FORUM AND ITS IMPLEMENTATION

Christian Urban

A Thesis Submitted for the Degree of MPhil
at the
University of St Andrews

1997

FORUM and its Implementation

Christian Urban

March 13, 1997

Presented to the University of St Andrews in partial fulfilment of the requirements for the degree of Master of Philosophy.

Supervisor: Dr Roy Dyckhoff (St Andrews)

Examiners: Prof. Ursula Martin (St Andrews)
Dr David Pym (London)

Th C228

Abstract

Miller presented Forum as a specification logic: Forum extends several existing logic programming languages, for example λProlog, LO and Lolli. The crucial change in Forum is the extension from single succedent sequents, as in intuitionistic logic, to multiple succedent sequents, as in classical logic, with a corresponding extension of the notion of uniform proof.

Forum uses the connectives of linear logic. Languages based on linear logic offer extra expressivity (in comparison with traditional logic languages), but also present new implementation challenges. One such challenge is that of context management, because the multiplicative linear connectives '$\otimes$', '$⅋$' and '$\multimap$' require context splitting. Hodas and Miller presented a solution (the IO model) to this in 1991 for the language Lolli based on minimal linear logic. This thesis presents a technique which is an adaptation of the aforementioned approach for the language Forum and following a suggestion of Miller that the '?' constant be treated as primitive in order to avoid looping problems arising from its use as a derived symbol. Cervesato, Hodas and Pfenning have presented a technique for managing the '$\top$' constant, dividing each input context into a "slack" part and a "strict" part; the main novel contribution of this thesis is to modify this technique, by dividing instead the output context. This leads to a proof system with fewer rules (and consequent ease of implementation) but enhanced performance, for which we present some experimental evidence.

Declaration

I, Christian Urban, hereby certify that this thesis, which is approximately 26000 words in length, has been written by me, that it is the record of work carried out by me, and that it has not been submitted in any previous application for a higher degree.

date 13th March 1997 *signature of candidate* _

I was admitted as a research student in September 1995 and as a candidate for the degree of Master of Philosophy in September 1995; the higher study of which this is a record was carried out in the University of St Andrews between 1995 and 1996.

date 13th March 1997 *signature of candidate* _

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Master of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

date 18th March 1997 *signature of supervisor* _

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker.

date 13th March 1997 *signature of candidate* _

Acknowledgements

First, and above all, I must thank Dr Roy Dyckhoff, my supervisor, who brought this project on its way; without his guidance and help, this thesis would never have been written. I greatly benefited from his insight and introduction to various researchers, amongst them Prof. Philip Scott, Dr David Pym, Dr Gavin Bierman and Dr Martin Hyland with whom I have had many interesting discussions. I am happy to thank Dr Dyckhoff for his immense help in improving my English and my style of writing technical texts. Additionally, the delightful personal contacts I had with him and his family provide many pleasant memories.

I am happy to thank Prof. Steffen Hölldobler, Prof. Werner Daniel and Dr Barbara Daniel, who, in diverse ways, taught me that science is fun and that research is a lifelong fascinating activity.

I appreciate the discussion with Dr Iliano Cervesato; I am happy to thank him for his valuable comments on an earlier version of the box calculus. I thank Prof. Joshua Hodas who made unpublished papers on FORUM and Lolli available for me.

In May 1996, I was invited by Prof. Dale Miller to spend a month at the University of Pennsylvania. I learnt much from this visit and the thesis owes many ideas to Prof. Miller. I am happy to thank him for his kind invitation and the discussions with him on Forum.

Prof. Gopalan Nadathur and Philip Wickline gave me support that allowed me to implement the calculi of this thesis in the logic programming language Terzo.

I thank Jacob Howe and Tom Kelsey for very carefully proofreading earlier versions of this thesis.

Special thanks to Alison Craig Frantz, Mai Spurlock, Melissa McPherson, Mark Killian Brewer, Chris King, Gabriele De Anna, and John Serrati (my friends in Deans Court) for their tremendous help in improving my English.

Finally, I thank my parents and my brother who gave me endless support—thank you for all they have done for me is not even close to adequate.

Funding: I received financial support from the Chamber of Commerce in Germany, from the University of Dresden, from the University of St Andrews and from the EU ESPRIT Basic Research Action group 7232 "Gentzen".

Contents

1 Introduction **5**

1.1 Background on Linear Logic . . . 5

1.2 Motivation . . . 8

1.3 Terminology and Notation . . . 10

1.4 Outline and Results . . . 10

2 Foundations of Logic Programming and FORUM **12**

2.1 Single Succedent Logics and Uniform Proofs . . . 12

2.1.1 Horn Logic and its Formulae . . . 15

2.1.2 First-Order Hereditary Harrop Formulae . . . 15

2.1.3 Lolli and its Formulae . . . 16

2.2 Extension of Uniform Proofs for FORUM . . . 16

2.3 Miller's Proof System $\mathcal{F}$ for FORUM . . . 19

2.3.1 The Use of Contexts in Sequents . . . 21

2.3.2 Behaviour of the Non-Primitive Connectives . . . 23

2.4 Modified Proof System $\mathcal{F}'$ Including an Atomic-Rule . . . 24

2.5 An Operational Reading of the Rules in $\mathcal{F}'$ . . . 25

2.6 The Approach in Lygon . . . 31

2.7 Types and Terms in FORUM . . . 32

2.7.1 Kinds and Types . . . 33

2.7.2 Simply Typed λ-Terms . . . 35

2.7.3 Definition of FORUM's Syntax . . . 36

3 Efficient Context Management **38**

3.1 Input-Output Model and the Box Calculus $\mathcal{B}$ . . . 38

3.1.1 Soundness . . . 41

3.1.2 Completeness . . . 42

3.1.3 Implementation . . . 44

3.2 A New Box Calculus $\mathcal{B}'$ . . . 51

3.2.1 Soundness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$ . . . 52

3.2.2 Completeness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$ . . . 57

3.2.3 Implementation . . . 64
3.3 Consideration of the Choose Rules . . . 71
3.4 Problems with Empty Headed Implications . . . 72

4 Examples **74**
4.1 A Simple Object Logic . . . 74
4.2 A Conjunctive Planner in FORUM . . . 80
4.3 A Program for Computing the Fibonacci Series . . . 82
4.4 A Program for Finding Paths in Cyclic Graphs . . . 84

5 Related Work **85**
5.1 Context Management in Lolli . . . 85
5.2 Hodas' and Polakow's Approach Towards a FORUM Implementation . . . 87
5.3 Context Management in Lygon . . . 88

6 Conclusion, Open Problems and Further Work **90**
6.1 Conclusion . . . 90
6.2 Open Problems . . . 90
6.3 Further Work . . . 91

A Proofs **96**
A.1 Soundness of $\mathcal{B}$ w.r.t. $\mathcal{F}$' . . . 96
A.2 Modification of a $\mathcal{B}$-Proof . . . 101
A.3 Completeness of $\mathcal{B}$ w.r.t. $\mathcal{F}'$ . . . 106
A.4 Soundness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$ . . . 110
A.5 Completeness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$ . . . 113

B Source Code **117**
B.1 Module aux.mod . . . 117
B.2 Module forumlog.mod . . . 119
B.3 Module choosetest.mod . . . 120
B.4 Module forum.mod (Calculus $\mathcal{F}'$) . . . 121
B.5 Module forum.mod (Box Calculus $\mathcal{B}$) . . . 124
B.6 Module forum.mod (Box Calculus $\mathcal{B}'$) . . . 128

Chapter 1

Introduction

1.1 Background on Linear Logic

An important problem of designing programming languages is to provide techniques for the programmers to reason about the computation in their programs. Forum is presented as a specification logic in [Miller, 1994], [Miller, 1995] and [Miller, 1996]. The use of specification logics is to tell programmers precisely what they implement and also enables them to reason about computation (e.g., in state and transition systems). Two detailed Forum specifications can be found in [Chirimar, 1995]: in this work, Forum is employed to model computations of programs and to encode states and transitions using formulae and proofs. For this purpose Forum is rather convenient because it is based on linear logic which offers extra expressivity (in comparison with traditional logics)[1], and it includes a natural extension of first-order logic to higher-order logic, as used in λProlog.

The present thesis is an approach towards an implementation of Forum. The specification language Forum incorporates a certain amount of non-determinism, and therefore, for its implementation, the set of proofs is restricted in such a way that the construction of proofs can be carried out efficiently by a computer. Hence, our Forum implementation can express only a fragment of the problems that one could specify using it as a specification logic owing to the restrictions of the underlying proof search strategy. In order to make this difference explicit from now on, the term FORUM is used rather than Miller's term Forum.

FORUM is based on linear logic; this will be introduced in the remainder of this section. However, a detailed treatment of linear logic is beyond the scope of the thesis and the reader is referred to [Girard, 1987], [Lincoln, 1992] or [Scedrov, 1993]. The latter includes an extensive bibliography on the topic.

One of the main contributions of [Girard, 1987] is the presentation of linear logic as a sequent-style calculus. Roughly speaking, linear logic is a logic that analyses resources and resource use. For expository purposes, it is convenient to use a sequent calculus for classical logic (CL), such as the LK^{cut} calculus, as a point of the conceptual departure. (LK is regarded as a cut-free calculus, and LK^{cut} combines LK and the 'Cut' rule.) A three-step modification of LK^{cut} leads to the sequent calculus for classical linear logic (CLL).

Firstly, the structural rules 'Contraction-L/-R' and 'Weakening-L/-R' are banned from

[1] In this context, expressivity means the ability to support useful programming constructs rather than the expressivity in terms of, for example, Turing-completeness.

LK^{cut} since these rules are responsible for the loss of resource control in CL. The rules are as follows:

$$\frac{\Delta \Longrightarrow \Gamma}{\Delta, B \Longrightarrow \Gamma}\ \text{Weakening-L} \qquad \frac{\Delta, B, B \Longrightarrow \Gamma}{\Delta, B \Longrightarrow \Gamma}\ \text{Contraction-L}$$

$$\frac{\Delta \Longrightarrow \Gamma}{\Delta \Longrightarrow \Gamma, B}\ \text{Weakening-R} \qquad \frac{\Delta \Longrightarrow \Gamma, B, B}{\Delta \Longrightarrow \Gamma, B}\ \text{Contraction-R}$$

The *contexts* Δ and Γ are regarded as multisets since this permits the suppression of the structural rules 'Exchange-L/-R' in the sequent calculus. The 'Weakening-L/-R' rules allow one to "throw away" some formulae, and the duplication of formulae is possible using the 'Contraction-L/-R' rules. Without these four structural rules and an identity axiom formulated as follows:

$$\frac{}{B \Longrightarrow B}\ \text{Identity,}$$

Girard obtained a *linear system* where it is necessary to use each assumption exactly once.

Secondly, the lack of 'Contraction-L/-R' and 'Weakening-L/-R' causes a distinction between two forms of conjunction; similarly for disjunction and implication. In CL, the two formulations, for example, of the conjunction rules:

$$\frac{\Delta \Longrightarrow B, \Gamma \quad \Delta \Longrightarrow C, \Gamma}{\Delta \Longrightarrow B \wedge C, \Gamma}\ \wedge\text{-R}_a \qquad \frac{\Delta_1 \Longrightarrow B, \Gamma_1 \quad \Delta_2 \Longrightarrow C, \Gamma_2}{\Delta_1, \Delta_2 \Longrightarrow B \wedge C, \Gamma_1, \Gamma_2}\ \wedge\text{-R}_m$$

are equivalent, since it is possible to simulate the '$\wedge$-R$_a$' rule by liberal application of the 'Contraction-L/-R' rules and a subsequent use of the '$\wedge$-R$_m$' rule as shown in the following proof fragment:

$$\cfrac{\cfrac{\cfrac{\Delta \Longrightarrow B, \Gamma \quad \Delta \Longrightarrow C, \Gamma}{\Delta, \Delta \Longrightarrow B \wedge C, \Gamma, \Gamma}\ \wedge\text{-R}_m}{\vdots}\ \text{Contraction-L}}{\Delta \Longrightarrow B \wedge C, \Gamma}\ \text{Contraction-R}$$

The *vice-versa* case can be achieved by using the 'Weakening-L/-R' rules and the '$\wedge$-R$_a$' rule. However in CLL, the '$\wedge$-R$_a$' rule, the '$\wedge$-R$_m$' rule and the like become distinct, and accordingly, the corresponding connectives of CL are divided into the *multiplicative* and into the *additive* connectives. Observe that none of them is quite the same as the corresponding connective in CL. The contexts Δ and Γ contain all passive formulae of a rule. The rules of multiplicative and additive connectives treat these contexts in a different manner. The multiplicative connectives forbid a sharing of resources in a rule that branches the proof. The contexts in the *conclusion* (see Section 1.3) are split into two parts. Each *premise* receives a different portion from these contexts in the conclusion. In contrast, an additive rule that branches the proof tree requires a complete sharing of the contexts in the premises. That means both premises have the same context. Nevertheless, the quantifiers and the corresponding inference rules of CLL correspond to those in CL.

Thirdly, two new operators are introduced into the linear system, called *exponentials*. In effect, these operators provide a "controlled" contraction and weakening, i.e., the use of 'Contraction-L/-R' and 'Weakening-L/-R' is restricted to formulae decorated with an exponential. Along with the introduction of the exponentials come natural mappings of all CL's formulae into formulae in CLL (see for instance [Troelstra, 1992]); there are similar mappings for intuitionistic logic (IL) in [Girard, 1987], [Bierman, 1994] and [Negri, 1995]. This observation has led to the remark "Linear logic is a logic behind logics" [Lincoln, 1992].

In short, classical linear logic is based on the following connectives and logical constants:

multiplicative connectives and constants:	$\multimap$, $\otimes$, $⅋$, $\mathbf{1}$, $\bot$;
additive connectives and constants:	$\leadsto$, $\&$, $\oplus$, $\top$, $\mathbf{0}$;
exponentials:	$!$, $?$;
quantifiers:	$\exists, \forall$.

At this point, the linear negation is left out, but will be introduced later. The multiplicative connectives are called 'linear implication' or 'lollipop', 'times' and 'par'. The corresponding logical constants are 'one' and 'bottom'. The additive connectives are 'additive implication'[2], 'with' and 'plus'. The corresponding logical constants are 'top' and 'zero'. The exponentials are named 'bang' and 'query'. As in CL, each quantifier is either an 'existential quantifier' or a 'universal quantifier'.

Some differences between CL's and CLL's connectives are illustrated by an example taken from [Perrier, 1995]. Accordingly, the fact that an identifier x has the value 5 is encoded as the proposition $(x = 5)$. The change of state is modelled by the intuitionistic implication. However, an attempt to reassign a new value to the identifier x, i.e. a change of state, fails in CL because the following sequent is provable:

$$(x = 2)\ ,\ (x = 2) \supset (x = 5) \implies (x = 2) \wedge (x = 5)$$

On the other hand, it is possible to encode the reassignment in CLL using the '$\multimap$' connective, and thus represent the notion of state. In CLL, the following sequent is provable:

$$(x = 2)\ ,\ (x = 2)\multimap(x = 5) \implies (x = 5),$$

$$\begin{array}{rl} \text{but neither} & (x = 2)\ ,\ (x = 2)\multimap(x = 5) \implies (x = 2) \otimes (x = 5) \\ \text{nor} & (x = 2)\ ,\ (x = 2)\multimap(x = 5) \implies (x = 2)\,\&\,(x = 5) \end{array}$$

is provable. This is what is meant by the aforementioned extra expressivity of linear logic. The notions of state, action and change as described above fit naturally into a linear logic setting.

In [Lincoln et al., 1992], it is shown that propositional linear logic is undecidable. By contrast, the propositional fragment of CL is decidable. The undecidability seems to be a discouraging result, but on the other hand, it is a symptom of the expressiveness of linear logic which is illustrated by some examples below.

An application which fits nicely into the propositional fragment of linear logic is the encoding of *Petri nets* into formulae and proofs (for a survey see [Peterson, 1981], but our presentation follows [Cervesato, 1995]). Petri nets consist of two kinds of static component (places and transitions) and one kind of dynamic component (tokens). A Petri net is generally presented as a directed graph, but here a more abstract presentation is used. It is

[2]The '$\leadsto$' connective is often disregarded since it lacks some properties of an implication, e.g. $A \leadsto A$ is not provable. That is why it has no specific name.

assumed that places are stores for possibly infinitely many tokens. The state, when a certain place labelled with a unique name p stores three tokens, is represented as the multiset $\{\!|p, p, p|\!\}$. The transitions, on the other hand, are the connection links between the states. Each transition has a *premultiset* and a *postmultiset*: each of them is a collection of place names. A transition is *enabled* if it satisfies the condition that for each occurrence of a place in the premultiset there exists a token on that place. In the following example, the transition t is enabled:

$$t\text{'s premultiset:} \quad \{\!|p, p, q|\!\}$$

$$t\text{'s postmultiset:} \quad \{\!|q|\!\}$$

in any state with at least two tokens are on p and at least one is on q. Each of these states is represented as $\{\!|p, p, q, \ldots|\!\}$. (The dots stand for an arbitrary but fixed multiset of tokens stored in the Petri net.) When an enabled transition *fires*, tokens are removed from places according to the premultiset and added to places according to the postmultiset. This can be expressed in the example above by rewriting the multiset $\{\!|p, p, q, \ldots|\!\}$ to $\{\!|q, \ldots|\!\}$ where the other tokens in it are unchanged.

In linear logic, the tokens on a place are encoded as atomic formulae. The transitions are encoded as linear implications; the transition t from above is encoded as follows:

$$!(p \otimes p \otimes q \multimap q)$$

where the implicans corresponds to the premultiset and the implicatum corresponds to the postmultiset. This implication is decorated with a bang because each time a transition is enabled the transition should be able to fire (i.e., it is a reusable resource). The elements of the pre- and postmultisets are connected by '$\otimes$' in the encoding. An empty multiset is represented by the constant '**1**'.

A Petri net is a simple system which appears to be very inexpressive. However, Petri nets are a powerful and flexible tool for describing concurrent processes. The permitted states of a concurrent system can be represented as the states of the Petri net that are reachable. ('Reachable' here means that a multiset of tokens which represents an initial state can by rewritten by firing transitions to a multiset of tokens which represent one of the aforementioned permitted states.) In terms of the encoding in linear logic, the reachability problem is represented as the following sequent:

$$\mathit{transition}_1, \ldots, \mathit{transition}_k, \mathit{token}_{s_1}, \ldots, \mathit{token}_{s_n} \Longrightarrow \mathit{token}_{e_1} \otimes \ldots \otimes \mathit{token}_{e_m}.$$

The tokens s_i $(1 \leq i \leq n)$ encode the initial state and the tokens e_j $(1 \leq j \leq m)$ represent the final state. The above sequent is provable if and only if there are transitions that transform the token multiset $\{\!|s_1, \ldots, s_n|\!\}$ into the token multiset $\{\!|e_1, \ldots, e_m|\!\}$ (i.e., $\{\!|e_1, \ldots, e_m|\!\}$ is a reachable state). These observations are well-studied in [Martí-Oliet & Meseguer, 1989], [Brown, 1990] and [Cervesato, 1995].

1.2 Motivation

Forum combines sequents with multiple succedents and an approach for a corresponding notion of uniform proof. Along with the multiple succedents come a logic language which is rather expressive. In effect, FORUM is sound and complete with respect to CLL. That means that sequents of CLL are provable if and only if the corresponding sequent is provable in FORUM[3]. This observation is not obvious because the proof search in FORUM is rather restricted. However, Miller has shown the difficult part (the completeness) via a one-to-one conversion of FORUM proofs into a proof system developed by [Andreoli, 1992]. The proof can be found in [Miller, 1996]. Nevertheless, Miller did not introduce FORUM as a logic programming language. He called FORUM a "multiple conclusion specification logic". There are still unsolved problems (e.g., $\bot$-headed implications occurring in the classical context of the antecedent) preventing its complete implementation as a logic programming language.

The main problem, the context splitting during proof search, is solved for other linear languages (it does not occur in traditional languages based on fragments of CL). The multiplicative rules of FORUM present an implementation challenge because the context splitting is a rather costly operation. The complication is best exemplified by the multiplicative conjunction rule, '$\otimes$-R' (although, this formulation of the rule is not present in FORUM):

$$\frac{\Delta_1 \Longrightarrow B, \Gamma_1 \quad \Delta_2 \Longrightarrow C, \Gamma_2}{\Delta_1, \Delta_2 \Longrightarrow B \otimes C, \Gamma_1, \Gamma_2} \otimes\text{-R}$$

Suppose that there are n formulae in a linear context. So, there are 2^n different ways to split this context. A naïve implementation of the splitting operation as a "don't know" choice where all possible splits are "explored" until one is found with the desired properties is too inefficient for a non-trivial proof search. In [Hodas & Miller, 1991], the first work concerning this inefficiency was presented for Lolli.

This thesis attempts to provide a basis from which further problems concerning FORUM can be investigated. Therefore, the main emphasis of this thesis is an expressive prototype implementation which can be easily modified.

The calculus of FORUM has been used already in the literature. Amongst the examples of FORUM programs, Cervesato encoded Petri nets in several programming languages based on linear logic (see [Cervesato, 1995]). He observed that FORUM with multiple succedents is the most appropriate calculus (of those he studied) to embed Petri nets. The convenience of FORUM in the field of concurrency can also be illustrated by a translation of Milner's π-*calculus*, a powerful tool to describe concurrency. The multiple succedents in FORUM provide a mechanism to express concurrency on the level of sequents. As a result, in [Miller, 1992] a preliminary report is given which encodes a fragment of Milner's π-calculus as a FORUM theory. In the disjunctive translation, the non-deterministic choice operator '+' is translated into a '$\oplus$' and the parallel constructor '|' into a '$⅋$'. The conjunctive translation is completely dual and consists of a translation using '&' and '$\otimes$', respectively.

Other examples are approaches towards the design of object-oriented programming languages in FORUM. In [Delzanno & Martelli, 1995], progress was made towards a calculus of an object-oriented programming language in a sublanguage of FORUM, called 'Forum and Objects' ($F\&O$). In this approach, objects are represented as sets of atoms including an additional unique identifier for each object and the methods are encoded as program formulae. The classes are templates to create an object; the encapsulation of data and methods is achieved by universal quantification as in λProlog for data abstraction. This calculus is an interesting proposal towards an integration of object-orientation and logic programming.

[3] A more detailed explanation including the non-primitive connectives can be found in Section 2.3.2.

1.3 Terminology and Notation

In the present thesis, we often refer to the various parts of sequents, formulae and inference rules. A precise terminology is introduced in order to distinguish between them. The structures are usually written as the following:

$$\frac{\text{premises}}{\text{conclusion}} \quad \text{antecedent} \Longrightarrow \text{succedent}$$

$$\text{implicans} \supset \text{implicatum}$$

An inference rule consists of some *premises* and a *conclusion*, a sequent consists of an *antecedent* and a *succedent* and an implication consists of an *implicans* and an *implicatum*.

The thesis mainly follows the notation generally in use. However, there are three minor differences from the terminology introduced by Miller. FORUM was introduced as 'multiple conclusion logic'. To avoid clashes with our notation for inference rules and implications, it is called a 'multiple succedent logic'. The 'decide' rules are called 'choose' rules in the new calculi of this thesis. The motivation for calling these rules 'choose' rules rather than 'decide' rules is that the term "decide" suggests a deterministic behaviour rather than a "don't know" action. The terminology of the 'left' rules in Miller's calculus is changed into *stoup* rules, following [Girard, 1991].

The variable A stands for atomic formulae; the variables B and C stand for arbitrary formulae. The term '*higher-order logic*' is used for a framework which uses λ-terms and permits quantification over some predicates and functions.

1.4 Outline and Results

The thesis is structured as follows:

- Chapter 2 considers the notion of 'uniform proof' for single succedent logics. Three logics are briefly described for which single succedent uniform proofs are complete. Subsequently, the extension of the notion of 'uniform proof' for the multiple succedent logic FORUM is introduced. Miller's work is described in some detail, including an examination of FORUM's non-primitive connectives. Miller's calculus $\mathcal{F}$ is modified in Section 2.4 for more convenience in the following soundness and completeness proofs. Subsequently, we describe the operational reading of the modified sequent calculus $\mathcal{F}'$. Miller's approach towards multiple succedent uniform proofs is compared and contrasted with the principles of Lygon's proof search strategy. Finally, we consider the terms and types of FORUM and introduce a concrete syntax.

- Chapter 3 presents two "box calculi". The first, $\mathcal{B}$, removes the non-determinism of the context splitting in the '$⅋$-S' and '$\multimap$-S' rule. The second box calculus, $\mathcal{B}'$, is introduced in order to reduce the non-determinism of the context splitting in the '$\top$-R' rule. The soundness and completeness proofs are given. The implementations of the box calculi are illustrated by a partial examination of the boxes and the corresponding Terzo code (Terzo is a derivative of the logic programming language λProlog). Finally, the non-determinism in the 'choose' rules is addressed and an implementation of the rules is described.

- Chapter 4 considers two examples of FORUM programs. Firstly, an object-logic is implemented in FORUM using three different representations. Secondly, a small planning system illustrates FORUM's convenience in a resource sensitive domain. Finally, two small programs are translated into FORUM and they are used for illustrating the speed-ups of the box calculi.
- Chapter 5 compares and contrasts the approaches in Lolli and Lygon on the context management. Hodas' and Polakow's work on an implementation of FORUM as a logic programming language is studied.
- In Chapter 6, the thesis concludes by consideration of some open problems and suggestions for further work.

Results

- A sequent calculus $\mathcal{F}'$ based on multiset contexts in the succedent is derived from Miller's calculus $\mathcal{F}$. Both calculi include the '?' as primitive connective.
- A 'box calculus $\mathcal{B}$', so called because the contexts of sequent are formalised with lots of components to emphasise their different purposes, is presented; it is an extension of the IO model which deals efficiently with the context splitting. The soundness and completeness of this box calculus is proven relative to the sequent calculus $\mathcal{F}'$.
- A modified box calculus is presented which deals efficiently with the context splitting in the '$\top$-R' rule. In contrast to the earlier approaches addressing this inefficiency, the output context is split into a strict and a slack part. In this calculus, there is no need to introduce additional inference rules which would be forced when applying the earlier approaches.
- The sequent calculus $\mathcal{F}'$ and the box calculi are implemented in the logic programming language Terzo.
- An object-logic $\mathcal{O}$ (suggested by Miller), which includes the classical connectives '$\wedge$', ':-' and *true*, is implemented into FORUM in three different ways using the connectives '$\&$', '$\multimap$' and '$\top$'; '$\otimes$', '$\multimap$' and '$\top$' and '$⅋$', '$\bot$' and '$\multimap$'. The soundness and completeness for each of the three representations is proven.
- A small deductive planner for conjunctive planning problems is implemented in FORUM; this has some features that have not been addressed earlier in a representation using linear logic.

Chapter 2

Foundations of Logic Programming and FORUM

2.1 Single Succedent Logics and Uniform Proofs

At first, we investigate the foundation of logic programming which can be applied to languages that are based on either traditional logics or linear logics. We approach logic program execution by regarding it as a proof search method in a sequent calculus. Other views based on certain efficient calculi for rather restricted fragments of various logics (e.g. SLD-refutation in Prolog) are not considered. The sequent calculi are built upon certain sets of inference rules and grammars for formulae. The grammar serves, for example, for distinguishing *program* and *goal* formulae that appear on the left or on the right-hand side of sequents, respectively.

Definition 1 *A* sequent *is an expression of the form:*

$$\Delta \Longrightarrow \Gamma$$

where Δ *and* Γ *are syntactic variables for multisets of formulae called* program *formulae and* goal *formulae, respectively. A program formula is analysed by a* left rule*; a goal formula is analysed by a* right rule.

Note our reverse use of Gentzen's nomenclature [Gentzen, 1969] because Δ stands for a multiset of program ("definite") formulae and Γ stands for a multiset of goal formulae. In [Miller, 1989b], a proof in a sequent system is defined as follows:

Definition 2 *A* proof *of an arbitrary sequent* $\Delta \Longrightarrow \Gamma$ *is a finite tree whose nodes are labelled such that:*

1. *the root is labelled with the sequent* $\Delta \Longrightarrow \Gamma$*;*
2. *the inner nodes are instances of inference rules;*
3. *the leaf nodes are labelled with initial sequents, i.e., axioms or premise-free rules.*

A major foundation of all logic programming languages is searching in a certain space of proofs [Pym & Wallen, 1992]. In this thesis, the construction of a proof is carried out beginning from the root up to the leaves. Unfortunately, this construction of proofs incorporates much non-determinism. Suppose one wants to prove a sequent σ_0. Following [Andreoli, 1992], a simple proof search could be initiated with SEARCH(σ_0) where SEARCH is defined as follows:

Procedure SEARCH(σ)

```
1. Select an instance of an inference rule with the sequent $\sigma$ as the conclusion
   and the sequents $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) as the premises;
2. For each $k = 1, \ldots, n$ start SEARCH($\sigma_k$).
```

The procedure SEARCH is a *root-upward* construction of proofs. It starts with an end-sequent in the root and gradually completes the proof by proving the premises of the inference rules.

In general, Step 1 of SEARCH is not determined and an implementation of it has to explore all possible choices. This is a rather inefficient method for constructing proofs. An improvement of the search strategy employs some constraints which restrict the logic programming language to a certain fragment of the underlying logic. A key idea in logic programming language implementations is to use a cut-free formulation of the logic. The cut-elimination theorem ensures that the 'Cut' rule is admissible without a loss of expressivity. A 'Cut' rule looks as follows:

$$\frac{\Delta' \Longrightarrow \Gamma', B \qquad B, \Delta'' \Longrightarrow \Gamma''}{\Delta', \Delta'' \Longrightarrow \Gamma', \Gamma''}\ Cut.$$

It is noteworthy that the 'Cut' rule has a formula B, the *cut-formula*, which appears only in the premises and not in the conclusion. An instantiation in our root-upward construction of proofs has to guess non-deterministically at what the needed cut-formula might be. This guessing is too inefficient and that is why only cut-free logics are of interest in logic programming. However, this restriction is not serious because it is shown for: IL, CL ([Gentzen, 1969]), intuitionistic linear logic (ILL; [Bierman, 1994]), and CLL ([Girard, 1987]) that the 'Cut' rule is admissible.

A further restriction on the proof search construction deals with the selection of a formula that will be decomposed and the selection of a corresponding inference rule. For expository purposes, we first deal with logic programming languages which are based on single succedent logics. The sequents of a single succedent logic are constrained to have exactly one goal formula, i.e., the multiset Γ in Definition 1 consists only of one formula[1].

The notion of uniform proof introduced in [Miller et al., 1991] restricts the proof construction for logic programming languages based on a single succedent logic (the definition, slightly modified, is taken from [Hodas, 1994]):

Definition 3 *In a cut-free single succedent sequent calculus, a proof is* uniform *if for every occurrence of a sequent in the proof with a non-atomic goal formula, that occurrence is the conclusion of a right rule.*

[1]Note that many sequent calculi for intuitionistic logics are constrained to have at most one formula in the succedent. However, the empty succedents can be avoided when the calculus is rephrased with an explicit logical constant f, called absurdity.

This restriction can affect the completeness of proofs with respect to the underlying logic. In IL, for example, there exists no uniform proof for the sequent $p \vee q \Longrightarrow p \vee q$, but a non-uniform proof is as follows:

$$\dfrac{\dfrac{\overline{p \Longrightarrow p}\ Ax}{p \Longrightarrow p \vee q}\ \vee\text{-R} \qquad \dfrac{\overline{q \Longrightarrow q}\ Ax}{q \Longrightarrow p \vee q}\ \vee\text{-R}}{p \vee q \Longrightarrow p \vee q}\ \vee\text{-L} \tag{2.1}$$

The reason that a logic programming language becomes incomplete after the restriction to the uniform proofs is that certain left and right rules do not permute over each other. In the intuitionistic proof above, for example, the '$\vee$-L' rule lies under some '$\vee$-R' rules and the proof cannot be rewritten such that the right rules lay under the left rule. The '$\exists$-L' rule has the same troublesome property. However in a significant fragment of the logic, the proofs can be rewritten by permutations to uniform proofs. The following example shows two intuitionistic proofs of the same endsequent. In the first proof, a left rule lies under a right rule[2]:

$$\dfrac{\dfrac{\dfrac{\overline{p \Longrightarrow p}\ Ax}{p, q \Longrightarrow p}\ \text{Weakening-L} \qquad \dfrac{\overline{q \Longrightarrow q}\ Ax}{p, q \Longrightarrow q}\ \text{Weakening-L}}{p, q \Longrightarrow p \wedge q}\ \wedge\text{-R}}{p \wedge q \Longrightarrow p \wedge q}\ \wedge\text{-L}$$

The proof can be permuted to a uniform proof:

$$\dfrac{\dfrac{\dfrac{\overline{p \Longrightarrow p}\ Ax}{p, q \Longrightarrow p}\ \text{Weakening-L}}{p \wedge q \Longrightarrow p}\ \wedge\text{-L} \qquad \dfrac{\dfrac{\overline{q \Longrightarrow q}\ Ax}{p, q \Longrightarrow q}\ \text{Weakening-L}}{p \wedge q \Longrightarrow q}\ \wedge\text{-L}}{p \wedge q \Longrightarrow p \wedge q}\ \wedge\text{-R}$$

After identification of a fragment where uniform proofs exist for every provable sequent, the method SEARCH described above can be modified for the search for uniform proofs. SEARCH$'$ is as follows:

Procedure SEARCH$'(\sigma)$

1. If the goal formula of σ is non-atomic, then select an instance of an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ, that analyses the outermost connective of the goal formula;

1' otherwise, select an instance of an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ, that analyses a program formula;

2. For each $k = 1, \ldots, n$ start SEARCH$'(\sigma_k)$.

SEARCH$'$ focuses first on the goal formula and applies right rules until a sequent with an atomic goal formula is reached. This focusing on the goal formula, when analysing it, is called *goal-directed* or *goal-conducted* proof search. It reduces the amount of non-determinism during the proof construction. The procedure SEARCH$'$ is *complete* for a logic where for every provable sequent there exists a uniform proof. The restriction to uniform proof leads to the definition of an "abstract logic programming language" which can be found in [Miller et al., 1991] and [Hodas, 1994]. The following definition is adapted from the former.

[2]The example is taken from [Pym & Harland, 1994].

Definition 4 *A logic with a single succedent sequent calculus proof system is an* abstract logic programming language *if the restriction to uniform proofs does not lose completeness.*

2.1.1 Horn Logic and its Formulae

The logic programming language Prolog is based on Horn logic which is a simple but inexpressive fragment of IL. In Horn logic, the sequents are as follows:

$$H_1, \ldots, H_2 \Longrightarrow G$$

where H_i are Horn formulae and G is a goal formula. The Horn formulae are organised so that a disjunction or an existential quantifier do not occur during proof search as outermost connective of a formula on the left-hand side of sequents. The Horn restrictions on formulae are necessary in order to maintain the completeness with uniform proofs. On the other hand, the goal formulae of Horn logic are rather restricted as well (these restrictions can be relaxed as seen in the following section). As a result of these restrictions, a very efficient proof search strategy is developed for Horn logic.

The division of the sequent into Horn and goal formulae offers a convenient way to introduce a grammar for formulae of Horn logic.

$$G ::= \top \mid A \mid G \wedge G \mid G \vee G \mid \exists x G$$

$$H ::= A \mid G \supset H \mid H \wedge H \mid \forall x H$$

where A ranges over atomic formulae; G are called goal formulae; H are called Horn formulae which are also called *definite formulae*. The problematic inference rules (i.e., '$\vee$-L', '$\exists$-L') of IL can be omitted because the grammar does not permit an occurrence of a disjunction or an existential quantifier on the left-hand side of a sequent. As a result, in [Miller et al., 1991] it is shown that uniform proofs are complete for Horn logic. Consequently, Horn logic is an abstract logic programming language. In the following section, it is shown that not all the imposed constraints are necessary in order to maintain the completeness of uniform proofs.

