: $(A \to B \wedge C) \to (A \to B)$
71.	$(\langle a^\times\rangle p \to p)$	MP: 69, 70
72.	$(\langle\langle a^\times\rangle p!\rangle p \leftrightarrow (\langle a^\times\rangle p \vee p))$	Ax9 !
73.	$(\langle\langle a^\times\rangle p!\rangle p \to (\langle a^\times\rangle p \vee p))$	BE_L: 72
74.	$((\langle a^\times\rangle p \to p) \to ((\langle\langle a^\times\rangle p!\rangle p \to (\langle a^\times\rangle p \vee p)) \to (\langle\langle a^\times\rangle p!\rangle p \to p)))$	PL: $(A \to B) \to ((C \to A \vee B) \to (C \to B))$
75.	$((\langle\langle a^\times\rangle p!\rangle p \to (\langle a^\times\rangle p \vee p)) \to (\langle\langle a^\times\rangle p!\rangle p \to p))$	MP: 71, 74
76.	$(\langle\langle a^\times\rangle p!\rangle p \to p)$	MP: 73, 75
77.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \to \langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	Mon: 76
78.	$(\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	Mon: 77
79.	$((\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)) \to (\langle a^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p))$	PL: $(A \to B \wedge C) \to (A \to C)$
80.	$(\langle a^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle\langle\langle a^\times\rangle p!\rangle p)$	MP: 72, 79
81.	$(\langle a^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 80, 78
82.	$((\langle a^\times\rangle p \to p) \to ((\langle a^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to (\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$
83.	$((\langle a^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to (\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)))$	MP: 71, 82
84.	$(\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	MP: 81, 83
85.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \leftrightarrow (p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	Ax8 ×
86.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	BE_L: 85
87.	$((\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)) \to (\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	PL: $(A \to B \wedge C) \to (A \to C)$
88.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	MP: 86, 87
89.	$(\langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \leftrightarrow \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	Ax2 ;
90.	$(\langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	BE_L: 89
91.	$(\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \leftrightarrow (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	Ax9 !
92.	$(\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	BE_L: 91
93.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 88, 90
94.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	PL_{CUT}: 93, 92
95.	$((\langle a^\times\rangle p \to (p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)) \to ((\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to ((p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))))$	PL: $(A \to B \wedge C) \to ((A \vee C) \to ((B \wedge C) \vee C))$
96.	$((\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to ((p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	MP: 84, 95
97.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to ((p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	PL_{CUT}: 94, 96
98.	$(((p \wedge \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL: $((A \wedge B) \vee B) \to B$
99.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 96, 98
100.	$(\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle a^\times\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	$\mathrm{BarInd}^\times$: 99
101.	$(\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to (\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))$	PL: $A \to A \vee B$
102.	$((\langle a^\times\rangle p \vee \langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	BE_R: 92
103.	$(\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 101, 102
104.	$(\langle\langle a^\times\rangle p!\rangle\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	BE_R: 90
105.	$(\langle a\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 103, 104
106.	$(\langle a^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p)$	PL_{CUT}: 82, 105
107.	$((\langle a^\times\rangle p \to p) \to ((\langle a^\times\rangle p \to \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p) \to (\langle a^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!; a)\rangle\langle(\langle a^\times\rangle p!; a)^\times\rangle p))))$	PL: $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$

108.	$((\langle a^\times\rangle p \to \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p) \to$ $(\langle a^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p)))$	MP: $72, 107$
109.	$(\langle a^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p))$	MP: $106, 108$
110.	$(\langle(\langle a^\times\rangle p!;a)^\times\rangle p \leftrightarrow (p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p))$	Ax8 $\times$
111.	$((p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p) \to \langle(\langle a^\times\rangle p!;a)^\times\rangle p)$	BE_R: 110
112.	$(\langle a^\times\rangle p \to \langle(\langle a^\times\rangle p!;a)^\times\rangle p)$	PL_{CUT}: $109, 111$
113.	$((\langle(\langle a^\times\rangle p!;a)^\times\rangle p \to (p \wedge \langle(\langle a^\times\rangle p!;a)\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p)) \to$ $(\langle(\langle a^\times\rangle p!;a)^\times\rangle p \to p))$	PL: $(A \to B \wedge C) \to (A \to B)$
114.	$(\langle(\langle a^\times\rangle p!;a)^\times\rangle p \to p)$	MP: $87, 113$
115.	$(\langle a^\times\rangle p \to \langle a^\times\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p)$	PL_{CUT}: $112, 100$
116.	$(\langle a^\times\rangle\langle(\langle a^\times\rangle p!;a)^\times\rangle p \to \langle a^\times\rangle p)$	Mon: 114
117.	$(\langle a^\times\rangle p \to \langle a^\times\rangle p)$	PL_{CUT}: $115, 116$
118.	$\langle a^{d^*}\rangle \neg p \vee \langle a^\times\rangle p$	$\mathrm{PL}_{C;D}$: 117

X Rascal Code Listings

Listing 1: "GtoParTool.rsc"

```
module GtoParTool

/*
  This modules contains all the main functions to parse a proof in G and output a proof in
    Par (and LaTeX).
*/

import GSyntax;
import GLAST;
import CST2AST;
import LaTeXOutput;
import GtoPar;
import DeepInferenceRules;
import StrongInductionRule;

import util::IDE;
import ParseTree;

// Function to be called in the Rascal terminal to activate keyword highlighting in .txt
    files for G proofs
void IDE() {
  start[SGProof] g(str src, loc l) {
    return parse(#start[SGProof], src, l);
  }

  registerLanguage("G", "txt", g);
}

// Returns the path to an input file
loc inputLoc(str fileName) {
  return (|project://GtoPar/input| + fileName)[extension=".txt"];
}

// Returns the path to an output file
loc outputLoc(str fileName) {
  return (|project://GtoPar/output| + fileName)[extension=".tex"];
}

// Returns an abstract syntax tree given an input filename of a G proof
GProof getGAST(str fileName){
  loc l = inputLoc(fileName);
  start[SGProof] cst = parse(#start[SGProof], l);
  return cst2astG(cst);
}

// Constructs a LaTeX file from a G proof
void G2LaTeX(GProof p, str out){
  loc l = outputLoc(out);
  LaTeXOutput(p, l);
}

// Constructs a LaTeX file from a Par proof
void Par2LaTeX(list[ParLine] p, str out){
  loc l = outputLoc(out);
  LaTeXOutput(p, l);
}

// Constructs a LaTeX file from a G input file
void input2latex(str \in,  str out){
  G2LaTeX(getGAST(\in), out);
}

// Main function to transform a G proof into a Par proof given an input filename
```

```
void GtoPar_Tool(str fileName){
  GProof ast = getGAST(fileName);
  list[ParLine] p = GtoPar(ast);
  Par2LaTeX(p, fileName);
}
```

Listing 2: “GSyntax.rsc”

```
module GSyntax
/*
 * Module containing the concrete syntax of the proof system G
 */

extend lang::std::Layout;
extend lang::std::Id;

// Proof file starts with "G" and wrapped with {}
start syntax SGProof
  = "G" "{" SGPart* ps "}";

/*
 * A file of a G proof is a list of G expressions, each wrapped in square brackets [],
    followed by the rule.
 *
 * In case of the conjunction rule, the proof tree splits into two branches wrapped in curly
    brackets {}
 * where both sides of the 'and' are derived
 *
 * The proof tree and each branch is closed by a 'leaf'
 */
syntax SGPart
  = "[" SGExpr* seq "]" SGRule rule ";"
  | "[" SGExpr* seq "]" "and" ";" "{" SGPart* psL "}" "{" SGPart* psR "}"
  | "leaf" ";";

// Syntax definition for list of rules defined for G, excluding the conjunction rule
syntax SGRule
  = "Ax"
  | "weak"
  | "modm"
  | "or"
  | "iter"
  | "dIter"
  | "choice"
  | "dChoice"
  | "test"
  | "dTest"
  | "monG"
  | "monF"
  | "concatd"
  | "inds";

// Syntax definition of normal form game logic expression in G
syntax SGExpr
  = prop: SId p
  | not: "~" SId p
  > left parent: "(" SGExpr ex ")"
  > right strat: "\<" SGGame g "\>" SGExpr ex
  > left and: SGExpr exL "&" SGExpr exR
  > left or: SGExpr exL "|" SGExpr exR;

// Syntax definition of normal form game expression in G
syntax SGGame
  = agame: SId g
  | dual: SGGame g "^d"
  > left parent: "(" SGGame ga ")"
  > left \test: SGExpr ga "?"
  > left dTest: SGExpr ga "!"