2.1.2 First-Order Hereditary Harrop Formulae

First-order Hereditary Harrop formulae are an extension of the notion of Horn formula. As shown in Example 2.1, uniform proofs cannot be complete with IL because there does not exist a uniform proof for every intuitionistic provable sequent. However, all right rules of IL can be permitted in a logic programming language, and therefore, the limitation on goal formulae in Horn logic can be weakened. This extension supports some very useful programming constructs such as modules; see [Miller, 1989b] and [Miller, 1993]. The formulae in first-order Hereditary Harrop logic are defined as follows:

$$G ::= \top \mid A \mid G \wedge G \mid G \vee G \mid \exists x G \mid \forall x G \mid D \supset G$$

$$D ::= A \mid G \supset D \mid D \wedge D \mid \forall x D$$

where A ranges over atomic formulae; G formulae are called goal formulae; D formulae are called program formulae (also called definite formulae). The implicans of an implicative goal formula is restricted to be of the form of a definite formula since it only appears on the left-hand side of a sequent during the proof construction. On the other hand, the implicans of an implicative definite formula can be of the form G since it only appears on the right-hand side of a sequent.

In [Miller et al., 1991], it is shown that uniform proofs are complete with respect to the class of first-order Hereditary Harrop formulae. Some further restrictions force the unification of atomic formulae in higher-order Hereditary Harrop logic, but they will not be discussed here[3]. First-order Hereditary Harrop logic is the logic underlying a sublanguage of λProlog.

2.1.3 Lolli and its Formulae

Lolli is a language based on a fragment of ILL; it can be regarded as a linear refinement of first-order Hereditary Harrop logic [Hodas, 1992]. The goal and program formulae, respectively, in Lolli are defined as follows:

$$G ::= \top \mid \mathbf{1} \mid A \mid G \otimes G \mid G \& G \mid G \oplus G \mid !G \mid D \multimap G \mid D \supset G \mid \forall x G \mid \exists x G$$

$$D ::= \top \mid A \mid D \& D \mid \forall x D \mid G \multimap D \mid G \supset D$$

Remarks similar to those given for first-order Hereditary Harrop formulae can be made for the formulae defined above. For example, the '$\otimes$-L' rule cannot be permuted over all right rules. An example is the following proof:

$$\dfrac{\dfrac{\dfrac{}{p \Longrightarrow p}\ \text{Ax} \qquad \dfrac{}{q \Longrightarrow q}\ \text{Ax}}{p, q \Longrightarrow p \otimes q}\ \otimes\text{-R}}{p \otimes q \Longrightarrow p \otimes q}\ \otimes\text{-L} \tag{2.2}$$

Consequently, '$\otimes$' is not used for constructing D-formulae. The '⅋' connective is left out in the goal formulae because it is not meaningful in Lolli's single succedent calculus. However, some presentations of ILL include the '⅋' connective on the right hand-side of sequents; for example [Hyland & de Paiva, 1993]. (Beth, Maehara and Takeuti presented a similar formulation of IL which allows multiple succedents; see for example [Beth, 1965].)

In [Hodas, 1994], it is proven that for every provable sequent using the formulae defined above there exists a uniform proof, i.e., this fragment of ILL is an abstract logic programming language. It is implemented as the logic programming language "Lolli".

2.2 Extension of Uniform Proofs for Forum

So far, the search procedure SEARCH$'$ deals only with one goal formula. In the multiple succedent logic Forum, however, the succedent of a sequent might consist of more than one goal formula. The thesis is restricted to an extension of uniform proofs for multiple succedent logics based on linear logic. Nevertheless, in [Nadathur, 1995] there is an approach towards uniform proofs in CL. It encodes a fragment of CL into a suitable system where uniform and classical provability coincide.

[3]See [Miller et al., 1991] for a complete presentation of the completeness of uniform proofs of higher-order Hereditary Harrop logic.

The basic principles of uniform proofs in fragments of CLL are studied in [Andreoli, 1992] and [Pym & Harland, 1994]. The later work explores the permutability properties of CLL's inference rules and focuses on an implementation of a large fragment of CLL as a logic programming language. Accordingly, in this fragment of CLL left rules can be permuted over right rules. Following this approach, the SEARCH$'$ method can be augmented for multiple succedents as follows:

Procedure SEARCH$''(\sigma)$

1. If at least one goal formula in σ is non-atomic, then select an instance of an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ, that analyses the outermost connective of one non-atomic goal formula;

1' otherwise, select an instance of an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ, that analyses a program formula;

2. For each $k = 1, \ldots, n$ start SEARCH$''(\sigma_k)$.

The search procedure above focuses first on the goal side. However, along with the change in Step 1 comes a new source of non-determinism because the interpreter has to select a goal formula. In general, there might be some interdependencies between the goals as shown in the following proof (the analysed formula is underlined in each sequent):

$$\frac{\dfrac{\dfrac{\dfrac{}{p \Longrightarrow p}\,Ax \quad \dfrac{}{q \Longrightarrow q}\,Ax}{\underline{p \parr q} \Longrightarrow q, p}\,\parr\text{-L} \qquad \dfrac{\dfrac{}{p \Longrightarrow p}\,Ax}{\underline{p\&q} \Longrightarrow p}\,\&\text{-L}}{p\&q, p \parr q \Longrightarrow \underline{p \otimes q}, p}\,\otimes\text{-R} \qquad \dfrac{\dfrac{\dfrac{}{p \Longrightarrow p}\,Ax \quad \dfrac{}{q \Longrightarrow q}\,Ax}{\underline{p \parr q} \Longrightarrow p, q}\,\parr\text{-L} \qquad \dfrac{\dfrac{}{q \Longrightarrow q}\,Ax}{\underline{p\&q} \Longrightarrow q}\,\&\text{-L}}{p\&q, p \parr q \Longrightarrow \underline{p \otimes q}, q}\,\otimes\text{-R}}{p\&q, p \parr q \Longrightarrow p \otimes q, \underline{p\&q}}\,\&\text{-R}$$

The proof is only constructible if the second goal $p\&q$ is chosen first. The interdependencies between goals are caused by the impermutability of certain pairs of right rules. In the case above, the '&-R' and '$\otimes$-R' rule are impermutable. To ensure that the SEARCH$''$ procedure is complete, all possible ways of selecting a goal formula may have to be explored. This is very inefficient and would exceed the efficiency criterion for a logic programming language. However, there are two different approaches for a large fragment of CLL which focus on this difficulty and on an implementation of a logic programming language for a multiple succedent linear logic.

Firstly, the logic programming language Lygon (see [Harland & Winikoff, 1995b] and [Harland & Winikoff, 1996b]) is presented; this is implemented as a single-sided calculus with multiple goal formulae. In Section 2.6, this programming language will be compared with FORUM. Secondly, Miller described an extension of the notion of uniform proof in multiple succedent logics. This approach led to the definition of FORUM (see [Miller, 1994]). The key idea is that two proofs which differ in the order of right rule applications are regarded as equivalent proofs. Consequently, the order of selecting the goal formulae does not matter. The following constraints are imposed on the inference rules of FORUM in order to implement this idea:

- all left rules have to permute over all right rules (to allow a proof search for uniform proofs), and
- all right rules have to permute over each other.

As a result, the logic behind **Forum** is rather restricted to meet all these constraints. Forum's connectives, a fragment of CLL, and the corresponding inference rules come from [Andreoli, 1992] which is an investigation of permutation properties in a single sided sequent calculus. The first study of permutability properties of inference rules for CL appeared in [Kleene, 1952]. [Lincoln, 1991] presented first the complete permutation properties of a single-sided linear sequent calculus which are also studied in [Galmiche & Perrier, 1994].

A permutation is an exchange of two inference rules that has only local effects on the proof tree. A situation where two inferences are permutable is characterised by the following general pattern:

$$\dfrac{\dfrac{\dfrac{\vdots}{\sigma_1}\cdots\dfrac{\vdots}{\sigma_n}}{\sigma}\,S}{\sigma_0}\,R \quad\Longleftrightarrow\quad \dfrac{\dfrac{\dfrac{\vdots}{\sigma_1}}{\sigma_1'}\,R\cdots\dfrac{\dfrac{\vdots}{\sigma_n}}{\sigma_n'}\,R}{\sigma_0}\,S \tag{2.3}$$

where the premises σ_k $(k = 1, \ldots, n)$ are obtained from the same endsequent σ_0 in two different ways.

In [Andreoli, 1992], the "asynchronous" connectives meet the property that the associated right rules permute over each other. They are as follows:

$$\bot, \top, ⅋, \&, \forall \text{ and } ?.$$

For expository purposes the "synchronous" connectives are introduced as well. They are as follows:

$$\mathbf{1}, \mathbf{0}, \otimes, \oplus, \exists \text{ and } !.$$

Miller added the implications '$\multimap$' and '$\supset$' to the set of "asynchronous" connectives; they do not occur in Andreoli's single sided calculus. In Forum, however, their right-rules also permute over all other right-rules. The uniform proofs for multiple succedent sequents are defined as follows (slightly changed taken from [Miller, 1996]):

Definition 5 *A cut-free proof Ξ is* uniform *if and only if, for every subproof Ξ' of Ξ and for every non-atomic formula occurrence B in the succedent of the endsequent of Ξ', there exists a proof Ξ'' such that:*

1. *the height of Ξ'' is not higher than the height of Ξ',*
2. *Ξ'' is equal to Ξ' up to some permutations of inference rules,*
3. *the top-level logical connective of B is introduced by the last inference step of Ξ''.*

The first property is not included in Miller's definition, but is important for the termination of the inductive definition. However in this context, it follows directly from the construction of Ξ'' because both proofs differ only in the order of inference rules and have the same height. (The permutations are carried out with respect to the pattern in Display 2.3.)

For the set of connectives described above and the corresponding notion of uniform proofs, it is possible to reformulate the SEARCH″ procedure as follows:

Procedure SEARCH$'''(\sigma)$

1. If one goal formula of σ is non-atomic then select a non-atomic goal formula that is analysed by an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ;

1' otherwise, select an instance of an inference rule, with premises $\sigma_1, \ldots, \sigma_n$ (with $n \geq 0$) and conclusion σ, that analyses a program formula;

2. For each $k = 1, \ldots, n$ start SEARCH$'''(\sigma_k)$.

Step 1 is only slightly changed and is still not deterministic. However, the choice of a particular goal formula does not matter because of the permutability property. Such non-determinism is called "don't care" non-determinism in contrast to the "don't know" non-determinism in Step 1 of the procedure SEARCH$''$. The crucial improvement of SEARCH$'''$ arises from the fact that one can omit the inefficient backtracking mechanism for exploring all possible choices in Step 1 (but not in Step 1'). The "don't care" non-determinism causes no inefficiency in an implementation.

2.3 Miller's Proof System $\mathcal{F}$ for FORUM

This section is based mainly on work taken from Miller's papers on FORUM; [Miller, 1994] and [Miller, 1996]. He has presented[4] FORUM with $\top$, $\bot$, $\&$, $⅋$, $\multimap$, $\supset$, $\forall$ and $?$ as primitive connectives. All formulae of FORUM can be freely generated from this set. Thus, the formulae of FORUM are defined as follows:

$$F ::= \top \mid \bot \mid A \mid F \& F \mid F ⅋ F \mid F \multimap F \mid F \supset F \mid \forall x F \mid ?F \tag{2.4}$$

where the syntactic variable A ranges over atoms; the logical constants '$\top$' and '$\bot$' are regarded as non-atomic formulae.

The rather restricted fragment of CLL using the formulae F (2.4) satisfies the uniform proof condition given in Definition 5. However, this does not mean that the expressivity of the language is restricted as well. The linear negation can be introduced using the primitive connectives since, in CLL, the formulae $B^\perp$ and $B \multimap \bot$ are equivalent. In what follows, the implications of the form $B \multimap \bot$ are called '$\bot$-headed implications' (see [Hodas & Polakow, 1996]).

The '$\otimes$' connective, for example, is excluded from the set of primitive connectives because its right rule does not meet the extended uniform proof condition. (An example was given for Lolli in Section 2.1.3 where a '$\otimes$-L' rule must be applied at first; the same argument can be applied in FORUM.) However, Miller reintroduces the missing connectives of CLL using the linear negation, '$^\perp$'. The following logical equivalences hold:

$$\begin{array}{ccc} B^\perp \equiv B \multimap \bot & \mathbf{0} \equiv \top \multimap \bot & \mathbf{1} \equiv \bot \multimap \bot \\ !B \equiv (B \supset \bot) \multimap \bot & B \oplus C \equiv (B^\perp \& C^\perp)^\perp & B \otimes C \equiv (B^\perp ⅋ C^\perp)^\perp \\ & \exists x.B \equiv (\forall x.B^\perp)^\perp. & \end{array}$$

This thesis uses the more recent version of the sequent calculus $\mathcal{F}$ for FORUM (see [Miller, 1995, Miller, 1996]). This allows the introduction of the exponential '?' as a primitive connective rather than as an abbreviation. The *right, decide* and *left* rules of $\mathcal{F}$ are

[4]To be consistent with our notation we change Miller's $\Rightarrow$ into $\supset$.

given in Figures 2.1, 2.2 and 2.3. The components of the sequents are described in Section 2.3.1.

$$\frac{}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, \top, \Gamma; \Upsilon}\ \top\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B, \Gamma; \Upsilon \quad \Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, C, \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B\&C, \Gamma; \Upsilon}\ \&\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, \bot, \Gamma; \Upsilon}\ \bot\text{-R} \qquad \frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B, C, \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B\parr C, \Gamma; \Upsilon}\ \parr\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta, B \Longrightarrow \mathcal{A}, C, \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B\multimap C, \Gamma; \Upsilon}\ \multimap\text{-R} \qquad \frac{\Sigma\colon \Psi, B;\Delta \Longrightarrow \mathcal{A}, C, \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B\supset C, \Gamma; \Upsilon}\ \supset\text{-R}$$

$$\frac{y:\tau, \Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B[x\mapsto y], \Gamma; \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, \forall_\tau x B, \Gamma; \Upsilon}\ \forall\text{-R} \qquad \frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, \Gamma; B, \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, ?B, \Gamma; \Upsilon}\ ?\text{-R}$$

y is not declared in Σ

Figure 2.1: The 'right' rules in Miller's proof system $\mathcal{F}$.

$$\frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}, B; B, \Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A}; B, \Upsilon}\ \textit{decide?}$$

$$\frac{\Sigma\colon \Psi;\Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi;B,\Delta \Longrightarrow \mathcal{A}; \Upsilon}\ \textit{decide} \qquad \frac{\Sigma\colon B,\Psi;\Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon B,\Psi;\Delta \Longrightarrow \mathcal{A}; \Upsilon}\ \textit{decide!}$$

Figure 2.2: The 'decide' rules in Miller's proof system $\mathcal{F}$.

$$\frac{}{\Sigma\colon \Psi;\emptyset \overset{A}{\Longrightarrow} A; \Upsilon}\ \textit{initial} \qquad \frac{}{\Sigma\colon \Psi;\emptyset \overset{A}{\Longrightarrow} [\,]; A, \Upsilon}\ \textit{initial?}$$

$$\frac{}{\Sigma\colon \Psi;\emptyset \overset{\bot}{\Longrightarrow} [\,]; \Upsilon}\ \bot\text{-L} \qquad \frac{\Sigma\colon \Psi;\Delta \overset{B_i}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi;\Delta \overset{B_1\&B_2}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \&\text{-L}_i$$

$$\frac{\Sigma\colon \Psi;B \Longrightarrow [\,]; \Upsilon}{\Sigma\colon \Psi;\emptyset \overset{?B}{\Longrightarrow} [\,]; \Upsilon}\ ?\text{-L} \qquad \frac{\Sigma\colon \Psi;\Delta \overset{B[x\mapsto t]}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi;\Delta \overset{\forall_\tau x B}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \forall\text{-L}$$

t is a Σ-term of type τ

$$\frac{\Sigma\colon \Psi;\Delta_1 \overset{B}{\Longrightarrow} \mathcal{A}_1; \Upsilon \quad \Sigma\colon \Psi;\Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \overset{B\parr C}{\Longrightarrow} \mathcal{A}_1 + \mathcal{A}_2; \Upsilon}\ \parr\text{-L}$$

$$\frac{\Sigma\colon \Psi;\Delta_1 \Longrightarrow \mathcal{A}_1, B; \Upsilon \quad \Sigma\colon \Psi;\Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \overset{B\multimap C}{\Longrightarrow} \mathcal{A}_1 + \mathcal{A}_2; \Upsilon}\ \multimap\text{-L}$$

$$\frac{\Sigma\colon \Psi;\emptyset \Longrightarrow B; \Upsilon \quad \Sigma\colon \Psi;\Delta \overset{C}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi;\Delta \overset{B\supset C}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \supset\text{-L}$$

Figure 2.3: The 'left' rules in Miller's proof system $\mathcal{F}$.

In [Miller, 1996], the representation of sequents for FORUM is non-standard (sequents are introduced in Definition 1). In Miller's calculus $\mathcal{F}$ there are two kinds of sequents, one of which has a *stoup* formula written over the arrow as in [Girard, 1991], and thus, we distinguish them by calling them *stoup* sequents and *non-stoup* sequents. A separate part Σ, which contains the type information of constants in the λ-calculus, is added to the sequents.

The terms in FORUM are strongly typed using a fragment from Church's simple theory of types (see Section 2.7.1). The Σ part is separated by a colon from the rest of the sequent. The antecedent and succedent are separated, as usual, by the sequent arrow, but both of them are divided into *zones*. This will be described in more detail in Section 2.3.1.

Each 'right' rule in $\mathcal{F}$ analyses a goal formula in the succedent. The 'decide' rules are used when atomic goals are reached in the succedents. The 'decide?' rule serves for choosing a new goal formula from the classical context of the succedent. The other two 'decide' rules pick up a formula from the antecedent which goes subsequently into the *stoup*. These two rules correspond to the selection of a program formula (Step 1' in SEARCH'''). The 'left' rules analyse the *stoup* formula; it will be decomposed until atoms are reached. In effect, the proof search in $\mathcal{F}$ for a *stoup*-sequent is focused (in the sense of [Andreoli, 1992]) on one program formula. That means that a program formula that is selected has to lead to a proof where all leaves match with either a 'initial' rules or the '$\perp$-R' rule. However, three rules are exceptions to this scheme. In the case where the *stoup* formula matches with $?B$, the program is augmented with B; another program formula will be chosen to go into the *stoup* or the succedent can be augmented by the 'decide?' rule. This means that the focus of the rather restricted *stoup*-sequents is lost. However, that is necessary since it maintains the completeness with respect to CLL (an example is given in Section 2.5). The other exceptions are the two rules that analyse an implication in the *stoup*. They branch the proof tree and in one premise the implicans is added to the succedent as an additional goal formula which will be analysed by a 'right' or 'decide' rule.

2.3.1 The Use of Contexts in Sequents

In linear logic, formulae in the antecedent with a '!' as an outermost connective are dischargeable; the other formulae in the antecedent are usable exactly once (the same behaviour in the succedent is achieved by the exponential '?'). That means the structural rules for weakening and contraction are restricted to formulae with an exponential. These rules are defined in CLL as follows:

$$\frac{\Delta \Longrightarrow \Gamma}{\Delta, !B \Longrightarrow \Gamma}\ \text{Weakening!-L} \qquad \frac{\Delta \Longrightarrow \Gamma}{\Delta \Longrightarrow ?B, \Gamma}\ \text{Weakening?-R}$$

$$\frac{\Delta, !B, !B \Longrightarrow \Gamma}{\Delta, !B \Longrightarrow \Gamma}\ \text{Contraction!-L} \qquad \frac{\Delta \Longrightarrow ?B, ?B, \Gamma}{\Delta \Longrightarrow ?B, \Gamma}\ \text{Contraction?-R}$$

Consequently, the formulae decorated with an exponential behave to some extent as if they were classical formulae. In [Girard, 1993] this observation is used and a sequent calculus, called LU, is presented in order to unify CL, IL and CLL in one calculus. The sequents are divided into different zones—into *classical* and *linear* zones. Each zone allows a different usage of the structural rules weakening and contraction. In the classical zones, these rules are permitted; however, they are implemented implicitly in the inference rules. In the linear zones, both structural rules are forbidden.

A similar approach is used in the *dyadic* single-sided calculus of [Andreoli, 1992] in order to normalise proofs. Given that the proof search is performed from the root upwards to the leaves, the application of the structural rules is delayed in this calculus as long as possible. That means the structural rules are permuted so that they occur in proofs as late as possible. The weakening rule can be permuted until leaves are reached and is implicitly built into the rules that terminate a proof branch. The contraction rule can be permuted until an occurrence of certain connectives, and it is built into the corresponding inference rules.

In logic programming languages, for example in Lolli, the implicit representation of the structural rules is preferred because of the fact that the 'Contraction!-L', 'Weakening!-L' and '?-L' rule analysing formulae with an exponential violate the uniform proof condition. Consider the following proof in CLL (the example is from [Pym and Harland, 1994]):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \Longrightarrow p}\,Ax}{!p \Longrightarrow p}\,!\text{-L} \quad \dfrac{\dfrac{}{p \Longrightarrow p}\,Ax}{!p \Longrightarrow p}\,!\text{-L}}{!p, !p \Longrightarrow p \otimes p}\,\otimes\text{-R}}{!p \Longrightarrow p \otimes p}\,\text{Contraction!-L}$$

The proof can not be rewritten such that the '⊗-R' rule lies under all left rules. Using an implicit formulation, the structural rules are admissible and can be simulated with the new zones. The '⊗-R' rule with zones could be formulated as follows (see [Girard, 1993]):

$$\dfrac{\Psi; \Delta_1 \Longrightarrow B, \Gamma_1; \Upsilon \quad \Psi; \Delta_2 \Longrightarrow C, \Gamma_2; \Upsilon}{\Psi; \Delta_1, \Delta_2 \Longrightarrow B \otimes C, \Gamma_1, \Gamma_2; \Upsilon}\,\otimes\text{-R}'$$

The formulae in the zones Ψ and Υ are given to both premises. In case where they are superfluous, it is permitted to "weaken" them. Miller's calculus $\mathcal{F}$, for example, is formulated using sequents with zone; these zones are also called *contexts*[5]. An extension of this idea is described in [Hodas, 1993] for Lolli where four contexts exist: for intuitionistic, relevant, affine, and linear logic. These contexts differ in the usage of the structural rules.

In FORUM, each antecedent and each succedent is divided into a *classical* and a *linear* context. The classical contexts represent reusable formulae; the formulae in the linear context are usable exactly once. Miller presented the sequents as follows:

$$\Sigma\colon \Psi; \Delta \Longrightarrow \Gamma; \Upsilon \qquad \text{and} \qquad \Sigma\colon \Psi; \Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon$$

where the linear contexts (Δ, Γ and $\mathcal{A}$) are grouped around the sequent arrow, and the classical contexts (Ψ, Υ) are grouped at the edge of the sequent (the purpose of Σ will be described in detail in Section 2.7). The linear contexts Δ in the antecedent are multisets of formulae; the classical contexts Ψ and Υ are sets of formulae. The antecedent Γ in the sequents above is introduced as a list. The list starts with a part that consists only of atoms (the variable $\mathcal{A}$ stands for this part) and is followed by a part that consists of atomic or non-atomic formulae ($\mathcal{B}$ stands for this part). The part $\mathcal{B}$ of Γ is always empty in *stoup* sequents. The *stoup* sequents (on the right-hand side above) have a *stoup* formula B written over the sequent arrow. Omitting the type information in Σ, the sequents are intended to behave like the following sequents in CLL:

$$!\Psi, \Delta \Longrightarrow \Gamma, ?\Upsilon \qquad \text{and} \qquad !\Psi, \Delta, B \Longrightarrow \Gamma, ?\Upsilon$$

where $!\Psi$ and $?\Upsilon$ is a shorthand that stands for multisets where each formula is decorated with a '!' or a '?', respectively.

[5] Another reason for the usage of the contexts (i.e., zones) in linear logic programming is that they simplify the splitting of the contexts in the multiplicative rules. Usually, formulae have to be distributed over the two premises by an expensive operation that explores all possibilities. The formulae in the classical contexts are excluded from this expensive operation, since they can be given safely to both branches.

2.3.2 Behaviour of the Non-Primitive Connectives

The non-primitive connectives, for example the '$\otimes$' connective, are excluded from the set of primitive connectives because their left rules cannot be permuted over all right rules or their right rules do not permute over all other right rules. They are reintroduced by some logical equivalences. Because of this fact, there are different opinions on whether FORUM should count as a logic programming language for full CLL or not. Certainly, FORUM has a more direct relationship to CLL than logic languages, such as PROLOG, since the completeness is achieved without the help of a theory. All connectives of FORUM including the non-primitive ones fulfil the extended uniform proof condition (Definition 5). Consequently, FORUM can be seen as an abstract logic programming language as given in Definition 3.

However in FORUM, only for a provable sequent using primitive connectives in the succedent does there exist a uniform proof which is goal-directed (see Step 1 in SEARCH$'$). That means the corresponding formula can be decomposed efficiently using only right rules until atomic formulae are reached. Such a proof may not exist for the non-primitive connectives. The translation of the non-primitive connectives ensures that they fit into the scheme of uniform proof search. However, an occurrence of a non-primitive connective in the succedent cannot be treated entirely with the "don't care" non-determinism as in Step 1 of SEARCH$'''$. The translation into a $\bot$-headed implication for formulae with an outermost non-primitive connective has the effect that an analysis of such a formula adds another formula to the antecedent. Then, this additional formula in the antecedent is treated with less efficient "don't know" choices. Therefore, only the core of FORUM (formulae with primitive connectives) counts as a logic programming language if one has in mind that for all provable sequents there exists a goal directed proof.

In the following example, which is taken from page 17, a FORUM proof is given for a formula with the non-primitive connective '$\otimes$' in the succedent. The proof is presented using the calculus $\mathcal{F}$ of FORUM. The endsequent that will be proven is as follows:

$$\Sigma\colon \emptyset; p\&q, p\parr q \Longrightarrow p\&q, p\otimes q; \emptyset.$$

It is assumed that Σ contains the appropriate declarations for p, q and the used connectives. With a usual formulation of the '$\otimes$-R' rule the goal formula $p\otimes q$ has to be analysed first in order to obtain a proof. However in the sequent with the translated form of the $\otimes$-formula, the order in which the goals appear is not significant. The sequent using the logically equivalent formula for $p\otimes q$ is as follows:

$$\Sigma\colon \emptyset; p\&q, p\parr q \Longrightarrow p\&q, (p^{\perp}\parr q^{\perp})\multimap\bot; \emptyset$$

where the negations can be expressed by $\bot$-headed implications. This sequent will be proven in FORUM by analysing the $\&$-formula first.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Xi_1}{\Sigma\colon \emptyset; p^{\perp}\parr q^{\perp}, p\&q, p\parr q \Longrightarrow p; \emptyset}}{\Sigma\colon \emptyset; p^{\perp}\parr q^{\perp}, p\&q, p\parr q \Longrightarrow p, \underline{\bot}; \emptyset}\ \bot\text{-R}}{\Sigma\colon \emptyset; p\&q, p\parr q \Longrightarrow p, \underline{(p^{\perp}\parr q^{\perp})\multimap\bot}; \emptyset}\ \multimap\text{-R} \qquad \dfrac{\dfrac{\dfrac{\Xi_2}{\Sigma\colon \emptyset; p^{\perp}\parr q^{\perp}, p\&q, p\parr q \Longrightarrow q; \emptyset}}{\Sigma\colon \emptyset; p^{\perp}\parr q^{\perp}, p\&q, p\parr q \Longrightarrow q, \underline{\bot}; \emptyset}\ \bot\text{-R}}{\Sigma\colon \emptyset; p\&q, p\parr q \Longrightarrow q, \underline{(p^{\perp}\parr q^{\perp})\multimap\bot}; \emptyset}\ \multimap\text{-R}}{\Sigma\colon \emptyset; p\&q, p\parr q \Longrightarrow \underline{p\&q}, (p^{\perp}\parr q^{\perp})\multimap\bot; \emptyset}\ \&\text{-R}$$

In the subproof Ξ_1, the new program formula $p^{\perp}\parr q^{\perp}$ must be chosen in order to find a proof for the endsequent. This is a "don't know" choice and FORUM has to explore all three possibilities. However, FORUM uses a simple heuristic: new program formulae, that were added by an implication in the succedent, are considered first. In this case, it leads immediately to the successful proof, but this is not necessarily always the case. The subproof Ξ_1 is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma:\emptyset;\emptyset\xRightarrow{p}p;\emptyset}\,\mathit{initial}}{\Sigma:\emptyset;\emptyset\xRightarrow{p\&q}p;\emptyset}\,\&\text{-L}_1}{\Sigma:\emptyset;p\&q\Longrightarrow p;\emptyset}\,\mathit{decide!}}{\Sigma:\emptyset;p\&q\xRightarrow{p\multimap\bot}[];\emptyset}\,\multimap\text{-L}^{\sharp}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma:\emptyset;\emptyset\xRightarrow{p}p;\emptyset}\,\mathit{initial}\quad\dfrac{}{\Sigma:\emptyset;\emptyset\xRightarrow{q}q;\emptyset}\,\mathit{initial}}{\Sigma:\emptyset;\emptyset\xRightarrow{p\parr q}p,q;\emptyset}\,\parr\text{-L}}{\Sigma:\emptyset;p\parr q\Longrightarrow p,q;\emptyset}\,\mathit{decide!}}{\Sigma:\emptyset;p\parr q\xRightarrow{q\multimap\bot}p;\emptyset}\,\multimap\text{-L}^{\sharp}}{\dfrac{\Sigma:\emptyset;p\&q,p\parr q\xRightarrow{p^{\bot}\parr q^{\bot}}p;\emptyset}{\Sigma:\emptyset;p^{\bot}\parr q^{\bot},p\&q,p\parr q\Longrightarrow p;\emptyset}\,\mathit{decide!}}\,\parr\text{-L}$$

The second premise in both '$\multimap$-L$^{\sharp}$' inference rules is omitted. These premises have the logical constant '$\bot$' in the *stoup* and empty linear contexts; they can be proven immediately by the rule '$\bot$-L'. The subproof Ξ_2 is similar to Ξ_1, and so is omitted.

In [Hodas & Polakow, 1996] and [Harland & Winikoff, 1996b], similar remarks to those above are made: FORUM does not have goal-directed proofs for all sequents of CLL, but only for sequents having formulae with primitive connectives; the logic underlying FORUM is as expressive as CLL because all of CLL's sequents can be mapped into FORUM sequents via some logical equivalences and the provability is preserved when restricting to uniform proofs as given in Definition 5. However, the implementation of FORUM is not only restricted to a uniform proof search but also to a depth-first proof search. Therefore, FORUM for which we will present an implementation is not as expressive as the underlying logic. Further remarks on this aspect will be given in Section 2.5 and 3.4.

2.4 Modified Proof System $\mathcal{F}'$ Including an Atomic-Rule

In the sequents of the calculus $\mathcal{F}$, the variable Γ stands for a list and the expression $\mathcal{A}_1+\mathcal{A}_2$ stands for a list obtained by interleaving $\mathcal{A}_1$ and $\mathcal{A}_2$. That means the operator '+' preserves the order of both argument lists in the resulting list. However, for the soundness and completeness proofs, which will be given in Chapter 3, it is more convenient to use a multiset notation instead[6]. Miller chose a list structure for the goal formulae because the proof search in such a system is easier to implement. However, that is an arbitrary decision: Miller proved in Corollary 4 of [Miller, 1996] that the order of goal formulae is not crucial for the provability. Precisely, if for a non-*stoup* sequent there exists a proof, then for all sequents consisting of a permutation of the goal list there exists a proof as well. In Lemma 2 of the same paper, he also showed that the order of atoms in *stoup*- and non-*stoup* sequents is not significant for the provability of the sequent. Hence, all possible orders of goal achievements lead to the same result. The proof can be shown by the fact that all right rules of FORUM permute over each other (its connectives are chosen so that they satisfy the extended uniform proof definition and hence satisfy this condition).

[6] Thus, an interpreter has to incorporate the following operations over multisets: selection of an element, multiset union and splitting of a multiset into two parts.

Consequently, the calculus $\mathcal{F}'$ (derived from Miller's $\mathcal{F}$) is used where the list structure of Γ is replaced by two multisets. The multisets ease any structural inductions because one does not have to pay attention to the new calculus preserving the list order as would be necessary for a proof using $\mathcal{F}$.

One multiset ($\mathcal{A}$) contains only atomic goals, and the other one ($\mathcal{B}$)[7] contains all other goals which have not been analysed yet. The sequents of $\mathcal{F}$:

$$\Sigma\colon \Psi;\Delta \Longrightarrow \Gamma;\Upsilon \qquad \text{and} \qquad \Sigma\colon \Psi;\Delta \xRightarrow{B} \mathcal{A};\Upsilon$$

are replaced in $\mathcal{F}'$ by the following sequents:

$$\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};\Upsilon \qquad \text{and} \qquad \Sigma\colon \Psi;\Delta \xRightarrow{B} \mathcal{A};\Upsilon.$$

The multiset $\mathcal{B}$ in *stoup* sequents (on the right-hand side) is always empty because of the proof search method used in FORUM, and therefore, this part is omitted in $\mathcal{F}'$. An '*atomic*-R' rule is newly introduced into the calculus in order to move an atomic goal formula from $\mathcal{B}$ to $\mathcal{A}$.

The proof of soundness and completeness of $\mathcal{F}'$ with respect to $\mathcal{F}$ is omitted: the proof can be achieved with the results from Miller; the additional '*atomic*-R' rule of $\mathcal{F}'$ is embedded implicitly in Miller's list notation. The use of the 'decide' rules in $\mathcal{F}$ is triggered when the list Γ consists only of atomic formulae. In the new calculus $\mathcal{F}'$, a 'choose' rule is applied when the multiset $\mathcal{B}$ is empty. The new inference rules of $\mathcal{F}'$ can be found in Figures 2.4, 2.5, and 2.6.

$$\frac{}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\top,\mathcal{B};\Upsilon}\ \top\text{-R} \qquad \frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A},A;\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};A,\mathcal{B};\Upsilon}\ \textit{atomic}\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B,\mathcal{B};\Upsilon \quad \Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};C,\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B\&C,\mathcal{B};\Upsilon}\ \&\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\bot,\mathcal{B};\Upsilon}\ \bot\text{-R} \qquad \frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B,C,\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B\parr C,\mathcal{B};\Upsilon}\ \parr\text{-R}$$

$$\frac{\Sigma\colon \Psi;\Delta,B \Longrightarrow \mathcal{A};C,\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B\multimap C,\mathcal{B};\Upsilon}\ \multimap\text{-R} \qquad \frac{\Sigma\colon \Psi,B;\Delta \Longrightarrow \mathcal{A};C,\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B\supset C,\mathcal{B};\Upsilon}\ \supset\text{-R}$$

$$\begin{array}{c}\dfrac{y:\tau,\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};B[x\mapsto y],\mathcal{B};\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\forall_\tau x B,\mathcal{B};\Upsilon}\ \forall\text{-R}\\ y \text{ is not declared in } \Sigma\end{array} \qquad \frac{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};B,\Upsilon}{\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};?B,\mathcal{B};\Upsilon}\ ?\text{-R}$$

Figure 2.4: The modified 'right' rules in $\mathcal{F}'$.