```

```
  > left iter: SGGame ga "*"
  > left dIter: SGGame ga "^x"
  > left concat: SGGame gaL ";" SGGame gaR
  > left dChoice: SGGame gaL "&&" SGGame gaR
  > left choice: SGGame gaL "||" SGGame gaR;

// Syntax definition of a named ID
// Used for atomic games, atomic propositions
syntax SId = Id n;

// Regular Expression definition for an integer that can be used for an ID subscript
lexical Int
  = [1-9][0-9]*
  | [0];
```

Listing 3: "GLAST.rsc"

```
module GLAST
/*
 * Module defining all Abstract Syntax Types for G, Par and Game Logic formulas/games
 */

/*
 * Abstract Syntax for a G Proof
 */

/*
  A GProof is a proof tree consisting of either
    a leaf,
    a unary inference with a sequent, a rule and the GProof inference,
    a binary inference from a conjunction rule with a left branch and right branch
*/
data GProof(loc src = |tmp:///|)
  = GLeaf()
  | GUnaryInf(list[GameLog] seq, GRule rule, GProof inf)
  | GBinaryInf(list[GameLog] seq, GProof infL, GProof infR);

/*
  Abstract types of all rules in G
*/
data GRule(loc src = |tmp:///|)
  = ax()
  | weak()
  | modm()
  | or()
  | and()
  | iter()
  | dIter()
  | choice()
  | dChoice()
  | \test()
  | dTest()
  | mongd()
  | monfd()
  | concatD()
  | inds()
  | deepInf();

/*
 * Abstract Syntax for a Par Proof
 */

/*
  A Par proof is a list of ParLines, a ParLine is either:
    a ParStat(ement) with a list of disjunctive formulas 'disjs' and an associated Par rule,
    a ParStat as before but with an additional colour for the LaTeX output,
    a ParStatStart with an additional G rule indicating the begin of a G rule derivation in
    Par,
```

```
    a ParStatStart with an additional G rule indicating the end of a G rule derivation in Par
*/
data ParLine(loc src = |tmp:///|)
  = ParStat(list[GameLog] disjs, ParRule rule)
  | ParStat(list[GameLog] disjs, ParRule rule, str colour)
  | ParStatStart(list[GameLog] disjs, ParRule rule, GRule startRule)
  | ParStatEnd(list[GameLog] disjs, ParRule rule, GRule endRule);

/*
  Abstract types of all Par rules and used tautologies
*/
data ParRule(loc src = |tmp:///|)
  = concatAx()         // <a;b>p <-> <a><b>p
  | choiceAx()         // <a || b>p <-> <a>p v <b>p
  | iterAx()           // <a*>p <-> p v <a><a*>p
  | testAx()           // <q?>p <-> q ^ p
  | dualAx()           // <g^d>p <-> ~<g>~p
  | dChoiceAx()        // <a && b>p <-> <a>p ^ <b>p
  | dIterAx()          // <a^x>p <-> p ^ <a><a^x>p
  | dTestAx()          // <q!>p <-> q v p
  | mp(int var, int cond)   // (p, p -> q) -> q
  | mon(int ref)         // (p -> q) -> (<y>p -> <y>q)
  | barInd(int ref)      // (<y>p -> p) -> (<y*>p -> p)
  | dBarInd(int ref)       // (p -> <y>p) -> (p -> <y^x>p)
  | condSIL(int ref)       // Phi v premise -> (~Phi -> premise)
  | condSEL(int ref)       // (~Phi -> premise) -> (Phi v premise)
  | cut(int l, int r)      // (A -> B) ^ (B -> C) -> (A -> C)
  | ci()             // A -> (B -> (A ^ B))
  | bicondL(int ref)       // (A <-> B) -> (A -> B)
  | bicondR(int ref)       // (A <-> B) -> (B -> A)
  | commOr()           // A v B -> B v A
  | condDL()           // (A -> B) -> ((C -> A v B) -> (C -> B))
  | condIL()           // A v B -> (~A -> B)
  | condIR()           // A v B -> (~B -> A)
  | condEL()           // (~A -> B) -> A v B
  | condCL()           // (A -> B ^ C) -> (A -> B)
  | condCR()           // (A -> B ^ C) -> (A -> C)
  | condCI()           // (A -> B) -> ((A -> C) -> (A -> B ^ C))
  | condDER()          // (A v B -> C) -> (B -> C)
  | condDEL()          // (A v B -> C) -> (A -> C)
  | condDIL()          // (A -> B) -> (A -> B v C)
  | condDIR()          // (A -> B) -> (A -> C v B)
  | condDIDR()         // (A -> B) -> (A v C -> B v C)
  | condDIDL()         // (A -> B) -> (C v A -> C v B)
  | condCIR()          // (A -> B) -> (A ^ C -> B ^ C)
  | condCIL()          // (A -> B) -> (C ^ A -> C ^ B)
  | bicondWR()         // (A <-> B) -> (B v C -> A v C)
  | exMidPL()          // A v ~A
  | weakPL()           // A -> A v B
  | weakLPL()          // A -> B v A
  | andPL()          // A -> (B -> A ^ B)
  | mongdChoice()        // ((A -> B v D) ^ (C -> B v D)) -> (A v C -> B v D)
  | mpAndL()           // ((A ^ B) -> ((A -> C) -> (C ^ B))
  | mpAndR()           // ((A ^ B) -> ((B -> C) -> (A ^ C))
  | mpOrL()          // ((A v B) -> ((A -> C) -> (C v B))
  | mpOrR()          // ((A v B) -> ((B -> C) -> (A v C))
  | mpCondL()          // ((A -> B) -> ((A -> C) -> (B -> C))
  | mpCondR()          // ((A -> B) -> ((C -> A) -> (C -> B))
  | mpBicondL()        // ((A -> B) -> ((A <-> C) -> (B <-> C))
  | mpBicondR()        // ((A -> B) -> ((C <-> A) -> (C <-> B))
  | monAB()          // A -> ((A -> B) -> B)
  | orPL()           // (A v B v C) -> (A v (B v C))
  | strong()           // (A -> B ^ C) -> ((A v C) -> ((B ^ C) v C))
  | andOr();           // ((A ^ B) v B) -> B

/*
 * Abstract Syntax for a Game Logic Formula
```

```
 */

data GameLog(loc src = |tmp:///|)
  = atomP(Prop p)
  | neg(GameLog pr)
  | \mod(Game g, GameLog pr)
  | and(GameLog pL, GameLog pR)
  | or(GameLog pL, GameLog pR)
  | cond(GameLog pL, GameLog pR)
  | bicond(GameLog pL, GameLog pR)
  | condS(list[GameLog] pLs, GameLog pR)
  | orS(list[GameLog] pLs, GameLog pR)
  | negS(list[GameLog] pLs);

data Game(loc src = |tmp:///|)
  = atomG(AGame g)
  | dual(Game ga)
  | \test(GameLog p)
  | dTest(GameLog p)
  | iter(Game ga)
  | dIter(Game ga)
  | concat(Game gL, Game gR)
  | choice(Game gL, Game gR)
  | dChoice(Game gL, Game gR);

/*
 * Abstract syntax for atomic games and propositions
 */

data AGame(loc src = |tmp:///|)
  = agame(str l)
  | agameS(str l, int sub);

data Prop(loc src = |tmp:///|)
  = prop(str l)
  | propS(str l, int sub);
```

Listing 4: “CST2AST.rsc”

```
module CST2AST
/*
 * Module containing the functions to transform the Concrete Syntax Tree of a G proof into
    an Abstract Syntax Tree
 */

import ParseTree;
import String;

import GSyntax;
import GLAST;
import List;

/*
  Main function to for conversion to an AST

  Input: SGProof, concrete syntax tree
  Output: GProof, abstract syntax tree
 */
GProof cst2astG(start[SGProof] sp){
  SGProof p = sp.top;

  return cst2astG([ pp | SGPart pp <- p.ps ]);
}

/*
  Function that converts a concrete syntax G Part into an abstract syntax GProof.
  Recursively calls the rest of the proof in every GUnaryInf and GBinaryInf.

```

```
  Input: Concrete syntax of an expression
  Output: Abstract syntax of an expression
 */
GProof cst2astG(list[SGPart] ps){
  switch(head(ps)){
    case (SGPart)`leaf;`:
      return GLeaf();
    case (SGPart)`[ <SGExpr* seq> ] <SGRule rule> ;`:
      return GUnaryInf([cst2astG(e) | SGExpr e <- seq ],
                cst2astG(rule),
                cst2astG(tail(ps)));
    case (SGPart)`[ <SGExpr* seq> ] and ; { <SGPart* psL> } { <SGPart* psR> }`:
      return GBinaryInf([cst2astG(e) | SGExpr e <- seq ],
                cst2astG([ pp | SGPart pp <- psL ]),
                cst2astG([ pp | SGPart pp <- psR ]));
    default: throw "Unsupported G Proof";
  }
}

/*
  Function that converts a concrete syntax Rule into an abstract syntax Rule.

  Input: Concrete syntax of a rule
  Output: Abstract syntax of a rule
 */
GRule cst2astG(SGRule r){
  switch(r){
    case (SGRule)`Ax`: return ax();
    case (SGRule)`weak`: return weak();
    case (SGRule)`modm`: return modm();
    case (SGRule)`or`: return or();
    case (SGRule)`iter`: return iter();
    case (SGRule)`dIter`: return dIter();
    case (SGRule)`choice`: return choice();
    case (SGRule)`dChoice`: return dChoice();
    case (SGRule)`test`: return \test();
    case (SGRule)`dTest`: return dTest();
    case (SGRule)`monG`: return mongd();
    case (SGRule)`monF`: return monfd();
    case (SGRule)`concatd`: return concatD();
    case (SGRule)`inds`: return inds();
    default: throw "Unsupported G Rule";
  }
}

/*
  Function that converts a concrete syntax Expression into an abstract syntax Expression.

  Input: Concrete syntax of an expression
  Output: Abstract syntax of an expression
 */
GameLog cst2astG(SGExpr exp){
  switch (exp){
    case(SGExpr)`~ <SId p>`:
      return neg(atomP(id2prop(p)));
    case(SGExpr)`<SId p>`:
      return atomP(id2prop(p));
    case(SGExpr)`( <SGExpr ex> )`:
      return cst2astG(ex);
    case(SGExpr)`\< <SGGame g> \> <SGExpr ex>`:
      return \mod(cst2astG(g), cst2astG(ex));
    case(SGExpr)`<SGExpr exL> & <SGExpr exR>`:
      return and(cst2astG(exL), cst2astG(exR));
    case(SGExpr)`<SGExpr exL> | <SGExpr exR>`:
      return or(cst2astG(exL), cst2astG(exR));
    default: throw "Unsupported Game Logic Formula";
  }
}
```

```
97
98  /*
99    Function that converts a concrete syntax Game into an abstract syntax Game.
100
101   Input: Concrete syntax of a game
102   Output: Abstract syntax of a game
103  */
104 Game cst2astG(SGGame g){
105   switch (g){
106     case (SGGame)`<SGGame ga> ^d`:
107       return dual(cst2astG(ga));
108     case (SGGame)`<SId a>`:
109       return atomG(id2agame(a));
110     case (SGGame)`( <SGGame ga> )`:
111       return cst2astG(ga);
112     case (SGGame)`<SGExpr ex> ?`:
113       return \test(cst2astG(ex));
114     case (SGGame)`<SGExpr ex> !`:
115       return dTest(cst2astG(ex));
116     case (SGGame)`<SGGame ga> *`:
117       return iter(cst2astG(ga));
118     case (SGGame)`<SGGame ga> ^x`:
119       return dIter(cst2astG(ga));
120     case (SGGame)`<SGGame gaL> ; <SGGame gaR>`:
121       return concat(cst2astG(gaL), cst2astG(gaR));
122     case (SGGame)`<SGGame gaL> && <SGGame gaR>`:
123       return dChoice(cst2astG(gaL), cst2astG(gaR));
124     case (SGGame)`<SGGame gaL> || <SGGame gaR>`:
125       return choice(cst2astG(gaL), cst2astG(gaR));
126     default: throw "Unsupported Game Formula";
127   }
128 }
129
130 // Function that converts an atomic proposition
131 Prop id2prop(SId p){
132   switch (p){
133     case (SId)`<Id id>`:
134       return prop("<id>");
135     default: throw "Unsupported Proposition";
136   }
137 }
138
139 // Function that converts an atomic game
140 AGame id2agame(SId a){
141   switch (a){
142     case (SId)`<Id id>`:
143       return agame("<id>");
144     default: throw "Unsupported Proposition";
145   }
146 }
```

Listing 5: “GtoPar.rsc”

```
1  module GtoPar
2
3  /*
4    Module containing all functions that handle the transformation for the basic rules and
       game operation rules
5
6    Every function returns a list of proof lines in Par that correspond with a specific G rule
       .
7    Case distinction is based on the G rule by passing a GProof to the function GtoPar,
8    where every GProof has a GRule parameter that determines the Par lines that need to be
       constructed.
9
10   Function template: list[ParLine] GtoPar(GUnaryInf(sequent: list[GameLog] seq, G rule:
       GRule rule, rest of proof tree: GProof inf);
11
```

```
  Every (almost every) function identifies the active formula in both seq and inf.