2.5 An Operational Reading of the Rules in $\mathcal{F}'$

The proof construction using SEARCH''', as the name implies, is not completely determined: there is "don't know" and "don't care" non-determinism. Moving from the non-deterministic proof search of SEARCH''' to a deterministic proof search that can be implemented is a difficult

[7] Not to be confused with the B that stands for the box calculus.

$$\frac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B; B, \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \emptyset; B, \Upsilon}\ \mathit{choose?}$$

$$\frac{\Sigma\colon \Psi; \Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; B, \Delta \Longrightarrow \mathcal{A}; \emptyset; \Upsilon}\ \mathit{choose} \qquad \frac{\Sigma\colon B, \Psi; \Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon B, \Psi; \Delta \Longrightarrow \mathcal{A}; \emptyset; \Upsilon}\ \mathit{choose!}$$

Figure 2.5: The 'choose' rules in $\mathcal{F}'$.

$$\frac{}{\Sigma\colon \Psi; \emptyset \overset{A}{\Longrightarrow} A; \Upsilon}\ \mathit{initial} \qquad \frac{}{\Sigma\colon \Psi; \emptyset \overset{A}{\Longrightarrow} \emptyset; A, \Upsilon}\ \mathit{initial?}$$

$$\frac{}{\Sigma\colon \Psi; \emptyset \overset{\bot}{\Longrightarrow} \emptyset; \Upsilon}\ \bot\text{-S} \qquad \frac{\Sigma\colon \Psi; \Delta \overset{B_i}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; \Delta \overset{B_1 \& B_2}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \&\text{-S}_i$$

$$\frac{\Sigma\colon \Psi; B \Longrightarrow \emptyset; \emptyset; \Upsilon}{\Sigma\colon \Psi; \emptyset \overset{?B}{\Longrightarrow} \emptyset; \Upsilon}\ ?\text{-S} \qquad \frac{\Sigma\colon \Psi; \Delta \overset{B[x \mapsto t]}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; \Delta \overset{\forall_\tau x B}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \forall\text{-S}$$

t is a Σ-term of type τ

$$\frac{\Sigma\colon \Psi; \Delta_1 \overset{B}{\Longrightarrow} \mathcal{A}_1; \Upsilon \quad \Sigma\colon \Psi; \Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi; \Delta_1, \Delta_2 \overset{B \parr C}{\Longrightarrow} \mathcal{A}_1, \mathcal{A}_2; \Upsilon}\ \parr\text{-S}$$

$$\frac{\Sigma\colon \Psi; \Delta_1 \Longrightarrow \mathcal{A}_1; B; \Upsilon \quad \Sigma\colon \Psi; \Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi; \Delta_1, \Delta_2 \overset{B \multimap C}{\Longrightarrow} \mathcal{A}_1, \mathcal{A}_2; \Upsilon}\ \multimap\text{-S}$$

$$\frac{\Sigma\colon \Psi; \emptyset \Longrightarrow \emptyset; B; \Upsilon \quad \Sigma\colon \Psi; \Delta \overset{C}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; \Delta \overset{B \supset C}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \supset\text{-S}$$

Figure 2.6: The *stoup* rules in $\mathcal{F}'$.

task. This section is focused on the choices which have to be made during a construction of a proof and on further constraints which have to be imposed on FORUM for its implementation.

When "don't care" non-determinism occurs, an arbitrary choice can be made and the final result is independent of this choice. On the other hand, when "don't know" non-determinism occurs, a specific choice has to be made out of many possibilities, and the final result is dependent on this choice. Consequently, all possibilities must be explored in order to find the choice that leads to the desired result. The extended uniform proof condition is used in the proof search of FORUM in order to minimise the non-determinism or to transform the "don't know" into a "don't care" non-determinism.

In logic programming where a sequence of choices has to be made, the exploration of the choices is traditionally carried out with a *depth-first* search. Suppose, all possible choices are represented in a tree where the nodes represent the choices and the edges represent the causal relations between the choices. Then, a depth-first search makes a choice which is "deeper" (i.e, the next choice is made out of the most recent choice) whenever possible. In case no choice can be made, the search "backtracks" and considers other choices. By contrast, the *breadth-first* search "discovers" every possible choice which can be made out of the most recent one. However in the average-case, the depth-first search leads to a result faster (to a proof) than the breadth-first search. Furthermore, the *breadth-first* search requires too much space. That is why the usage of a depth-first search is usually an efficiency criterion for a logic programming language.

There are two kinds of choices, called *conjunctive* and *disjunctive* choice. A disjunctive choice occurs when one possibility has to be chosen out of many; on the other hand, a conjunctive choice occurs when several possibilities have to be chosen. However, the chosen possibilities have to appear in a certain order. The disjunctive choices have to be made in Step 1 and 1' of SEARCH'''—the conjunctive choices in Step 2.

In Step 1, a disjunctive choice has to be made as to which goal formula will be analysed subsequently. As mentioned earlier, this choice can be made efficiently because of the extended uniform proof condition. The first major source of troublesome disjunctive choices occur in Step 1', when a 'choose' rule has to be applied. FORUM is implemented so that it attempts to prove the corresponding sequent where the choose rules are applied in the following order: '*choose*', '*choose*!', '*choose*?. This order is due to the proviso of linear proofs where all linear resources (i.e., formulae in the linear contexts) must be "consumed"[8].

In case of the '*choose*' and '*choose*!' rule, a program formula has to be selected that goes into the *stoup* and will be analysed subsequently. The program formulae are ordered[9] and they are selected according to this order. The interplay of this method and the depth-first search causes problems and can compromise the completeness. Consider the following sequent where the program formulae are ordered from the left to the right (it is assumed that Σ contains the appropriate declarations):

$$\Sigma: p\multimap p, p; \emptyset \Longrightarrow p; \emptyset; \emptyset$$

The implemented proof search of FORUM fails in this case because it selects always the formula $p\multimap p$ first as the following proof fragment shows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Sigma: p\multimap p, p; \emptyset \Longrightarrow \emptyset; p; \emptyset} \quad \dfrac{}{\Sigma: p\multimap p, p; \emptyset \xRightarrow{p} p; \emptyset}\,initial}{\Sigma: p\multimap p, p; \emptyset \xRightarrow{p\multimap p} p; \emptyset}\,\multimap\text{-S}}{\Sigma: \underline{p\multimap p}, p; \emptyset \Longrightarrow p; \emptyset; \emptyset}\,choose!}{\Sigma: p\multimap p, p; \emptyset \Longrightarrow \emptyset; p; \emptyset}\,atomic\text{-R} \qquad \dfrac{}{\Sigma: p\multimap p, p; \emptyset \xRightarrow{p} p; \emptyset}\,initial}{\dfrac{\Sigma: p\multimap p, p; \emptyset \xRightarrow{p\multimap p} p; \emptyset}{\Sigma: \underline{p\multimap p}, p; \emptyset \Longrightarrow p; \emptyset; \emptyset}\,choose!}\,\multimap\text{-S}$$

However, a proof exists in FORUM and can be constructed if the second program formula is selected. It would lead to the rather simple proof:

$$\dfrac{\dfrac{}{\Sigma: p\multimap p, p; \emptyset \xRightarrow{p} p; \emptyset}\,initial}{\Sigma: p\multimap p, p; \emptyset \Longrightarrow p; \emptyset; \emptyset}\,choose!$$

Consequently, the implemented proof search is incomplete with respect to the proofs in FORUM, and it is the responsibility of the programmer to provide the appropriate order of the program formulae.

Also, the '&-S$_i$' rule causes a disjunctive choice in Step 1' in a root upward search. This rule stands essentially for two rules that have the same conclusion but different premises.

[8]The '*choose*? is considered as last because it introduces new formulae into the $\mathcal{B}$ context. On the other hand, the '*choose*' rule is applied first because it "consumes" eventually the *stoup* formula from the context Δ and a formula from the context $\mathcal{A}$ if the proof branch terminates, for example, with the '*initial*' rule.

[9]The sequents in $\mathcal{F}'$ are represented using multisets for more convenience in the following proofs. However, they are implemented based on a list structure.

When the '&-S$_i$' rule is applied in the proof search, a component of the *stoup* formula has to be chosen that will be analysed subsequently. The implementation of FORUM attempts at first to find a proof with the first component of the &-formula and, after failing, to find one with the second component. (The disjunctive choice that arises when the contexts of the conclusion must be split is addressed in Section 3.)

The conjunctive choices occur in Step 2 of SEARCH$'''$; the inference rules of FORUM which are selected in Steps 1 and 1' are as follows:

$$\frac{\sigma_1 \ldots \sigma_n}{\sigma}$$

where:

- the premise-free rules have $n = 0$;
- the proof branching rules have two premises (i.e. $n = 2$);
- otherwise, the rules have one premise (i.e. $n = 1$).

In the case where n=2, a conjunctive choice must be made as to which premise is attempted to be proven first. A naïve ordering can cause problems and the proof construction fails because of a loop that could be avoided by a different ordering of the premises. Suppose the following sequent:

$$\Sigma : p \supset p; \emptyset \Longrightarrow p, q; \emptyset; \emptyset.$$

The sequent is not provable, but the depth-first search and the left-to-right order of the premises in the '⊃-S' inference rule results in a loop rather than giving the answer: "not provable". In this particular case, a naïve implementation of FORUM behaves as the following proof fragment illustrates:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Sigma : p \supset p; \emptyset \Longrightarrow p, q; \emptyset; \emptyset} \quad \Sigma : p \supset p; \emptyset \overset{p}{\Longrightarrow} p, q; \emptyset}{\Sigma : p \supset p; \emptyset \overset{p \supset p}{\Longrightarrow} p; \emptyset}\ \supset\text{-S}}{\Sigma : p \supset p; \emptyset \Longrightarrow p, q; \emptyset; \emptyset}\ \mathit{choose!}$$

It attempts to prove the sequent on the left-hand side and finds always a program formula which is applicable. An attempt to prove the sequent on the right-hand side first would lead directly to the desired result that the endsequent is not provable. In our implementation, the order of the premises is changed such that the *stoup* sequent is proven at first before the non-*stoup* sequent is proven because the proof search for a *stoup* sequent is more restricted (focused) than the proof search for a non-*stoup* sequent

Proof search in the implementation is carried out in two modes. One mode deals with non-*stoup* sequents and is focused on the goal formulae; the other mode deals with *stoup* sequents and analyses the formula in the *stoup*. Therefore, the modes are called *right* and *left* mode. The proof search starts in the right mode. The '*choose*' and '*choose!*' rules switch from the right mode into the left mode. The left rules that analyse an implication switch from the left mode into the right mode when they attempt to prove the premise that is a non-*stoup* sequent. The '?-S' switches from the left mode into the right mode as well, otherwise FORUM would become incomplete. Consider the following proof [Miller, 1996]:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma\colon \emptyset;\emptyset \overset{p}{\Longrightarrow} p;p}\;\textit{initial}}{\Sigma\colon \emptyset;p \Longrightarrow p;\emptyset;p}\;\textit{choose}}{\Sigma\colon \emptyset;p \Longrightarrow \emptyset;p;p}\;\textit{atomic}}{\Sigma\colon \emptyset;p \Longrightarrow \emptyset;\emptyset;p}\;\textit{choose}?}{\Sigma\colon \emptyset;\emptyset \overset{?p}{\Longrightarrow} \emptyset;p}\;?\text{-S}}{\Sigma\colon \emptyset;?p \Longrightarrow \emptyset;\emptyset;p}\;\textit{choose}}{\Sigma\colon \emptyset;?p \Longrightarrow \emptyset;?p;\emptyset}\;?\text{-R}$$

where Σ contains the appropriate declarations. A lower application of the '*choose*?' rule does not lead to a proof because the '?-S' rule is only applicable if the linear contexts are empty.

In what follows, the formulae contain only primitive connectives (a formula which contains a non-primitive connective can be replaced by its logically equivalent formula).

Right mode

Proof search while in the right mode analyses a sequent that is as follows:

$$\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};\Upsilon,$$

and a formula B is selected from $\mathcal{B}$. In the case where B is:

A	and A is atomic; the '*atomic*-R' rule is applied which moves A from $\mathcal{B}$ to $\mathcal{A}$;
$\top$	the '$\top$-R' rule is applied; this terminates the proof branch;
$B \& C$	the '&-R' rule is applied and it is attempted to prove both sequents where the formula $B\&C$ is replaced by B and C, respectively;
$\bot$	the '$\bot$-R' rule is applied which removes the '$\bot$' connective from $\mathcal{B}$;
$B \parr C$	the '$\parr$-R' rule is applied and the formula $B\&C$ is replaced by B, C in $\mathcal{B}$;
$B \multimap C$	the '$\multimap$-R' rule is applied and the implicans B is removed to Δ and $B\multimap C$ is replaced by C in $\mathcal{B}$;
$B \supset C$	the '$\supset$-R' rule is applied and the implicans B is removed to Υ and $B\multimap C$ is replaced by C in $\mathcal{B}$;
$?B$	the '?-R' rule is applied and the formula $?B$ is removed to Υ (without the '?' connective);
$\forall_\tau x B$	the '$\forall$-R' rule is applied and the signature Σ is augmented by a fresh variable y of type τ and the variable x is replaced by y in the formula B.

In case the multiset $\mathcal{B}$ is empty, the following three rules are applicable:

choose a formula B is chosen from Δ and removed into the *stoup*, the mode of proof search switches into the left mode and the following sequent (Δ' is the result of removing B from Δ) is analysed:

$$\Sigma\colon \Psi;\Delta' \overset{B}{\Longrightarrow} \mathcal{A};\Upsilon;$$

choose! a formula B is chosen from Ψ and copied into the *stoup*, the mode of proof search switches into the left mode and the following sequent is analysed:

$$\Sigma \colon \Psi; \Delta \xrightarrow{B} \mathcal{A}; \Upsilon;$$

choose? a formula B is chosen from Υ and copied to $\mathcal{B}$ and is analysed by a 'right' rule.

Left Mode

Proof search in the left mode analyses a sequent that is as follows:

$$\Sigma \colon \Psi; \Delta \xrightarrow{B} \mathcal{A}; \Upsilon,$$

and the formula B is considered. In the case where B is:

A and A is atomic, and Δ is empty; the proof terminates with the '*initial*' rule if $\mathcal{A}$ is just the formula A;

A A is atomic, and Δ and $\mathcal{A}$ are empty; the proof terminates with the '*initial*?' rule if A is contained in Υ;

$\bot$ the proof terminates with the '$\bot$-S' rule if $\mathcal{A}$ and Δ are both empty;

$B\&C$ the '&-S' rule is applied; either B or C is chosen, and $B\&C$ is replaced by the chosen formula;

$?B$ if $\mathcal{A}$ and Δ are empty then the '?-S' rule is applied; $?B$ is removed from the *stoup*, and Δ is set to $\{\!|B|\!\}$; the mode of proof search switches into the right mode, and the following sequent is analysed:

$$\Sigma \colon \Psi; B \Longrightarrow \emptyset; \emptyset; \Upsilon;$$

$\forall_\tau x B$ the '$\forall$ -S' rule is applied and the variable x in the *stoup* formula B is substituted by a term t of the type τ;

$B \parr C$ the '$\parr$-S' rule is applied; Δ and $\mathcal{A}$ are split into the multisets $\mathcal{A}_1$, $\mathcal{A}_2$ and Δ_1, Δ_2, respectively; both of the following sequents must be proven:

$$\Sigma \colon \Psi; \Delta_1 \xrightarrow{B} \mathcal{A}_1; \Upsilon$$

$$\Sigma \colon \Psi; \Delta_2 \xrightarrow{C} \mathcal{A}_2; \Upsilon;$$

$B \multimap C$ the '$\multimap$-S' rule is applied; Δ and $\mathcal{A}$ are split into the multisets $\mathcal{A}_1$, $\mathcal{A}_2$ and Δ_1, Δ_2, respectively; the following sequent must be proven in the left mode:

$$\Sigma \colon \Psi; \Delta_2 \xrightarrow{C} \mathcal{A}_2; \Upsilon;$$

and the following in the right mode:

$$\Sigma \colon \Psi; \Delta_1 \Longrightarrow \mathcal{A}_1; B; \Upsilon$$

$B \supset C$ the '$\supset$-S' rule is applied; the following sequent must be proven in the left mode:

$$\Sigma \colon \Psi; \Delta \xrightarrow{C} \mathcal{A}; \Upsilon;$$

and the following in the right mode:

$$\Sigma \colon \Psi; \emptyset \Longrightarrow \emptyset; B; \Upsilon.$$

2.6 The Approach in Lygon

The logic programming language Lygon relates to FORUM in two ways; this section is focused on one of them, the foundations of the logic and proof search in Lygon (the other, the context management, will be compared with the present approach in Section 5.3). Lygon is based on a systematic investigation of goal-directed proofs in CLL as given in [Pym & Harland, 1994]. The complete language is introduced first in order to illustrate its principles and working assumptions. However in the later part of this section, the presentation is restricted to a subset of Lygon which represents the actual implementation.

For the multiple succedents in Lygon the notion of goal-directed proof is characterised by the set of *simple locally LR proofs*. This notion is slightly weaker than the notion of goal-directed proof in intuitionistic logics where any non-atomic goal formula must be decomposed by a right rule. The simple locally LR proofs allow certain occurrences of left rules which lie under right rules and cannot be permuted over right rules (amongst these left rules are the '$\multimap$-L', '$\otimes$-L', and 'C!-L' rules).

The two-sided sequents in Lygon are as introduced in Definition 1 (Page 12). It is proven that for the following two classes, goal formulae G and program formulae D,

$$G ::= A \mid \mathbf{1} \mid \bot \mid \top \mid G\&G \mid G \otimes G \mid G⅋G \mid G \oplus G \mid \forall x G \mid \exists x G \mid !G \mid ?G \mid D \multimap G$$

$$D ::= A \mid \mathbf{1} \mid \bot \mid D\&D \mid D \otimes D \mid D⅋D \mid \forall x D \mid !D \mid G \multimap A \mid G \multimap \bot$$

there exists for each provable sequent a simple locally LR proof (see Corollary 2.10, Lemma 2.15 and Proposition 2.14 in [Pym & Harland, 1994]).

A calculus is introduced for this fragment of CLL; the inference rules in this calculus are designed so that a notion of goal-directed proof search is meaningful. For example, a formula $p \otimes q$ occurring in the antecedent will be replaced by the components p and q, and thus, a goal-directed proof exists in this calculus for the sequent of Display 2.2 (Page 16). All program formulae in the calculus are transformed into a clause form. (The reader is referred to Definition 3.1 in [Pym & Harland, 1994] for the complete definition of the clausal decomposition.) As a result of this clause form, it is possible to replace all left rules by a single rule, called *backchaining rule* (it is called originally 'resolution' rule).

In what follows, a fragment of Lygon is considered which exists as implementation (see [Harland & Winikoff, 1996b]). The reason for the restriction is the fact that for this implementation exists an approach on the selection of the active goal formula. This approach is of most interest with respect to the approach in FORUM. The calculus for this fragment of Lygon is formulated as a single-sided calculus. The sequents are defined as follows:

$$\vdash F_1, \ldots F_n$$

Negated formulae are permitted as goal formulae and a 'backchaining' rule is as follows:

$$\frac{\vdash ?\Delta, G, \Gamma}{\vdash ?\Delta, A, \Gamma}\ BC$$

where a program formula is of the form $\forall x_1 \ldots \forall x_n (A \leftarrow G)$. The goal formulae are defined as follows:

$$G ::= A \mid A^{\perp} \mid \mathbf{1} \mid \bot \mid \top \mid G_1\&G_2 \mid G_1 \otimes G_2 \mid G_1⅋G_2 \mid G_1 \oplus G_2 \mid \forall x G \mid \exists x G \mid !G \mid ?G$$

The clauses are of the form:

$$\forall x_1 \ldots \forall x_n (A \leftarrow G).$$

The single-sided calculus defined above represents a multiple succedent logic. Therefore, the proof search has to select a goal formula which will be analysed subsequently. As illustrated in the proof on Page 17, there may be interdependencies between the selections. FORUM is designed so that the selection can be done with "don't care" non-determinism. This yields to the rather restricted logic which is based on the formulae using FORUM's primitive connectives. In Lygon, the opposite decision was made. As a result, Lygon permits a larger set of primitive connectives, but the selection of a goal formula cannot be made using "don't care" non-determinism and may require backtracking. However, some formulae can be analysed safely by "don't care" non-determinism. These formulae have as outermost connective those which are identified in [Andreoli, 1992] as 'asynchronous' connectives (see Page 18). The remaining connectives of Lygon are those which Andreoli called 'synchronous' connectives. The selection of a goal formula is illustrated by the following two Lygon proofs[10]:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash q^\perp, q}\,Ax}{\vdash q^\perp, \perp, q}\,\perp}{\vdash q^\perp \parr \perp, q}\,\parr}{\vdash q^\perp \parr \perp, p \oplus q}\,\oplus \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash q^\perp, q}\,Ax}{\vdash q^\perp, p \oplus q}\,\oplus}{\vdash q^\perp, \perp, p \oplus q}\,\perp}{\vdash q^\perp \parr \perp, p \oplus q}\,\parr$$

Since $\parr$ is a asynchronous connective, there exists a proof where the corresponding formula is decomposed first. As a result, whenever a formula with an outermost asynchronous connective occurs, it can be selected with the efficient "don't care" non-determinism.

To sum up the comparison of FORUM and Lygon, the main difference is that in FORUM any right rule must permute over the other right rules, and in contrast, in Lygon only some right rules need to permute over other right rules. However, both languages can be seen to use in essence a very similar proof search strategy. Lygon decomposes first all formulae with an asynchronous connective as outermost connective in a sequent with "don't care" non-determinism. Next, Lygon decomposes all formulae with a synchronous connective using "don't know" non-determinism and then Lygon backtracks over all choices. FORUM, however, decomposes the formulae in the succedent with "don't care" non-determinism, but it translates each formula with a synchronous connective to a logically equivalent formula which places some components in the antecedent during proof search. The formulae in the antecedent are analysed with "don't know" non-determinism.

2.7 Types and Terms in FORUM

Miller founded the underlying formal system of FORUM on a typed λ-calculus similar to that used in λProlog. It is based on two syntactic components (simple types and λ-terms) and is derived from Church's formulation of the simple theory of types [Church, 1951]. The groundwork of this system is well-studied for λProlog for which it is invaluable. Apart from some basic principles which will be introduced below, the reader is referred to the work

[10] The examples come from the paper [Harland & Winikoff, 1996b]. They are slightly changed because the originals require unification which is not yet addressed in this thesis.

([Miller & Nadathur, 1986], [Miller, 1989a] and [Nadathur & Miller, 1994]) on λProlog for a more detailed treatment. The section concludes with a presentation of the concrete syntax employed for FORUM.

2.7.1 Kinds and Types

The terms in FORUM are built from typed constants and variables. The types that are employed in FORUM are first-order types referred to as *simple types*. The first-order types are built from *primitive types*, *type constructors*, *type variables* and *functional types*. The *kind declaration* gives to each primitive type and type constructor an arity. As examples, consider the declarations:

```
kind o      type.
kind int    type.
kind list   type -> type.
```

where the -> operator associates to the right; nested occurrences of this operator of the form `(A -> B) -> C` are not permitted. In the examples above, the identifiers `o` and `int` are names of primitive types; `list` is the name of a type constructor. Each type is associated with an arity which is a non-negative integer and which is one less than the occurrences of `type` in the corresponding kind declaration. That means, the primitive types `o` and `int` have the arity 0 and the type constructor `list` has the arity 1.

Definition 6 *The* types *in* FORUM *are inductively defined as:*

- *primitive types;*
- *type expressions (written as $c\ \sigma_1 \ldots \sigma_n$ $(n \geq 1)$ where c is a type constructor of arity n and σ_i are types);*
- *type variables (written with a capital letter) and*
- *functional types (written as $\sigma \rightarrow \tau$ where σ and τ are types).*

The functional type constructor $\rightarrow$ is right associative; parentheses can be used in order to avoid ambiguities. The type variables allow polymorphism in the usual way. Each type σ can be written as:

$$\sigma_1 \rightarrow \ldots \sigma_n \rightarrow \tau$$

where τ is either a primitive type, a type expression or a type variable. The type τ is called the *target type* of σ; the types σ_i are called the *argument types* of σ. As examples, consider the types below[11]:

```
o, int, int -> int -> int, int -> o, A -> list A -> o.
```

Following Church, Miller chose the primitive type `o` for FORUM's formulae. A type which has the target type `o` and in the argument types there is no occurrence of the type `o` is called a *predicate type*; it is used for typing a *predicate*.

[11]The functional operator $\rightarrow$ is written as ->. From the context it is always clear whether the operator for functional types or the operator in the kind expressions is meant.

The terms of FORUM (see next section) are built upon *free* and *bound* variables and *constants*. A type must be given to each constant using a *type declaration*. (The type of a variable is inferred from the context.) The following examples are type declarations that declare the *non-logical* constants `a, f, p` and `member`:

```
type a        int.
type f        int -> int -> int.
type p        int -> o.
type member   A -> list A -> o.
```

FORUM's primitive connectives are declared using the primitive type `o`: the type of formulae. These connectives are represented using the following ASCII-sequences:

FORUM's primitive connectives	$\top$	$\bot$	$\parr$	$\&$	$\multimap$	$\supset$	?	$\forall$
ASCII-sequences	`top`	`bot`	`|`	`@`	`--o`	`==>`	`?`	forall

The connectives (or *logical constants*) are declared as follows:

```
type top        o.
type bot        o.
type |          o -> o -> o.
type @          o -> o -> o.
type --o,o--    o -> o -> o.
type ==>,<==    o -> o -> o.                    (2.5)
type ?          o -> o.
type forall     (A -> o) -> o.
type x          o -> o -> o.
type neg        o -> o.
```

The logical constants `x` and `neg` are declared in addition to the primitive connectives and represent the non-primitive connective $\otimes$ and the negation (see Page 19), respectively. (They are used in some example programs.) The reverse implications are declared for a better readability of FORUM programs. The logical constants `|`, `@`, `--o`, `o--`, `==>`, `<==` and `x` are the only constants which are written as infix symbols[12]. (The constants `|`, `&` and `x` bind more tightly than the implications. However, parentheses should be used in order to avoid ambiguities.) The constants `?`, `forall` and `neg` are written as prefix symbols.

Definition 7 *A* signature *is a list of kind and type declarations and is defined inductively:*

1. *A list that consists of the kind declaration for the simple type* o *and the type declarations for the logical constants of Display 2.5 is a signature.*

2. *If Σ is a signature, k is a kind declaration that declares a type τ and τ is not declared in Σ, then Σ, k is a signature.*

3. *If Σ is a signature, τ is declared by a kind declaration in Σ, t is a type declaration that declares a constant c with a type τ and c is not declared in Σ, then Σ, t is a signature.*

[12]The symbol for abstraction is also written as infix symbol, but it is neither a logical nor a non-logical constant.

2.7.2 Simply Typed λ-Terms

The λ-terms in FORUM are constructed by two operations: *abstraction* and *application*. Following [Nadathur & Miller, 1994], the definition of the λ-terms associated with a type is as follows:

Definition 8 *A* simply typed λ-term *(or short* term*) is inductively defined:*

- *a constant or a variable of type σ is a term with the type σ;*
- *if x is a variable of type σ and F is a term of type τ then $(\lambda x.F)$ is a term of type $\sigma \rightarrow \tau$: this operation is called* abstraction*; the variable x is bound and its scope is F with λx as its* binder*;*
- *if F_1 is a term of type $\sigma \rightarrow \tau$ and F_2 is a term of type σ then $(F_1 F_2)$ is a term of type τ; this operation is called* application.

The term $(\lambda x.F)$ constructed by an abstraction is written in an ASCII-style as: `(X\F)`. λProlog and FORUM use a *curried* syntax form for λ-terms. Suppose, the constants `a` and `b` are declared with the type σ and a predicate `h` is declared with the functional type $\sigma \rightarrow \sigma \rightarrow$ `o`. Then, it is possible to form the λ-term `(h a)` by applying the constant `a` to the term `h`. This new term is of the functional type $\sigma \rightarrow$ `o`. A further application using the constant `b` to this term leads to the term `((h a) b)` with the type `o`. The last term can be shortly written as `(h a b)` (i.e., the application is left associative). A similar term using a Prolog convention would look as $h(a, b)$.

The λ-terms are convenient for distinguishing between occurrences of *bound* and *free variables*. As introduced above, an occurrence of a variable x is bound if it occurs in a scope of a term that is abstracted using the variable x; all non-bound variable occurrences in a term are free. Consequently, there are three syntactic categories which have to be distinguished: constants, free variables and bound variables. By convention, the constants are written starting with a lower-case letter; the free variables are written starting with an upper-case letter; the bound variables can be written starting with either a lower or an upper-case letter because they can be identified by their binders.

Definition 9 *A* substitution *for a free variable x of type σ in a term s (where term t is substituted for a variable x of the same type σ) is written as $s[x \mapsto t]$ where all free variables of t are distinct from the bound variables in s. A substitution may be viewed as a type preserving mapping on variables that is the identity everywhere except on the variable x which is mapped on the term t.*

In [Miller, 1991], three conversion rules over λ-terms are defined as follows:

Definition 10 *Conversion rules for λ-terms*

1. *A term $\lambda x.s$ α-converts to the term $\lambda y.s[x \mapsto y]$ provided that x is free in s and y is not free in s.*
2. *A term $(\lambda x.s)t$ β-converts to the term $s[x \mapsto t]$ provided that the free variables of t are not bound in s.*
3. *A term $\lambda x.(s\ x)$ η-converts to s, provided that x is not free in s.*

Two terms s and t are regarded as equivalent if there is a sequence of conversions that transforms s into t. The reader is referred to [Nadathur & Miller, 1994] for a detailed treatment of a unification algorithm and further restrictions imposed on λ-terms in order to obtain a language where an implementation is feasible. Since we implement our calculi in Terzo, the unification, for example, is already provided by this language.

2.7.3 Definition of Forum's Syntax

In the following, a BNF-style grammar is introduced for Forum. The syntactic category of identifiers is divided into two classes: lower-case identifier (`lcid`) and upper-case identifier (`ucid`). The other words written using a `typewriter font` are the reserved words, and as such, may not be used as identifiers.

The types are declared using kind declarations:

```
kind_decl  ::=  kind id_list arity.

id_list    ::=  lcid
             |  lcid, id_list

arity      ::=  type
             |  type -> arity
```

The constants are declared using type declarations:

```
type_decl  ::=  type id_list type.

type       ::=  type -> type
             |  type   type
             |  ( type )
             |  lcid | ucid
```

The terms are specified as:

```
term       ::=  term   term
             |  term   infix_op   term
             |  prefix_op   term
             |  ( term )
             |  top | bot
             |  lcid | ucid

infix_op   ::=  ⅋ | @ | x | --o | o-- | ==> | <==
             |  \ (abstraction)

prefix_op  ::=  forall
             |  neg
             |  ?
```

Note '⅋' is written as '|' in typewriter font, but appears as itself in the grammar rule to avoid confusion with the BNF '|'.

The goal and program formulae are specified as:

goal_frm	`::=`	*term*
	|	*goal_frm* , *goal_frm*
program_frm	`::=`	*term*
	|	`linear` *term*
	|	*program_frm* . *program_frm*

In order to separate goal and program formulae, a comma is used for goal formulae and a period for program formulae. The reserved word `linear` indicates which context accommodates the corresponding formula.

Consider the following simple example (proposed in [Große et al., 1992]; see also Section 4.2). The constants `d`, `q` and `l` stand for a dollar, a quarter and a lemonade, respectively. The program is as follows:

```
type d,q,l o.
d --o q x q x q x q
q x q x q --o l
linear d.
linear c.
```

The intended meaning of the implications is getting four quarters for a dollar and getting a lemonade for three quarters. The last two lines of the program mean having a dollar and quarter. A goal could be as follows:

```
l x q x q
```

which stands for having a lemonade and two quarters. The program and the goal translates into the following FORUM sequent[13]:

$$\Sigma, \{d, q, l : o\}:\ d \multimap q \otimes q \otimes q \otimes q,\ q \otimes q \otimes q \multimap l\ ;\ d, q \Longrightarrow \emptyset;\ l \otimes q \otimes q\ ;\emptyset$$

In what follows, free variables which occur in a program formula are assumed to be universal quantified. The quantifiers are omitted in order to improve the readability. The free variables in a goal formula are essential existential quantified, i.e., the proof system looks for an instantiation of such variables.

[13] Σ contains all declarations for the connectives.

Chapter 3

Efficient Context Management

3.1 Input-Output Model and the Box Calculus $\mathcal{B}$

The multiplicative rules present an implementation challenge because the context splitting is a rather costly operation during proof search. The first work concerning the context splitting in Lolli appeared in [Hodas & Miller, 1991]. Let us restate the problem in terms of the proof system $\mathcal{F}'$. There, the '$\wp$-S' rule, for example, is presented as follows:

$$\frac{\Sigma\colon \Psi; \Delta_1 \overset{\mathcal{B}}{\Longrightarrow} \mathcal{A}_1; \Upsilon \quad \Sigma\colon \Psi; \Delta_2 \overset{\mathcal{C}}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi; \Delta_1, \Delta_2 \overset{\mathcal{B}\wp\mathcal{C}}{\Longrightarrow} \mathcal{A}_1, \mathcal{A}_2; \Upsilon}\ \wp\text{-S}$$

The rule is problematic since in the root-upward proof construction of SEARCH''' the contexts are not split, respectively, into Δ_1, Δ_2 and $\mathcal{A}_1, \mathcal{A}_2$. An interpreter has to find non-deterministically a partition of the contexts that leads to a proof. The number of partitions grows exponentially with the number of formulae, and therefore, a naïve implementation of the splitting operation as a "don't know" choice is too inefficient for a proof search by a computer.

In the following, a calculus (we call it a *box calculus* ($\mathcal{B}$) since each sequent is represented as a box with lots of components) will be introduced to avoid the aforementioned problem concerning the splitting of the linear contexts. The box notation is chosen because it presents the contexts clearly separated, and it preserves the structure that the program formulae appear on the left-hand side and the goal formulae on the right-hand side[1]. The non-*stoup* and *stoup* sequents of $\mathcal{F}'$:

$$\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \mathcal{B}; \Upsilon \quad \text{and} \quad \Sigma\colon \Psi; \Delta \overset{\mathcal{B}}{\Longrightarrow} \mathcal{A}; \Upsilon$$

are replaced in $\mathcal{B}$ by boxes which are called *non-stoup boxes* and *stoup boxes*. They are as follows:

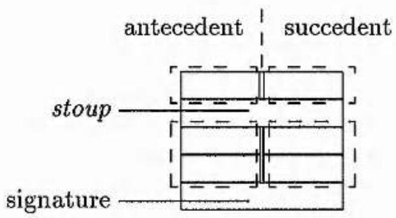
[1]The sequent formalisation of [Cervesato et al., 1996] appeared to be less suitable for FORUM with its several contexts. The representation of the boxes is similar to the sequent formalisation that is established in [Harland & Winikoff, 1996a] except that our input and output parts are arranged vertically instead of written horizontally.

Ψ	Υ	
$\mathcal{M}^0, \mathcal{M}^1$	$\mathcal{A}^0, \mathcal{A}^1$	$\mathcal{B}^0, \mathcal{B}^1$
$\mathcal{M}^1$	$\mathcal{A}^1$	$\mathcal{B}^1$
Σ		

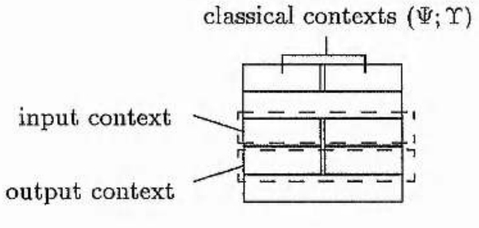
and

Ψ	Υ
B	
$\mathcal{M}^0, \mathcal{M}^1$	$\mathcal{A}^0, \mathcal{A}^1$
$\mathcal{M}^1$	$\mathcal{A}^1$
Σ	

The boxes contain an *antecedent* and a *succedent*; these are separated by double, vertical lines. The linear contexts $\mathcal{M}$ in the antecedent and the linear contexts $\mathcal{A}$, $\mathcal{B}$ are multisets of formulae ($\mathcal{A}$ contains only atomic formulae); the classical contexts Ψ and Υ are sets of formulae. According to the two types of sequent, there are two types of box representing a sequent. The boxes that represent non-*stoup* sequents have no *stoup* field; the boxes representing a sequent with a *stoup* formula B have the formula B in the *stoup*. The *stoup* boxes have similarly only an atomic context in the antecedent like the corresponding *stoup* sequents in $\mathcal{F}'$.