  The active formula in seq is the conclusion of the rule, the one in inf.seq is the premise
    /hypothesis.
  Then, by recursion, a list of Par lines is returned by first calling GtoPar(inf) on the
    rest of the G proof,
  followed by a list of Par lines on the current rule to be transformed.

  In most of the functions the distinction is made between having side formulas or not.
  If a sequent has side formulas, then some more Par lines are needed to operate on the
    active formula within the sequent.
*/

import List;
import GLAST;
import DeepInferenceRules;
import StrongInductionRule;

/*
  Function returning an empty list in a recursive iteration for a deep inference rule
  Serves as a relative root in a recursive Par derivation

  input: GProof with as rule "deepInf()" and as inf a "GLeaf()"
  output: empty list, terminating the recursion in a deep inference rule
*/
list[ParLine] GtoPar(GUnaryInf(_,deepInf(),GLeaf())) = [];

/*
  ----Transformations of basic rules----
*/

/*
  Ax rule: just return the same line in Par

  input: GProof with as rule "ax()"
  output: one Par line with the sequent and ParRule the "law of excluded middle"
*/
list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, ax(), GProof inf)) = [ParStat(seq,exMidPL
    ())];

/*
  weak rule: always has side formulas

  input: GProof with as rule "weak()"
  output: Transformed Par proof
*/
list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, weak(), GProof inf)) {
  GameLog w = getOneFrom(seq - inf.seq);
  return GtoPar(inf) +
       ParStatStart([condS(inf.seq,orS(inf.seq,w))], weakPL(), weak()) +
       ParStatEnd(inf.seq + w, mp(2,1), weak());
}

/*
  or rule:

  input: GProof with as rule "or()"
  output: Transformed Par proof
*/
list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, or(), GProof inf)) {
  list[GameLog] ors = inf.seq - seq;
  list[GameLog] side = inf.seq - ors;
  GameLog orLog = getOneFrom(seq - inf.seq);
  if(side == []) {
    return GtoPar(inf);
  }
  return GtoPar(inf) +
       ParStatStart([condS(inf.seq,orS(side,orLog))], orPL(), or()) +
       ParStatEnd([orS(side,orLog)], mp(2,1), or());
```

```
}

/*
  and rule:

  input: GProof that is a GBinaryInf, meaning it splits into two GProof branches
  output: Perform recursion on both proof branches, add both to the transformed Par proof
*/
list[ParLine] GtoPar(GBinaryInf(list[GameLog] seq, GProof infL, GProof infR)) {
  GameLog antL = getOneFrom(infL.seq - seq);
  GameLog antR = getOneFrom(infR.seq - seq);
  list[GameLog] side = infL.seq - (infL.seq - infR.seq);

  list[ParLine] left = GtoPar(infL);
  list[ParLine] right = GtoPar(infR);

  // no side formulas
  if(side == []) {
    return left +
         right +
         ParStatStart([cond(antL,cond(antR,and(antL,antR)))], ci(), and()) +
         ParStat([cond(antR,and(antL,antR))], mp(3,1)) +
         ParStatEnd([and(antL,antR)], mp(3,1), and());
  }
  // with side formulas
  return left +
       right +
       ParStatStart([cond(negS(side),antL)], condSIL(size(right)+1), and()) +
       ParStat([cond(negS(side),antR)], condSIL(2)) +
       ParStat([cond(cond(negS(side),antL),cond(cond(negS(side),antR),cond(negS(side),and(
    antL,antR))))], condCI()) +
       ParStat([cond(cond(negS(side),antR),cond(negS(side),and(antL,antR)))], mp(3,1)) +
       ParStat([cond(negS(side),and(antL,antR))], mp(3,1)) +
       ParStatEnd([orS(side,and(antL,antR))], condSEL(1), and());
}


/*
  Returns some games and gamelogs for modm transformation
*/
tuple[Game,Game,GameLog,GameLog,bool] getGames(\mod(g,phi),\mod(dual(g),psi)) {
  return <g,dual(g),phi,psi,false>;
}
tuple[Game,Game,GameLog,GameLog,bool] getGames(\mod(dual(g),psi),\mod(g,phi)) {
  return <g,dual(g),phi,psi,true>;
}

/*
  modm rule:

  input: GProof with as rule "modm()"
  output: Transformed Par proof
*/
list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, modm(), GProof inf)) {
  list[GameLog] con = seq - inf.seq;
  tuple[Game g, Game gd, GameLog phi, GameLog psi, bool switched] t = getGames(con[0],con
    [1]);

  GameLog phiG = \mod(t.g,t.phi);
  GameLog dualG = \mod(t.gd,t.psi);
  GameLog negAngG = neg(\mod(t.g,neg(t.psi)));

  list[ParLine] proof = [];

  if(!t.switched) {
    return GtoPar(inf) +
         ParStatStart([or(t.phi,t.psi),or(t.psi,t.phi)], commOr(), modm()) +
         ParStat([or(t.psi,t.phi)], mp(2,1)) +
```

```
142           ParStat([cond(neg(t.psi),t.phi)], condSIL(1)) +
143             ParStat([cond(\mod(t.g,neg(t.psi)), phiG)], mon(1)) +
144             ParStat([or(negAngG, phiG)], condSEL(1)) +
145             ParStat([bicond(dualG, negAngG)], dualAx()) +
146             ParStat([cond(bicond(dualG, negAngG), cond(or(negAngG,phiG), or(dualG,phiG)))],
    bicondWR()) +
147           ParStat([cond(or(negAngG,phiG), or(dualG,phiG))], mp(2,1)) +
148           ParStat([or(dualG,phiG)], mp(4,1)) +
149           ParStat([cond(or(dualG,phiG),or(phiG,dualG))], commOr()) +
150           ParStatEnd([or(phiG,dualG)], mp(2,1), modm());
151   }
152
153   return GtoPar(inf) +
154          ParStatStart([cond(neg(t.psi),t.phi)], condSIL(1), modm()) +
155        ParStat([cond(\mod(t.g,neg(t.psi)), phiG)], mon(1)) +
156          ParStat([or(negAngG, phiG)], condSEL(1)) +
157          ParStat([bicond(dualG, negAngG)], dualAx()) +
158          ParStat([cond(bicond(dualG, negAngG), cond(or(negAngG,phiG), or(dualG,phiG)))],
    bicondWR()) +
159          ParStat([cond(or(negAngG,phiG), or(dualG,phiG))], mp(2,1)) +
160          ParStatEnd([or(dualG,phiG)], mp(4,1), modm());
161 }
162
163
164 /*
165   ----Transformations of game operation rules----
166 */
167
168
169 /*
170   Function that transforms a Game Operation rule; functionality is equal for every game
      operator
171
172   input: G proof with a Game Operation rule
173   output: Transformed Par proof
174 */
175 list[ParLine] GtoPar(list[GameLog] seq, GRule rule, GProof inf) {
176   ParRule axiom = getGameOpParAx(rule);
177   GameLog ant = getOneFrom(inf.seq - seq);
178   GameLog con = getOneFrom(seq - inf.seq);
179   list[GameLog] sideInf = inf.seq - ant;
180   // no side formulas
181   if(sideInf == []) {
182     return GtoPar(inf) +
183          ParStatStart([bicond(con,ant)], axiom, rule) +
184            ParStat([cond(ant,con)], bicondR(1)) +
185            ParStatEnd(seq, mp(3,1), rule);
186   }
187   // with side formulas
188   return GtoPar(inf) +
189        ParStatStart([cond(negS(sideInf),ant)], condSIL(1), rule) +
190        ParStat([bicond(con,ant)], axiom) +
191        ParStat([cond(ant,con)], bicondR(1)) +
192        ParStat([cond(negS(sideInf),con)], cut(3,1)) +
193        ParStatEnd(seq, condSEL(1), rule);
194 }
195
196 // Function for all Game Operation rules that all call the same general Game Op
      transformation function listed here above
197 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, iter(), GProof inf)) = GtoPar(seq,iter(),
      inf);
198 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, dIter(), GProof inf)) = GtoPar(seq,dIter()
      ,inf);
199 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, choice(), GProof inf)) = GtoPar(seq,choice
      (),inf);
200 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, dChoice(), GProof inf)) = GtoPar(seq,
      dChoice(),inf);
201 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, \test(), GProof inf)) = GtoPar(seq,\test()
```

```
    ,inf);
list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, dTest(), GProof inf)) = GtoPar(seq,dTest()
    ,inf);

// Functions to get the corresponding Par axiom from a Game Operation G rule
ParRule getGameOpParAx(iter()) = iterAx();
ParRule getGameOpParAx(dIter()) = dIterAx();
ParRule getGameOpParAx(choice()) = choiceAx();
ParRule getGameOpParAx(dChoice()) = dChoiceAx();
ParRule getGameOpParAx(\test()) = testAx();
ParRule getGameOpParAx(dTest()) = dTestAx();
```

Listing 6: "DeepInferenceRules.rsc"

```
module DeepInferenceRules

/*
  Module containing all functions that handle the transformation for the deep inference
    rules Mongd, Monfd and ;d

  Every function returns a list of proof lines in Par that correspond with a specific Deep
    inference rule.
  Case distinction is based on the G rule by passing a GProof to the function GtoPar,
  where every GProof has a GRule parameter that determines the Par lines that need to be
    constructed.

  Function template: list[ParLine] GtoPar(GUnaryInf(sequent: list[GameLog] seq, G rule:
    GRule rule, rest of proof tree: GProof inf);

  Every (almost every) function identifies the active formula in both seq and inf.
  The active formula in seq is the conclusion of the rule, the one in inf.seq is the premise
    /hypothesis.
  Then, by recursion, a list of Par lines is returned by first calling GtoPar(inf) on the
    rest of the G proof,
  followed by a list of Par lines on the current rule to be transformed.
*/

import List;

import GLAST;
import GtoPar;
import StrongInductionRule;

/*
  General function for all deep inference rules, handles the initiation of the rule
    transformation

  Input: GProof/Sequent with a certain rule
  Output: Par proof/derivation of that rule
*/
list[ParLine] DeepInfToPar(list[GameLog] seq, GRule rule, GProof inf) {
  // stop recursion when antecedent and consequent are equal
  if(seq == inf.seq) return [];

  // active formulas of antecedent/consequent
  GameLog ant = getOneFrom(inf.seq - seq);
  GameLog con = getOneFrom(seq - inf.seq);

  // side formulas of sequent, might be empty
  list[GameLog] side = inf.seq - ant;

  list[ParLine] proof = DeepInfToPar(ant,con,rule);

  if(side != []) {
    return GtoPar(inf) +
         ParStat([cond(negS(side),ant)], condSIL(1)) +
         proof +
         ParStat([cond(negS(side),con)], cut(size(proof)+1,1)) +
```

```
48          ParStat([orS(side,con)], condSEL(1));
49    }
50    if(inf.rule != deepInf()) {
51      return GtoPar(inf) +
52          proof +
53          ParStat([con], mp(size(proof)+1,1));
54    }
55    return GtoPar(inf) + proof;
56  }
57
58  /*
59    These three functions call the above general function with their G rule as a parameter
60  */
61  list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, mongd(), GProof inf))
62    = DeepInfToPar(seq, mongd(), inf);
63  list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, monfd(), GProof inf))
64    = DeepInfToPar(seq, monfd(), inf);
65  list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, concatD(), GProof inf))
66    = DeepInfToPar(seq, concatD(), inf);
67
68  /*
69    Mongd atomic case: game and deep game are equal
70
71    Input: antecedent/premise, consequent/conclusion, mongd() rule
72    Output: Par proof of mongd atomic case, from premise to conclusion
73  */
74  list[ParLine] DeepInfToPar(\mod(ga,phi), \mod(concat(dTest(chi),ga),phiC), mongd()) {
75    GameLog ant = \mod(ga,phi);
76    GameLog seq = \mod(dTest(chi),ant);
77    GameLog conc = \mod(concat(dTest(chi),ga),phi);
78    str atomicColour = "brown";
79
80    return
81    [
82      ParStat([cond(ant, or(chi,ant))], weakLPL(), atomicColour),
83      ParStat([bicond(seq,or(chi,ant))], dTestAx(), atomicColour),
84      ParStat([cond(or(chi,ant),seq)], bicondR(1), atomicColour),
85      ParStat([cond(ant,seq)], cut(3,1), atomicColour),
86      ParStat([bicond(conc,seq)], concatAx(), atomicColour),
87      ParStat([cond(seq,conc)], bicondR(1), atomicColour),
88      ParStat([cond(ant,conc)], cut(3,1), atomicColour)
89    ];
90  }
91
92  /*
93    Monfd atomic case: gamelog and deep gamelog are equal
94
95    Input: antecedent/premise, consequent/conclusion, monfd() rule
96    Output: Par proof of monfd atomic case, from premise to conclusion
97  */
98  list[ParLine] DeepInfToPar(GameLog phi, \mod(dTest(chi),phi), monfd()) {
99    return [ParStat([cond(phi,or(chi,phi))], weakLPL()),
100       ParStat([bicond(\mod(dTest(chi),phi),or(chi,phi))], dTestAx()),
101       ParStat([cond(or(chi,phi),\mod(dTest(chi),phi))], bicondR(1)),
102       ParStat([cond(phi,\mod(dTest(chi),phi))], cut(3,1))];
103 }
104
105 /*
106   ;d/concatd atomic case: gamelog and deep gamelog are equal
107
108   Input: antecedent/premise, consequent/conclusion, concatD() rule
109   Output: Par proof of ;d atomic case, from premise to conclusion
110 */
111 list[ParLine] DeepInfToPar(\mod(l,\mod(r,phi)), \mod(concat(l,r),phi), concatD()) {
112   return [ParStat([bicond(\mod(concat(l,r),phi),\mod(l,\mod(r,phi)))], concatAx()),
113       ParStat([cond(\mod(l,\mod(r,phi)),\mod(concat(l,r),phi))], bicondR(1))];
114 }
115
```

```
/*
  Adds a list of Par lines for every Mon rule that needs to be applied
  to the Game formula that has been transformed.

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of applications of the Mon rule on the formula where the deep
     inf rule is applied
*/
list[ParLine] DeepInfToPar(\mod(ga,pla), \mod(ga,plc), GRule rule) {
  return DeepInfToPar(pla,plc,rule) +
       ParStat([cond(\mod(ga,pla),\mod(ga,plc))], mon(1));
}

/*
  ! case: adds the needed Par lines to the Par proof in case of a demonic test

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on ! and the recursive step that operates on the
     game inside the !
*/
list[ParLine] DeepInfToPar(\mod(dTest(glA),phi), \mod(dTest(glC),plC), GRule rule) {
  // 'rec' is the recursive step that returns the Par proof for the game inside !
  GProof inf = GUnaryInf([glA],deepInf(),GLeaf());
  list[ParLine] rec = GtoPar(GUnaryInf([glC],rule,inf));
  str testColour = "cyan";

  return
    [
      ParStat([bicond(\mod(dTest(glA),phi),or(glA,phi))], dTestAx(), testColour),
      ParStat([cond(\mod(dTest(glA),phi),or(glA,phi))], bicondL(1), testColour)
    ]
    + rec +
    [
      ParStat([cond(cond(glA,glC),cond(or(glA,phi),or(glC,phi)))], condDIDR(), testColour),
      ParStat([cond(or(glA,phi),or(glC,phi))], mp(2,1), testColour),
      ParStat([cond(\mod(dTest(glA),phi),or(glC,phi))], cut(size(rec)+3,1), testColour),
      ParStat([bicond(\mod(dTest(glC),phi),or(glC,phi))], dTestAx(), testColour),
      ParStat([cond(or(glC,phi),\mod(dTest(glC),phi))], bicondR(1), testColour),
      ParStat([cond(\mod(dTest(glA),phi),\mod(dTest(glC),phi))], cut(3,1), testColour)
    ]
  ;
}

/*
  ? case: adds the needed Par lines to the Par proof in case of an angelic test

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on ? and the recursive step that operates on the
     game inside the ?
*/
list[ParLine] DeepInfToPar(\mod(\test(glA),phi), \mod(\test(glC),plC), GRule rule) {
  // 'rec' is the recursive step that returns the Par proof for the game inside ?
  GProof inf = GUnaryInf([glA],deepInf(),GLeaf());
  list[ParLine] rec = GtoPar(GUnaryInf([glC],rule,inf));
  str dTestColour = "red";

  return
    [
      ParStat([bicond(\mod(\test(glA),phi),and(glA,phi))], testAx(), dTestColour),
      ParStat([cond(\mod(\test(glA),phi),and(glA,phi))], bicondL(1), dTestColour)
    ]
    + rec +
    [
      ParStat([cond(cond(glA,glC),cond(and(glA,phi),and(glC,phi)))], condCIR(), dTestColour)
    ,
      ParStat([cond(and(glA,phi),and(glC,phi))], mp(2,1), dTestColour),
      ParStat([cond(\mod(\test(glA),phi),and(glC,phi))], cut(size(rec)+3,1), dTestColour),
      ParStat([bicond(\mod(\test(glC),phi),and(glC,phi))], testAx(), dTestColour),
```

```
      ParStat([cond(and(glC,phi),\mod(\test(glC),phi))], bicondR(1), dTestColour),
      ParStat([cond(\mod(\test(glA),phi),\mod(\test(glC),phi))], cut(3,1), dTestColour)
    ]
  ;
}

/*
  * case: adds the needed Par lines to the Par proof in case of an angelic iteration

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on * and the recursive step that operates on the
     game inside the *
*/
list[ParLine] DeepInfToPar(\mod(iter(ga),phi), \mod(iter(gc),plC), GRule rule) {
  // 'rec' is the recursive step that returns the Par proof for the game inside *
  GProof inf = GUnaryInf([\mod(ga,\mod(iter(gc),phi))],deepInf(),GLeaf());
  list[ParLine] rec = GtoPar(GUnaryInf([\mod(gc,\mod(iter(gc),phi))],rule,inf));
  str iterColour = "teal";

  return
    [
      ParStat([bicond(\mod(iter(gc),phi),or(phi,\mod(gc,\mod(iter(gc),phi))))], iterAx(),
    iterColour),
      ParStat([cond(or(phi,\mod(gc,\mod(iter(gc),phi))),\mod(iter(gc),phi))], bicondR(1),
    iterColour),
      ParStat([cond(
            cond(or(phi,\mod(gc,\mod(iter(gc),phi))),\mod(iter(gc),phi)),
            cond(\mod(gc,\mod(iter(gc),phi)),\mod(iter(gc),phi)))], condDER(), iterColour),
      ParStat([cond(\mod(gc,\mod(iter(gc),phi)),\mod(iter(gc),phi))], mp(2,1), iterColour)
    ]
    + rec +
    [
      ParStat([cond(\mod(ga,\mod(iter(gc),phi)),\mod(iter(gc),phi))], cut(1,size(rec)+1),
    iterColour),
      ParStat([cond(\mod(iter(ga),\mod(iter(gc),phi)),\mod(iter(gc),phi))], barInd(1),
    iterColour),
      ParStat([cond(
            cond(or(phi,\mod(gc,\mod(iter(gc),phi))),\mod(iter(gc),phi)),
            cond(phi,\mod(iter(gc),phi)))], condDEL(), iterColour),
      ParStat([cond(phi,\mod(iter(gc),phi))], mp(size(rec)+6,1), iterColour),
      ParStat([cond(\mod(iter(ga),phi),\mod(iter(ga),\mod(iter(gc),phi)))], mon(1),
    iterColour),
      ParStat([cond(\mod(iter(ga),phi),\mod(iter(gc),phi))], cut(1,4), iterColour)
    ]
  ;
}

/*
  x case: adds the needed Par lines to the Par proof in case of demonic iteration

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on x and the recursive step that operates on the
     game inside the x
*/
list[ParLine] DeepInfToPar(\mod(dIter(ga),phi), \mod(dIter(gc),plC), GRule rule) {
  // 'rec' is the recursive step that returns the Par proof for the game inside x
  GProof inf = GUnaryInf([\mod(ga,\mod(dIter(ga),phi))],deepInf(),GLeaf());
  list[ParLine] rec = GtoPar(GUnaryInf([\mod(gc,\mod(dIter(ga),phi))],rule,inf));
  str dIterColour = "violet";

  return
    [
      ParStat([bicond(\mod(dIter(ga),phi),and(phi,\mod(ga,\mod(dIter(ga),phi))))], dIterAx()
    , dIterColour),
      ParStat([cond(\mod(dIter(ga),phi),and(phi,\mod(ga,\mod(dIter(ga),phi))))], bicondL(1),
     dIterColour),
      ParStat([cond(
            cond(\mod(dIter(ga),phi),and(phi,\mod(ga,\mod(dIter(ga),phi)))),
```

```
            cond(\mod(dIter(ga),phi),\mod(ga,\mod(dIter(ga),phi)))
            )], condCR(), dIterColour),
      ParStat([cond(\mod(dIter(ga),phi),\mod(ga,\mod(dIter(ga),phi)))], mp(2,1), dIterColour
    )
    ]
    + rec +
    [
      ParStat([cond(\mod(dIter(ga),phi),\mod(gc,\mod(dIter(ga),phi)))], cut(size(rec)+1,1),
    dIterColour),
      ParStat([cond(\mod(dIter(ga),phi),\mod(dIter(gc),\mod(dIter(ga),phi)))], dBarInd(1),
    dIterColour),
      ParStat([cond(
            cond(\mod(dIter(ga),phi),and(phi,\mod(ga,\mod(dIter(ga),phi)))),
            cond(\mod(dIter(ga),phi),phi)
            )], condCL(), dIterColour),
      ParStat([cond(\mod(dIter(ga),phi),phi)], mp(size(rec)+6,1), dIterColour),
      ParStat([cond(\mod(dIter(gc),\mod(dIter(ga),phi)),\mod(dIter(gc),phi))], mon(1),
    dIterColour),
      ParStat([cond(\mod(dIter(ga),phi),\mod(dIter(gc),phi))], cut(1,4), dIterColour)