Each linear context of a sequent is represented as an *input context* and an *output context* in the corresponding box of the new calculus. The purpose of the input-output notion is, roughly speaking, that the resources "consumed" by a proof form the multiset difference between the input and output context of the endbox. Each input context is represented using two multisets. The first multiset with the superscript '0' represents the formulae that are "consumed" in the proof branch of a box. On the other hand, the second multiset (with superscript '1') represents the formulae that are not used in the proof branch. Therefore, this multiset forms the output context of the box.



The full system of inference rules is presented in figures 3.1, 3.2 and 3.3 (pp. 48). In order to reduce the amount of repetition, the inference rules of the box calculus are categorised into four groups. Each group represents an essential operation of the box calculus. These groups are named *passing* rules, *returning* rules, *splitting* rules and *sharing* rule according to the operation which they perform in the box calculus. The inference rules fall into the groups as follows:

passing rules:	*atomic*-R, $\bot$-R, $⅋$-R, $\multimap$-R, $\supset$-R, $\forall$-R, ?-R, *choose*, *choose*!, *choose*?, &-S$_i$, $\forall$-S, $\supset$-S;
returning rules:	$\top$-R, *initial*, *initial*?, $\bot$-S, ?-S;
splitting rules:	$⅋$-S, $\multimap$-S;
sharing rule:	&-R.

When considering these groups, it is convenient to employ an operational point of view. However, the rules themselves do not rely on such an interpretation: the rules presented in figures 3.1, 3.2 and 3.3 are fully declarative.

The passing rules. Each right rule of this group picks one formula from the input context of the conclusion. This formula will be decomposed, and the components are passed to the input context of the premise. On the other hand, the *stoup* rules analyse the formula in the *stoup*. The formulae of the input context in the conclusion are passed to the input context of the premise. Subsequently, the formulae that are returned as output context of the premise are passed to the corresponding output context of the conclusion. Some rules of this group are discussed in more detail since they possess some features which vary from this general scheme. The '*choose*' and '*choose*!' rule pass the $\mathcal{B}$ part of the input context directly to the corresponding output context of the conclusion since the proof branch cannot use any formula of it (both rules in the sequent calculus $\mathcal{F}'$ require an empty $\mathcal{B}$ context). In a similar fashion, the '*choose*?' rule returns the $\mathcal{B}$ part of input context, but it also passes the selected formula B to the input context of the premise and expects that the corresponding output context is empty. The '$\supset$-S' inference rule requires that the implicans of the stoup formula is proven with empty input and output contexts.

The returning rules. The '*initial*', *initial*?' and '$\bot$-S' rules of the sequent calculus $\mathcal{F}'$ require some empty contexts, i.e., they do not consume any resources from them. This behaviour is modelled by direct returning of the input context to the output context. The '$\top$-R' rule consumes some formulae from the input context and returns the remaining formulae to the output contexts of the conclusion. The '?-S' rule also returns directly its input context, but has a box as premise that consist of the formula B and empty input and output contexts.

The splitting rules. These rules pass the input context from the conclusion to the left premise. The returned output context of the left premise is passed to the right premise for use as input context. The returned formulae of the right premise are passed to the corresponding output contexts of the conclusion. Intuitively, the premise on the left-hand side consumes some resources from the input context and passes the remaining resources to the premise on the right-hand side. The resources that remain from both subproofs of the premises are given as output to the conclusion.

The sharing rule. For expository purposes, the associated sequent inference of $\mathcal{F}'$ will be presented:

$$\frac{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B, \mathcal{B}; \Upsilon \quad \Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B\&C, \mathcal{B}; \Upsilon}\ \&\text{-R}$$

Both premises have the same contexts Δ, $\mathcal{A}$ and $\mathcal{B}$. In resemblance to the previously presented rules and to the intended meaning of the input-output contexts, the '&-R' can be formalised as follows:

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & C,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\&C,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-R}$$

In terms of the box calculus, both premises receive the same input context (apart from the component of the analysed formula $B\&C$) and must produce the same output context. However, there might be many proofs that fail because the output contexts differ. This difference caused by consuming different formulae from the input context can only be detected after both subproofs return their output contexts. An improvement of the rule for the additive conjunction is suggested in [Cervesato et al., 1996] for Lolli. The proof of the right-hand premise receives exactly those formulae which are consumed by the proof of the left premise. Consequently, the formulae not used in the first proof are inaccessible for the second proof. Thus, the output context of the premise on the right-hand side must be empty. Accordingly, the '&-R' rule is stated as follows:

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\&C,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-R}$$

Before proceeding further in our study, it is necessary to prove the soundness and completeness of $\mathcal{B}$ with respect to the proof system $\mathcal{F}'$. Both proofs are carried out by a translation of the proofs into proofs of the other calculus.

3.1.1 Soundness

Proposition 11 (Soundness)

1. *For every proof of a box of the form:* *there exists a proof of the sequent:*

$$\begin{array}{|c||c|c|}\hline \multicolumn{1}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta,X & \mathcal{A},Y & \mathcal{B},Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \Sigma : \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};\Upsilon\ .$$

and

2. *For every proof of a box of the form:* *there exists a proof of the sequent:*

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta,X & \mathcal{A},Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \qquad \Sigma : \Psi;\Delta \overset{B}{\Longrightarrow} \mathcal{A};\Upsilon\ ,$$

The proposition is divided into two parts according to the two kinds of boxes and sequents. However, in a proof of the first part there might be occurrences of boxes of the second kind and *vice versa.* That is why, the proof will be shown by a simultaneous induction on the height of proofs.

Proof: Simultaneous induction on the heights of the proofs. The transformations of the inference rules are given in Appendix A.1.

3.1.2 Completeness

For the convenience of the completeness proof, a technical result is proven first. Intuitively, it is possible to add arbitrary multisets of formulae to both input and output context, and this does not affect the provability in the box calculus. If necessary, the signature Σ had to be augmented in an appropriate way in order to type the additional formulae. However in the translations given below, the signatures are translated in such a way that formulae are well-typed. Therefore, we omit all aspects of the signatures wherever it does not lead to conflicts.

Proposition 12

1. For all multisets X, Z of formulae,
and for every multiset Y of atomic formulae,
and for every proof of a box of the form:

$$\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}$$

there exists a proof of the box:

$$\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1,X & \mathcal{A}^0,\mathcal{A}^1,Y & \mathcal{B}^0,\mathcal{B}^1,Z \\ \hline \mathcal{M}^1,X & \mathcal{A}^1,Y & \mathcal{B}^1,Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}$$

and vice versa, *and*

2. For every multiset X of formulae,
and for every multiset Y of atomic formulae,
and for every proof of a box of the form:

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

there exists a proof of the box:

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^1,X & \mathcal{A}^0,\mathcal{A}^1,Y \\ \hline \mathcal{M}^1,X & \mathcal{A}^1,Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

and vice versa.

Proof: Simultaneous induction on the heights of the proofs. The transformations are given in Appendix A.2.

Remark: Notice that in each part of the proposition both boxes represent the same sequent. The variables in the sequent calculus $\mathcal{F}'$ stand for multisets of formulae which are "consumed" in the corresponding proof branch. The simultaneous augmentation of both the input and output contexts protects the consumption of an additional formula in the modified proof branch. The corresponding sequents are:

$$\begin{array}{ll} 1. & \Sigma : \Psi; \mathcal{M}_0 \Longrightarrow \mathcal{A}_0; \mathcal{B}_0; \Upsilon \\ 2. & \Sigma : \Psi; \mathcal{M}_0 \overset{B}{\Longrightarrow} \mathcal{A}_0; \Upsilon \end{array}$$

The proof of the completeness will be shown by an induction on the height of proofs; the corresponding endboxes will be proven starting with an empty output context. However at the stage where a splitting rule is applied, the input and output context of the box in the left premise must be augmented using Proposition 12 in order to construct the desired box proof.

Proposition 13 (Completeness)

1. *For every proof of a sequent of the form:* *there exists a proof of the box:*

$$\Sigma\colon \Psi;\Delta \Longrightarrow \mathcal{A};\mathcal{B};\Upsilon \qquad \begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}$$

and

2. *For every proof of a sequent of the form:* *there exists a proof of the box:*

$$\Sigma\colon \Psi;\Delta \overset{B}{\Longrightarrow} \mathcal{A};\Upsilon \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

Proof: Simultaneous induction on the height of the proofs. The transformations of the trivial cases are given in Appendix A.3. The non-trivial cases are given below.

Case 1 (⅋-S): The translation is as follows:

$$\dfrac{\Sigma\colon \Psi;\Delta_1 \overset{B}{\Longrightarrow} \mathcal{A}_1;\Upsilon \quad \Sigma\colon \Psi;\Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2;\Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \overset{B⅋C}{\Longrightarrow} \mathcal{A}_1,\mathcal{A}_2;\Upsilon}\ ⅋\text{-S} \;\Rightarrow\; \dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta_1,\Delta_2 & \mathcal{A}_1,\mathcal{A}_2 \\ \hline \Delta_2 & \mathcal{A}_2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \Delta_2 & \mathcal{A}_2 \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B⅋C} \\ \hline \Delta_1,\Delta_2 & \mathcal{A}_1,\mathcal{A}_2 \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ ⅋\text{-S}$$

The difficulty in the translation step above is that the box of the left premise has a possible non-empty output context Δ_2 and $\mathcal{A}_2$. Accordingly, the given translation rules do not match with such boxes. However, a proof can be found first with an empty output context and then with the help of Proposition 12, this proof can be changed such that the endbox of the proof matches with the left premise of the '⅋-S' rule of the box calculus.

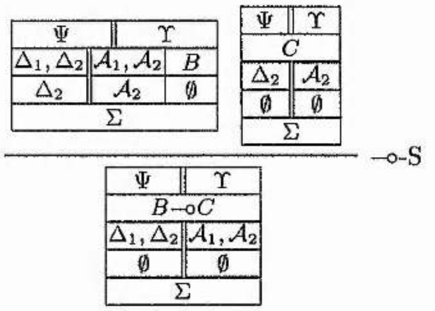
A proof of the box: is given by the proof of the box below and using Proposition 12 with $X = \Delta_2$ and $Y = \mathcal{A}_2$:

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta_1,\Delta_2 & \mathcal{A}_1,\mathcal{A}_2 \\ \hline \Delta_2 & \mathcal{A}_2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \qquad\qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta_1 & \mathcal{A}_1 \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

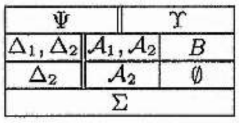
The remark to proposition 12 ensures that both box proofs have the same associated sequent proof.

Case 2 (—o-S): Similarly, the translation of the '—o-S' inference rule is given as follows:

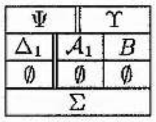
$$\frac{\Sigma\colon \Psi;\Delta_1 \Longrightarrow \mathcal{A}_1;B;\emptyset\Upsilon \quad \Sigma\colon \Psi;\Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2;\Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \overset{B\multimap C}{\Longrightarrow} \mathcal{A}_1,\mathcal{A}_2;\Upsilon} \multimap\text{-S} \Rightarrow$$



Again, the proof of the box:



is given by the proof of the box below and using Proposition 12 with $X = \Delta_2$, $Y = \mathcal{A}_2$ and $Z = \emptyset$:



3.1.3 Implementation

The box calculus $\mathcal{B}$ introduced previously is implemented in Terzo, a derivative of λProlog. The usage of Terzo (or λProlog) provides constructs which permit an implementation of the complete box calculus. If a logic programming language which, for example, lacks λ-abstraction, such as PROLOG, were used, only the propositional fragment could be implemented because these languages have no natural representation of an object-level quantification. However, the quantification over variables, functions and predicates results in a very expressive logic programming language.

FORUM allows the goal formulae to be analysed in any order. On the other hand, the program formulae need to be in a fixed order for the depth-first proof search. This order should coincide with the order of the written FORUM programs. Consequently, it is feasible to represent the used multisets of $\mathcal{B}$ by the lists provided by Terzo. However, $\mathcal{B}$ incorporates several constraints and side conditions which must be expressed adequately in the implementation.

The formulae of FORUM are implemented so that they inhabit the primitive type o. Another choice could be made; however, this type is appropriate for our purposes. The representation of the connectives and formulae is described in Section 2.7.1.

The following predicates are significant for the translation of the box calculus. Their main purpose is to check some side conditions in the implementation which are built into the box inference rules (these predicates are implemented in an auxiliary module; see B.1).

`atomic o -> o.` and `non_atomic o -> o.`

These predicates determine whether a formula is atomic or non-atomic. The `non-atomic` predicate is explicitly defined by presenting some terms with a connective as outermost constant. The `atomic` predicate succeeds if `non-atomic` fails.

`memb A -> list A -> o.` and `member A -> list A -> o.`

The predicate `memb F L` succeeds if the formula `F` is an element of the list `L`. In contrast to `member`, it will succeed only once: for the first occurrence of `F` in `L`. The predicate `member F L` also succeeds if `F` is an element of `L`. However, it succeeds as often as `F` occurs in `L`.

`membNrest A -> list A -> list A -> o.`

This predicate owes its name from "member-and-rest". As the name implies, it picks up a formula from the list and returns the remaining list as an argument.

`removed A -> list A -> list A -> o.`

This predicate is important for maintaining the soundness of the implementation relative to $\mathcal{B}$. The components of a formula that is analysed by right rules must not appear in the output context. Thus, the predicate serves for checking the number of a certain formula's occurrences in the input context which must be no less than the number in the output context.

`diff list A -> list A -> list A -> o.`

The `diff` predicate computes the multiset difference, i.e., it deletes all elements occurring in the smaller multiset (list) in the bigger multiset (list). In the implementation, the order of the arguments is fixed (bigger list, smaller list, result list).

`split list A -> list A -> list A -> o.`

The predicate `split` generates all possible two partitions of a context. It is used for a subsequent test if it is a partition which leads to a proof. Therefore, the predicate is part of a non-deterministic (and inefficient) generate-and-test algorithm.

The implementation of the 'choose' rules will be addressed separately in Section 3.3. The signature part of the boxes is completely embedded into the variable and quantification system of λProlog. Therefore, the signature Σ vanishes in the implementation. The provability of the two types of box are implemented with the predicates *right* and *stoup*, respectively. These predicates have the following types:

```
type right  list o ->                  % Ψ (classical)
            list o -> list o ->        % M (linear)
            list o -> list o ->        % A (linear)
            list o -> list o ->        % B (linear)
            list o -> o.               % Υ (classical)

type stoup  list o ->                  % Ψ (classical)
            list o -> list o ->        % M (linear)
            o ->                       % stoup
            list o -> list o ->        % A (linear)
            list o -> o.               % Υ (classical)
```

atomic-R **rule:**

$$\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & A,\mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & A,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{atomic}\text{-R}$$

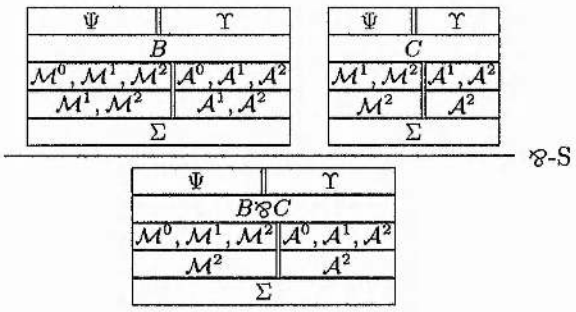
```
% atomic-R rule
right Psi M0M1 M1 A0A1 A1 (A::B0B1) B1 Upsilon :-
  atomic A,
  right Psi M0M1 M1 (A::A0A1) A1 B0B1 B1 Upsilon ,
  removed A (A::A0A1) A1.
```

⊤-**R rule:**

$$\frac{}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \top,\mathcal{B}^0,\mathcal{B}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R}$$

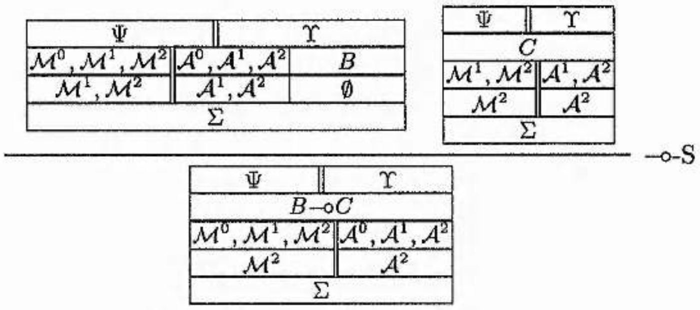
```
% top-R rule
right Psi M0M1 M1 A0A1 A1 (top::B0B1) B1 Upsilon :-
  split M0M1 M0 M1,
  split A0A1 A0 A1,
  split B0B1 B0 B1.
```

⅋-S rule:


Ψ
Υ
B
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^2$
$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^2$
$\mathcal{M}^1, \mathcal{M}^2$
$\mathcal{A}^1, \mathcal{A}^2$
Σ
C
$\mathcal{M}^2$
$\mathcal{A}^2$
⅋-S
B⅋C

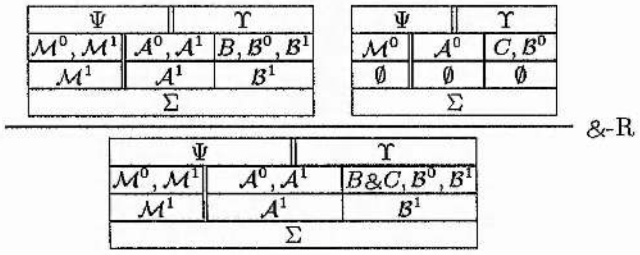
```
% Par-S rule (|-S rule)
stoup Psi M0M1M2 M2 (B | C) A0A1A2 A2 Upsilon :-
  stoup Psi M0M1M2 M1M2 B A0A1A2 A1A2 Upsilon,
  stoup Psi M1M2 M2 C A1A2 A2 Upsilon.
```

—o-S rule:


Ψ
Υ
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^2$
$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^2$
B
$\mathcal{M}^1, \mathcal{M}^2$
$\mathcal{A}^1, \mathcal{A}^2$
∅
Σ
C
$\mathcal{M}^2$
$\mathcal{A}^2$
—o-S
B—oC

```
% Lolli-S rule (--o-S rule)
stoup Psi M0M1M2 M2 (B --o C) A0A1A2 A2 Upsilon :-
  stoup Psi M0M1M2 M1M2 C A0A1A2 A1A2 Upsilon,
  right Psi M1M2 M2 A1A2 A2 (B::nil) nil Upsilon.
```

&-R rule:


Ψ
Υ
$\mathcal{M}^0, \mathcal{M}^1$
$\mathcal{A}^0, \mathcal{A}^1$
$B, \mathcal{B}^0, \mathcal{B}^1$
$\mathcal{M}^1$
$\mathcal{A}^1$
$\mathcal{B}^1$
Σ
$\mathcal{M}^0$
$\mathcal{A}^0$
$C, \mathcal{B}^0$
∅
&-R
$B\&C, \mathcal{B}^0, \mathcal{B}^1$

```
% With-R rule (@-R rule)
right Psi M0M1 M1 A0A1 A1 ((B @ C)::B0B1) B1 Upsilon :-
  right Psi M0M1 M1 A0A1 A1 (B::B0B1) B1 Upsilon,
  removed B (B::B0B1) B1,
  diff M0M1 M1 M0,
  diff A0A1 A1 A0,
  diff B0B1 B1 B0,
  right Psi M0 nil A0 nil (C::B0) nil Upsilon.
```

$$\frac{}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \top,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \top\text{-R}
\qquad
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & A,\mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & A,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \mathit{atomic}\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}
\quad
\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0 & \mathcal{A}^0 & C,\mathcal{B}^0 \\\hline \emptyset & \emptyset & \emptyset \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\&C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \&\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \bot,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \bot\text{-R}
\qquad
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B,C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Sigma & \multicolumn{2}{c|}{\Psi} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\parr C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \parr\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline B,\mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\multimap C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \multimap\text{-R}
\qquad
\frac{\begin{array}{|c||c|c|}\hline \Psi,B & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B\supset C,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \supset\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & B[x\mapsto y],\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{y:\tau,\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \forall_\tau xB,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ \forall\text{-R}
\qquad
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & ?B,\mathcal{B}^0,\mathcal{B}^1 \\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}}\ ?\text{-R}$$

y is not declared in Σ

Figure 3.1: The right rules in the box calculus $\mathcal{B}$.

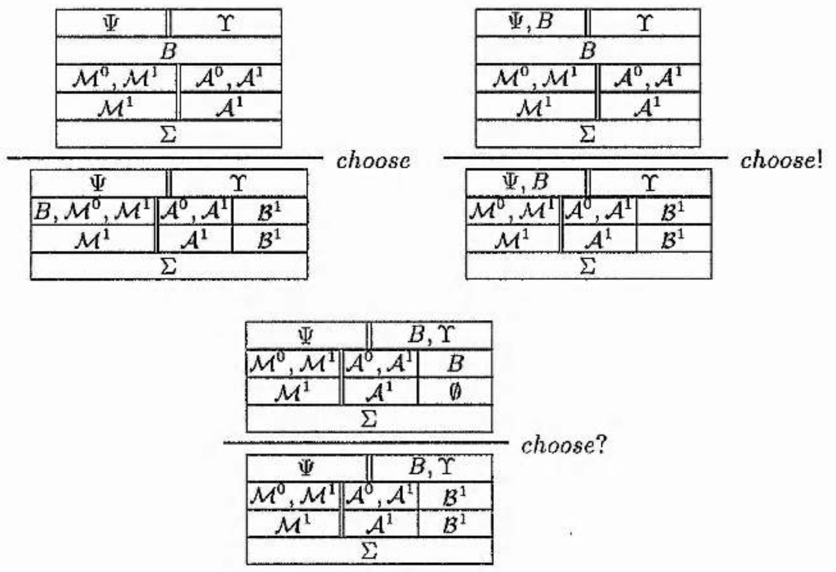


Figure 3.2: The choose rules in the box calculus $\mathcal{B}$.

$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}^1 & A,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial} \qquad \dfrac{}{\begin{array}{|c||c|}\hline \Psi & A,\Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial?} \qquad \dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\bot} \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-S}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_i} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_1 \& B_2} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-S}_i \qquad \dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline B & \emptyset & \emptyset \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{?B} \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ ?\text{-S} \qquad \dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B[x \mapsto t]} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\forall_\tau x B} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \forall\text{-S}$$

t is a Σ-term of type τ

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^2 \\ \hline \mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^1,\mathcal{A}^2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^1,\mathcal{A}^2 \\ \hline \mathcal{M}^2 & \mathcal{A}^2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \parr C} \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^2 \\ \hline \mathcal{M}^2 & \mathcal{A}^2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-S}$$

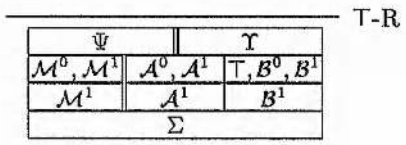
$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^2 & B \\ \hline \mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^1,\mathcal{A}^2 & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^1,\mathcal{A}^2 \\ \hline \mathcal{M}^2 & \mathcal{A}^2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \multimap C} \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^2 & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^2 \\ \hline \mathcal{M}^2 & \mathcal{A}^2 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \multimap\text{-S}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \emptyset & \emptyset & B \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \supset C} \\ \hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \supset\text{-S}$$

Figure 3.3: The *stoup* rules in the box calculus $\mathcal{B}$.

3.2 A New Box Calculus $\mathcal{B}'$

While the box calculus $\mathcal{B}$ introduced above removes the most serious portion of the non-determinism arising from the context splitting, the context management is not completely deterministic. Consider the formulation of the '⊤-R' rule:

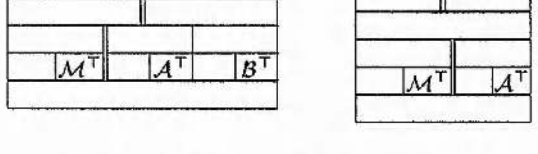

⊤-R
Ψ
Υ
$\mathcal{M}^0, \mathcal{M}^1$
$\mathcal{A}^0, \mathcal{A}^1$
$\top, \mathcal{B}^0, \mathcal{B}^1$
$\mathcal{M}^1$
$\mathcal{A}^1$
$\mathcal{B}^1$
Σ

An interpreter has to split each input context into two parts. One part is consumed by the '⊤-R' rule, and the other part is passed on as the output context. As may be seen in the implementation (see Section 3.1.3), the splitting is achieved by an inefficient generate-and-test operation. This remaining source of non-determinism is particularly problematic when it occurs as part of a goal that fails because of another reason. In this case, the interpreter is enforced to explore all partitions which grow exponentially with the number of formulae in the contexts.

The key idea, for removing the aforementioned non-determinism in the '⊤-R' rule, is the modification of its active role in consuming formulae from the input context. That means that other rules are permitted from now on to consume formulae which were previously consumed by the '⊤-R' inference rule. This idea is adapted from the approaches in [Hodas, 1994] and [Cervesato et al., 1996].

The boxes of $\mathcal{B}$ are replaced by the following boxes of the *modified box calculus* $\mathcal{B}'$. The boxes contain an additional part which is called the *slack* context. The name is derived from Hodas' term *slack indicator*. The counterpart of this context is called *strict* context. Due to the complete separation of the slack context from the strict context there is no need to introduce an additional flag (as in [Hodas, 1994]) which indicates when some formulae can be consumed from the output context.

The output context of the new boxes is divided into a strict part and a slack part as shown below. The variables which stand for multisets containing formulae of the slack context have the symbol '$^\top$' as superscript. The multiset $\mathcal{A}^\top$ contains only atomic formulae.



The modified rules can be found in figures 3.4, 3.5 and 3.6 (pp. 68). In most of the rules, we maintain the declarative presentation with several multisets as in the previously introduced calculus. However, we change slightly the style of presentation in the splitting and sharing rules. Although it is possible to formalize them completely declaratively including several multisets assigned to each different portion, a more operational view is employed which eases the readability of the inference rules.

The passing rules. These rules have an additional component: the new slack output context. It is passed from the output context of the premise to the corresponding output context in the conclusion. The left premise of the '⊃-S' rule expects an empty context.

The returning rules. Apart from the '⊤-R' rule, these inference rules are modified so that they pass their remaining formulae (i.e., the unused formulae) directly to the strict part of the output context and set the slack part to the empty multiset. On the other hand, the '⊤-R' rule passes all formulae disregarding the '⊤' connective from the input context to the slack part of the output context. The strict part of the output context is set to the empty multiset.

The splitting rules. The left premises of these rules receive the input context from the input context of the conclusion. Some formulae will be consumed by the corresponding proof branch. However, it will return a strict and a slack output context. Both parts are given subsequently to the right premise for use as input context. This proof branch also consumes a portion from the input context and produces a strict and a slack output context. The strict output context of the conclusion is the multiset intersection of the strict output contexts of both premises. The slack output context of the conclusion is the slack output of the right premise and, additionally, the multiset intersection of the slack output of the left premise and the strict output of the right premise. Roughly speaking, the strict output context of the conclusion is formed by the formulae that are part of the strict output context in both premises. On the other hand, the slack output context is formed by formulae that appeared at least in one slack output context of the premises.

The sharing rule. The left premise of this rule receives the input context of the conclusion; it consumes a portion from this context and produces a strict and a slack output context. The right premise receives all formulae which are consumed by the proof branch of the left premise and its slack output context. The right premise consumes some formulae and has the remaining formulae as strict or slack output. Three side-conditions ensure that the right premise consumes all formulae which are consumed in the left premise. The strict output of the conclusion is formed by both strict output contexts of the premises. The slack part of the conclusion is the multiset intersection of both slack output contexts of the premises. That means roughly that only formulae which appear in both slack output contexts are passed to the slack output context of the conclusion; the remaining formulae are consumed by this inference rule. This behaviour is because the rule has to consume, apart from the two components of the analysed formula, the same formulae in both proof branches.

3.2.1 Soundness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$

Before we begin with the soundness proof of $\mathcal{B}'$ w.r.t. $\mathcal{B}$, an observation is stated which simplifies the proof. Intuitively, a formula from the slack output context of a box can be consumed in the corresponding proof branch. Hence, we can modify a $\mathcal{B}'$-proof so that a particular formula from the original slack output context is consumed by a '⊤-R' rule in the corresponding proof branch and thus does not appear in the output context. The following proposition is used when a '&-R' rule is translated since this rule passes the intersection of each slack output context of the premises to the slack output context of the conclusion. That means the formulae which do not appear in both slack output contexts are "consumed" by the '&-R' rule.

Proposition 14 (Modification of a $\mathcal{B}'$-proof)

1. A $\mathcal{B}'$-proof of a box of the form ($\mathcal{M}$ and $\mathcal{B}$ are multisets of formulae, $\mathcal{A}$ is a multiset of atomic formulae):

$$\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A},\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B},\mathcal{B}^\top} \\ \hline
\mathcal{M}^1 & \mathcal{M},\mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A},\mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B},\mathcal{B}^\top \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}$$

can be modified such that the end box is of the form:

$$\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A},\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B},\mathcal{B}^\top} \\ \hline
\mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}$$

and

2. A $\mathcal{B}'$-proof of a box of the form ($\mathcal{M}$ is a multiset of formulae, $\mathcal{A}$ is a multiset of atomic formulae):

$$\begin{array}{|c|c||c|c|}
\hline
\multicolumn{4}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{4}{|c|}{B} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A},\mathcal{A}^\top} \\ \hline
\mathcal{M}^1 & \mathcal{M},\mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A},\mathcal{A}^\top \\ \hline
\multicolumn{4}{|c|}{\Sigma} \\ \hline
\end{array}$$

can be modified such that the end box is of the form:

$$\begin{array}{|c|c||c|c|}
\hline
\multicolumn{4}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{4}{|c|}{B} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A},\mathcal{A}^\top} \\ \hline
\mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top \\ \hline
\multicolumn{4}{|c|}{\Sigma} \\ \hline
\end{array}$$

Proof: Simultaneous induction on the height of the $\mathcal{B}'$-proof. For expository purposes the proof is given only for formulae from the slack context $\mathcal{M}, \mathcal{M}^\top$.The other slack contexts can be treated similarly.

This modification of a $\mathcal{B}'$-proof does not change its height. The $\mathcal{B}'$ proof is modified such that the formulae $\mathcal{M}$ of the slack output context are consumed by the '$\top$-**R**' rule where these formulae are passed to the slack output context. The translation of the trivial cases is omitted since the corresponding inference rules do not change the slack and strict contexts. The non-trivial cases are given below:

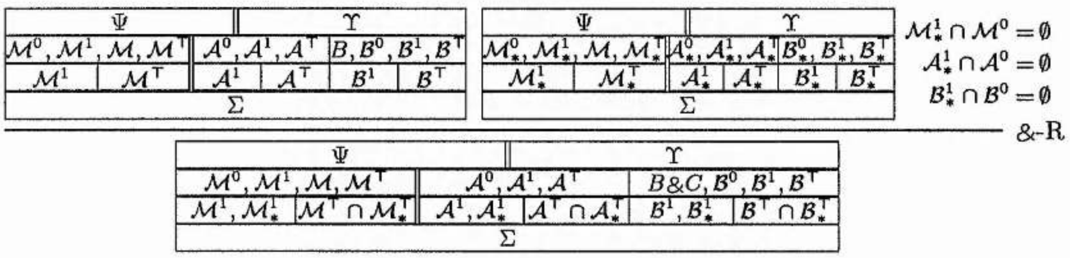
Case 1 ($\top$-R):

$$\dfrac{\qquad}{\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^\top} & \multicolumn{2}{c|}{\top,\mathcal{B}^\top} \\ \hline
\emptyset & \mathcal{M},\mathcal{M}^\top & \emptyset & \mathcal{A}^\top & \emptyset & \mathcal{B}^\top \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}}\ \top\text{-R}
\quad\Rightarrow\quad
\dfrac{\qquad}{\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^\top} & \multicolumn{2}{c|}{\top,\mathcal{B}^\top} \\ \hline
\emptyset & \mathcal{M}^\top & \emptyset & \mathcal{A}^\top & \emptyset & \mathcal{B}^\top \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}}\ \top\text{-R}$$

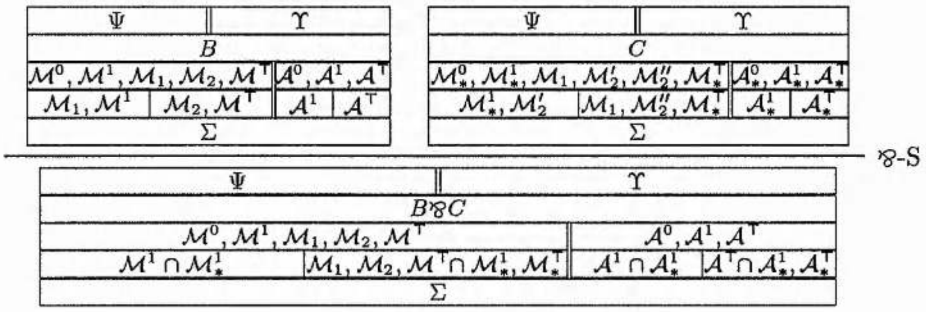
Case 2 (&-R):

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline
\mathcal{M}^1 & \mathcal{M},\mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}
\quad
\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M},\mathcal{M}^\top_*} & \multicolumn{2}{c|}{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*} & \multicolumn{2}{c|}{\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*} \\ \hline
\mathcal{M}^1_* & \mathcal{M},\mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* & \mathcal{B}^1_* & \mathcal{B}^\top_* \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}
\quad
\begin{array}{r}
\mathcal{M}^1_* \cap \mathcal{M}^0 = \emptyset \\
\mathcal{A}^1_* \cap \mathcal{A}^0 = \emptyset \\
\mathcal{B}^1_* \cap \mathcal{B}^0 = \emptyset
\end{array}}{\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{6}{|c|}{\Psi \quad\|\quad \Upsilon} \\ \hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M},\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\&C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline
\mathcal{M}^1,\mathcal{M}^1_* & \mathcal{M},\mathcal{M}^\top \cap \mathcal{M}^\top_* & \mathcal{A}^1,\mathcal{A}^1_* & \mathcal{A}^\top \cap \mathcal{A}^\top_* & \mathcal{B}^1,\mathcal{B}^1_* & \mathcal{B}^\top \cap \mathcal{B}^\top_* \\ \hline
\multicolumn{6}{|c|}{\Sigma} \\ \hline
\end{array}}\ \&\text{-R}$$

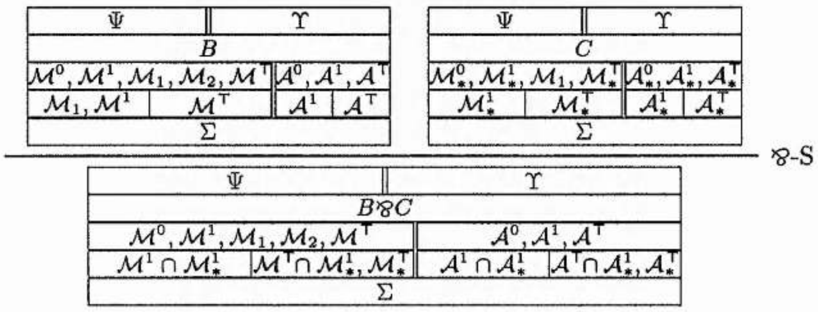
$$\Downarrow$$



Case 3 ($\parr$-S and $\multimap$-S): For the translation of the '$\parr$-S' rule the multiset $\mathcal{M}$ is divided into two parts. One part ($\mathcal{M}_1$) appears in the strict output context of the left premise and the second part ($\mathcal{M}_2$) appears in the slack output context of the left premise. Hence, the first part is consumed by the proof branch of the right premise and the second one is consumed in the proof branch of the left premise. The following rule ($\mathcal{M}_2 = \mathcal{M}'_2, \mathcal{M}''_2$):



is translated as follows:

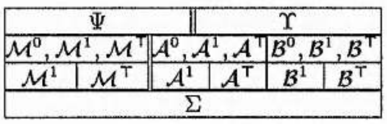


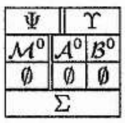
The translation of the '$\multimap$-S' rule is similar but omitted.