    ]
  ;
}

/*
  ; case: adds the needed Par lines to the Par proof in case of concatenation

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on ; and the recursive step that operates on the
     left or right game inside the ;
*/
list[ParLine] DeepInfToPar(\mod(concat(gla,gra),phi), \mod(concat(glc,grc),plC), GRule rule)
     {
  str concatColour = "olive";

  // in both left and right cases these two Par lines are needed
  list[ParLine] beforeRec =
      [ParStat([bicond(\mod(concat(gla,gra),phi),\mod(gla,\mod(gra,phi)))], concatAx(),
    concatColour),
      ParStat([cond(\mod(concat(gla,gra),phi),\mod(gla,\mod(gra,phi)))], bicondL(1),
    concatColour)];

  // rule is applied in left game
  if(gra == grc) {
    // 'rec' is the recursive step that returns the Par proof for the left game inside ;
    GProof infLeft = GUnaryInf([\mod(gla,\mod(gra,phi))],deepInf(),GLeaf());
    list[ParLine] recLeft = GtoPar(GUnaryInf([\mod(glc,\mod(gra,phi))],rule,infLeft));

    return beforeRec +
         recLeft +
         ParStat([cond(\mod(concat(gla,gra),phi),\mod(glc,\mod(gra,phi)))], cut(size(recLeft
    )+1,1), concatColour) +
         ParStat([bicond(\mod(concat(glc,gra),phi),\mod(glc,\mod(gra,phi)))], concatAx(),
    concatColour) +
         ParStat([cond(\mod(glc,\mod(gra,phi)),\mod(concat(glc,gra),phi))], bicondR(1),
    concatColour) +
         ParStat([cond(\mod(concat(gla,gra),phi),\mod(concat(glc,gra),phi))], cut(3,1),
    concatColour)
    ;
  }

  // rule is applied in right game
  // 'rec' is the recursive step that returns the Par proof for the right game inside ;
  GProof infRight = GUnaryInf([\mod(gra,phi)],deepInf(),GLeaf());
  list[ParLine] recRight = GtoPar(GUnaryInf([\mod(grc,phi)],rule,infRight));


  return beforeRec +
       recRight +
```

```
295         ParStat([cond(\mod(gla,\mod(gra,phi)),\mod(gla,\mod(grc,phi)))], mon(1), concatColour
    ) +
296         ParStat([cond(\mod(concat(gla,gra),phi),\mod(gla,\mod(grc,phi)))], cut(size(recRight)
    +2,1), concatColour) +
297         ParStat([bicond(\mod(concat(gla,grc),phi),\mod(gla,\mod(grc,phi)))], concatAx(),
    concatColour) +
298         ParStat([cond(\mod(gla,\mod(grc,phi)),\mod(concat(gla,grc),phi))], bicondR(1),
    concatColour) +
299         ParStat([cond(\mod(concat(gla,gra),phi),\mod(concat(gla,grc),phi))], cut(3,1),
    concatColour)
300   ;
301 }
302
303 /*
304   choice case: adds the needed Par lines to the Par proof in case of angelic choice
305
306   Input: antecedent/premise, consequent/conclusion, G rule
307   Output: Par proof consisting of operating on 'choice' and the recursive step that operates
       on the left or right game inside 'choice'
308 */
309 list[ParLine] DeepInfToPar(\mod(choice(gla,gra),phi), \mod(choice(glc,grc),plC), GRule rule)
      {
310   str choiceColour = "magenta";
311
312   // in both left and right cases these two Par lines are needed
313   list[ParLine] beforeRec =
314       [ParStat([bicond(\mod(choice(gla,gra),phi),or(\mod(gla,phi),\mod(gra,phi)))], choiceAx
    (), choiceColour),
315        ParStat([cond(\mod(choice(gla,gra),phi),or(\mod(gla,phi),\mod(gra,phi)))], bicondL(1),
     choiceColour)];
316
317   // rule is applied in left game
318   if(gra == grc) {
319     // 'rec' is the recursive step that returns the Par proof for the left game inside '
    choice'
320     GProof infLeft = GUnaryInf([\mod(gla,phi)],deepInf(),GLeaf());
321     list[ParLine] recLeft = GtoPar(GUnaryInf([\mod(glc,phi)],rule,infLeft));
322
323     return beforeRec +
324          recLeft +
325          ParStat([cond(cond(\mod(gla,phi),\mod(glc,phi)),cond(or(\mod(gla,phi),\mod(gra,phi)
    ),or(\mod(glc,phi),\mod(gra,phi))))], condDIDR(), choiceColour) +
326          ParStat([cond(or(\mod(gla,phi),\mod(gra,phi)),or(\mod(glc,phi),\mod(gra,phi)))], mp
    (2,1), choiceColour) +
327          ParStat([cond(\mod(choice(gla,gra),phi),or(\mod(glc,phi),\mod(gra,phi)))], cut(size
    (recLeft)+3,1), choiceColour) +
328          ParStat([bicond(\mod(choice(glc,gra),phi),or(\mod(glc,phi),\mod(gra,phi)))],
    choiceAx(), choiceColour) +
329          ParStat([cond(or(\mod(glc,phi),\mod(gra,phi)),\mod(choice(glc,gra),phi))], bicondR
    (1), choiceColour) +
330          ParStat([cond(\mod(choice(gla,gra),phi),\mod(choice(glc,gra),phi))], cut(3,1),
    choiceColour)
331     ;
332   }
333
334   // rule is applied in right game
335   // 'rec' is the recursive step that returns the Par proof for the right game inside '
    choice'
336   GProof infRight = GUnaryInf([\mod(gra,phi)],deepInf(),GLeaf());
337   list[ParLine] recRight = GtoPar(GUnaryInf([\mod(grc,phi)],rule,infRight));
338
339   return beforeRec +
340        recRight +
341        ParStat([cond(cond(\mod(gra,phi),\mod(grc,phi)),cond(or(\mod(gla,phi),\mod(gra,phi)),
    or(\mod(gla,phi),\mod(grc,phi))))], condDIDL(), choiceColour) +
342        ParStat([cond(or(\mod(gla,phi),\mod(gra,phi)),or(\mod(gla,phi),\mod(grc,phi)))], mp
    (2,1), choiceColour) +
343        ParStat([cond(\mod(choice(gla,gra),phi),or(\mod(gla,phi),\mod(grc,phi)))], cut(size(
```

```
    recRight)+3,1), choiceColour) +
344       ParStat([bicond(\mod(choice(gla,grc),phi),or(\mod(gla,phi),\mod(grc,phi)))], choiceAx
    (), choiceColour) +
345       ParStat([cond(or(\mod(gla,phi),\mod(grc,phi)),\mod(choice(gla,grc),phi))], bicondR(1)
    , choiceColour) +
346       ParStat([cond(\mod(choice(gla,gra),phi),\mod(choice(gla,grc),phi))], cut(3,1),
    choiceColour)
347   ;
348 }
349
350 /*
351   demonic choice case: adds the needed Par lines to the Par proof in case of demonic choice
352
353   Input: antecedent/premise, consequent/conclusion, G rule
354   Output: Par proof consisting of operating on 'demonic choice' and the recursive step that
      operates on the left or right game inside 'demonic choice'
355 */
356 list[ParLine] DeepInfToPar(\mod(dChoice(gla,gra),phi), \mod(dChoice(glc,grc),plC), GRule
    rule) {
357   str dChoiceColour = "pink";
358
359   // in both left and right cases these two Par lines are needed
360   list[ParLine] beforeRec =
361       [ParStat([bicond(\mod(dChoice(gla,gra),phi),and(\mod(gla,phi),\mod(gra,phi)))],
    dChoiceAx(), dChoiceColour),
362       ParStat([cond(\mod(dChoice(gla,gra),phi),and(\mod(gla,phi),\mod(gra,phi)))], bicondL
    (1), dChoiceColour)];
363
364   // rule is applied in left game
365   if(gra == grc) {
366     // 'rec' is the recursive step that returns the Par proof for the left game inside '
    dChoice'
367     GProof infLeft = GUnaryInf([\mod(gla,phi)],deepInf(),GLeaf());
368     list[ParLine] recLeft = GtoPar(GUnaryInf([\mod(glc,phi)],rule,infLeft));
369
370     return beforeRec +
371          recLeft +
372          ParStat([cond(cond(\mod(gla,phi),\mod(glc,phi)),cond(and(\mod(gla,phi),\mod(gra,phi
    )),and(\mod(glc,phi),\mod(gra,phi))))], condCIR(), dChoiceColour) +
373          ParStat([cond(and(\mod(gla,phi),\mod(gra,phi)),and(\mod(glc,phi),\mod(gra,phi)))],
    mp(2,1), dChoiceColour) +
374          ParStat([cond(\mod(dChoice(gla,gra),phi),and(\mod(glc,phi),\mod(gra,phi)))], cut(
    size(recLeft)+1,1), dChoiceColour) +
375          ParStat([bicond(\mod(dChoice(glc,gra),phi),and(\mod(glc,phi),\mod(gra,phi)))],
    dChoiceAx(), dChoiceColour) +
376          ParStat([cond(and(\mod(glc,phi),\mod(gra,phi)),\mod(dChoice(glc,gra),phi))],
    bicondR(1), dChoiceColour) +
377          ParStat([cond(\mod(dChoice(gla,gra),phi),\mod(dChoice(glc,gra),phi))], cut(3,1),
    dChoiceColour)
378     ;
379   }
380
381   // rule is applied in right game
382   // 'rec' is the recursive step that returns the Par proof for the right game inside '
    dChoice'
383   GProof infRight = GUnaryInf([\mod(gra,phi)],deepInf(),GLeaf());
384   list[ParLine] recRight = GtoPar(GUnaryInf([\mod(grc,phi)],rule,infRight));
385
386   return beforeRec +
387       recRight +
388       ParStat([cond(cond(\mod(gra,phi),\mod(grc,phi)),cond(and(\mod(gla,phi),\mod(gra,phi))
    ,and(\mod(gla,phi),\mod(grc,phi))))], condCIL(), dChoiceColour) +
389       ParStat([cond(and(\mod(gla,phi),\mod(gra,phi)),and(\mod(gla,phi),\mod(grc,phi)))], mp
    (2,1), dChoiceColour) +
390       ParStat([cond(\mod(dChoice(gla,gra),phi),and(\mod(gla,phi),\mod(grc,phi)))], cut(size
    (recRight)+1,1), dChoiceColour) +
391       ParStat([bicond(\mod(dChoice(gla,grc),phi),and(\mod(gla,phi),\mod(grc,phi)))],
    dChoiceAx(), dChoiceColour) +
```

```
392         ParStat([cond(and(\mod(gla,phi),\mod(grc,phi)),\mod(dChoice(gla,grc),phi))], bicondR
    (1), dChoiceColour) +
393         ParStat([cond(\mod(dChoice(gla,gra),phi),\mod(dChoice(gla,grc),phi))], cut(3,1),
    dChoiceColour)
394   ;
395 }
396
397 /*
398   and case: adds the needed Par lines to the Par proof in case of conjunction
399
400   Input: antecedent/premise, consequent/conclusion, G rule
401   Output: Par proof consisting of operating on 'and' and the recursive step that operates on
       the left or right gamelog of conjunction
402 */
403 list[ParLine] DeepInfToPar(and(l,r), and(lc,rc), GRule rule) {
404   // game(log) is in left branch of conjunction
405   if(r == rc) {
406     // 'rec' is the recursive step that returns the Par proof for the left gamelog inside '
    and'
407     GProof infLeft = GUnaryInf([l],deepInf(),GLeaf());
408     list[ParLine] recLeft = GtoPar(GUnaryInf([lc],rule,infLeft));
409
410     return recLeft +
411            ParStat([cond(cond(l,lc),cond(and(l,r),and(lc,rc)))], mpAndL()) +
412            ParStat([cond(and(l,r),and(lc,rc))], mp(2,1));
413   }
414
415   // game(log) is in right branch of conjunction
416   // 'rec' is the recursive step that returns the Par proof for the right gamelog inside '
    and'
417   GProof infRight = GUnaryInf([r],deepInf(),GLeaf());
418   list[ParLine] recRight = GtoPar(GUnaryInf([rc],rule,infRight));
419
420   return recRight +
421          ParStat([cond(cond(r,rc),cond(and(l,r),and(lc,rc)))], mpAndR()) +
422          ParStat([cond(and(l,r),and(lc,rc))], mp(2,1));
423
424 }
425
426 /*
427   and case: adds the needed Par lines to the Par proof in case of disjunction
428
429   Input: antecedent/premise, consequent/conclusion, G rule
430   Output: Par proof consisting of operating on 'or' and the recursive step that operates on
      the left or right gamelog of disjunction
431 */
432 list[ParLine] DeepInfToPar(or(l,r), or(lc,rc), GRule rule) {
433   // game(log) is in left branch of disjunction
434   if(r == rc) {
435     // 'rec' is the recursive step that returns the Par proof for the left gamelog inside '
    or'
436     GProof infLeft = GUnaryInf([l],deepInf(),GLeaf());
437     list[ParLine] recLeft = GtoPar(GUnaryInf([lc],rule,infLeft));
438
439     return recLeft +
440          ParStat([cond(cond(l,lc),cond(or(l,r),or(lc,r)))], condDIDR()) +
441          ParStat([cond(or(l,r),or(lc,r))], mp(2,1));
442   }
443
444   // 'rec' is the recursive step that returns the Par proof for the right gamelog inside 'or
    '
445   GProof infRight = GUnaryInf([r],deepInf(),GLeaf());
446   list[ParLine] recRight = GtoPar(GUnaryInf([rc],rule,infRight));
447
448   // game(log) is in right branch of disjunction
449   return recRight +
450        ParStat([cond(cond(r,rc),cond(or(l,r),or(l,rc)))], condDIDL()) +
451        ParStat([cond(or(l,r),or(l,rc))], mp(2,1));
```

```
}

/*
  conditional case: adds the needed Par lines to the Par proof in case of conditional

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on 'cond' and the recursive step that operates
    on the left or right gamelog of conditional
*/
list[ParLine] DeepInfToPar(cond(l,r), cond(lc,rc), GRule rule) {
  // game(log) is in left branch of conditional
  if(r == rc) {
    // 'rec' is the recursive step that returns the Par proof for the left gamelog inside '
    conditional'
    GProof infLeft = GUnaryInf([l],deepInf(),GLeaf());
    list[ParLine] recLeft = GtoPar(GUnaryInf([lc],rule,infLeft));

    return recLeft +
         ParStat([cond(cond(l,lc),cond(cond(l,r),cond(lc,r)))], mpCondL()) +
         ParStat([cond(or(l,r),or(lc,r))], mp(2,1));
  }

  // game(log) is in right branch of conditional
  // 'rec' is the recursive step that returns the Par proof for the right gamelog inside '
    conditional'
  GProof infRight = GUnaryInf([r],deepInf(),GLeaf());
  list[ParLine] recRight = GtoPar(GUnaryInf([rc],rule,infRight));

  return recRight +
       ParStat([cond(cond(r,rc),cond(or(l,r),or(l,rc)))], mpCondR()) +
       ParStat([cond(or(l,r),or(l,rc))], mp(2,1));

}

/*
  biconditional case: adds the needed Par lines to the Par proof in case of biconditional

  Input: antecedent/premise, consequent/conclusion, G rule
  Output: Par proof consisting of operating on 'bicond' and the recursive step that operates
     on the left or right gamelog of biconditional
*/
list[ParLine] DeepInfToPar(bicond(l,r), bicond(lc,rc), GRule rule) {
  // game(log) is in left branch of biconditional
  if(r == rc) {
    // 'rec' is the recursive step that returns the Par proof for the left gamelog inside '
    biconditional'
    GProof infLeft = GUnaryInf([l],deepInf(),GLeaf());
    list[ParLine] recLeft = GtoPar(GUnaryInf([lc],rule,infLeft));

    return recLeft +
           ParStat([cond(cond(l,lc),cond(bicond(l,r),bicond(lc,r)))], mpBicondL()) +
           ParStat([cond(bicond(l,r),bicond(lc,r))], mp(2,1));
  }

  // game(log) is in right branch of biconditional
  // 'rec' is the recursive step that returns the Par proof for the right gamelog inside '
    biconditional'
  GProof infRight = GUnaryInf([r],deepInf(),GLeaf());
  list[ParLine] recRight = GtoPar(GUnaryInf([rc],rule,infRight));

  return recRight +
         ParStat([cond(cond(r,rc),cond(bicond(l,r),bicond(l,rc)))], mpBicondR()) +
         ParStat([cond(bicond(l,r),bicond(l,rc))], mp(2,1));
}
```

Listing 7: "StrongInductionRule.rsc"

```
module StrongInductionRule
```

```
2
3  /*
4    Module containing all functions that handle the transformation of the strengthened
       induction rule (inds)
5
6  */
7
8  import List;
9
10 import GLAST;
11 import GtoPar;
12 import DeepInferenceRules;
13
14 // Function that returns the formula phi of an unfolded demonic iteration
15 GameLog getPhi(and(l,_)) = l;
16
17 // Function that returns the unfolded right formula of an unfolded demonic iteration
18 GameLog getUnfold(and(_,r)) = r;
19
20 // Function that returns the game of a modality
21 Game getGame(\mod(g,_)) {
22   return g;
23 }
24
25 // Function that returns the negated side formula(s) of the inds() rule
26 GameLog getSide(\mod(_,\mod(dIter(concat(dTest(side),_)),_))) {
27   return side;
28 }
29
30 /*
31   Function that returns a Par proof for the strengthened induction rule
32
33   input: GProof of the strengthened induction rule (inds)
34   output: Par proof of inds()
35 */
36 list[ParLine] GtoPar(GUnaryInf(list[GameLog] seq, inds(), GProof inf)) {
37   // active formula in antecedent/premise
38   GameLog ant = getOneFrom(inf.seq - seq);
39
40   list[GameLog] side = inf.seq - ant;
41
42   // helper variables to make the Par derivation less cluttered
43   GameLog phi = getPhi(ant);
44   GameLog unfold = getUnfold(ant);
45   GameLog negSide = getSide(unfold);
46   Game game = getGame(unfold);
47   GameLog concatPart = \mod(dIter(concat(dTest(negSide),game)),phi);
48   GameLog gConcatPart = \mod(game,concatPart);
49   GameLog testPart = \mod(dTest(negSide),phi);
50   GameLog dblConcatPart = \mod(concat(dTest(negSide),game),concatPart);
51   GameLog testGameConcat = \mod(dTest(negSide),gConcatPart);
52   GameLog gxConcatPart = \mod(dIter(game),concatPart);
53
54   list[ParLine] proof = [];
55
56   if(side == []) {
57     proof += [ParStatStart([or(atomP(prop("p")),neg(atomP(prop("p"))))], exMidPL(), inds()),
58          ParStat([cond(or(atomP(prop("p")),neg(atomP(prop("p")))),or(or(atomP(prop("p")),neg
       (atomP(prop("p")))),ant))], weakPL()),
59          ParStat([or(or(atomP(prop("p")),neg(atomP(prop("p")))),ant)], mp(2,1)),
60          ParStat([cond(negS([atomP(prop("p")),neg(atomP(prop("p")))]),ant)], condSIL(1))];
61   } else {
62     proof += ParStatStart([cond(negSide,ant)], condSIL(1), inds());
63   }
64
65   proof +=
66     [ParStat([cond(cond(negSide,ant),cond(negSide,phi))], condCL()),
67       ParStat([cond(negSide,phi)], mp(2,1)),
```

```
      ParStat([bicond(testPart,or(negSide,phi))], dTestAx()),
      ParStat([cond(testPart,or(negSide,phi))], bicondL(1)),
      ParStat([cond(cond(negSide,phi),cond(cond(testPart,or(negSide,phi)),cond(testPart,phi)
    ))], condDL()),
      ParStat([cond(cond(testPart,or(negSide,phi)),cond(testPart,phi))], mp(4,1)),
      ParStat([cond(testPart,phi)], mp(3,1)),
      ParStat([cond(\mod(dIter(concat(dTest(negSide),game)),testPart),\mod(dIter(concat(
    dTest(negSide),game)),phi))], mon(1)),
      ParStat([cond(unfold,gConcatPart)], mon(1)),
      ParStat([cond(cond(negSide,ant),cond(negSide,unfold))], condCR()),
      ParStat([cond(negSide,unfold)], mp(8,1)),
      ParStat([cond(negSide,gConcatPart)], cut(1,3)),
      ParStat([cond(cond(negSide,phi),cond(cond(negSide,gConcatPart),cond(negSide,and(phi,
    gConcatPart))))], condCI()),
      ParStat([cond(cond(negSide,gConcatPart),cond(negSide,and(phi,gConcatPart)))], mp(12,1)
    ),
      ParStat([cond(negSide,and(phi,gConcatPart))], mp(3,1)),
      ParStat([bicond(concatPart,and(phi,dblConcatPart))], dIterAx()),
      ParStat([cond(concatPart,and(phi,dblConcatPart))], bicondL(1)),
      ParStat([cond(cond(concatPart,and(phi,dblConcatPart)),cond(concatPart,dblConcatPart))
    ], condCR()),
      ParStat([cond(concatPart,dblConcatPart)], mp(2,1)),
      ParStat([bicond(dblConcatPart,testGameConcat)], concatAx()),
      ParStat([cond(dblConcatPart,testGameConcat)], bicondL(1)),
      ParStat([bicond(testGameConcat,or(negSide,gConcatPart))], dTestAx()),
      ParStat([cond(testGameConcat,or(negSide,gConcatPart))], bicondL(1)),
      ParStat([cond(concatPart,testGameConcat)], cut(5,3)),
      ParStat([cond(concatPart,or(negSide,gConcatPart))], cut(1,2)),
      ParStat([cond(cond(negSide,and(phi,gConcatPart)),cond(or(negSide,gConcatPart),or(and(
    phi,gConcatPart),gConcatPart)))], strong()),
      ParStat([cond(or(negSide,gConcatPart),or(and(phi,gConcatPart),gConcatPart))], mp(12,1)
    ),
      ParStat([cond(concatPart,or(and(phi,gConcatPart),gConcatPart))], cut(3,1)),
      ParStat([cond(or(and(phi,gConcatPart),gConcatPart),gConcatPart)], andOr()),
      ParStat([cond(concatPart,gConcatPart)], cut(3,1)),
      ParStat([cond(concatPart,gxConcatPart)], dBarInd(1)),
      ParStat([cond(gConcatPart,or(negSide,gConcatPart))], weakPL()),
      ParStat([cond(or(negSide,gConcatPart),testGameConcat)], bicondR(10)),
      ParStat([cond(gConcatPart,testGameConcat)], cut(2,1)),
      ParStat([cond(testGameConcat,dblConcatPart)], bicondR(14)),
      ParStat([cond(gConcatPart,dblConcatPart)], cut(2,1)),
      ParStat([cond(negSide,dblConcatPart)], cut(24,1)),
      ParStat([cond(cond(negSide,phi),cond(cond(negSide,dblConcatPart),cond(negSide,and(phi,
    dblConcatPart))))], condCI()),
      ParStat([cond(cond(negSide,dblConcatPart),cond(negSide,and(phi,dblConcatPart)))], mp
    (36,1)),
      ParStat([cond(negSide,and(phi,dblConcatPart))], mp(3,1)),
      ParStat([bicond(concatPart,and(phi,dblConcatPart))], dIterAx()),
      ParStat([cond(and(phi,dblConcatPart),concatPart)], bicondR(1)),
      ParStat([cond(negSide,concatPart)], cut(3,1)),
      ParStat([cond(cond(concatPart,and(phi,dblConcatPart)),cond(concatPart,phi))], condCL()
    ),
      ParStat([cond(concatPart,phi)], mp(27,1)),
      ParStat([cond(negSide,gxConcatPart)], cut(3,15)),
      ParStat([cond(gxConcatPart,\mod(dIter(game),phi))], mon(2)),
      ParStat([cond(negSide,\mod(dIter(game),phi))], cut(2,1))
      ];

  if(side == []) {
    return GtoPar(inf) + proof + ParStatEnd(seq, mp(size(proof),1), inds());
  }
  return GtoPar(inf) + proof + ParStatEnd(seq, condSEL(1), inds());
}
```

Listing 8: “LaTeXOutput.rsc”

```
module LaTeXOutput
/*
```

```
 * Module containing the functions to transform a Par proof to a proof in LaTeX
 */

import GLAST;

import IO;
import List;

// These two functions take a G proof and a Par proof, respectively, and file location, and
    then write the LaTeX output to the file.