The soundness proof will be shown by translating a $\mathcal{B}'$-proof into a $\mathcal{B}$-proof which has the consumed formulae from the $\mathcal{B}'$-proof as input context. An exception is the '&-R' rule where the left-premise returns some formulae which must be consumed in order to obtain a $\mathcal{B}$-proof. Therefore, Proposition 14 is used which modifies a $\mathcal{B}'$-proof such that it consumes some additional formulae.

Proposition 15 (Soundness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$)

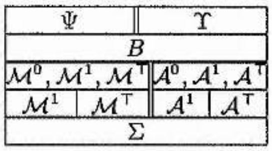
1. *For every $\mathcal{B}'$-proof of a box of the form:* *there exists a $\mathcal{B}$-proof of the box:*

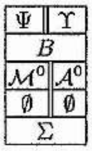




and

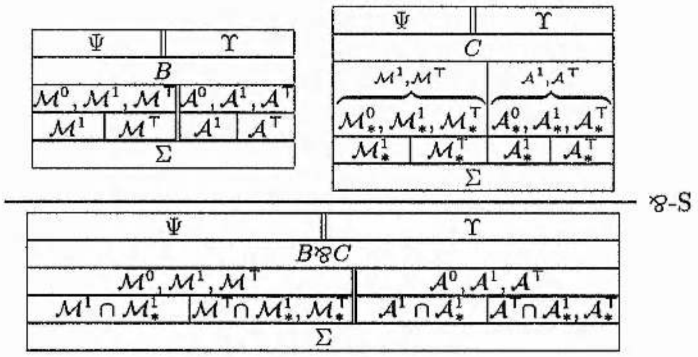
2. For every $\mathcal{B}'$-proof of a box of the form: *there exists a $\mathcal{B}$-proof of the box:*


Ψ
Υ
B
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top$
$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^\top$
$\mathcal{M}^1$
$\mathcal{M}^\top$
$\mathcal{A}^1$
$\mathcal{A}^\top$
Σ



Proof: Simultaneous induction on the heights of the $\mathcal{B}'$-proofs. The trivial cases of the transformations are given in Appendix A.4. The non-trivial rules that split the proof branch are examined below:

Case 1 ($\parr$-S): The inference rule in $\mathcal{B}'$ is as follows:


Ψ
Υ
B
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top$
$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^\top$
$\mathcal{M}^1$
$\mathcal{M}^\top$
$\mathcal{A}^1$
$\mathcal{A}^\top$
Σ
Ψ
Υ
C
$\mathcal{M}^1, \mathcal{M}^\top$
$\mathcal{A}^1, \mathcal{A}^\top$
$\mathcal{M}^0_*, \mathcal{M}^1_*, \mathcal{M}^\top_*$
$\mathcal{A}^0_*, \mathcal{A}^1_*, \mathcal{A}^\top_*$
$\mathcal{M}^1_*$
$\mathcal{M}^\top_*$
$\mathcal{A}^1_*$
$\mathcal{A}^\top_*$
Σ
$\parr$-S
Ψ
Υ
$B \parr C$
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top$
$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^\top$
$\mathcal{M}^1 \cap \mathcal{M}^1_*$
$\mathcal{M}^\top \cap \mathcal{M}^1_*, \mathcal{M}^\top_*$
$\mathcal{A}^1 \cap \mathcal{A}^1_*$
$\mathcal{A}^\top \cap \mathcal{A}^1_*, \mathcal{A}^\top_*$
Σ

Although the presentation is partially ambiguous, it is chosen since it improves the readability. The names of the multisets $\mathcal{M}^0$ and $\mathcal{A}^0$ in the conclusion suggest that they stand for the fragments of the input context that will be consumed. However, they stand only for the fragments which are consumed by the proof of the left premise. The actual consumed part $\mathcal{C}$ is a result of the multiset difference between the input context and the output context. Hence[2]

$$\begin{aligned}
\mathcal{C} &= \mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top - \mathcal{M}^1 \cap \mathcal{M}^1_* - (\mathcal{M}^\top \cap \mathcal{M}^1_*, \mathcal{M}^\top_*) \\
&= \mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top - (\mathcal{M}^1 \cap \mathcal{M}^1_*, \mathcal{M}^\top \cap \mathcal{M}^1_*) - \mathcal{M}^\top_* && (1) \\
&= \mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top - \mathcal{M}^1_* - \mathcal{M}^\top_* && (2) \\
&= \mathcal{M}^0, \mathcal{M}^0_*.
\end{aligned}$$

Step 1 is a valid transformation since $\mathcal{M}^1_*$ is a subset of $\mathcal{M}^1, \mathcal{M}^\top$. Step 2 is a valid transformation because the input context of the right premise is formed by the output context of the left premise. Hence $\mathcal{M}^1, \mathcal{M}^\top = \mathcal{M}^0_*, \mathcal{M}^1_*, \mathcal{M}^\top_*$ and therefore $\mathcal{M}^0_* = (\mathcal{M}^1, \mathcal{M}^\top) - \mathcal{M}^1_* - \mathcal{M}^\top_*$. Similar equations can be stated for the other contexts; the translation is as follows:

[2] A comma stands for a multiset union; a $-$ stands for the multiset difference (the ',' binds tighter than the '$-$').

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^0_* & \mathcal{A}^0,\mathcal{A}^0_* \\ \hline \mathcal{M}^0_* & \mathcal{A}^0_* \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^0_*, & \mathcal{A}^0_*, \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \parr C} \\ \hline \mathcal{M}^0,\mathcal{M}^0_* & \mathcal{A}^0,\mathcal{A}^0_* \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \ \parr\text{-S}$$

The translated proof of the left premise can be obtained using Proposition 12 which constructs a $\mathcal{B}$-proof where $\mathcal{M}^0_*$ and $\mathcal{A}^0_*$ are added to the contexts.

Case 2 ($\multimap$-S): The inference rule is translated similarly to the case shown above:

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{C} \\ \hline \multicolumn{2}{|c|}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^1,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^1,\mathcal{A}^\top}} \\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B \multimap C} \\ \hline \multicolumn{2}{|c|}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} \\ \hline \mathcal{M}^1 \cap \mathcal{M}^1_* & \mathcal{M}^\top \cap \mathcal{M}^1_*,\mathcal{M}^\top_* & \mathcal{A}^1 \cap \mathcal{A}^1_* & \mathcal{A}^\top \cap \mathcal{A}^1_*,\mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} \ \multimap\text{-S}$$

is translated to:

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_* & \mathcal{A}^0,\mathcal{A}^0_* & B \\ \hline \mathcal{M}^0_* & \mathcal{A}^0_* & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^0_* & \mathcal{A}^0_* \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \multimap C} \\ \hline \mathcal{M}^0,\mathcal{M}^0_* & \mathcal{A}^0,\mathcal{A}^0_* \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \ \multimap\text{-S}$$

Case 3 (&-R): The inference rule in $\mathcal{B}'$ is as follows:

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^0,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^0,\mathcal{A}^\top}} & \multicolumn{2}{c|}{\overbrace{C,\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*}^{C,\mathcal{B}^0,\mathcal{B}^\top}} \\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* & \mathcal{B}^1_* & \mathcal{B}^\top_* \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{r} \mathcal{M}^1_* \cap \mathcal{M}^0 = \emptyset \\ \mathcal{A}^1_* \cap \mathcal{A}^0 = \emptyset \\ \mathcal{B}^1_* \cap \mathcal{B}^0 = \emptyset \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B \& C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1,\mathcal{M}^1_* & \mathcal{M}^\top \cap \mathcal{M}^\top_* & \mathcal{A}^1,\mathcal{A}^1_* & \mathcal{A}^\top \cap \mathcal{A}^\top_* & \mathcal{B}^1,\mathcal{B}^1_* & \mathcal{B}^\top \cap \mathcal{B}^\top_* \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}} \ \&\text{-R}$$

In this case, the multiset $\mathcal{M}^0$, $\mathcal{A}^0$ and $\mathcal{B}^0$ stand only for the formulae that are consumed by the proof of the left premise. The right premise consumes possibly some additional formulae which are returned as slack output from the left premise. However, the corresponding $\mathcal{B}$-proof has to consume the same formulae. The $\mathcal{B}'$-proof of the left premise using Proposition 14 is modified so that the corresponding proof consumes the additional formulae, namely the formulae of the multisets $\mathcal{M}^0_* \cap \mathcal{M}^\top$, $\mathcal{A}^0_* \cap \mathcal{A}^\top$ and $\mathcal{B}^0_* \cap \mathcal{B}^\top$. The modification of the endbox is as follows:

$$
\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\
\hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top_0,\mathcal{M}^\top-\mathcal{M}^\top_0} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top_0,\mathcal{A}^\top-\mathcal{A}^\top_0} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top_0,\mathcal{B}^\top-\mathcal{B}^\top_0} \\
\hline
\mathcal{M}^1 & \mathcal{M}^\top_0,\mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top_0,\mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top_0,\mathcal{B}^\top \\
\hline
\multicolumn{6}{|c|}{\Sigma} \\
\hline
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{|c|c||c|c|c|c|}
\hline
\multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\
\hline
\multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top_0,\mathcal{M}^\top-\mathcal{M}^\top_0} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top_0,\mathcal{A}^\top-\mathcal{A}^\top_0} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top_0,\mathcal{B}^\top-\mathcal{B}^\top_0} \\
\hline
\mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\
\hline
\multicolumn{6}{|c|}{\Sigma} \\
\hline
\end{array}
$$

where $\mathcal{M}^\top_0 = \mathcal{M}^0_* \cap \mathcal{M}^\top$; $\mathcal{A}^\top_0 = \mathcal{A}^0_* \cap \mathcal{A}^\top$ and $\mathcal{B}^\top_0 = \mathcal{B}^0_* \cap \mathcal{B}^\top$. Then, the translation of the proof branch which corresponds to the left premise is obtained by translating the modified $\mathcal{B}'$-proof into a $\mathcal{B}$-proof. The inference rule of $\mathcal{B}$ is as follows:

$$
\dfrac{
\begin{array}{|c||c|c|}
\hline
\Psi & \multicolumn{2}{c|}{\Upsilon} \\
\hline
\mathcal{M}^0,\mathcal{M}^\top_0 & \mathcal{A}^0,\mathcal{A}^\top_0 & B,\mathcal{B}^0,\mathcal{B}^\top_0 \\
\hline
\emptyset & \emptyset & \emptyset \\
\hline
\multicolumn{3}{|c|}{\Sigma} \\
\hline
\end{array}
\qquad
\begin{array}{|c||c|c|}
\hline
\Psi & \multicolumn{2}{c|}{\Upsilon} \\
\hline
\mathcal{M}^0_* & \mathcal{A}^0_* & C,\mathcal{B}^0_* \\
\hline
\emptyset & \emptyset & \emptyset \\
\hline
\multicolumn{3}{|c|}{\Sigma} \\
\hline
\end{array}
}{
\begin{array}{|c||c|c|}
\hline
\Psi & \multicolumn{2}{c|}{\Upsilon} \\
\hline
\mathcal{M}^0,\mathcal{M}^\top_0 & \mathcal{A}^0,\mathcal{A}^\top_0 & B\&C,\mathcal{B}^0,\mathcal{B}^\top_0 \\
\hline
\emptyset & \emptyset & \emptyset \\
\hline
\multicolumn{3}{|c|}{\Sigma} \\
\hline
\end{array}
}\ \&\text{-R}
$$

3.2.2 Completeness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$

The completeness proof is achieved by translating each $\mathcal{B}$-proof into a $\mathcal{B}'$-proof. However, we have to identify the formulae which appear in the slack output context of the translated $\mathcal{B}'$-proof. The '$\top$-part', which is given by the following proposition, represents those formulae. Some of these formulae come from the consumed part (i.e., $\mathcal{M}^0$, $\mathcal{A}^0$ and $\mathcal{B}^0$) and the other come from the returned part (i.e., $\mathcal{M}^1$, $\mathcal{A}^1$ and $\mathcal{B}^1$).

Proposition 16 **($\top$-Part)**

1. *From each multiset in a box of a $\mathcal{B}$-proof with the endbox of the form:*

$$
\begin{array}{|c||c|c|}
\hline
\Psi & \multicolumn{2}{c|}{\Upsilon} \\
\hline
\mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \mathcal{B}^0,\mathcal{B}^1 \\
\hline
\mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1 \\
\hline
\multicolumn{3}{|c|}{\Sigma} \\
\hline
\end{array}
$$

a multiset can be divided so that the endbox of the $\mathcal{B}$-proof is of the form:

$$
\begin{array}{|c||c|c|}
\hline
\Psi & \multicolumn{2}{c|}{\Upsilon} \\
\hline
\mathcal{M}'^0,\mathcal{M}^0_\top,\mathcal{M}'^1,\mathcal{M}^1_\top & \mathcal{A}'^0,\mathcal{A}^0_\top,\mathcal{A}'^1,\mathcal{A}^1_\top & \mathcal{B}'^0,\mathcal{B}^0_\top,\mathcal{B}'^1,\mathcal{B}^1_\top \\
\hline
\mathcal{M}'^1,\mathcal{M}^1_\top & \mathcal{A}'^1,\mathcal{A}^1_\top & \mathcal{B}'^1,\mathcal{B}^1_\top \\
\hline
\multicolumn{3}{|c|}{\Sigma} \\
\hline
\end{array}
$$

and

2. From each multiset in a box of $\mathcal{B}$-proof with the endbox of the form:

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0, \mathcal{M}^1 & \mathcal{A}^0, \mathcal{A}^1 \\ \hline \mathcal{M}^1 & \mathcal{A}^1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

a multiset can be divided so that the endbox of the $\mathcal{B}$-proof is of the form:

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}'^0, \mathcal{M}^0_\top, \mathcal{M}'^1, \mathcal{M}^1_\top & \mathcal{A}'^0, \mathcal{A}^0_\top, \mathcal{A}'^1, \mathcal{A}^1_\top \\ \hline \mathcal{M}'^1, \mathcal{M}^1_\top & \mathcal{A}'^1, \mathcal{A}^1_\top \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

Proof: The aforementioned division of each multiset is given by describing the $\top$-part; the remaining formulae are members of the other part.

A box in a $\mathcal{B}$-proof is a conclusion of:

- a $\top$-R rule, then all formulae of each multiset are in the $\top$-parts;
- a *initial*, *initial*?, ?-S or $\bot$-S rule, then the $\top$-parts are empty (the multisets $\mathcal{B}^1$ in each *choose* rule are treated similarly);
- a $\invamp$-S or $\multimap$-S rule, then the $\top$-part of each multiset consists of formulae which appear at least in one of the corresponding $\top$-parts of the premises;
- a $\&$-R rule, then the $\top$-part of each multiset consists of formulae which are in both of the corresponding $\top$-parts of the premises;
- a $\supset$-S rule, then the $\top$-part of each multiset of each multiset is equal to the corresponding $\top$-part of the right premise.

In all other cases, each $\top$-part of the conclusion is equal to the corresponding $\top$-part of the premise. From now on, we use a subscript '$\top$' for representing a '$\top$-part' of a multiset in a $\mathcal{B}$-proof. □

In the remaining part of this section, we consider only $\mathcal{B}$-proofs where the multisets are divided according to Proposition 16. The formulae which are members of a $\top$-part are distributed (in the completeness proof) across the strict and slack output context, with the $\top$-parts going into the latter. However, before we start with the actual translation, a technical result is proven because we give the translation for boxes with an empty output context. Therefore, the contexts have to be modified so that we obtain $\mathcal{B}'$-proofs.

The following proposition is used for modifying contexts in $\mathcal{B}'$-proofs. Intuitively, it is possible to add arbitrary multisets of formulae to both input and output context, and these additional formulae do not affect the provability. This addition of formulae is similar to the modification of $\mathcal{B}$-proofs using Proposition 12. However, there is one difference: the added formulae can appear in either the strict or the slack output context. Therefore the multisets which are added are split into two parts: one part is returned as strict output (i.e., X^1, Y^1 and Z^1 in the proposition below) and the other part is returned as slack output (i.e., $X^1_\top, Y^1_\top$ and $Z^1_\top$). In the following proposition, we use a *corresponding* $\mathcal{B}$-proof. Such a $\mathcal{B}$-proof has the same structure as the $\mathcal{B}'$-proof (i.e., the order of inference steps is equal). The proposition is used when translating a '$\invamp$-S' and a '$\multimap$-S' rule in the completeness proof.

Proposition 17 (Modification of $\mathcal{B}'$-proofs)

1. Let P' be a $\mathcal{B}'$-proof with the endbox of the form:

Ψ			Υ		
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^0_\top, \mathcal{M}^1_\top$		$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^0_\top, \mathcal{A}^1_\top$		$\mathcal{B}^0, \mathcal{B}^1, \mathcal{B}^1_\top, \mathcal{B}^1_\top$	
$\mathcal{M}^1$	$\mathcal{M}^0_\top, \mathcal{M}^1_\top$	$\mathcal{A}^1$	$\mathcal{A}^0_\top, \mathcal{A}^1_\top$	$\mathcal{B}^1$	$\mathcal{B}^0_\top, \mathcal{B}^1_\top$
Σ					

let P be a corresponding $\mathcal{B}$-proof with the endbox of the form:

Ψ	Υ	
$\mathcal{M}^0, \mathcal{M}^0_\top, \mathcal{M}^1, \mathcal{M}^1_\top, X^1, X^1_\top$	$\mathcal{A}^0, \mathcal{A}^0_\top, \mathcal{A}^1, \mathcal{A}^1_\top, Y^1, Y^1_\top$	$\mathcal{B}^0, \mathcal{B}^0_\top, \mathcal{B}^1, \mathcal{B}^1_\top, Z^1, Z^1_\top$
$\mathcal{M}^1, \mathcal{M}^1_\top, X^1, X^1_\top$	$\mathcal{A}^1, \mathcal{A}^1_\top, Y^1, Y^1_\top$	$\mathcal{B}^1, \mathcal{B}^1_\top, Z^1, Z^1_\top$
Σ		

then there exists a $\mathcal{B}'$-proof of the box:

Ψ			Υ		
$\mathcal{M}^0, \mathcal{M}^1, X^1, \mathcal{M}^0_\top, \mathcal{M}^1_\top, X^1_\top$		$\mathcal{A}^0, \mathcal{A}^1, Y^1, \mathcal{A}^0_\top, \mathcal{A}^1_\top, Y^1_\top$		$\mathcal{B}^0, \mathcal{B}^1, Z^1, \mathcal{B}^1_\top, \mathcal{B}^1_\top, Z^1_\top$	
$\mathcal{M}^1, X^1$	$\mathcal{M}^0_\top, \mathcal{M}^1_\top, X^1_\top$	$\mathcal{A}^1, Y^1$	$\mathcal{A}^0_\top, \mathcal{A}^1_\top, Y^1_\top$	$\mathcal{B}^1, Z^1$	$\mathcal{B}^0_\top, \mathcal{B}^1_\top, Z^1_\top$
Σ					

and

2. Let P' be a $\mathcal{B}'$-proof with the endbox of the form:

Ψ		Υ	
B			
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^0_\top, \mathcal{M}^1_\top$		$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^0_\top, \mathcal{A}^1_\top$	
$\mathcal{M}^1$	$\mathcal{M}^0_\top, \mathcal{M}^1_\top$	$\mathcal{A}^1$	$\mathcal{A}^0_\top, \mathcal{A}^1_\top$
Σ			

let P be a corresponding $\mathcal{B}$-proof with the endbox of the form:

Ψ	Υ
B	
$\mathcal{M}^0, \mathcal{M}^0_\top, \mathcal{M}^1, \mathcal{M}^1_\top, X^1, X^1_\top$	$\mathcal{A}^0, \mathcal{A}^0_\top, \mathcal{A}^1, \mathcal{A}^1_\top, Y^1, Y^1_\top$
$\mathcal{M}^1, \mathcal{M}^1_\top, X^1, X^1_\top$	$\mathcal{A}^1, \mathcal{A}^1_\top, Y^1, Y^1_\top$
Σ	

then there exists a $\mathcal{B}'$-proof of the box:

Ψ		Υ	
B			
$\mathcal{M}^0, \mathcal{M}^1, X^1, \mathcal{M}^0_\top, \mathcal{M}^1_\top, X^1_\top$		$\mathcal{A}^0, \mathcal{A}^1, Y^1, \mathcal{A}^0_\top, \mathcal{A}^1_\top, Y^1_\top$	
$\mathcal{M}^1, X^1$	$\mathcal{M}^0_\top, \mathcal{M}^1_\top, X^1_\top$	$\mathcal{A}^1, Y^1$	$\mathcal{A}^0_\top, \mathcal{A}^1_\top, Y^1_\top$
Σ			

Proof Simultaneous induction on the heights of $\mathcal{B}'$-proofs. The proof consists of the translations of the inference rules; we give only the non-trivial cases.

Case 1 ($\invamp$-S) The inference rule is as follows (where $\mathcal{M}^1, \mathcal{M}^\top = \mathcal{M}^0_*, \mathcal{M}^1_*, \mathcal{M}^\top_*$):

Ψ		Υ	
B			
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top$		$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^\top$	
$\mathcal{M}^1$	$\mathcal{M}^\top$	$\mathcal{A}^1$	$\mathcal{A}^\top$
Σ			

Ψ		Υ	
C			
$\mathcal{M}^0_*, \mathcal{M}^1_*, \mathcal{M}^\top_*$		$\mathcal{A}^0_*, \mathcal{A}^1_*, \mathcal{A}^\top_*$	
$\mathcal{M}^1_*$	$\mathcal{M}^\top_*$	$\mathcal{A}^1_*$	$\mathcal{A}^\top_*$
Σ			

$\invamp$-S

Ψ		Υ	
$B \invamp C$			
$\mathcal{M}^0, \mathcal{M}^1, \mathcal{M}^\top$		$\mathcal{A}^0, \mathcal{A}^1, \mathcal{A}^\top$	
$\mathcal{M}^1 \cap \mathcal{M}^1_*$	$\mathcal{M}^\top \cap \mathcal{M}^1_*, \mathcal{M}^\top_*$	$\mathcal{A}^1 \cap \mathcal{A}^1_*$	$\mathcal{A}^\top \cap \mathcal{A}^1_*, \mathcal{A}^\top_*$
Σ			

with the corresponding proof step of the $\mathcal{B}$-proof (where $X^2_\top = X^2$ and $Y^2_\top = Y^2$, but the formulae of X^2 and Y^2 are not members of a $\top$-part) as follows:

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\\hline \multicolumn{2}{|c|}{B} \\\hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top,X^2_\top & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top,Y^2_\top \\\hline \mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top,X^2_\top & \mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top,Y^2_\top \\\hline \multicolumn{2}{|c|}{\Sigma} \\\hline \end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\\hline \multicolumn{2}{|c|}{C} \\\hline \mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*,X^1,X^1_\top,X^2 & \mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1,Y^1_\top,Y^2 \\\hline \mathcal{M}^1_*,\mathcal{M}^\top_*,X^1,X^1_\top,X^2 & \mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1,Y^1_\top,Y^2 \\\hline \multicolumn{2}{|c|}{\Sigma} \\\hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\\hline \multicolumn{2}{|c|}{B \parr C} \\\hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top,X^2_\top & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top,Y^2_\top \\\hline \mathcal{M}^1\cap\mathcal{M}^1_*,\mathcal{M}^\top\cap\mathcal{M}^1_*,\mathcal{M}^\top_*,X^1,X^1_\top,X^2_\top & \mathcal{A}^1\cap\mathcal{A}^1_*,\mathcal{A}^\top\cap\mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1,Y^1_\top,Y^2_\top \\\hline \multicolumn{2}{|c|}{\Sigma} \\\hline \end{array}} \parr\text{-S}$$

is translated as follows:

$$\dfrac{\begin{array}{c} \begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\\hline \multicolumn{4}{|c|}{B} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top,X^2_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top,Y^2_\top} \\\hline \mathcal{M}^1,X^1 & \mathcal{M}^\top,X^1_\top,X^2_\top & \mathcal{A}^1,Y^1 & \mathcal{A}^\top,Y^1_\top,Y^2_\top \\\hline \multicolumn{4}{|c|}{\Sigma} \\\hline \end{array} \\ \begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\\hline \multicolumn{4}{|c|}{C} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*,X^1,X^1_\top,X^2_\top} & \multicolumn{2}{c|}{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1,Y^1_\top,Y^2_\top} \\\hline \mathcal{M}^1_*,X^1,X^2 & \mathcal{M}^\top_*,X^1_\top & \mathcal{A}^1_*,Y^1,Y^2_\top & \mathcal{A}^\top_*,Y^1_\top \\\hline \multicolumn{4}{|c|}{\Sigma} \\\hline \end{array} \end{array}}{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\\hline \multicolumn{4}{|c|}{B \parr C} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top,X^2_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top,Y^2_\top} \\\hline \mathcal{M}^1\cap\mathcal{M}^1_*,X^1 & \mathcal{M}^\top\cap\mathcal{M}^1_*,\mathcal{M}^\top_*,X^1_\top,X^2_\top & \mathcal{A}^1\cap\mathcal{A}^1_*,Y^1 & \mathcal{A}^\top\cap\mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1_\top,Y^2_\top \\\hline \multicolumn{4}{|c|}{\Sigma} \\\hline \end{array}} \parr\text{-S}$$

The translation of the '$\multimap$-S' rule is similar, and therefore it is omitted.

Case 2 (&-R) The inference rule is as follows:

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\\hline \multicolumn{6}{|c|}{\Sigma} \\\hline \end{array} \quad \begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*} & \multicolumn{2}{c|}{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*} & \multicolumn{2}{c|}{C,\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*} \\\hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* & \mathcal{B}^1_* & \mathcal{B}^\top_* \\\hline \multicolumn{6}{|c|}{\Sigma} \\\hline \end{array} \quad \begin{array}{c} \mathcal{M}^1_*\cap\mathcal{M}^0=\emptyset \\ \mathcal{A}^1_*\cap\mathcal{A}^0=\emptyset \\ \mathcal{B}^1_*\cap\mathcal{B}^0=\emptyset \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\&C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\\hline \mathcal{M}^1,\mathcal{M}^1_* & \mathcal{M}^\top\cap\mathcal{M}^\top_* & \mathcal{A}^1,\mathcal{A}^1_* & \mathcal{A}^\top\cap\mathcal{A}^\top_* & \mathcal{B}^1,\mathcal{B}^1_* & \mathcal{B}^\top\cap\mathcal{B}^\top_* \\\hline \multicolumn{6}{|c|}{\Sigma} \\\hline \end{array}} \&\text{-R}$$

with the endbox of the corresponding $\mathcal{B}$-proof which is as follows:

$$\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\\hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top & B\&C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top,Z^1,Z^1_\top \\\hline \mathcal{M}^1,\mathcal{M}^1_*,\mathcal{M}^\top\cap\mathcal{M}^\top_*,X^1,X^1_\top & \mathcal{A}^1,\mathcal{A}^1_*,\mathcal{A}^\top\cap\mathcal{A}^\top_*,Y^1,Y^1_\top & \mathcal{B}^1,\mathcal{B}^1_*,\mathcal{B}^\top\cap\mathcal{B}^\top_*,Z^1,Z^1_\top \\\hline \multicolumn{3}{|c|}{\Sigma} \\\hline \end{array}$$

is translated as follows:

$$\dfrac{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top,Z^1,Z^1_\top}\\ \hline \mathcal{M}^1,X^1 & \mathcal{M}^\top,X^1_\top & \mathcal{A}^1,Y^1 & \mathcal{A}^\top,Y^1_\top & \mathcal{B}^1,Z^1 & \mathcal{B}^\top,Z^1_\top\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array} \qquad \begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*,X^1_\top} & \multicolumn{2}{c|}{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*,Y^1_\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*,Z^1_\top}\\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_*,X^1_\top & \mathcal{A}^1_* & \mathcal{A}^\top_*,Y^1_\top & \mathcal{B}^1_* & \mathcal{B}^\top_*,Z^1_\top\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array} \quad \begin{array}{r} \mathcal{M}^1_*\cap\mathcal{M}^0=\emptyset\\ \mathcal{A}^1_*\cap\mathcal{A}^0=\emptyset\\ \mathcal{B}^1_*\cap\mathcal{B}^0=\emptyset \end{array}}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top,X^1,X^1_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top,Y^1,Y^1_\top} & \multicolumn{2}{c|}{B\&C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top,Z^1,Z^1_\top}\\ \hline \mathcal{M}^1,\mathcal{M}^1_*,X^1 & \mathcal{M}^\top\cap\mathcal{M}^\top_*,X^1_\top & \mathcal{A}^1,\mathcal{A}^1_*,Y^1 & \mathcal{A}^\top\cap\mathcal{A}^\top_*,Y^1_\top & \mathcal{B}^1,\mathcal{B}^1_*,Z^1 & \mathcal{B}^\top\cap\mathcal{B}^\top_*,Z^1_\top\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array}} \ \&\text{-R}$$

□

The key idea behind the completeness proof is a translation of those inference steps of $\mathcal{B}$ that have an empty output context. This simplifies the translation because the corresponding inference rules of $\mathcal{B}'$ also have an empty strict output context. However in some cases, the output contexts have to consist of some formulae in order to obtain a $\mathcal{B}'$-proof. This is achieved by a modification of $\mathcal{B}'$-proofs using the proposition stated above.

A $\top$-part is divided from the multisets which appear in the boxes of the $\mathcal{B}$ calculus (on the left-hand side below) according to Proposition 16. The $\top$-parts (i.e., $\mathcal{M}^0_\top$, $\mathcal{A}^0_\top$ and $\mathcal{B}^0_\top$) represent formulae which are consumed by '$\top$-R' rules.

Proposition 18 (Completeness $\mathcal{B}'$ w.r.t. $\mathcal{B}$)

1. *For every $\mathcal{B}$-proof of a box of the form:* *there exists a $\mathcal{B}'$-proof of the box:*

$$\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & \mathcal{B}^0,\mathcal{B}^0_\top\\ \hline \emptyset & \emptyset & \emptyset\\ \hline \multicolumn{3}{|c|}{\Sigma}\\ \hline \end{array} \qquad\qquad \begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^0_\top}\\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array}$$

and

2. *For every $\mathcal{B}$-proof of a box of the form:* *there exists a $\mathcal{B}'$-proof of the box:*

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon\\ \hline \multicolumn{2}{|c|}{B}\\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top\\ \hline \emptyset & \emptyset\\ \hline \multicolumn{2}{|c|}{\Sigma}\\ \hline \end{array} \qquad\qquad \begin{array}{|cc||cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\ \hline \multicolumn{4}{|c|}{B}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top}\\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top\\ \hline \multicolumn{4}{|c|}{\Sigma}\\ \hline \end{array}$$

Proof: Simultaneous induction on the heights of the $\mathcal{B}$-proofs. The trivial cases are given in Appendix A.5 and the non-trivial cases are given below.

Case 1 ($\parr$-S): The inference rule of $\mathcal{B}$ is:

$$\frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top,\mathcal{M}^1,\mathcal{M}^1_\top & \mathcal{A}^0,\mathcal{A}^0_\top,\mathcal{A}^1,\mathcal{A}^1_\top \\ \hline \mathcal{M}^1,\mathcal{M}^1_\top & \mathcal{A}^1,\mathcal{A}^1_\top \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}'^0,\mathcal{M}'^0_\top & \mathcal{A}'^0,\mathcal{A}'^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \parr C} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top,\mathcal{M}^1,\mathcal{M}^1_\top & \mathcal{A}^0,\mathcal{A}^0_\top,\mathcal{A}^1,\mathcal{A}^1_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \parr\text{-S} \tag{3.1}$$

where the input contexts of the right premise are given by $\mathcal{M}^1, \mathcal{M}^1_\top = \mathcal{M}'^0, \mathcal{M}'^0_\top$ and $\mathcal{A}^1, \mathcal{A}^1_\top = \mathcal{A}'^0, \mathcal{A}'^0_\top$. This inference rule is translated into the following inference rule of $\mathcal{B}'$:

$$\frac{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top,\mathcal{M}^1,\mathcal{M}^1_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top,\mathcal{A}^1,\mathcal{A}^1_\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^0_\top,\mathcal{M}^1_\top & \mathcal{A}^1 & \mathcal{A}^0_\top,\mathcal{A}^1_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}'^0,\mathcal{M}'^0_\top,\mathcal{M}^1_*,\mathcal{M}^\top_*} & \multicolumn{2}{c|}{\mathcal{A}'^0,\mathcal{A}'^0_\top,\mathcal{A}^1_*,\mathcal{A}^\top_*} \\ \hline \mathcal{M}^1_* & \mathcal{M}'^0_\top,\mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}'^0_\top,\mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B \parr C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,,\mathcal{M}^0_\top,\mathcal{M}^1,\mathcal{M}^1_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top,\mathcal{A}^1,\mathcal{A}^1_\top} \\ \hline \emptyset & \mathcal{M}'^0_\top,\mathcal{M}^\top_* & \emptyset & \mathcal{A}'^0_\top,\mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} \parr\text{-S}$$

where the output contexts of the left premise and the input contexts of the right premise are given by the following equations:

$$\begin{array}{cc} \mathcal{M}^1, \mathcal{M}^1_\top = \mathcal{M}'^0, \mathcal{M}'^0_\top & \mathcal{A}^1, \mathcal{A}^1_\top = \mathcal{A}'^0, \mathcal{A}'^0_\top \\ \mathcal{M}^0_\top = \mathcal{M}^1_*, \mathcal{M}^\top_* & \mathcal{A}^0_\top = \mathcal{A}^1_*, \mathcal{A}^\top_*. \end{array}$$

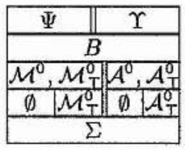
The boxes of the premises have non-empty output contexts. In order to construct the corresponding $\mathcal{B}'$-proof some modifications of the $\mathcal{B}$-proof and $\mathcal{B}'$-proof are necessary. The $\mathcal{B}'$-proof of the left premise in the translation is constructed as follows. First we have the $\mathcal{B}$-proof with the endbox (left premise of the inference rule of $\mathcal{B}$):

$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top,\mathcal{M}^1,\mathcal{M}^1_\top & \mathcal{A}^0,\mathcal{A}^0_\top,\mathcal{A}^1,\mathcal{A}^1_\top \\ \hline \mathcal{M}^1,\mathcal{M}^1_\top & \mathcal{A}^1,\mathcal{A}^1_\top \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array} \tag{3.2}$$

However, it has non-empty output contexts; a $\mathcal{B}$-proof of such a box cannot be translated directly to a $\mathcal{B}'$-proof using the given translations. Using Proposition 12, this $\mathcal{B}$-proof can be modified so that the endbox is:

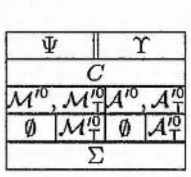
$$\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}$$

Then, since the output contexts are empty, the corresponding proof can be translated into the $\mathcal{B}'$-proof with the endbox:

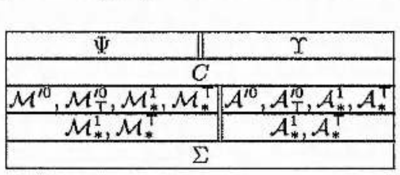


Using Proposition 17 with the $\mathcal{B}$-proof of (3.2) the desired $\mathcal{B}'$-proof is constructible with the left premise as the endbox.

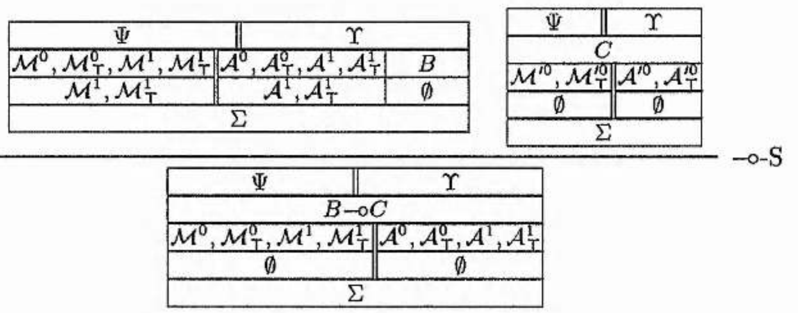
The $\mathcal{B}'$-proof of the right premise can be obtained by translating the $\mathcal{B}$-proof of the right premise to a $\mathcal{B}'$-proof with the following endbox.



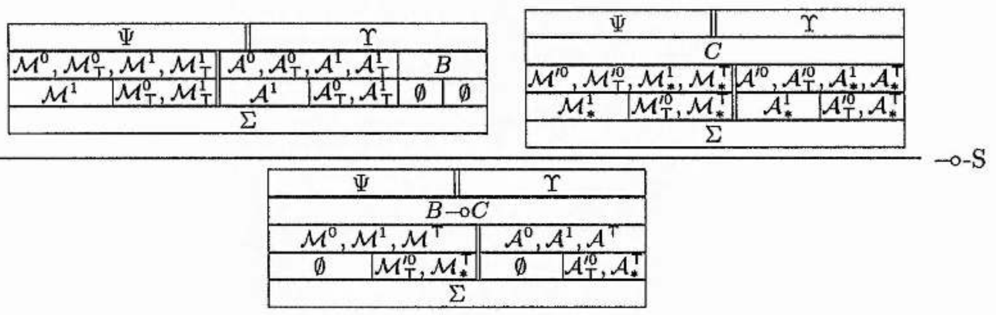
This $\mathcal{B}'$-proof can be modified using Proposition 17. The corresponding $\mathcal{B}$-proof can be obtained by the $\mathcal{B}$-proof of the right premise of (3.1) with a modification using Proposition 12 and the multisets $\mathcal{M}^1_*, \mathcal{M}^\top_*$ and $\mathcal{A}^1_*, \mathcal{A}^\top_*$. The endbox of the desired $\mathcal{B}$-proof is as follows:



Case 2 ($-\!\circ$-S): The inference rule of $\mathcal{B}$ is as follows ($\mathcal{M}^1, \mathcal{M}^1_\top = \mathcal{M}'^0, \mathcal{M}'^0_\top$; $\mathcal{A}^1, \mathcal{A}^1_\top = \mathcal{A}'^0, \mathcal{A}'^0_\top$):



which is translated into the following inference rule of $\mathcal{B}'$:



The $\mathcal{B}'$-proofs of the premises can be constructed similarly to Case 1.

Case 3 (&-R): The inference rule of $\mathcal{B}$ is:

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top & B, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \mathcal{M}'^0, \mathcal{M}'^0_\top & \mathcal{A}{,}^0, \mathcal{A}'^0_\top & C, \mathcal{B}^0, \mathcal{B}'^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top & B\&C, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \ \&\text{-R}$$

where the input contexts of the right premise are given by $\mathcal{M}^0, \mathcal{M}^0_\top = \mathcal{M}'^0, \mathcal{M}'^0_\top$; $\mathcal{A}^0, \mathcal{A}^0_\top = \mathcal{A}'^0, \mathcal{A}'^0_\top$ and $\mathcal{B}^0, \mathcal{B}^0_\top = \mathcal{B}'^0, \mathcal{B}'^0_\top$. This rule is translated as follows:

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{6}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} & \multicolumn{2}{c|}{B, \mathcal{B}^0, \mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{6}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}'^0, \mathcal{M}'^0_\top} & \multicolumn{2}{c|}{\mathcal{A}'^0, \mathcal{A}'^0_\top} & \multicolumn{2}{c|}{C, \mathcal{B}'^0, \mathcal{B}'^0_\top} \\ \hline \emptyset & \mathcal{M}'^0_\top & \emptyset & \mathcal{A}'^0_\top & \emptyset & \mathcal{B}'^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{6}{|c|}{\Psi \quad \| \quad \Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} & \multicolumn{2}{c|}{B\&C, \mathcal{B}^0, \mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top \cap \mathcal{M}'^0_\top & \emptyset & \mathcal{A}^0_\top \cap \mathcal{A}'^0_\top & \emptyset & \mathcal{B}^0_\top \cap \mathcal{B}'^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}} \ \&\text{-R}$$

where the output and the input contexts of the premises are given (as in the $\mathcal{B}$-proof) by:

$$\begin{array}{rcl} \mathcal{M}'^0, \mathcal{M}'^0_\top & = & \mathcal{M}^0, \mathcal{M}^0_\top \\ \mathcal{A}'^0, \mathcal{A}'^0_\top & = & \mathcal{A}^0, \mathcal{A}^0_\top \qquad \text{and} \\ \mathcal{B}'^0, \mathcal{B}'^0_\top & = & \mathcal{B}^0, \mathcal{B}^0_\top \end{array}$$

The side conditions which appear in the '&-R' rule of $\mathcal{B}'$ are satisfied since all strict output contexts are empty. □

3.2.3 Implementation

The main novelty of the calculus $\mathcal{B}'$ appears in the treatment of the '$\top$-R'. The generate-and-test operation of $\mathcal{B}$ which determines the part that is actively consumed by this rule is removed. The new inference rule returns all formulae (apart from the $\top$ connective) and the other rules have permission to consume some formulae from the slack output context. In the implementation of $\mathcal{B}$, the predicate `removed` checked that a component of the analysed formulae does not appear in the output contexts. On the other hand, a predicate, named with `remove`, is declared in the implementation of $\mathcal{B}'$ which also checks the occurrences of a component of the analysed formula, but can consume it if it appears in the slack output context.

```
remove A -> list A -> list A -> list A -> list A -> o.
```

This predicate succeeds in two cases. Firstly, `remove F X Y Z Z` succeeds if the lists `Y` and `Z` together contain less occurrences of the formula `F` than the list `X`. Secondly, `remove F X Y Z W` succeeds if the lists `Y` and `Z` together contain the same number of occurrences of the formula `F` as the list `X`. In this case, the list `W` is instantiated with the a list that contains all elements of `Z` apart from one occurrence of the formula `F`.

The provability of boxes of the calculus $\mathcal{B}'$ is implemented with the predicates *right* and *stoup*, respectively. These predicates have the following types:

```
type right  list o ->                             % Ψ (classical)
            list o -> list o -> list o            % M (linear)
            list o -> list o -> list o            % A (linear)
            list o -> list o -> list o            % B (linear)
            list o -> o.                          % Υ (classical)

type stoup  list o ->                             % Ψ (classical)
            list o -> list o -> list o            % M (linear)
            o ->                                  % stoup
            list o -> list o -> list o            % A (linear)
            list o -> o.                          % Υ (classical)
```

The complete implementation can be found in Appendix B.6. In the following, some implementations of the interesting rules are given together with their specification.

atomic-R rule:

$$\frac{\begin{array}{c}\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline\end{array}\\ \begin{array}{|c||c|c|}\hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top & A,\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top & \mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|c||c|c|c|c|}\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|}\hline \Sigma \\ \hline\end{array}\end{array}}{\begin{array}{c}\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline\end{array}\\ \begin{array}{|c||c|c|}\hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top & \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top & A,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|c||c|c|c|c|}\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|}\hline \Sigma \\ \hline\end{array}\end{array}}\ \textit{atomic}\text{-R}$$

```
% atomic-R rule
right Psi M0M1MT M1 MT A0A1AT A1 AT (A::B0B1BT) B1 BT Upsilon :-
  atomic A,
  right Psi M0M1MT M1 MT (A::A0A1AT) A1 AT' B0B1BT B1 BT Upsilon,
  remove A (A::A0A1AT) A1 AT' AT.
```

⊤-R rule:

$$\frac{}{\begin{array}{c}\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline\end{array}\\ \begin{array}{|c||c|c|}\hline \mathcal{M}^\top & \mathcal{A}^\top & \top,\mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|c||c|c|c|c|}\hline \emptyset & \mathcal{M}^\top & \emptyset & \mathcal{A}^\top & \emptyset & \mathcal{B}^\top \\ \hline\end{array}\\ \begin{array}{|c|}\hline \Sigma \\ \hline\end{array}\end{array}}\ \top\text{-R}$$

```
% top-R rule

right Psi MT nil MT AT nil AT (top::BT) nil BT Upsilon.
```

⅋-S rule:

$$\dfrac{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{C} \\ \hline \multicolumn{2}{|c|}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^1,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^1,\mathcal{A}^\top}} \\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B⅋C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} \\ \hline \mathcal{M}^1\cap\mathcal{M}^1_* & \mathcal{M}^\top\cap\mathcal{M}^1_*,\mathcal{M}^\top_* & \mathcal{A}^1\cap\mathcal{A}^1_* & \mathcal{A}^\top\cap\mathcal{A}^1_*,\mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} ⅋\text{-S}$$

```
% Par-S rule (|-S rule)

stoup Psi MOM1MT M1^M1* MT^M1*MT* (B | C) AOA1AT A1^A1* AT^A1*AT* Upsilon :-
  stoup  Psi MOM1MT M1 MT B AOA1AT A1 AT Upsilon,
  append M1 MT MO*M1*MT*,
  append A1 AT AO*A1*AT*,
  stoup Psi MO*M1*MT* M1* MT* C AO*A1*AT* A1* AT* Upsilon,
  inter M1 M1* M1^M1*,
  inter A1 A1* A1^A1*,
  inter MT M1* MT^M1*,
  inter AT A1* AT^A1*,
  append MT^M1* MT* MT^M1*MT*,
  append AT^A1* AT* AT^A1*AT*.
```

⊸-S rule:

$$\dfrac{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{C} \\ \hline \multicolumn{2}{|c|}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^1,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^1,\mathcal{A}^\top}} \\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B\multimap C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} \\ \hline \mathcal{M}^1\cap\mathcal{M}^1_* & \mathcal{M}^\top\cap\mathcal{M}^1_*,\mathcal{M}^\top_* & \mathcal{A}^1\cap\mathcal{A}^1_* & \mathcal{A}^\top\cap\mathcal{A}^1_*,\mathcal{A}^\top_* \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} \multimap\text{-S}$$

```
% Lolli-S rule (--o-S rule)
stoup Psi MOM1MT M1^M1* MT^M1*MT* (B --o C) AOA1AT A1^A1* AT^A1*AT* Upsilon :-
  stoup  Psi MOM1MT M1 MT C AOA1AT A1 AT Upsilon,
  append M1 MT MO*M1*MT*,
  append A1 AT AO*A1*AT*,
  right Psi MO*M1*MT* M1* MT* AO*A1*AT* A1* AT* (B::nil) nil nil Upsilon,
  inter M1 M1* M1^M1*,
  inter A1 A1* A1^A1*,
  inter MT M1* MT^M1*,
  inter AT A1* AT^A1*,
  append MT^M1* MT* MT^M1*MT*,
  append AT^A1* AT* AT^A1*AT*.
```

&-R rule:

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array} \quad \begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^0,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^0,\mathcal{A}^\top}} & \multicolumn{2}{c|}{\overbrace{C,\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*}^{C,\mathcal{B}^0,\mathcal{B}^\top}}\\ \hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* & \mathcal{B}^1_* & \mathcal{B}^\top_*\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array} \quad \begin{array}{r} \mathcal{M}^1_*\cap\mathcal{M}^0=\emptyset\\ \mathcal{A}^1_*\cap\mathcal{A}^0=\emptyset\\ \mathcal{B}^1_*\cap\mathcal{B}^0=\emptyset \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\&C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\ \hline \mathcal{M}^1,\mathcal{M}^1_* & \mathcal{M}^\top\cap\mathcal{M}^\top_* & \mathcal{A}^1,\mathcal{A}^1_* & \mathcal{A}^\top\cap\mathcal{A}^\top_* & \mathcal{B}^1,\mathcal{B}^1_* & \mathcal{B}^\top\cap\mathcal{B}^\top_*\\ \hline \multicolumn{6}{|c|}{\Sigma}\\ \hline \end{array}}\ \&\text{-R}$$

```
% With-R rule (@-R rule)
right Psi MOM1MT M1M1* MT^MT* AOA1AT A1A1* AT^AT*
                                       ((B @ C)::BOB1BT) B1B1* BT^BT* Upsilon :-
  right Psi MOM1MT M1 MT  AOA1AT A1 AT (B::BOB1BT) B1 BT' Upsilon,
  remove B (B::BOB1BT) B1 BT' BT,
  diff MOM1MT M1 MO,
  diff AOA1AT A1 AO,
  diff BOB1BT B1 BO,
  append MO MT MO*M1*MT*,
  append AO AT AO*A1*AT*,
  append BO BT BO*B1*BT*,
  right Psi MO*M1*MT* M1* MT* AO*A1*AT* A1* AT* (C::BO*B1*BT*) B1* BT* Upsilon,
  inter MO M1* nil,
  inter AO A1* nil,
  inter BO B1* nil,
  append M1 M1* M1M1*,
  append A1 A1* A1A1*,
  append B1 B1* B1B1*,
  inter MT MT* MT^MT*,
  inter AT AT* AT^AT*,
  inter BT BT* BT^BT*.
```

$$\dfrac{}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^\top} & \multicolumn{2}{c|}{\top,\mathcal{B}^\top}\\\hline \emptyset & \mathcal{M}^\top & \emptyset & \mathcal{A}^\top & \emptyset & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \top\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{A,\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{A,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{atomic}\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\bot,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Sigma} & \multicolumn{4}{c|}{\Psi}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B⅋C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ ⅋\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{B,\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\multimap C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \multimap\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi,B} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\supset C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \supset\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B[x\mapsto y],\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{y:\tau,\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\forall_\tau xB,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \forall\text{-R}$$

y is not declared in Σ

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{B,\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{?B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ ?\text{-R}$$

$$\dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}\quad \begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\overbrace{\mathcal{M}^0_*,\mathcal{M}^1_*,\mathcal{M}^\top_*}^{\mathcal{M}^0,\mathcal{M}^\top}} & \multicolumn{2}{c|}{\overbrace{\mathcal{A}^0_*,\mathcal{A}^1_*,\mathcal{A}^\top_*}^{\mathcal{A}^0,\mathcal{A}^\top}} & \multicolumn{2}{c|}{\overbrace{C,\mathcal{B}^0_*,\mathcal{B}^1_*,\mathcal{B}^\top_*}^{C,\mathcal{B}^0,\mathcal{B}^\top}}\\\hline \mathcal{M}^1_* & \mathcal{M}^\top_* & \mathcal{A}^1_* & \mathcal{A}^\top_* & \mathcal{B}^1_* & \mathcal{B}^\top_*\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}\quad \begin{array}{r}\mathcal{M}^1_*\cap\mathcal{M}^0=\emptyset\\ \mathcal{A}^1_*\cap\mathcal{A}^0=\emptyset\\ \mathcal{B}^1_*\cap\mathcal{B}^0=\emptyset\end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\& C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top}\\\hline \mathcal{M}^1,\mathcal{M}^1_* & \mathcal{M}^\top\cap\mathcal{M}^\top_* & \mathcal{A}^1,\mathcal{A}^1_* & \mathcal{A}^\top\cap\mathcal{A}^\top_* & \mathcal{B}^1,\mathcal{B}^1_* & \mathcal{B}^\top\cap\mathcal{B}^\top_*\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-R}$$

Figure 3.4: The right rules in the modified box calculus $\mathcal{B}'$.

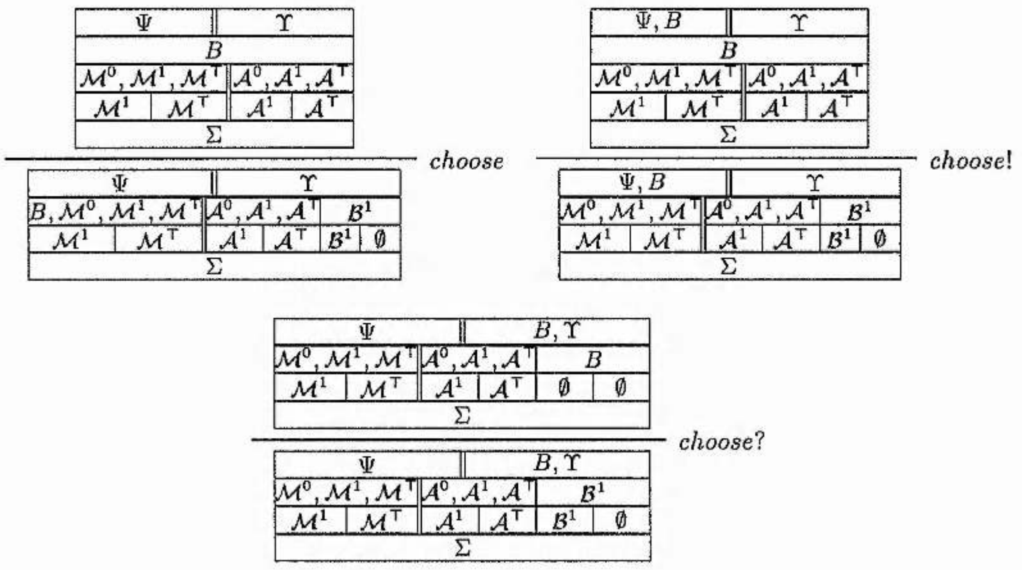


Figure 3.5: The *choose* rules in the modified box calculus $\mathcal{B}'$.

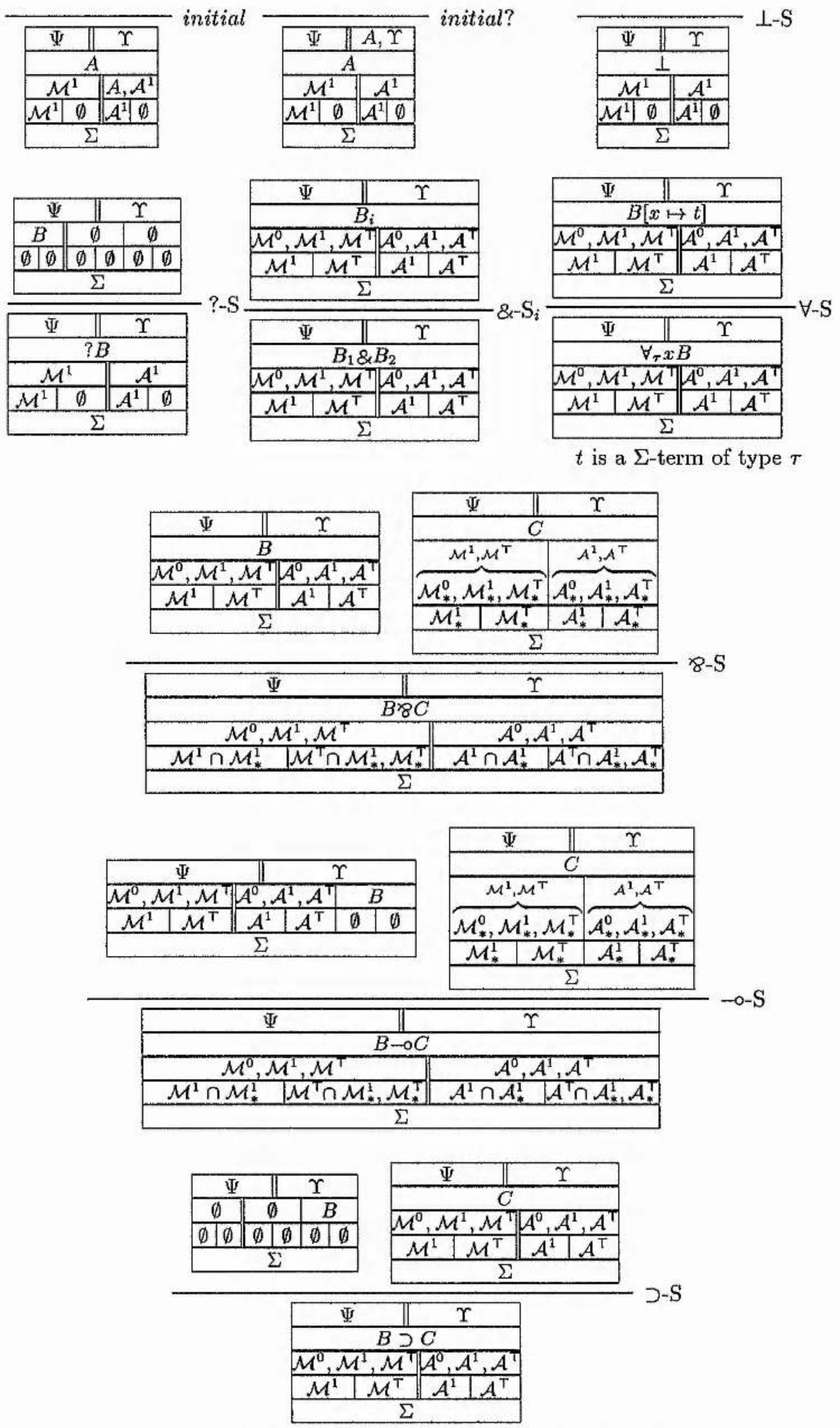


Figure 3.6: The *stoup* rules in the modified box calculus $\mathcal{B}'$.

3.3 Consideration of the Choose Rules

In the previous sections, we addressed the non-determinism that arises from the context splitting. However, there is another significant source of non-determinism which appears in the 'choose' rules. Consider the following implementation of the choose rules:

```
% choose  rule
% choose! rule
% choose? rule

right Psi BMOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  membNrest B BMOM1 MOM1,
  stoup Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Psi,
  stoup  Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Upsilon,
  right Psi MOM1 M1 AOA1 A1 (B::nil) nil Upsilon.
```

In Section 2.5, we presented an explanation why we implemented these rules using the order as illustrated in the program code. The 'choose' and 'choose!' rule serve for choosing a program formula which will be analysed subsequently as *stoup* formula; the 'choose?' rule chooses a formula from the classical context Υ of the antecedent which will be considered as a goal formula.

All three rules use a generate-and-test operation for choosing a formula (i.e., `member`, `membNrest`). This strategy is particularly inefficient since the proof construction (which follows when a choice is made) is rather expensive. In case of the 'choose' and 'choose!' rule, however, it is feasible to decide whether the chosen formula is a candidate for a proof or it is predictable that the construction will fail.

The idea behind this *choose test* is the fact that a *stoup* formula (apart from some exceptions) has to lead to a proof where it is focused on that particular formula. Furthermore in these cases, the proof has to terminate with either the 'initial', 'initial?' or '$\perp$-S' rule. Therefore, it is possible to decompose recursively the chosen *stoup* formula independently from the proof search.

We can say: a formula F is a *candidate* for a proof if it passes the following test. By case analysis:

- F is an atom, then it must unify with an atom from the succedent;
- F is '$\perp$', then it passes the test;
- F is of the the form $B \parr C$, then both components must pass the test;
- F is of the form $B \& C$, then at least one component must pass the test;
- F is an implication, then the implicatum must pass the test;
- F is of the form $\forall x B$, then the formula $B[x \mapsto y]$ must pass the test where y is a fresh variable;

- F is composed with an outermost non-primitive connective, then the translated form of B must pass the test (e.g., $B^{\perp}$ then $(B \multimap \perp)$ is considered);

- F is of the form $?B$, then it passes the test since this formula is removed to the linear context of the succedent and possibly another *stoup* formula will be chosen and

- F is $\top$, then the test fails.

The implementation of the 'choose test' is given in Appendix B.3. The modified 'choose' rules are as follows:

```
% choose  rule
% choose! rule
% choose? rule

right Psi BMOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  membNrest B BMOM1 MOM1,
   append AOA1 Upsilon Rightside,
   choosetest B Rightside,
  stoup Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Psi,
   append AOA1 Upsilon Rightside,
   choosetest B Rightside,
  stoup  Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Upsilon,
  right Psi MOM1 M1 AOA1 A1 (B::nil) nil Upsilon.
```

The implementation of the 'choose' rules for $\mathcal{B}'$ is similar to the presented implementation for $\mathcal{B}$.

3.4 Problems with Empty Headed Implications

Let us consider what is achieved so far. Miller proved for FORUM that it can be seen as an abstract logic programming language, in the sense that uniform proofs (Definition 5) are complete for the language. The box calculi presented here provide a deterministic method for the context splitting which improves significantly the efficiency compared with $\mathcal{F}$ and $\mathcal{F}'$. However, Miller avoided calling FORUM a 'logic programming language' and generally referred to FORUM as a *specification logic.* The problem concerning an implementation of FORUM as a logic programming language arises from the $\perp$-headed implications which appear to be problematic when they occur in the classical context of the antecedent. Consider, the following proof fragment (for expository purposes we present this fragment using the sequent calculus $\mathcal{F}'$; however, the same problem appears, thus far, in all presented calculi for FORUM):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Sigma: A\multimap\bot;\emptyset \Longrightarrow A;A;\emptyset} \quad \dfrac{}{\Sigma: A\multimap\bot;\emptyset \xRightarrow{\bot} \emptyset;\emptyset}\,\bot\text{-S}}{\Sigma: A\multimap\bot;\emptyset \xRightarrow{A\multimap\bot} A;\emptyset}\,\multimap\text{-S}}{\Sigma: A\multimap\bot;\emptyset \Longrightarrow A;\emptyset;\emptyset}\,\textit{choose}}{\Sigma: A\multimap\bot;\emptyset \Longrightarrow A;A;\emptyset}\,\textit{atomic}\text{-R} \qquad \dfrac{}{\Sigma: A\multimap\bot;\emptyset \xRightarrow{\bot} \emptyset;\emptyset}\,\bot\text{-S}}{\dfrac{\Sigma: A\multimap\bot;\emptyset \xRightarrow{A\multimap\bot} A;\emptyset}{\Sigma: A\multimap\bot;\emptyset \Longrightarrow A;\emptyset;\emptyset}\,\textit{choose}}\,\multimap\text{-S}$$

It is always possible to choose $\bot$-headed implication for the *stoup* position. However, it does not help in proving the sequent since the right premise expects empty contexts and does not consume any formula from the contexts. On the other hand, an elimination of such formulae is not desired since they play an important role when proving a non-primitive connective (see Section 2.3.2). Thus far, there is no satisfactory solution described in the literature concerning this problem.

Chapter 4

Examples

4.1 A Simple Object Logic

FORUM's expressivity can be illustrated by specifying and implementing object-logics (many examples can be found in [Miller, 1996]). In the following, a very simple object logic $\mathcal{O}$ based on single succedents is implemented in FORUM (suggested by Miller). The language of formulae ($\mathcal{L}$) in $\mathcal{O}$ is defined as follows:

$$\begin{aligned}\mathcal{L}:\quad G &::= \mathit{true} \mid A \mid G_1 \wedge G_2\\ D &::= A{:}\text{-}G\end{aligned}$$

where A stands for atomic formulae, G for goal formulae and D for program formulae. The symbol ':-' represents the reverse intuitionistic implication and is borrowed from the concrete syntax of PROLOG in order to distinguish it from the implications used in FORUM. The sequents of $\mathcal{O}$ are:

$$\Delta \Longrightarrow G$$

where Δ is a multiset of program formulae and G is a goal formula. A program formula of the form 'A:-*true*' is called a *fact*. (Usually, facts are written as 'A' in logic programming languages. However for convenience of what follows, the facts are written as implications.) The inference rules of $\mathcal{O}$ are as follows:

$$\frac{}{\Delta \Longrightarrow \mathit{true}}\ \mathit{Axiom} \qquad \frac{A{:}\text{-}G, \Delta \Longrightarrow G}{A{:}\text{-}G, \Delta \Longrightarrow A}\ {:}\text{-L} \qquad \frac{\Delta \Longrightarrow B \quad \Delta \Longrightarrow C}{\Delta \Longrightarrow B \wedge C}\ \wedge\text{-R}$$

In the following, three implementations of $\mathcal{O}$ are given which represent the connectives and formulae of $\mathcal{O}$ using the appropriate connectives of FORUM. The atomic formulae of $\mathcal{O}$ are translated into atomic formulae which inhabit the type `o`. The variable Σ stands for a set of type declarations. In what follows, Σ (in FORUM sequents) contains all the type declarations of atomic formulae and the used connectives.

For expository purposes, each implementation has a triple as name which represents the corresponding FORUM connectives. It is shown in each case that the implementation is a faithful representation of the object-logic $\mathcal{O}$:

object logic $\mathcal{O}$:	implementations:
$(\mathit{true}, \wedge, {:}\text{-})$	$(\top, \&, \multimap)$, $(\top, \otimes, \multimap)$ $(\bot, \parr, \multimap)$

Representation ($\top, \&, \multimap$):

The formulae of $\mathcal{O}$ are translated as follows:

$$\begin{aligned} \mathit{true}^\circ &\stackrel{\text{def}}{=} \top; \\ A^\circ &\stackrel{\text{def}}{=} A, \text{ where } A \text{ is atomic;} \\ (B \wedge C)^\circ &\stackrel{\text{def}}{=} B^\circ \& C^\circ; \\ (C\text{:-}B)^\circ &\stackrel{\text{def}}{=} B^\circ \multimap C^\circ. \end{aligned}$$

Proposition 19 *A sequent $\Delta \Longrightarrow G$ of $\mathcal{O}$ is provable if and only if the* FORUM *sequent $\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; G^\circ; \emptyset$ is provable. (In what follows Δ° stands for a multiset where each formula is translated.)*

Proof: Induction on the height of proofs. The translations of the inference rules are given below. The order of FORUM's inference rules at the right-hand side is forced by the proof search strategy. Therefore, the vice-versa translation follows the same scheme.

$$\frac{}{\Delta \Longrightarrow \mathit{true}}\ \mathit{Axiom} \qquad \Leftrightarrow \qquad \frac{}{\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; \top; \emptyset}\ \top\text{-R}$$

$$\frac{\Delta \Longrightarrow B \quad \Delta \Longrightarrow C}{\Delta \Longrightarrow B \wedge C}\ \wedge\text{-R} \qquad \Leftrightarrow \qquad \frac{\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; B^\circ; \emptyset \quad \Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; C^\circ; \emptyset}{\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; B^\circ \& C^\circ; \emptyset}\ \&\text{-R}$$

$$\frac{A\text{:-}G, \Delta \Longrightarrow G}{A\text{:-}G, \Delta \Longrightarrow A}\ \text{:-L} \Leftrightarrow \qquad \cfrac{\cfrac{\cfrac{\Sigma\colon G^\circ \multimap A, \Delta^\circ; \emptyset \Longrightarrow \emptyset; G^\circ; \emptyset \quad \cfrac{}{\Sigma\colon G^\circ \multimap A, \Delta^\circ; \emptyset \overset{A}{\Longrightarrow} A; \emptyset}\ \mathit{initial}}{\Sigma\colon G^\circ \multimap A, \Delta^\circ; \emptyset \overset{G^\circ \multimap A}{\Longrightarrow} A; \emptyset}\ \multimap\text{-S}}{\Sigma\colon G^\circ \multimap A, \Delta^\circ; \emptyset \Longrightarrow A; \emptyset; \emptyset}\ \mathit{choose!}}{\Sigma\colon G^\circ \multimap A, \Delta^\circ; \emptyset \Longrightarrow \emptyset; A; \emptyset}\ \mathit{atomic}\text{-R}$$

Representation ($\top, \otimes, \multimap$):

The formulae of $\mathcal{O}$ are translated using the following translation:

$$\begin{aligned} \mathit{true}^\circ &\stackrel{\text{def}}{=} \top; \\ A^\circ &\stackrel{\text{def}}{=} A, \text{ where } A \text{ is atomic;} \\ (B \wedge C)^\circ &\stackrel{\text{def}}{=} B^\circ \otimes C^\circ; \\ (C\text{:-}B)^\circ &\stackrel{\text{def}}{=} B^\circ \multimap C^\circ. \end{aligned}$$

Proposition 20 *A sequent $\Delta \Longrightarrow G$ of $\mathcal{O}$ is provable if and only if the* FORUM *sequent $\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; G^\circ; \emptyset$ is provable.*

Proof: Induction on the height of proofs. For more convenience, a '$\otimes$-R' rule is defined which is a shorthand for the following proof fragment (the rule '*Trans*' stands for the translation of a non-primitive into a primitive connective):

$$\frac{\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; B^\circ; \emptyset \quad \Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; C^\circ; \emptyset}{\Sigma\colon \Delta^\circ; \emptyset \Longrightarrow \emptyset; B^\circ \otimes C^\circ; \emptyset}\ \otimes\text{-R}$$

$$\Updownarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; B^\circ;\emptyset \quad \dfrac{}{\Sigma\colon \Delta^\circ;\emptyset \xRightarrow{\perp} \emptyset;\emptyset}\,\perp\text{-S}}{\Sigma\colon \Delta^\circ;\emptyset \xRightarrow{B^\circ \multimap \perp} \emptyset;\emptyset}\,\multimap\text{-S} \quad \dfrac{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; C^\circ;\emptyset \quad \dfrac{}{\Sigma\colon \Delta^\circ;\emptyset \xRightarrow{\perp} \emptyset;\emptyset}\,\perp\text{-S}}{\Sigma\colon \Delta^\circ;\emptyset \xRightarrow{C^\circ \multimap \perp} \emptyset;\emptyset}\,\multimap\text{-S}}{\Sigma\colon \Delta^\circ;\emptyset \xRightarrow{(B^\circ \multimap \perp)\parr(C^\circ \multimap \perp)} \emptyset;\emptyset}\,\parr\text{-S}}{\Sigma\colon \Delta^\circ;(B^\circ \multimap \perp)\parr(C^\circ \multimap \perp) \Longrightarrow \emptyset;\emptyset;\emptyset}\,\mathit{choose}}{\Sigma\colon \Delta^\circ;(B^\circ \multimap \perp)\parr(C^\circ \multimap \perp) \Longrightarrow \emptyset;\perp;\emptyset}\,\perp\text{-R}}{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset;((B^\circ \multimap \perp)\parr(C^\circ \multimap \perp))\multimap\perp;\emptyset}\,\multimap\text{-R}}{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; B^\circ \otimes C^\circ;\emptyset}\,\mathit{Trans}$$

Translations (similar to the representation $(\top,\&,\multimap)$):

$$\dfrac{}{\Delta \Longrightarrow \mathit{true}}\,\mathit{Axiom} \qquad \Leftrightarrow \qquad \dfrac{}{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset;\top;\emptyset}\,\top\text{-R}$$

$$\dfrac{\Delta \Longrightarrow B \quad \Delta \Longrightarrow C}{\Delta \Longrightarrow B \wedge C}\,\wedge\text{-R} \quad \Leftrightarrow \quad \dfrac{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; B^\circ;\emptyset \quad \Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; C^\circ;\emptyset}{\Sigma\colon \Delta^\circ;\emptyset \Longrightarrow \emptyset; B^\circ \otimes C^\circ;\emptyset}\,\otimes\text{-R}$$

$$\dfrac{A\text{:-}G, \Delta \Longrightarrow G}{A\text{:-}G, \Delta \Longrightarrow A}\,\text{:-L} \;\Leftrightarrow\; \dfrac{\dfrac{\dfrac{\Sigma\colon G^\circ \multimap A, \Delta^\circ;\emptyset \Longrightarrow \emptyset; G^\circ;\emptyset \quad \dfrac{}{\Sigma\colon G^\circ \multimap A, \Delta^\circ;\emptyset \xRightarrow{A} A;\emptyset}\,\mathit{initial}}{\Sigma\colon G \multimap A, \Delta^\circ;\emptyset \xRightarrow{G^\circ \multimap A} A;\emptyset}\,\multimap\text{-S}}{\Sigma\colon G^\circ \multimap A, \Delta^\circ;\emptyset \Longrightarrow A;\emptyset;\emptyset}\,\mathit{choose!}}{\Sigma\colon G^\circ \multimap A, \Delta^\circ;\emptyset \Longrightarrow \emptyset; A;\emptyset}\,\mathit{atomic}\text{-R}$$

Both translations described above are traditional ways to represent the object-logic $\mathcal{O}$. The translation of the program formulae using '$\multimap$' could be replaced by a translation using FORUM's intuitionistic implication '$\supset$' since the program formulae are in the classical context.