void LaTeXOutput(GProof p, loc out){
  writeFile(out, LaTeXOutput(p));
}
void LaTeXOutput(list[ParLine] p, loc out){
  writeFile(out, LaTeXOutput(p));
}

// These two functions construct the LaTeX file settings and packages for a G Proof and a
    Par proof, respectively.
str LaTeXOutput(GProof p) =
  "\\documentclass{article}
  '\\usepackage[utf8]{inputenc}
  '\\usepackage{prooftrees}
  '
  '\\begin{document}
  '
  '\\begin{prooftree}
  '<LaTeXGTree(p)>
  '\\end{prooftree}
  '
  '\\end{document}";
str LaTeXOutput(list[ParLine] p) =
  "\\documentclass{article}
  '
  '% Packages
  '\\usepackage[utf8]{inputenc}
  '\\usepackage[a1paper]{geometry}
  '\\usepackage{amsmath}
  '\\usepackage{booktabs,array}
  '\\usepackage{longtable}
  '\\usepackage{xcolor}
  '
  '% Settings
  '\\newcolumntype{C}{\>{$}c\<{$}}
  '\\newcolumntype{L}{\>{$}l\<{$}}
  '
  '\\begin{document}
  '
  '\\begin{longtable}{C|LL}
  '<LaTeXParProof(p)>
  '\\end{longtable}
  '
  '\\end{document}";

/*
  Main function for constructing the Par proof in LaTeX, calls LaTeXParLine for every line
    in the proof

  input: list of Par lines, a Par proof
  output: a string containing the proof for LaTeX
*/
str LaTeXParProof(list[ParLine] p) {
  str proof = "";
  for(int i <- [0..size(p)]) {
    proof += LaTeXParLine(p[i],i+1);
  }
  return proof;
}
```

```
68
69 /*
70   Function to convert the first Par Line of the transformation of a G rule to a string
71
72   NOTE: Currently, this function does not differ from the regular LaTeXParLine function.
73       The idea is that in the LaTeX proof every transformed G rule can somehow be
       highlighted/recognized.
74
75   input: Par Line that is the start of a G rule transformation
76   output: a line in the LaTeX string with the justification, line number and formula.
77 */
78 str LaTeXParLine(ParStatStart(list[GameLog] disjs, ParRule rule, _), int i) {
79   return "<i>. & <(hasPar(head(disjs)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(
       disjs))> & <LaTeXParRule(rule,i)> <newLine()>";
80 }
81
82 /*
83   Function to convert a Par Line of the transformation of a G rule to a string
84
85   input: Par Line
86   output: a line in the LaTeX string with the justification, line number and formula.
87 */
88 str LaTeXParLine(ParStat(list[GameLog] disjs, ParRule rule), int i) {
89   return "<i>. & <(hasPar(head(disjs)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(
       disjs))> & <LaTeXParRule(rule,i)> <newLine()>";
90 }
91
92 /*
93   Function to convert a Par Line of the transformation of a G rule to a string with a colour
94
95   input: Par Line with a designated colour
96   output: a line in the LaTeX string with the justification, line number and coloured
       formula.
97 */
98 str LaTeXParLine(ParStat(list[GameLog] disjs, ParRule rule, str colour), int i) {
99   return "<i>. & \\color{<colour>}<(hasPar(head(disjs)) | it + " \\lor " + hasPar(g) |
       GameLog g <- tail(disjs))> & <LaTeXParRule(rule,i)> <newLine()>";
100 }
101
102 /*
103   Function to print the last Par Line of the transformation of a G rule
104
105   NOTE: Currently, this function does not differ from the regular LaTeXParLine function.
106       The idea is that in the LaTeX proof every transformed G rule can somehow be
       highlighted/recognized.
107
108   input: Par Line that is the end of a G rule transformation
109   output: a line in the LaTeX string with the justification, line number and formula.
110 */
111 str LaTeXParLine(ParStatEnd(list[GameLog] disjs, ParRule rule, _), int i) {
112   return "<i>. & <(hasPar(head(disjs)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(
       disjs))> & <LaTeXParRule(rule,i)> <newLine()>";
113 }
114
115 // Function to output the Par rule labels without line numbers
116 str LaTeXParRule(concatAx(),_) = "\\text{Ax2 } ; ";
117 str LaTeXParRule(choiceAx(),_) = "\\text{Ax3 } \\sqcup ";
118 str LaTeXParRule(iterAx(),_) = "\\text{Ax4 } * ";
119 str LaTeXParRule(testAx(),_) = "\\text{Ax5 } ? ";
120 str LaTeXParRule(dualAx(),_) = "\\text{Ax6 } dual ";
121 str LaTeXParRule(dChoiceAx(),_) = "\\text{Ax7 } \\sqcap ";
122 str LaTeXParRule(dIterAx(),_) = "\\text{Ax8 } \\times ";
123 str LaTeXParRule(dTestAx(),_) = "\\text{Ax9 } ! ";
124 str LaTeXParRule(ci(),_) = "\\text{PL: } A \\rightarrow (B \\rightarrow (A \\wedge B)) ";
125 str LaTeXParRule(commOr(),_) = "\\text{PL: } A \\lor B \\rightarrow B \\lor A ";
126 str LaTeXParRule(condIL(),_) = "\\text{PL: } A \\lor B \\rightarrow (\\neg A \\rightarrow B)
      ";
127 str LaTeXParRule(condIR(),_) = "\\text{PL: } A \\lor B \\rightarrow (\\neg B \\rightarrow A)
```

```
     ";
str LaTeXParRule(condEL(),_) = "\\text{PL: } (\\neg A \\rightarrow B) \\rightarrow A \\lor B
     ";
str LaTeXParRule(condCL(),_) = "\\text{PL: } (A \\rightarrow B \\wedge C) \\rightarrow (A \\
    rightarrow B) ";
str LaTeXParRule(condCR(),_) = "\\text{PL: } (A \\rightarrow B \\wedge C) \\rightarrow (A \\
    rightarrow C) ";
str LaTeXParRule(condCI(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow ((A \\
    rightarrow C) \\rightarrow (A \\rightarrow B \\wedge C)) ";
str LaTeXParRule(bicondWR(),_) = "\\text{PL: } (A \\leftrightarrow B) \\rightarrow (B \\lor
    C \\rightarrow A \\lor C) ";
str LaTeXParRule(exMidPL(),_) = "\\text{PL: } A \\lor \\neg A ";
str LaTeXParRule(weakPL(),_) = "\\text{PL: } A \\rightarrow A \\lor B ";
str LaTeXParRule(andPL(),_) = "\\text{PL: } A \\rightarrow (B \\rightarrow A \\wedge B) ";
str LaTeXParRule(condDIL(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (A \\
    rightarrow B \\lor C) ";
str LaTeXParRule(condDIR(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (A \\
    rightarrow C \\lor B) ";
str LaTeXParRule(condDER(),_) = "\\text{PL: } (A \\lor B \\rightarrow C) \\rightarrow (B \\
    rightarrow C) ";
str LaTeXParRule(condDEL(),_) = "\\text{PL: } (A \\lor B \\rightarrow C) \\rightarrow (A \\
    rightarrow C) ";
str LaTeXParRule(mongdChoice(),_) = "\\text{PL: } ((A \\rightarrow B \\lor D) \\wedge (C \\
    rightarrow B \\lor D)) \\rightarrow (A \\lor C \\rightarrow B \\lor D) ";
str LaTeXParRule(weakLPL(),_) = "\\text{PL: } A \\rightarrow B \\lor A ";
str LaTeXParRule(condDIDR(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (A \\lor C \\
    rightarrow B \\lor C) ";
str LaTeXParRule(condDIDL(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (C \\lor A \\
    rightarrow C \\lor B) ";
str LaTeXParRule(condCIR(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (A \\wedge C
    \\rightarrow B \\wedge C) ";
str LaTeXParRule(condCIL(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow (C \\wedge A
    \\rightarrow C \\wedge B) ";
str LaTeXParRule(orPL(),_) = "\\text{PL: } (A \\lor B \\lor C) \\rightarrow (A \\lor (B \\
    lor C))";
str LaTeXParRule(mpAndL(),_) = "\\text{PL: } ((A \\wedge B) \\wedge (A \\rightarrow C)) \\
    rightarrow (C \\wedge B) ";
str LaTeXParRule(mpAndR(),_) = "\\text{PL: } ((A \\wedge B) \\wedge (B \\rightarrow C)) \\
    rightarrow (A \\wedge C) ";