Representation $(\perp, \parr, \multimap)$:

A non-trivial embedding is described in the following. The formulae of $\mathcal{O}$ are translated using the translation:

$$\begin{aligned} \mathit{true}^\circ &\stackrel{\text{def}}{=} \perp; \\ A^\circ &\stackrel{\text{def}}{=} A, \text{ where } A \text{ is atomic;} \\ (B \wedge C)^\circ &\stackrel{\text{def}}{=} B^\circ \parr C^\circ; \\ (C\text{:-}B)^\circ &\stackrel{\text{def}}{=} B^\circ \multimap C^\circ. \end{aligned}$$

Proposition 21 *A sequent $\Delta \Longrightarrow G$ of $\mathcal{O}$ is provable if and only if the* FORUM *sequent $\Sigma\colon \Delta^\circ; \perp \Longrightarrow \emptyset; G^\circ; \emptyset$ is provable.*

This embedding does not preserve the structure of $\mathcal{O}$-proofs and uses a '$\perp$' as an additional linear program formula. Consequently, the proofs cannot be shown by a structural induction using a mapping of inference rules. The soundness and completeness proofs use an intermediate notation for $\mathcal{O}$, which is called *goal notation.* Intuitively, the goal notation represents a multiset of premises that have not been proven yet. The program Δ is unchanged in each inference rule of $\mathcal{O}$; therefore, it is possible to treat the program Δ independently from the goal formula. The multisets of the goal notation can be modified by some rewriting rules which correspond to the inference rules of $\mathcal{O}$. The corresponding pairs are given as follows (Γ stands for a multiset of goal formulae.):

$$\frac{}{\Delta \Longrightarrow true}\ Axiom \quad \Leftrightarrow \quad (true, \Gamma) \mapsto \Gamma$$

$$\frac{\Delta \Longrightarrow B \quad \Delta \Longrightarrow C}{\Delta \Longrightarrow B \wedge C}\ \wedge\text{-R} \quad \Leftrightarrow \quad (B \wedge C, \Gamma) \mapsto (B, C, \Gamma)$$

$$\frac{\Delta \Longrightarrow G}{\Delta \Longrightarrow A}\ \text{:--L} \quad \Leftrightarrow \quad (A, \Gamma) \mapsto (G, \Gamma)$$

where the program Δ contains a formula A:-G.

Definition 22 *Let Δ be a multiset of program formulae of $\mathcal{O}$. A Δ-sequence is a sequence of multiset rewritings where:*

1. *the last member is the empty multiset;*
2. *each step corresponds to either a splitting of a conjunction, removal of 'true' or a member of Δ.*

Consider the following proof in $\mathcal{O}$:

$$\frac{\dfrac{\dfrac{}{a\text{:-}true, b\text{:-}true \Longrightarrow true}\ \text{:--L}}{a\text{:-}true, b\text{:-}true \Longrightarrow a}\ \text{:--L} \quad \dfrac{\dfrac{}{a\text{:-}true, b\text{:-}true \Longrightarrow true}\ \text{:--L}}{a\text{:-}true, b\text{:-}true \Longrightarrow b}\ \text{:--L}}{a\text{:-}true, b\text{:-}true \Longrightarrow a \wedge b}\ \wedge\text{-R}$$

The following two Δ-sequences both correspond to this sequent proof:

$$(a \wedge b) \mapsto (a, b) \mapsto (true, b) \mapsto (b) \mapsto (true) \mapsto \emptyset$$

$$(a \wedge b) \mapsto (a, b) \mapsto (true, b) \mapsto (true, true) \mapsto (true) \mapsto \emptyset$$

Lemma 23 *A sequent $\Delta \Longrightarrow G$ of $\mathcal{O}$ is provable if and only if there is a Δ-sequence starting with $\{\!|G|\!\}$.*

Proof:

Soundness:	Induction on the length of goal sequences. A Δ-sequence of multisets that starts with $\{\!|G|\!\}$ and represents the open premises (premises that have not been proven yet) in the proof in $\mathcal{O}$. Thus, it is possible to translate the sequence into a $\mathcal{O}$-proof using the defined correspondences.
Completeness:	Structural induction on the height of $\mathcal{O}$-proofs using the defined correspondences.

The proof of Proposition 21 is still a difficult matter because the goal notation does not provide a one-to-one correspondence to Forum proofs. The difficulty is caused by Forum's rather restricted proof search. The particular problematic rule in a translation is the '$\multimap$-S' inference rule. In the goal notation, the corresponding rewriting rule is applicable whenever an atom occurs. On the other hand, the '$\multimap$-S' inference rule is only applicable if the succedent consists entirely of atomic formulae.

There are two ways to achieve the soundness and completeness of $\mathcal{O}$ with respect to Forum. A normal form of Δ-sequences in the goal notation could be introduced so that the normal Δ-sequences and Forum proofs are in a one-to-one correspondence. In this case, an appropriate notion of permutable Δ-sequences must be defined. However, the proof of Proposition 21 will be shown by translating any Δ-sequence into a CLL proof (for our purposes, we use a fragment of CLL). Subsequently, the complete proof with respect to FORUM can be achieved using the soundness (Theorem 1) and completeness (Theorem 3) theorem of FORUM relative to CLL [Miller, 1996]. These theorems provide the necessary translation between CLL and FORUM proofs.

Proof of Proposition 21: For soundness, we use structural induction on the length of Δ-sequences. For completeness, structural induction on the height of CLL proofs. (Note, that we use a fragment of CLL. The permitted formulae are specified above.)

Translations ($!(\Delta^\circ)$ stands for a multiset where each formula is decorated with a '!'):

$$(true) \mapsto \emptyset \quad \Leftrightarrow \quad \cfrac{\cfrac{}{\bot, !(\Delta^\circ) \Longrightarrow \emptyset}\ \bot\text{-L}}{\bot, !(\Delta^\circ) \Longrightarrow \bot}\ \bot\text{-R}$$

$$(true, \Gamma) \mapsto \Gamma \quad \Leftrightarrow \quad \cfrac{\bot, !(\Delta^\circ) \Longrightarrow \Gamma^\circ}{\bot, !(\Delta^\circ) \Longrightarrow \bot, \Gamma^\circ}\ \bot\text{-R}$$

$$(B \wedge C, \Gamma) \mapsto (B, C, \Gamma) \quad \Leftrightarrow \quad \cfrac{\bot, !(\Delta^\circ) \Longrightarrow B^\circ, C^\circ, \Gamma^\circ}{\bot, !(\Delta^\circ) \Longrightarrow B^\circ \parr C^\circ, \Gamma^\circ}\ \parr\text{-R}$$

$$(A, \Gamma) \mapsto (G, \Gamma) \quad \Leftrightarrow \quad \cfrac{\bot, !(\Delta^\circ) \Longrightarrow G^\circ, \Gamma^\circ}{\bot, !(\Delta^\circ) \Longrightarrow A, \Gamma^\circ}\ \text{BC}$$

where Δ° contains a formula $G^\circ \multimap A^\circ$. The 'BC' rule is a specific instance of '$\multimap$-L' rule. □

Two Examples for the representations of $\mathcal{O}$:

In the following, a program Δ and a goal G of the object-logic $\mathcal{O}$ will be represented in FORUM using the translations $(\top, \&, \multimap)$ and $(\bot, \parr, \multimap)$ (the translation using $(\top, \otimes, \multimap)$ is similar to the translation $(\top, \&, \multimap)$). The program Δ is as follows:

c:-$b \wedge a$.
b:-a.
a:-$true$.

The goal formula is $c \wedge a$; the sequent which will be proven is:

$$\Delta \Longrightarrow c \wedge a$$

In what follows, the right premise of any inference that can be proven in a single step is omitted in order to improve the readability of the given proofs. The proof of the sequent in $\mathcal{O}$ is as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\Delta \Longrightarrow true}\ Axiom}{\Delta \Longrightarrow a}\ \text{:-L}}{\Delta \Longrightarrow b}\ \text{:-L} \qquad \cfrac{\cfrac{}{\Delta \Longrightarrow true}\ Axiom}{\Delta \Longrightarrow a}\ \text{:-L}}{\Delta \Longrightarrow b \wedge a}\ \wedge\text{-R}}{\Delta \Longrightarrow c}\ \text{:-L} \qquad \cfrac{\cfrac{}{\Delta \Longrightarrow true}\ Axiom}{\Delta \Longrightarrow a}\ \text{:-L}}{\Delta \Longrightarrow c \wedge a}\ \wedge\text{-R}$$

Representation $(\top, \&, \multimap)$:
The translated program Δ° is: $b\&a\multimap c,\ a\multimap b,\ \top\multimap a$;
the translated goal G is: $c\&a$.
A proof in $\mathcal{F}'$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;\top;\emptyset}\,\top\text{-R}}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{\top\multimap a}a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{a\multimap b}b;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow b;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;b;\emptyset}\,\textit{atomic-R}\qquad\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;\top;\emptyset}\,\top\text{-R}}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{\top\multimap a}a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;b\&a;\emptyset}\,\&\text{-R}}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{b\&a\multimap c}c;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow c;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;c;\emptyset}\,\textit{atomic-R}\qquad\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;\top;\emptyset}\,\top\text{-R}}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{\top\multimap a}a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\emptyset\Longrightarrow\emptyset;c\&a;\emptyset}\,\&\text{-R}$$

Representation $(\bot, \parr, \multimap)$:
The translated program Δ° is: $b\parr a\multimap c,\ a\multimap b,\ \bot\multimap a$;
the translated goal G is: $c\parr a$.
A possible proof in $\mathcal{F}'$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma\colon\Delta^\circ;\emptyset\xRightarrow{\bot}\emptyset;\emptyset}\,\bot\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow\emptyset;\emptyset;\emptyset}\,\textit{choose}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow\emptyset;\bot;\emptyset}\,\bot\text{-R}}{\Sigma\colon\Delta^\circ;\bot\xRightarrow{\bot\multimap a}a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a;\bot;\emptyset}\,\bot\text{-R}}{\Sigma\colon\Delta^\circ;\bot\xRightarrow{\bot\multimap a}a,a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a,a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\bot\xRightarrow{a\multimap b}b,a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow b,a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow b,a;\bot;\emptyset}\,\bot\text{-R}}{\Sigma\colon\Delta^\circ;\bot\xRightarrow{\bot\multimap a}a,b,a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a,b,a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a,b;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a;b,a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow a;b\parr a;\emptyset}\,\parr\text{-R}}{\Sigma\colon\Delta^\circ;\bot\xRightarrow{b\parr a\multimap c}c,a;\emptyset}\,{\multimap}\text{-S}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow c,a;\emptyset;\emptyset}\,\textit{choose!}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow c;a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow\emptyset;c,a;\emptyset}\,\textit{atomic-R}}{\Sigma\colon\Delta^\circ;\bot\Longrightarrow\emptyset;c\parr a;\emptyset}\,\parr\text{-R}$$

To sum up, we finally compare the translations of $\mathcal{O}$. The proof search can be done more efficiently for the representations ($\top$,&,$\multimap$) and ($\bot$,⅋,$\multimap$) than for ($\top$,$\otimes$,$\multimap$) since they use only primitive connectives. As mentioned in Section 2.3.2, the proof search for primitive connectives is goal-conducted. While the first two representations (i.e., ($\top$,&,$\multimap$) and ($\top$,$\otimes$,$\multimap$)) are straightforward representations of $\mathcal{O}$ which strongly mirror $\mathcal{O}$-proofs, the last representation is to some extent very interesting. The multisets of the goal notation which can be easily implemented by a parallel algorithm using several processes is represented by FORUM's multiple succedents, i.e., this translation simulates the behaviour of some independent processes. Such a parallel behaviour is difficult to represent in a single succedent logic because the single goal represents only one "active" process and a technique is required in order to "suspend" and "activate" processes.

4.2 A Conjunctive Planner in FORUM

FORUM can be used for implementing a *deductive planning system*. A detailed treatment of planning problems and systems is far beyond this thesis (see [Weld, 1994] for a good survey); however, some important principles are introduced.

Our approach is a small *backward* planner (also called *regressive* planner) for conjunctive planning problems. A planning problem can be characterised by an *initial* situation, a *goal* situation and some *actions* that can be performed. The present implementation is restricted so that each situation consists of a multiset of atomic formulae. Similar to the description of transition in Petri-nets, the actions are characterised by two sets. One of them represents the *preconditions* and the other one the *effects*. An action may be performed in a situation only when its preconditions are satisfied, i.e., they form a submultiset of the current situation. Subsequently, the execution of an action replaces the preconditions by the action's effects.

A planning problem is completely symmetric; therefore, it does not matter if we start with the initial situation and attempt to find a plan for a goal situation or if we start with the goal and attempt to reach the initial situation. (In fact, this method is regarded as more efficient in the average case than the vice-versa method.) FORUM's uniform proofs are more appropriate for modelling a backward planner.

An early approach towards a translation of planning systems into linear logic appeared in [Masseron et al., 1990]. They used a fragment of ILL including the '$\otimes$' and '$\oplus$' connectives and their unit elements. The proofs are constructed using the rules for the connectives, some axioms and a 'Cut' rule. On the other hand, the authors of Lygon presented two examples of planning type problems in their logic programming language. This representation uses the linear connectives '$\otimes$', '&', and '$\multimap$'; [Harland & Winikoff, 1996c].

However, the major source for our approach comes from a planning system (introduced in [Hölldobler & Schneeberger, 1990]) using multiset terms and equational logic programming. This calculus is very expressive as far as planning problems are concerned. A detailed treatment of their work is omitted here since we only want to implement some features which can be represented in this system and which are not provided by the other approaches in linear logic. A survey of the planning systems in equational logic programming can be found in [Schneeberger, 1992].

For expository purposes, we introduce an example that is taken from [Große et al., 1992]. There, the initial situation is having a dollar note (d) and a quarter (q); the goal situation is getting a lemonade (l) which costs three quarters. Two actions can be performed which are called *get_change* and *get_lemonade* and may represent a cashier and a vending machine, respectively. The initial situation, the goal situation and the actions can be formalised as follows:

	preconditions:	effects:
get_change (g_c):	d	q, q, q, q
get_lemonade (g_l):	q, q, q	l
initial situation:	d, q	
goal situation:	l	

Clearly, a solution to this problem is to change the dollar note into four quarters and to buy subsequently a lemonade with three of the five quarters.

Similar to the Petri-net encoding from Section 1.1, the actions are represented by the linear implication '$\multimap$'. However, we use the '$⅋$' in order to represent a multiset of resources because this connective is primitive and can be treated more efficiently than the '$\otimes$' connective. This technique was briefly outlined in [Cervesato, 1995] for Petri-nets, but details were omitted. The translation into linear logic is as follows:

get_change (g_c):	$d \multimap q ⅋ q ⅋ q ⅋ q$
get_lemonade (g_l):	$q ⅋ q ⅋ q \multimap l$
initial situation:	$d ⅋ q$
goal situation:	l.

In our translation, the goal situation is regarded as a goal formula; the initial situation and the actions are regarded as program formulae. The initial situation and the goal situation are translated so that they can be considered as linear resources (i.e, they go into the linear context Δ and $\mathcal{B}$, respectively). However, the actions might be used more than once and therefore, have to be regarded as classical formulae which can be reused. Consequently, the actions are formulae of the classical context Ψ.

Nevertheless, a problem arises from the linear resources which are not necessary to achieve the goal situation. In our example, we need only three quarters when buying a lemonade, but have five quarters after changing the dollar note. However, the two remaining quarters must be consumed somehow in order to construct a proof. As a first attempt, we could encode the '$\top$' connective which would have the purpose to "consume" remaining resources in our translation. The aforementioned features which we want to model in our translation are the following; it should be possible to ask for a goal situation: 'Which are the remaining resources?' and 'Which resources are necessary for achieving a certain goal?'. The desired control over the "consumption" of formulae vanishes when translating the planning problem using the '$\top$' connective because it "consumes" the formulae somewhere inside the proof. Therefore a different approach is described in the following.

A variable X is introduced in the succedent which has the purpose of accommodating all remaining formulae. Thus the example above is represented by the following FORUM sequent (it is assumed that Σ contains the appropriate declarations[1]):

$$\Sigma : g_c, g_l ; d ⅋ q \Longrightarrow \emptyset ; l ⅋ X ; \emptyset$$

[1] i.e., the atomic formulae d, q, l have the type o

where the variable X can be of the form ('$\bot$' stands for "no remaining resources"):

$$X ::= T \mid X \parr X$$
$$T ::= q \mid d \mid \bot.$$

FORUM returns[2], when proving this sequent, the intended term $q \parr q$ for X. On the other hand, the following sequent represents a question: Which resources are necessary for a certain goal situation:

$$\Sigma\colon g_c, g_l; Y \Longrightarrow \emptyset; l; \emptyset$$

where the variable Y can be of the form:

$$Y ::= T \mid Y \parr Y$$
$$T ::= q \mid d.$$

FORUM returns the desired term $q \parr q \parr q$ for Y. In cases where a planning problem is not solvable, FORUM does not find a proof for the corresponding sequent (but a soundness and completeness theorem is omitted).

The problematic parts in this implementation come along with the restriction of the variables X and Y. In order to receive the desired answers, a predicate **Restrict** and **Ground** are defined on the metalevel of FORUM (which is in our case the actual implementation in Terzo). These predicates are defined for the variable X as follows:

```
Ground q.
Ground d.
Ground ⊥.
Restrict B :-Ground B.
Restrict (B⅋C) :-Restrict B, Ground C.
```

Clearly, our planner uses an inefficient 'generate-and-test' algorithm. However, it could be replaced by a proof search which delays the substitution of the variables X and Y as long as possible and where the substitutions can be restricted to a certain class of terms. A similar approach to that of [Nadathur & Miller, 1990] can be applied for the fragment of FORUM described above. Consequently, the unification can be computed efficiently, but these facilities are not yet provided by the Terzo implementation.

To sum up, much remains to be done in order to implement a deductive planner in FORUM which is comparable to the planner using equational logic programming. For example, a technique must be investigated which extracts a plan (i.e., a sequence of actions) out of the FORUM proof.

4.3 A Program for Computing the Fibonacci Series

The following two examples serve for illustrating some experimental evidence of the speed-up that is achieved by the new box calculi. The first example is taken from [Hodas, 1994]; it implements a computation of the Fibonacci series which stores each of the computed values

[2] Note, FORUM also returns $q \parr q \parr \bot$, $q \parr q \parr \bot \parr \bot$,... for X as next answers.

for a later inference. Thus, the values do not have to be recomputed as in the well-known but rather inefficient implementation of the Fibonacci series.

The implementation is accomplished in a continuation-passing style which is permitted by a language with the support of predicate quantification. This technique has become invaluable, for example, in λProlog and Lolli. The program uses a simple arithmetic of integers which are defined, in the usual way, using a constant **z** (zero), the successor-function **s** and a predicate **pl** (plus) The program is as follows:

```
% natural numbers and Fibonacci series with memoising
% based on a program which appeared in Hodas' PhD-thesis 1994
% (rewritten in Forum)                    last modified 26.09.96

kind  nat  type.

type z nat.
type s nat -> nat.

type pl    nat -> nat -> nat -> o.

pl X z X.
pl X (s Y) (s Z) o-- pl X Y Z.

type fib   nat -> nat -> o.
type fiba  nat -> nat -> o -> o.
type memo  nat -> nat -> o.

fib N F  o-- (memo z z --o memo (s z) (s z)
           --o memo (s z) (s z) --o fiba N F top).

fiba N F G o-- memo N F x G .
fiba (s(s N)) F G o-- ( fiba N F1
                      ( fiba (s N) F2
                          ( pl F1 F2 F x
                            (( memo (s (s N)) F x memo (s (s N)) F) --o G )
                          )
                       )
                    ).
```

Since the stored values are represented as linear resources which appear in the linear program context $\mathcal{M}$, the box calculi work more efficient than the calculus $\mathcal{F}'$ using the naïve generate-and-test algorithm for the context splitting. The box calculus $\mathcal{B}'$ is slower than $\mathcal{B}$ since the former incorporates some subtle operation and tests over contexts. The results of some tests are given in the following table (we used a Sun SPARC station 10 and the program `time` in order to determine the behaviors of the calculi; the format 'minutes:seconds' is used to present the times):

n		nth element	$\mathcal{F}'$	$\mathcal{B}$	$\mathcal{B}'$
0:	z	z	3.6	1.0	1.3
1:	sz	sz	2.3	1.0	1.2
2:	ssz	sz	11.0	3.7	4.1
3:	sssz	ssz	16.9	9.9	11.9
4:	ssssz	sssz	49.8	19.0	23.7
5:	sssssz	sssssz	2:24.0	37.2	46.0
6:	ssssssz	ssssssssz	9:34.7	1:15.1	1:32.8
7:	sssssssz	sssssssssssssz	45:18:5	2:38.9	3:08.6
8:	ssssssssz	s^{21}z	272:30.1	6:33.6	7:12.4

4.4 A Program for Finding Paths in Cyclic Graphs

The following program is taken from [Harland & Winikoff, 1996c] which is a presentation of various programs for Lygon. A graph is represented by some `points` (the nodes) and some directed `edges`. The predicate `path S E P` succeeds if there is a path between the points `S` and `E`. The variable `P` is instantiated with a list that represents a path (the lists are declared using the infix operator `::` and the constant `nil` which are provided by Terzo).

```
% Graph Problems
% based on a program which appeared in the paper "Some applications
% of the linear logic programming language lygon" by Winikoff and Harland
% (rewritten in Forum)                              last modified 26.09.96

kind point type.

type a point.
type b point.
type c point.
type d point.
type e point.
type f point.

type edge point -> point -> o.

type path point -> point -> list point -> o.

path X Y (X::Y::nil) o-- edge X Y.
path X Y (X::P) o-- (edge X Z x path Z Y P).

LINEAR edge a b.
LINEAR edge b c.
LINEAR edge c a.
LINEAR edge c d.
LINEAR edge c e.
LINEAR edge e f.
```

The edges are declared as linear resources, and therefore, they are consumed during the proof construction. Using this technique, the program can also find a path in a cyclic graph. However, some edges are not necessary to be traversed in a certain path. In order to consume these remaining linear resources (edges) the query contains a '$\top$' connective. The goal query can be stated as follows (`x` stands for the '$\otimes$'):

goal query 1: `path a f X x top`   solution: `X = a::b::c::e::f::nil`
goal query 2: `top x path a f X`

This example illustrates the speed-up which is achieved by the deterministic management of the contexts in the '$\top$-R' rule. The splitting of the contexts in the first goal query is not serious and the calculus $\mathcal{B}$ is slightly better than $\mathcal{B}'$. However, if we restate the goal query as given in the second line, the calculus $\mathcal{B}$ is approximately ten times slower as in the previous goal-query. On the other hand, the calculus $\mathcal{B}'$ which uses a deterministic resource management in a '$\top$-R' inference step shows no sign to be slower as in the first case. The results of the tests are given in the following table (we used similar condition as in the test of Section 4.3.):

goal	$\mathcal{F}'$	$\mathcal{B}$	$\mathcal{B}'$
`path a f X x top`	7:27.7	0:12,8	17,4
`top x path a f X`	6:14.3	2:13,1	17,3

Chapter 5

Related Work

5.1 Context Management in Lolli

[Hodas & Miller, 1991] presented the first solution of the problem of deterministic context management, which they called *input-output model (IO model)*, for the single succedent logic programming language Lolli. In this language, the '$\otimes$-R' rule appears to be critical because it has to split the linear context of the antecedent in a root-upward proof search. In Lolli, this rule is formalised as follows:

$$\frac{\Psi; \Delta_1 \Longrightarrow B \quad \Psi; \Delta_2 \Longrightarrow C}{\Psi; \Delta_1, \Delta_2 \Longrightarrow B \otimes C} \otimes\text{-R}$$

In their original work on the input-output model, Hodas and Miller used the sequent notation $I\{G\}O$ where G is a goal formula and I and O are the input and the output context of the corresponding proof branch, respectively. This system is specified in PROLOG; the contexts are represented as lists where a consumed formula is replaced by a special constant `del`. The IO model deals efficiently with the context splitting in the '$\otimes$-R' rule; however, it is not completely deterministic because the context in the '$\top$-R' is split with a generate-and-test algorithm. In order to remove the remaining non-determinism from the proof system, Hodas presented the *lazy input-output model ($IO^\top$ model)* in [Hodas, 1994]. The sequents in the refined system are as follows:

$$I[G](O, \bot) \qquad \text{and} \qquad I[G](O, \top)$$

where '$\bot$' and '$\top$' represent a flag, called *slack indicator*. This flag records whether the output context has to be consumed entirely ('$\bot$') or can contain some remaining formulae ('$\top$').

In [Cervesato et al., 1996], the treatment of the additive conjunction was improved and the initial presentation of the proof system (PROLOG specification) was changed to a system with a more proof theoretic flavour. (The contexts are represented as multisets and sequents are formalised with the usual notation using an arrow.)

In $\mathcal{RM}_3$ (the final version of their proof system), they use a sequent formulation with two input contexts. The sequents are presented as follows:

$$\Gamma; \Xi; \Delta^I \backslash \Delta^O \Longrightarrow_v G$$

where G is a goal formula; v is the slack indicator which is raised by the '$\top$-R' rule; Γ is the classical context; the rest of the context is divided into the input context Ξ and Δ^I and the output context Δ^O. Ξ contains formulae which have to be consumed in the proof branch above. Δ^I contains formulae which may be consumed. Δ^O contains those formulae which remain from the proof branch above. The separation of the input context into the parts Ξ and Δ^I forces the introduction of a new inference rule that selects a formula from the new context Ξ. Consequently, the corresponding rule d_r (of $\mathcal{RM}_1$ and $\mathcal{RM}_2$) is represented as $d^1_{rm_3}$ (for picking up a formula from Δ^I) and $d^2_{rm_3}$ (for a formula from Ξ) in $\mathcal{RM}_3$.

In [Cervesato et al., 1996], a *formula decomposition judgement* is introduced in order to find a program formula that will be analysed when an atomic formula a appears as goal formula. The judgement is as follows:

$$D \gg a \backslash G$$

where D is a program formula, a is an atomic formula and G is a goal formula. This judgement extracts from a program formula D a new goal formula G in a sense that G defines a; short $G \multimap a$. The formula decomposition judgement is formalised as follows:

$$\frac{}{\top \gg a \backslash 0}\ \top_d \qquad \frac{}{a' \gg a \backslash a' \doteq a}\ I_d$$

$$\frac{D \gg a \backslash G'}{G \multimap D \gg a \backslash G' \otimes G}\ \multimap_d \qquad \frac{D \gg a \backslash G'}{G \supset D \gg a \backslash G' \otimes !G}\ \supset_d$$

$$\frac{D_1 \gg a \backslash G_1 \quad D_2 \gg a \backslash G_2}{D_1 \& D_2 \gg a \backslash G_1 \oplus G_2}\ \&_d \qquad \frac{D \gg a \backslash G}{\forall x D \gg a \backslash \exists x G}\ \forall_d$$

where $a \doteq a'$ stands for the syntactic equality amongst atomic formulae. The formula decomposition judgement replaces Miller's and Hodas' function $\| \cdot \|$ which transforms a program formula into a (possible infinite) set. Our 'choose test' is similar to the usage of that formula decomposition judgement, however, in our presentation of FORUM it cannot be built into the proof search process and it appears as a separate operation which will be carried out before continuing the proof construction.

In summary, our box calculus does not separate the input context as in Hodas' IO$^\top$ because this involves an introduction of additional inference rules which analyse the same type of formulae in different input contexts. Along with such a separation comes, in the case of FORUM, an undesired significant expansion of set of inference rules. However, our approach separates the formulae of the output-context into two groups similar to the approach in [Cervesato et al., 1996]: one must be consumed and the other one might be consumed. This permits the omission of the slack indicator present in both approaches described above. Thus, it simplifies the implementation for which some evidence is provided (see Section 3.1.3 and 3.2.3).

[Peillon, 1991] introduced a system similar to Hodas' IO$^\top$ system for a theorem prover based on the intuitionistic fragment of MALL.

5.2 Hodas' and Polakow's Approach Towards a FORUM Implementation

Hodas and Polakow presented in [Hodas & Polakow, 1996] some preliminary results of their work on FORUM. They use Miller's calculus which does not include a classical context in the succedent (see [Miller, 1994]). Therefore, they cannot regard the '?' connective as primitive and have to use the logical equivalence:

$$?B \equiv (B \multimap \bot) \supset \bot$$

in order to analyse such a formula. This equivalence is problematic since its implicans $B \multimap \bot$ (a $\bot$-headed implication) eventually appears in the classical context of the antecedent. This causes problems particularly when the outlined heuristic is used (see Section 2.3.2). This heuristic considers newly added program formulae prior to other program formulae. Consequently an occurrence of a formula $?B$ (assume B is not an implication) in the succedent always leads to a loop where the interpreter selects the same formula over and over. Even if we do not make use of this heuristic, there is a significant number of cases where such an occurrence of a '?'-formula results in a loop of the interpreter (i.e., it cannot decide if a sequent is provable or not).

Hodas and Polakow introduced a *'backchain'* rule ('BC') which replaces all the *stoup* rules (left rules in Miller's calculus). This inference rule depends on the functions $||\cdot||$ and $||\cdot||'$. Suppose D is a program formula. Then, $||D||$ is defined inductively as follows:

- $\langle \emptyset, \emptyset, \{\!|D|\!\}\rangle \in ||D||$,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|\bot|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I}, \mathcal{L}, \Gamma\rangle \in ||D||$,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|B \parr C|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I}, \mathcal{L}, \{\!|B, C|\!\} \uplus \Gamma\rangle \in ||D||$,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|B \& C|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I}, \mathcal{L}, \{\!|B|\!\}\uplus, \Gamma\rangle \in ||D||$ and $\langle \mathcal{I}, \mathcal{L}, \{\!|C|\!\} \uplus \Gamma\rangle \in ||D||$,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|\forall x D|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I}, \mathcal{L}, \{\!|D[x \mapsto t]|\!\} \uplus \Gamma\rangle \in ||D||$ for all closed terms t,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|B \multimap C|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I}, \mathcal{L} \uplus \{\!|B|\!\}, \{\!|C|\!\} \uplus \Gamma\rangle \in ||D||$,
- $\langle \mathcal{I}, \mathcal{L}, \{\!|B \supset C|\!\} \uplus \Gamma\rangle \in ||D||$ implies $\langle \mathcal{I} \uplus \{\!|B|\!\}, \mathcal{L}, \{\!|C|\!\} \uplus \Gamma\rangle \in ||D||$.

Then, $||D||'$ is defined as $\{\langle \mathcal{I}, \mathcal{L}, \mathcal{A}\rangle | \langle \mathcal{I}, \mathcal{L}, \mathcal{A}\rangle \in ||D||$ where $\mathcal{A}$ is a multiset of atoms$\}$. Both functions convert a program formula into a set of triples. These triples can be regarded as program formulae defining a multiset ($\mathcal{A}$) of atoms (in [Hodas & Polakow, 1996] they are called the "true" head of clauses). However, the defined sets are generally infinite because of the "$\forall$ rule". The *formula decomposition judgement* described in [Cervesato et al., 1996] or our *choose test* provide a more syntactical and operational method which is easier to implement. The 'backchain' rule is presented as follows:

$$\frac{\Psi;\emptyset \Longrightarrow I_1 \quad \dots \quad \Psi;\emptyset \Longrightarrow I_m \quad \Psi;\Delta_1 \Longrightarrow L_1, \mathcal{A}_1 \quad \dots \quad \Psi;\Delta_n \Longrightarrow L_n, \mathcal{A}_n}{\Psi;\Delta_1,\dots,\Delta_n \overset{D}{\Longrightarrow} \mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_n} BC$$

where

- $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_n$ are lists of atomic formulae,
- $m, n \geq 0$ and
- $\langle \{I_1, \dots, I_m\}, \{L_1, \dots, L_n\}, \mathcal{A}\rangle \in ||D||'$.

In order to maintain the soundness, a side condition is introduced where 'BC' corresponds to an 'initial' rule. In this case, $\mathcal{I}$ and $\mathcal{L}$ are empty in the associated triple $\langle \mathcal{I}, \mathcal{L}, \mathcal{A} \rangle$ $(m, n = 0)$; the rule 'BC' is only applicable when $\mathcal{A}$ represents the entire list of the succedent.

In [Hodas & Polakow, 1996], an extension of the IO model is described as used in Lolli which does not include the improvements for the '$\top$' (as presented in a $\mathrm{IO}^\top$ model for Lolli) and for the '&' connective (as described in [Cervesato et al., 1996]). Thus, they present the modified 'BC' rule as follows (the premises are written on two lines):

$$\frac{\begin{array}{c}\Psi;\emptyset \Longrightarrow I_1 \quad \dots \quad \Psi;\emptyset \Longrightarrow I_m \\ \Psi;\Delta_I\backslash\Delta_{O_1} \Longrightarrow L_1, \mathcal{A}_I\backslash\mathcal{A}_{O_1} \quad \dots \quad \Psi;\Delta_{O_{n-1}}\backslash\Delta_O \Longrightarrow L_n, \mathcal{A}_{O_{n-1}}\backslash\mathcal{A}_O\end{array}}{\Psi;\Delta_I\backslash\Delta_O \overset{D}{\Longrightarrow} \mathcal{A}, \mathcal{A}_I\backslash\mathcal{A}_O} \ BC$$

where

- $\mathcal{A}, \mathcal{A}_I, \mathcal{A}_O, \mathcal{A}_{O_1}, \dots, \mathcal{A}_{O_{n-1}}$ are lists of atomic formulae
- $m, n \geq 0$ and $\langle \{I_1, \dots, I_m\}, \{L_1, \dots, L_n\}, \mathcal{A} \rangle \in ||D||'$

In case the 'BC' rule corresponds to an application of an 'initial' rule (i.e., the associated triple is $\langle \emptyset, \emptyset, \mathcal{A} \rangle$), the new 'BC' rule is as follows:

$$\frac{}{\Psi;\Delta_I\backslash\Delta_I \overset{D}{\Longrightarrow} \mathcal{A}, \mathcal{A}_I\backslash\mathcal{A}_I} \ BC$$

In [Hodas & Polakow, 1996], it is mentioned that the actual implementation follows the approach of [Cervesato et al., 1996]. No details are given of whether they have to introduce new inference rules for analysing formulae in the different contexts or not.

5.3 Context Management in Lygon

Independently of the work by Hodas and Miller, [Harland & Winikoff, 1996a] introduced a proof system for the multiple conclusion logic programming language Lygon (single-sided calculus) dealing deterministically with the context management (the first version of the calculus appeared as a technical report in 1994). The problematic rule in Lygon is as follows:

$$\frac{\delta : B, \Gamma_1 \quad \delta : C, \Gamma_2}{\delta : B \otimes C, \Gamma_1, \Gamma_2} \ \otimes\text{-R}$$

where δ consists entirely of nonlinear formulae and Γ_1 and Γ_2 are multisets of formulae. Harland's and Winikoff's approach uses the same idea as the IO model. It gives first all linear formulae to the left proof branch and all remaining unused formulae to the right proof branch. However, they maintain the soundness by giving a tag '$^\top$' to all formulae (apart from the components of the analysed formula) of the left premise. Subsequently, only tagged formulae can be passed to the right premise where one tag is removed from each formula (nested tags are permitted for nested occurrences of the '$\otimes$' connective). A '*Use*' rule is introduced which strips off all tags from a formula in order to analyse it.