str LaTeXParRule(mpOrL(),_) = "\\text{PL: } ((A \\lor B) \\wedge (A \\rightarrow C)) \\
    rightarrow (C \\lor B) ";
str LaTeXParRule(mpOrR(),_) = "\\text{PL: } ((A \\lor B) \\wedge (B \\rightarrow C)) \\
    rightarrow (A \\lor C) ";
str LaTeXParRule(mpCondL(),_) = "\\text{PL: } ((A \\rightarrow B) \\wedge (A \\rightarrow C)
    ) \\rightarrow (B \\rightarrow C) ";
str LaTeXParRule(mpCondR(),_) = "\\text{PL: } ((A \\rightarrow B) \\wedge (C \\rightarrow A)
    ) \\rightarrow (C \\rightarrow B) ";
str LaTeXParRule(mpBicondL(),_) = "\\text{PL: } ((A \\rightarrow B) \\wedge (A \\
    leftrightarrow C)) \\rightarrow (B \\leftrightarrow C) ";
str LaTeXParRule(mpBicondR(),_) = "\\text{PL: } ((A \\rightarrow B) \\wedge (C \\
    leftrightarrow A)) \\rightarrow (C \\leftrightarrow B) ";
str LaTeXParRule(monAB(),_) = "\\text{PL: } A \\rightarrow ((A \\rightarrow B) \\rightarrow
    B) ";
str LaTeXParRule(condDL(),_) = "\\text{PL: } (A \\rightarrow B) \\rightarrow ((C \\
    rightarrow A \\lor B) \\rightarrow (C \\rightarrow B)))";
str LaTeXParRule(strong(),_) = "\\text{PL: } (A \\rightarrow B \\wedge C) \\rightarrow ((A
    \\lor C) \\rightarrow ((B \\wedge C) \\lor C))";
str LaTeXParRule(andOr(),_) = "\\text{PL: } ((A \\wedge B) \\lor B) \\rightarrow B";

// Function to output the Par rule labels with line numbers
str LaTeXParRule(bicondL(ref),line) = "\\text{BE$_L$: } <getLine(line,ref)> ";
str LaTeXParRule(bicondR(ref),line) = "\\text{BE$_R$: } <getLine(line,ref)> ";
str LaTeXParRule(cut(l,r),line) = "\\text{PL$_{CUT}$: } <getLine(line,l)>,<getLine(line,r)>
    ";
str LaTeXParRule(mp(var,cond),line) = "\\text{MP: } <getLine(line,var)>,<getLine(line,cond)>
     ";
str LaTeXParRule(mon(ref),line) = ref > 0 ? "\\text{Mon: } <getLine(line,ref)> " : "\\text{
    Mon} ";
```

```
str LaTeXParRule(barInd(ref),line) = ref > 0 ? "\\text{BarInd: } <getLine(line,ref)> " : "\\text{BarInd} ";
str LaTeXParRule(dBarInd(ref),line) = ref > 0 ? "\\text{BarInd$^{\\times}$: } <getLine(line,ref)> " : "\\text{BarInd}^{\\times} ";
str LaTeXParRule(condSIL(ref),line) = "\\text{PL$_{D;C}$: } <getLine(line,ref)> ";
str LaTeXParRule(condSEL(ref),line) = "\\text{PL$_{C;D}$: } <getLine(line,ref)> ";

// Function that returns the justification reference
int getLine(int val, int sub) = val-sub;

// Function that returns a LaTeX newline
str newLine() = " \\\\";

// Function to output the game logic formulae in LaTeX maths mode
str LaTeXGameLog(atomP(Prop p)) = "<LaTeXGameLog(p)>";
str LaTeXGameLog(neg(GameLog pr)) = "\\neg <hasPar(pr)>";
str LaTeXGameLog(\mod(Game g, GameLog pr)) = "\\langle <hasPar(g)>\\rangle <hasPar(pr)>";
str LaTeXGameLog(and(GameLog pL, GameLog pR)) = "<hasPar(pL)>\\wedge <hasPar(pR)>";
str LaTeXGameLog(or(GameLog pL, GameLog pR)) = "<hasPar(pL)>\\lor <hasPar(pR)>";
str LaTeXGameLog(cond(GameLog pL, GameLog pR)) = "<hasPar(pL)>\\rightarrow <hasPar(pR)>";
str LaTeXGameLog(bicond(GameLog pL, GameLog pR)) = "<hasPar(pL)>\\leftrightarrow <hasPar(pR)>";
str LaTeXGameLog(condS(list[GameLog] pL, GameLog pR)) =
  "(<(hasPar(head(pL)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(pL))>) \\rightarrow <hasPar(pR)>";
str LaTeXGameLog(orS(list[GameLog] pL, GameLog pR)) =
  "<(hasPar(head(pL)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(pL))> \\lor <hasPar(pR)>";
str LaTeXGameLog(negS(list[GameLog] pL)) =
  "\\neg(<(hasPar(head(pL)) | it + " \\lor " + hasPar(g) | GameLog g <- tail(pL))>)";

// Function to output the game formulae in LaTeX maths mode
str LaTeXGameLog(atomG(AGame g)) = "<LaTeXGameLog(g)>";
str LaTeXGameLog(dual(Game ga)) = "{<hasPar(ga)>}^d";
str LaTeXGameLog(\test(GameLog g)) = "<hasPar(g)>?";
str LaTeXGameLog(dTest(GameLog g)) = "<hasPar(g)>!";
str LaTeXGameLog(iter(Game ga)) = "{<hasPar(ga)>}^{*}";
str LaTeXGameLog(dIter(Game ga)) = "{<hasPar(ga)>}^{\\times}";
str LaTeXGameLog(concat(Game gL, Game gR)) = "<hasPar(gL)>;<hasPar(gR)>";
str LaTeXGameLog(choice(Game gL, Game gR)) = "<hasPar(gL)>\\sqcup <hasPar(gR)>";
str LaTeXGameLog(dChoice(Game gL, Game gR)) = "<hasPar(gL)>\\sqcap <hasPar(gR)>";

// Function to put parentheses around only the binary connectives and not unary connectives or atomic formulae
str hasPar(atomP(Prop p)) = "<LaTeXGameLog(atomP(p))>";
str hasPar(neg(GameLog pr)) = "<LaTeXGameLog(neg(pr))>";
str hasPar(\mod(Game g, GameLog pr)) = "<LaTeXGameLog(\mod(g,pr))>";
str hasPar(and(GameLog pL, GameLog pR)) = "(<LaTeXGameLog(and(pL,pR))>)";
str hasPar(or(GameLog pL, GameLog pR)) = "(<LaTeXGameLog(or(pL,pR))>)";
str hasPar(cond(GameLog pL, GameLog pR)) = "(<LaTeXGameLog(cond(pL,pR))>)";
str hasPar(bicond(GameLog pL, GameLog pR)) = "(<LaTeXGameLog(bicond(pL,pR))>)";
str hasPar(atomG(AGame g)) = "<LaTeXGameLog(atomG(g))>";
str hasPar(dual(Game ga)) = "<LaTeXGameLog(dual(ga))>";
str hasPar(\test(GameLog g)) = "<LaTeXGameLog(\test(g))>";
str hasPar(dTest(GameLog g)) = "<LaTeXGameLog(dTest(g))>";
str hasPar(iter(Game ga)) = "<LaTeXGameLog(iter(ga))>";
str hasPar(dIter(Game ga)) = "<LaTeXGameLog(dIter(ga))>";
str hasPar(concat(Game gL, Game gR)) = "(<LaTeXGameLog(concat(gL,gR))>)";
str hasPar(choice(Game gL, Game gR)) = "(<LaTeXGameLog(choice(gL,gR))>)";
str hasPar(dChoice(Game gL, Game gR)) = "(<LaTeXGameLog(dChoice(gL,gR))>)";
str hasPar(condS(list[GameLog] pLs, GameLog pR)) = "<LaTeXGameLog(condS(pLs,pR))>";
str hasPar(orS(list[GameLog] pLs, GameLog pR)) = "(<LaTeXGameLog(orS(pLs,pR))>)";
str hasPar(negS(list[GameLog] pLs)) = "<LaTeXGameLog(negS(pLs))>";

// Functions to output the atomic propositions and games which can have a subscript
str LaTeXGameLog(agame(str n)) = "<n>";
str LaTeXGameLog(agameS(str n, int sub)) = "<n>_{<sub>}";
str LaTeXGameLog(prop(str n)) = n == "x" ? "\\chi" : "<n>";
```

```
str LaTeXGameLog(propS(str n, int sub)) = "<n>_{<sub>}";

// Function to output the LaTeX proof tree part for each G sequent and assocated rule label
str LaTeXGTree(GLeaf())
  = "\\AxiomC{}";
str LaTeXGTree(GUnaryInf(list[GameLog] seq, GRule rule, GProof inf)) =
  "<LaTeXGTree(inf)>
  '\\RightLabel{<LaTeXGTree(rule)>}
  '\\UnaryInfC{$<(LaTeXGameLog(head(seq)) | it + ", " + LaTeXGameLog(g) | GameLog g <- tail(
    seq))>$}";
str LaTeXGTree(GBinaryInf(list[GameLog] seq, GProof infL, GProof infR)) =
  "<LaTeXGTree(infL)>
  '<LaTeXGTree(infR)>
  '\\RightLabel{$\\wedge$}
  '\\BinaryInfC{$<(LaTeXGameLog(head(seq)) | it + ", " + LaTeXGameLog(g) | GameLog g <- tail
    (seq))>$}";

// Function to output the G rule labels
str LaTeXGTree(ax()) = "Ax";
str LaTeXGTree(modm()) = "mod$_{m}$";
str LaTeXGTree(and()) = "$\\wedge$";
str LaTeXGTree(or()) =  "$\\vee$";
str LaTeXGTree(choice()) =  "$\\sqcup$";
str LaTeXGTree(dChoice()) =  "$\\sqcap$";
str LaTeXGTree(weak()) =  "weak";
str LaTeXGTree(concatD()) =  "$;_d$";
str LaTeXGTree(iter()) =  "$*$";
str LaTeXGTree(\test()) =  "$?$";
str LaTeXGTree(dIter()) =  "$\\times$";
str LaTeXGTree(dTest()) =  "$!$";
str LaTeXGTree(mongd()) =  "mon$^g_d$";
str LaTeXGTree(monfd()) =  "mon$^f_d$";
str LaTeXGTree(inds()) =  "ind$_s$";
```