For dealing efficiently with the '$\top$-R' rule another tag ('?') is used. It prefixes formulae consumed by the '$\top$-R' rule, but which can become "unconsumed" in another part of the proof. In order to maintain the soundness (i.e., protect formulae to become "unconsumed" elsewhere in the proof when a '&-R' rule is applied) a notion of "⊤-like" proofs is defined. A flag is attached to the sequents which is either */true* (i.e., a proof branch is ⊤-like) or */false* (i.e., the proof is not ⊤-like). This flag is similar to Hodas' slack indicator.

We describe here the sequents of Harland's and Winikoff's approach, but omit the inference rules which can be found in the Appendix of [Harland & Winikoff, 1996a]. The sequents are as follows:

$$\delta : \Gamma, \Pi, \daleth \Longrightarrow \Sigma, \aleph / x$$

where δ is a multiset of nonlinear formulae, Γ is a multiset of formulae which have no tag or prefix, Σ and Π are multisets of formulae with the tag '$^\top$', $\daleth$ and $\aleph$ are multisets of formulae with the prefix '?' and '$/x$' is the aforementioned flag. The variables on the left-hand side of the sequent arrow stand for multisets of formulae given for usage to the proof branch and the variables on the right-hand side stand for multiset of formulae which can be consumed elsewhere.

The approach above differs from the IO model and our box calculi by using tags and prefixes instead of using clear separated contexts. It uses a flag similar to the slack indicator in order to maintain soundness in the calculus dealing efficiently with the '⊤-R' rule. Furthermore, the calculus is not refined according to the improvements for the '&-R' rule as introduced in [Cervesato et al., 1996].

Chapter 6

Conclusion, Open Problems and Further Work

6.1 Conclusion

We achieved in the second box calculus a more deterministic context management (in comparison with $\mathcal{B}$). However, the inefficient generate-and-test algorithm for finding the desired partition of the contexts is replaced by some subtle operations on contexts. That means we have to pay a rather high price because the context management now includes some costly operations (e.g., multiset intersection, multiset difference, shuffling formulae from one context to another one and exhaustive tests of multisets). However, the rules are completely declarative and presented in a proof theoretical style. We adapted the existing approaches of the IO model and developed a calculus, in our opinion more appropriate and elegant than the established calculi for an implementation of FORUM. In our approach, there is no need for an introduction of a slack indicator and for the additional inference rules which analyse formulae from different input contexts that would be required by the previous approaches. As shown in the implementation of the box calculi, each inference rule results in a single unit in the source code.

However, it should be noted that this is only a first step towards an implementation of FORUM as a logic programming language. Our approach provides not an implementation of the complete language FORUM as a logic programming language; it rather provides a basis for further investigations.

6.2 Open Problems

Much remains still to be done. Amongst the problems relative to FORUM as a logic programming language are the following:

- The most serious problem, when implementing FORUM as a logic programming language, arises from the $\perp$-headed implications occurring in the classical context of the antecedent. Under some circumstances, these program formulae result always in a loop (i.e., the interpreter of the language fails to answer whether a sequent is provable or not). A solution was suggested in [Hodas & Polakow, 1996] which delayed a selection of such a program formula as long as possible. However, this attempt will fail when one of several $\bot$-headed implications must be chosen since then it is not clear which formulae should be delayed and which chosen.

The problem also appears to be difficult because it is not feasible to restrict the logic so that these $\bot$-headed implications do not occur because they play an important role when analysing a formula with a non-primitive connective. Thus far, there is no satisfactory solution known for this problem.

- There is still less practical evidence whether FORUM is a useful logic programming language or not (the main purpose of the thesis is to provide an appropriate basis for further investigations). The usage of FORUM's primitive and non-primitive connectives must be proven to be a fruitful and an effective framework for representing and solving problems. In particular, the use of FORUM's non-primitive connectives must be defended from the criticism as an "abuse" of logical equivalences (citation from [Harland & Winikoff, 1995a]).
- FORUM is designed so that its right rules permute over each other. It is still unclear what is an appropriate semantics which identifies exactly these permutable proofs. A clear notion of permutable proofs has to be achieved which can justify the liberal use of permutations in the design of FORUM.

6.3 Further Work

There are many things for further work arising from the thesis and particularly from the open problems. Some of them are outlined below:

- Further work is needed in order to simplify our calculus following Hodas' and Polakow's description of a 'backchain' rule; it should determine efficiently whether a program formula is a candidate for the further proof construction.
- In our calculus, the order of the program formulae, which enables a programmer to predict the behaviour of the proof construction, is not completely maintained. Thus far, linear formulae are preferred relative to the formulae in the classical contexts. Therefore, we have to package all program contexts together and use a flag for each formula in order to determine if it is deleted or not.
- The field of an appropriate semantics which identifies FORUM's permutable proofs is fairly "unexplored". Further work will address this issue with the focus on obtaining some insight which proof are "necessary" in FORUM and which fragment of it can be implemented throughly as a logic programming language.

Bibliography

[Andreoli, 1992] Andreoli, J.-M. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):297–347.

[Beth, 1965] Beth, E. W. *The foundations of Mathematics.* North-Holland, Amsterdam, 2nd edition.

[Bierman, 1994] Bierman, G. *On Intuitionistic Linear Logic.* PhD thesis, University of Cambridge.

[Brown, 1990] Brown, C. *Linear Logic and Petri Nets: Categories, Algebra and Proof.* PhD thesis, University of Edinburgh. Technical Report ECS-LFCS-91-128.

[Cervesato, 1995] Cervesato, I. Petri Nets as Multiset Rewriting System in a Linear Framework. Technical report, Dipartimento di Informatica Universita di Torino, Italy.

[Cervesato et al., 1996] Cervesato, I., Hodas, J., and Pfenning, F. Efficient Resource Management for Linear Logic Proof Search. In Dyckhoff, R., Herre, H., and Schröder-Heister, P., editors, *Proceedings of the 5th International Workshop on Extensions of Logic Programming*, LNAI 1050, pages 67–81. Springer Verlag.

[Chirimar, 1995] Chirimar, J. *Proof Theoretic Approach to Specification Languages.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

[Church, 1951] Church, A. *A Formulation of the Simple Theory of Types*, volume 6 of *Annals of Mathematics Studies.* Princeton University Press, Princeton, NJ.

[Delzanno & Martelli, 1995] Delzanno, G. and Martelli, M. Objects in Forum. In *Proceedings of the International Logic Programming Symposium, Portland, Oregon*, pages 115–129. The MIT Press.

[Galmiche & Perrier, 1994] Galmiche, D. and Perrier, G. On Proof Normalisation in Linear Logic. *Theoretical Computer Science*, 135(1):67–110.

[Gentzen, 1969] Gentzen, G. *The collected Papers of Gerhard Gentzen.* North-Holland, Amsterdam. Ed. E. Szabo.

[Girard, 1987] Girard, J.-Y. Linear Logic. *Theoretical Computer Science*, 50:1–102.

[Girard, 1991] Girard, J.-Y. A New Constructive Logic: Classical Logic. *Mathematical Structures in Computer Science*, 1:255–296.

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

[Große et al., 1992] Große, G., Hölldobler, S., Schneeberger, J., Sigmund, U., and Thielscher, M. Equational Logic Programming, Actions, and Change. In Apt, K., editor, *International Conference and Symposium on Logic Programming*, pages 177–191. MIT Press.

[Harland & Winikoff, 1995a] Harland, J. and Winikoff, M. Deriving Logic Programming Languages. Technical Report 95/26, Department of Computer Science, University of Melburne.

[Harland & Winikoff, 1995b] Harland, J. and Winikoff, M. Implementation and Development Issues for the Linear Logic Programming Language Lygon. In *Proceedings of the 18th Australasian Computer Science Conference*, pages 563–572, Adelaide, Australia. Also available as Technical Report TR 95/6, Melbourne University, Department of Computer Science.

[Harland & Winikoff, 1996a] Harland, J. and Winikoff, M. Deterministic Resource Management for the Linear Logic Programming Language Lygon. In *Proceedings of the 19th Australasian Computer Science Confernce*, Melbourne, Australia.

[Harland & Winikoff, 1996b] Harland, J. and Winikoff, M. Programming in Lygon: An Overview. *Algebraic Methodology and Software Technology*, pages 391–405.

[Harland & Winikoff, 1996c] Harland, J. and Winikoff, M. Some Applications of the Linear Logic Programming Language Lygon. In *Proceedings of the 19th Australasian Computer Science Conference*, Melbourne, Australia.

[Hölldobler & Schneeberger, 1990] Hölldobler, S. and Schneeberger, J. A New Deductive Approach to Planning. *New Generation Computing*, 8:225–244. A short version appeared in the Proceedings of the German Workshop on Artificial Intelligence, Informatik Fachberichte *216*, pages 63–73, 1989.

[Hodas, 1992] Hodas, J. Lolli: An Extension of λProlog with Linear Context Management. In Miller, D., editor, *Workshop on the λProlog Programming Language*, pages 159–168, Philadelphia, Pennsylvania.

[Hodas, 1993] Hodas, J. Logic Programming with Multiple Context Management Schemes. In Dyckhoff, R., editor, *Proceedings of the 4th Workshop on Extensions of Logic Programming*, LNAI 798, pages 171–182. Springer Verlag.

[Hodas, 1994] Hodas, J. *Logic Programming in Intuitionistic Linear Logic: Theory, Design and Implementation*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science.

[Hodas & Miller, 1991] Hodas, J. and Miller, D. Logic Programming in a Fragment of Intuitionistic Linear Logic: Extended Abstract. In Kahn, G., editor, *6th Annual Symposium on Logic in Computer Science*, pages 32–42, Amsterdam. (superseded by [Hodas & Miller, 1994]).

[Hodas & Miller, 1994] Hodas, J. and Miller, D. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Information and Computation*, 110(2):327–365.

[Hodas & Polakow, 1996] Hodas, J. and Polakow, J. Forum as a Logic Programming Language: Preliminary Results and Observations. In *Linear Logic 96, extended abstracts and preliminary results*, Electronic Notes in Computer Science, Tokyo. Elsevier-North Holland.

[Hyland & de Paiva, 1993] Hyland, M. and de Paiva, V. Full Intuitionistic Linear Logic (Extended Abstract). *Annals of Pure and Applied Logic*, 64(3):273–291.

[Kleene, 1952] Kleene, S. C. Permutability of Inferences in Gentzen's calculi LK and LJ. *Memoirs of the American Mathematical Society*, 10:1–26.

[Lincoln, 1991] Lincoln, P. (Im)Permutabilities of LL. Appeared in the 'Linear' mailing list `linear@cs.stanford.edu` (30. November 1991).

[Lincoln, 1992] Lincoln, P. Linear Logic. *ACM SIGACT Notices*, 23:29–37.

[Lincoln et al., 1992] Lincoln, P., Mitchell, J., Scedrov, A., and Shankar, N. Decision Problems for Propositional Linear Logic. *Annals of Pure and Applied Logic*, 56:239–311.

[Martí-Oliet & Meseguer, 1989] Martí-Oliet, N. and Meseguer, J. From Petri Nets to Linear Logic. In Dybjer, P., Pitts, A. M., Pitt, D. H., Poigné, A., and Rydeheard, D. E., editors, *Proceedings of the Conference on Category Theory and Computer Science*, Springer-Verlag LNCS 389, pages 313–340, Manchester, United Kingdom.

[Masseron et al., 1990] Masseron, M., Tollu, C., and Vauzeilles, J. Plan Generation and Linear Logic. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 63–75, Bangalore, India. Springer-Verlag LNCS 472.

[Miller, 1989a] Miller, D. A Logic Programming Language with Lambda- Abstraction, Function Variables, and Simple Unification. In Schröder-Heister, P., editor, *Proceedings of the International Workshop on Proof-Theoretical Extensions of Logic Programming*, number 475 in LNAI, pages 253–281, Tübingen, Germany. Springer Verlag.

[Miller, 1989b] Miller, D. A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming*, pages 79–108.

[Miller, 1991] Miller, D. Unification of Simply Typed Lambda-Terms as Logic Programming. In Furukawa, K., editor, *8th International Conference on Logic Programming*, pages 255–269, Paris, France. MIT Press.

[Miller, 1992] Miller, D. The π-Calculus as a Theory in Linear Logic: Preliminary Results. In Lamma, E. and Mello, P., editors, *Proceedings of the Workshop on Extensions of Logic Programming*, pages 242–265. Springer-Verlag LNCS 660.

[Miller, 1993] Miller, D. A Proposal for Modules in λProlog. In Dyckhoff, R., editor, *Proceedings of the 4th International Workshop on Extensions of Logic Programming*, pages 206–221. Springer-Verlag LNAI 798.

[Miller, 1994] Miller, D. A Multiple-Conclusion Meta-Logic. In Abramsky, S., editor, *9th Annual Symposium on Logic in Computer Science*, pages 272–281, Paris, France. IEEE Computer Society Press.

[Miller, 1995] Miller, D. Course Material for the International Summer School (Marktoberdorf) on Logic of Computation. An Advanced Study Institute of the NATO Science Commitee and the Technische Universität München.

[Miller, 1996] Miller, D. Forum: A Multiple-Conclusion Specification Logic. *Theoretical Computer Science*, 165:201–232. ALP/PLILP meeting.

[Miller & Nadathur, 1986] Miller, D. and Nadathur, G. Some Uses of Higher-Order Logic in Computational Linguistics. In *24th Annual Meeting of the Association for Computational Linguistics, New York*, pages 247–255.

[Miller et al., 1991] Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure Applied Logic*, 51:125–157.

[Nadathur, 1995] Nadathur, G. Uniform Provability in Classical Logic. Technical Report TR-95-12, CS Department, University of Chicago.

[Nadathur & Miller, 1990] Nadathur, G. and Miller, D. Higher-Order Horn Clauses. *Journal of ACM*, 37(4):777–814.

[Nadathur & Miller, 1994] Nadathur, G. and Miller, D. Higher-Order Logic Programming. Technical Report CS-1994-38, Department of Computer Science, Duke University.

[Negri, 1995] Negri, S. Semantical observations on the embedding of Intuitionistic Logic into Intuitionistic Linear Logic. *Mathematical Structures in Computer Science*, 5:41–68.

[Peillon, 1991] Peillon, T. A Proof Editor For Linear Logic. Master thesis, University of St Andrews.

[Perrier, 1995] Perrier, G. *De la construction de preuves à la programmation parallèle en logique linéaire*. PhD thesis, L'Université Henri Poincaré – Nancy I.

[Peterson, 1981] Peterson, J. L. *Petri Net Theory and The Modeling of Systems*. Prentice-Hall.

[Pym & Harland, 1994] Pym, D. and Harland, J. A Uniform Proof-Theoretic Investigation of Linear Logic Programming. *Journal of Logic and Computation*, 4(2):175–207.

[Pym & Wallen, 1992] Pym, D. and Wallen, L. Logic Programming via Proof-valued Computations. In Broda, K., editor, *4th UK Conference on Logic Programming*, Workshop in Computing Series, pages 253–295. Springer-Verlag.

[Scedrov, 1993] Scedrov, A. A Brief Guide to Linear Logic. In Rozenberg, G. and Salomaa, A., editors, *Current Trends in Theoretical Computer Science*, pages 377–394. World Scientific Publishing Company. Also in Bulletin of the European Association for Theoretical Computer Science, volume 41, pages 154–165.

[Schneeberger, 1992] Schneeberger, J. *Plan generation by linear deduction*. PhD thesis, Technische Hochschule Darmstadt, Fachbereich Informatik.

[Troelstra, 1992] Troelstra, A. S. *Lectures on Linear Logic*. CSLI Lecture Notes 29, Stanford, California.

[Weld, 1994] Weld, D. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61.

Appendix A

Proofs

A.1 Soundness of $\mathcal{B}$ w.r.t. $\mathcal{F}$'

For each rule of $\mathcal{B}$, we give the translation into a rule of $\mathcal{F}'$.

$$\frac{}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & \top, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R} \quad \Rightarrow \quad \frac{}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; \top, \mathcal{B}; \Upsilon}\ \top\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & A, \mathcal{A}, Y & \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & A, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ atomic\text{-R} \Rightarrow \quad \frac{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}, A; \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; A, \mathcal{B}; \Upsilon}\ atomic\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & C, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B\&C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-R} \ \Rightarrow \ \frac{\begin{array}{c}\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon \\ \Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B, \mathcal{B}; \Upsilon\end{array}}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B\&C, \mathcal{B}; \Upsilon}\ \&\text{-R}$$

$$\frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & \bot, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R} \quad \Rightarrow \quad \frac{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; \bot, \mathcal{B}; \Upsilon}\ \bot\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B, C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Sigma} & \Psi \\ \hline \Delta, X & \mathcal{A}, Y & B \parr C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \parr\text{-R} \quad \Rightarrow \quad \dfrac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B, C, \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B \parr C, \mathcal{B}; \Upsilon} \parr\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X, B & \mathcal{A}, Y & C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B \multimap C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \multimap\text{-R} \quad \Rightarrow \quad \dfrac{\Sigma\colon \Psi; \Delta, B \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B \multimap C, \mathcal{B}; \Upsilon} \multimap\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi, B} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B \supset C, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \supset\text{-R} \quad \Rightarrow \quad \dfrac{\Sigma\colon \Psi, B; \Delta \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B \supset C, \mathcal{B}; \Upsilon} \supset\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & B[x \mapsto y], \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{y : \tau, \Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & \forall_\tau x B, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \forall\text{-R} \Rightarrow \dfrac{y : \tau, \Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B[x \mapsto y], \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \forall_\tau x B, \mathcal{B}; \Upsilon} \forall\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & B, \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta, X & \mathcal{A}, Y & ?B, \mathcal{B}, Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} ?\text{-R} \qquad \Rightarrow \quad \dfrac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \mathcal{B}; B, \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; ?B, \mathcal{B}; \Upsilon} ?\text{-R}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta, X & \mathcal{A}, Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta, B, X & \mathcal{A}, Y & Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \textit{ choose} \quad \Rightarrow \quad \dfrac{\Sigma\colon \Psi; \Delta \xRightarrow{B} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; B, \Delta \Longrightarrow \mathcal{A}; \emptyset; \Upsilon} \textit{ choose}$$

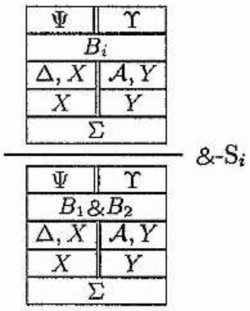
$$\dfrac{\begin{array}{|c||c|}\hline \Psi, B & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta, X & \mathcal{A}, Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi, B & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta, X & \mathcal{A}, Y & Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \textit{ choose!} \quad \Rightarrow \quad \dfrac{\Sigma\colon B, \Psi; \Delta \xRightarrow{B} \mathcal{A}; \Upsilon}{\Sigma\colon B, \Psi; \Delta \Longrightarrow \mathcal{A}; \emptyset; \Upsilon} \textit{ choose!}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \Delta, X & \mathcal{A}, Y & B \\ \hline X & Y & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \Delta, X & \mathcal{A}, Y & Z \\ \hline X & Y & Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \textit{ choose?} \quad \Rightarrow \quad \dfrac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B; B, \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \emptyset; B, \Upsilon} \textit{ choose?}$$

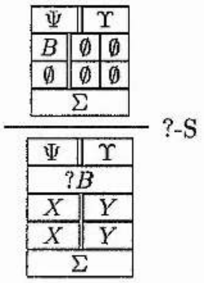
$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline X & A, Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \textit{ initial} \quad \Rightarrow \quad \dfrac{}{\Sigma\colon \Psi; \emptyset \xRightarrow{A} A; \Upsilon} \textit{ initial}$$

$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & A, \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline X & Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \textit{ initial?} \quad \Rightarrow \quad \dfrac{}{\Sigma\colon \Psi; \emptyset \xRightarrow{A} \emptyset; A, \Upsilon} \textit{ initial?}$$

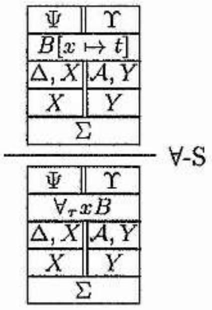
$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\bot} \\ \hline X & Y \\ \hline X & Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \ \bot\text{-S} \quad \Rightarrow \quad \dfrac{}{\Sigma\colon \Psi; \emptyset \xRightarrow{\bot} \emptyset; \Upsilon} \ \bot\text{-S}$$



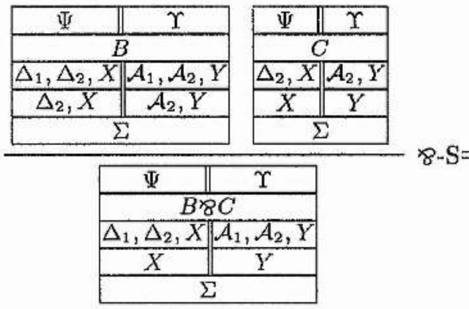
$$\Rightarrow \quad \frac{\Sigma : \Psi; \Delta \overset{B_i}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma : \Psi; \Delta \overset{B_1 \& B_2}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \&\text{-S}_i$$



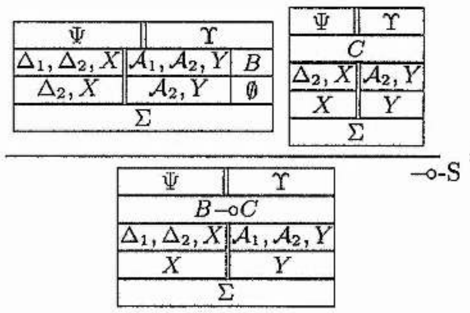
$$\Rightarrow \quad \frac{\Sigma : \Psi; B \Longrightarrow \emptyset; \emptyset; \Upsilon}{\Sigma : \Psi; \emptyset \overset{?B}{\Longrightarrow} \emptyset; \Upsilon}\ ?\text{-S}$$



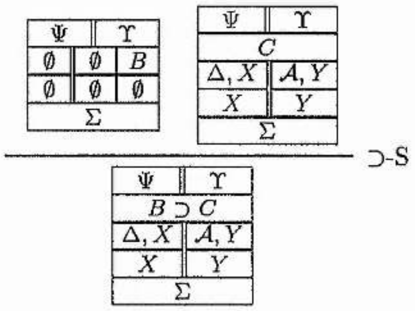
$$\Rightarrow \quad \frac{\Sigma : \Psi; \Delta \overset{B[x \mapsto t]}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma : \Psi; \Delta \overset{\forall_\tau x B}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \forall\text{-S}$$



$$\Rightarrow \frac{\Sigma : \Psi; \Delta_1 \overset{B}{\Longrightarrow} \mathcal{A}_1; \Upsilon \quad \Sigma : \Psi; \Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma : \Psi; \Delta_1, \Delta_2 \overset{B ⅋ C}{\Longrightarrow} \mathcal{A}_1, \mathcal{A}_2; \Upsilon}\ ⅋\text{-S}$$



$$\Rightarrow \frac{\Sigma\colon \Psi; \Delta_1 \Longrightarrow \mathcal{A}_1; B; \Upsilon \quad \Sigma\colon \Psi; \Delta_2 \overset{C}{\Longrightarrow} \mathcal{A}_2; \Upsilon}{\Sigma\colon \Psi; \Delta_1, \Delta_2 \overset{B \multimap C}{\Longrightarrow} \mathcal{A}_1, \mathcal{A}_2; \Upsilon} \multimap\text{-S}$$



$$\Rightarrow \frac{\Sigma\colon \Psi; \emptyset \Longrightarrow \emptyset; B; \Upsilon \quad \Sigma\colon \Psi; \Delta \overset{C}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma\colon \Psi; \Delta \overset{B \supset C}{\Longrightarrow} \mathcal{A}; \Upsilon} \supset\text{-S}$$

A.2 Modification of a $\mathcal{B}$-Proof

For each rule of $\mathcal{B}$, we give the translation into a rule of $\mathcal{B}$ which has some additional formulae in the contexts. The variables X, Y and Z are chosen in the endbox of the corresponding proof branch which is translated.

$$\dfrac{}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}^0,\mathcal{M}^1 & \mathcal{A}^0,\mathcal{A}^1 & \top,\mathcal{B}^0,\mathcal{B}^1\\\hline \mathcal{M}^1 & \mathcal{A}^1 & \mathcal{B}^1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \top\text{-R} \quad\Leftrightarrow\quad \dfrac{}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \top,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \top\text{-R}$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & A,\mathcal{A}_0,\mathcal{A}_1 & \mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & A,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{atomic}\text{-R}\Leftrightarrow \dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & A,\mathcal{A}_0,\mathcal{A}_1,Y & \mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & A,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{atomic}\text{-R}$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array} \quad \begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \ \| \ \Upsilon}\\\hline \mathcal{M}_0 & \mathcal{A}_0 & C,\mathcal{B}_0\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B\&C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-R}\Leftrightarrow \dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array} \quad \begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \ \| \ \Upsilon}\\\hline \mathcal{M}_0 & \mathcal{A}_0 & C,\mathcal{B}_0\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B\&C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-R}$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & \mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & \bot,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-R} \quad\Leftrightarrow\quad \dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \bot,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,X & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-R}$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B,C,\mathcal{B}_0,\mathcal{B}_1,\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Sigma \quad \| \quad \Psi}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B⅋C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ ⅋\text{-R}\Leftrightarrow \dfrac{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi \quad \| \quad \Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B,C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|}\hline \multicolumn{3}{|c|}{\Sigma \quad \| \quad \Psi}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B⅋C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ ⅋\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B,\mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B\multimap C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\multimap\text{-R}\Leftrightarrow\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B,\mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B\multimap C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\multimap\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi,B & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B\supset C,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\supset\text{-R}\Leftrightarrow\dfrac{\begin{array}{|c||c|c|}\hline \Psi,B & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B\supset C,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\supset\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & B[x\mapsto y],\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{y:\tau,\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & \forall_\tau xB,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\forall\text{-R}\Leftrightarrow\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & B[x\mapsto y],\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{y:\tau,\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \forall_\tau xB,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\forall\text{-R}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & \mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & ?B,\mathcal{B}_0,\mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}?\text{-R}\quad\Leftrightarrow\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & ?B,\mathcal{B}_0,\mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}?\text{-R}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B}\\\hline \mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B,\mathcal{M}_0,\mathcal{M}_1 & \mathcal{A}_0,\mathcal{A}_1 & \mathcal{B}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\mathit{choose}\Leftrightarrow\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B}\\\hline \mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B,\mathcal{M}_0,\mathcal{M}_1,X & \mathcal{A}_0,\mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \mathcal{M}_1,X & \mathcal{A}_1,Y & \mathcal{B}_1,Z\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\mathit{choose}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi, B & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi, B & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 & \mathcal{B}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \; choose! \quad \Leftrightarrow \quad \dfrac{\begin{array}{|c||c|}\hline \Psi, B & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi, B & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y & \mathcal{B}_1, Z \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y & \mathcal{B}_1, Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \; choose!$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 & B \\ \hline \mathcal{M}_1 & \mathcal{A}_1 & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 & \mathcal{B}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 & \mathcal{B}_1 \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \; choose? \quad \Leftrightarrow \quad \dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y & B \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y & \mathcal{B}_1, Z \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y & \mathcal{B}_1, Z \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \; choose?$$

$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}_1 & A, \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; initial \quad \Leftrightarrow \quad \dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}_1, X & A, \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; initial$$

$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & A, \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; initial? \quad \Leftrightarrow \quad \dfrac{}{\begin{array}{|c||c|}\hline \Psi & A, \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; initial?$$

$$\dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\bot} \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; \bot\text{-S} \quad \Leftrightarrow \quad \dfrac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\bot} \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; \bot\text{-S}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_i} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_1 \& B_2} \\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1 \\ \hline \mathcal{M}_1 & \mathcal{A}_1 \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; \&\text{-S}_i \quad \Leftrightarrow \quad \dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_i} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_1 \& B_2} \\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y \\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \; \&\text{-S}_i$$

$$
\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B & \emptyset & \emptyset\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{?B}\\\hline \mathcal{M}_1 & \mathcal{A}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \text{?-S}
\qquad\Leftrightarrow\qquad
\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B & \emptyset & \emptyset\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{?B}\\\hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\\hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \text{?-S}
$$

$$
\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B[x \mapsto t]}\\\hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{\forall_\tau x B}\\\hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1\\\hline \mathcal{M}_1 & \mathcal{A}_1\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \forall\text{-S}
\qquad\Leftrightarrow\qquad
\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B[x \mapsto t]}\\\hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y\\\hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{\forall_\tau x B}\\\hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y\\\hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \forall\text{-S}
$$

$$
\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2\\\hline \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_1, \mathcal{A}_2\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}\ \begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{C}\\\hline \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_1, \mathcal{A}_2\\\hline \mathcal{M}_1 & \mathcal{A}_2\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B \parr C}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2\\\hline \mathcal{M}_2 & \mathcal{A}_2\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \parr\text{-S}
\Leftrightarrow
\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}\ \begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{C}\\\hline \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \mathcal{M}_2, X & \mathcal{A}_2, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B \parr C}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \mathcal{M}_2, X & \mathcal{A}_2, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \parr\text{-S}
$$

$$
\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \Upsilon & \\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2 & B\\\hline \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_1, \mathcal{A}_2 & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}\ \begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{C}\\\hline \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_1, \mathcal{A}_2\\\hline \mathcal{M}_2 & \mathcal{A}_2\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B \multimap C}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2 & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2\\\hline \mathcal{M}_2 & \mathcal{A}_2\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \multimap\text{-S}
\Leftrightarrow
\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \Upsilon & \\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, Y & B\\\hline \mathcal{M}_1 \mathcal{M}_2, X & \mathcal{A}_1, \mathcal{A}_2, Y & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}\ \begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{C}\\\hline \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \mathcal{M}_2, X & \mathcal{A}_2, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B \multimap C}\\\hline \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, X & \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, Y\\\hline \mathcal{M}_2, X & \mathcal{A}_2, Y\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \multimap\text{-S}
$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\ \hline \emptyset & \emptyset & B\\ \hline \emptyset & \emptyset & \emptyset\\ \hline \multicolumn{3}{|c|}{\Sigma}\\ \hline\end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon\\ \hline \multicolumn{2}{|c|}{C}\\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1\\ \hline \mathcal{M}_1 & \mathcal{A}_1\\ \hline \multicolumn{2}{|c|}{\Sigma}\\ \hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\ \hline \multicolumn{2}{|c|}{B \supset C}\\ \hline \mathcal{M}_0, \mathcal{M}_1 & \mathcal{A}_0, \mathcal{A}_1\\ \hline \mathcal{M}_1 & \mathcal{A}_1\\ \hline \multicolumn{2}{|c|}{\Sigma}\\ \hline\end{array}} \supset\text{-S} \Leftrightarrow \dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\ \hline \emptyset & \emptyset & B\\ \hline \emptyset & \emptyset & \emptyset\\ \hline \multicolumn{3}{|c|}{\Sigma}\\ \hline\end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon\\ \hline \multicolumn{2}{|c|}{C}\\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y\\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\ \hline \multicolumn{2}{|c|}{\Sigma}\\ \hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\ \hline \multicolumn{2}{|c|}{B \supset C}\\ \hline \mathcal{M}_0, \mathcal{M}_1, X & \mathcal{A}_0, \mathcal{A}_1, Y\\ \hline \mathcal{M}_1, X & \mathcal{A}_1, Y\\ \hline \multicolumn{2}{|c|}{\Sigma}\\ \hline\end{array}} \supset\text{-S}$$

A.3 Completeness of $\mathcal{B}$ w.r.t. $\mathcal{F}'$

For each rule of $\mathcal{F}'$, we give the translation into a rule of $\mathcal{B}$.

$$\frac{}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \top, \mathcal{B}; \Upsilon}\ \top\text{-R} \quad \Rightarrow \quad \frac{}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & \top,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R}$$

$$\frac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}, A; \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; A, \mathcal{B}; \Upsilon}\ \mathit{atomic}\text{-R} \ \Rightarrow \ \frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & A,\mathcal{A} & \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & A,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \mathit{atomic}\text{-R}$$

$$\frac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B, \mathcal{B}; \Upsilon \quad \Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B\&C, \mathcal{B}; \Upsilon}\ \&\text{-R} \ \Rightarrow \ \frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & B,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \quad \begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & C,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & B\&C,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-R}$$

$$\frac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; \bot, \mathcal{B}; \Upsilon}\ \bot\text{-R} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & \bot,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R}$$

$$\frac{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B, C, \mathcal{B}; \Upsilon}{\Sigma\colon \Psi; \Delta \Longrightarrow \mathcal{A}; B \parr C, \mathcal{B}; \Upsilon}\ \parr\text{-R} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \Upsilon \\ \hline \Delta & \mathcal{A} & B,C,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \multicolumn{2}{|c||}{\Sigma} & \Psi \\ \hline \Delta & \mathcal{A} & B \parr C,\mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-R}$$

$$\frac{\Sigma : \Psi; \Delta, B \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B \multimap C, \mathcal{B}; \Upsilon} \multimap\text{-R} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta, B & \mathcal{A} & C, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & B \multimap C, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \multimap\text{-R}$$

$$\frac{\Sigma : \Psi, B; \Delta \Longrightarrow \mathcal{A}; C, \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B \supset C, \mathcal{B}; \Upsilon} \supset\text{-R} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \Psi, B & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & C, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & B \supset C, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \supset\text{-R}$$

$$\frac{y{:}\tau, \Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; B[x \mapsto y], \mathcal{B}; \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; \forall_\tau x B, \mathcal{B}; \Upsilon} \forall\text{-R} \Rightarrow \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & B[x \mapsto y], \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{y : \tau, \Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & \forall_\tau x B, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \forall\text{-R}$$

$$\frac{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; \mathcal{B}; B, \Upsilon}{\Sigma : \Psi; \Delta \Longrightarrow \mathcal{A}; ?B, \mathcal{B}; \Upsilon} ?\text{-R} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B, \Upsilon} \\ \hline \Delta & \mathcal{A} & \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta & \mathcal{A} & ?B, \mathcal{B} \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} ?\text{-R}$$

$$\frac{\Sigma : \Psi; \Delta \overset{B}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma : \Psi; B, \Delta \Longrightarrow \mathcal{A}; \emptyset; \Upsilon} \textit{choose} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \Delta, B & \mathcal{A} & \emptyset \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}} \textit{choose}$$

$$\frac{\Sigma: B,\Psi;\Delta\overset{B}{\Longrightarrow}\mathcal{A};\Upsilon}{\Sigma: B,\Psi;\Delta\Longrightarrow\mathcal{A};\emptyset;\Upsilon}\ \mathit{choose!}\qquad\Rightarrow\qquad\frac{\begin{array}{|c||c|}\hline \Psi,B & \Upsilon\\\hline \multicolumn{2}{|c|}{B}\\\hline \Delta & \mathcal{A}\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi,B & \multicolumn{2}{c|}{\Upsilon}\\\hline \Delta & \mathcal{A} & \emptyset\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{choose!}$$

$$\frac{\Sigma: \Psi;\Delta\Longrightarrow\mathcal{A};B;B,\Upsilon}{\Sigma: \Psi;\Delta\Longrightarrow\mathcal{A};\emptyset;B,\Upsilon}\ \mathit{choose?}\qquad\Rightarrow\qquad\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon}\\\hline \Delta & \mathcal{A} & B\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon}\\\hline \Delta & \mathcal{A} & \emptyset\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{choose?}$$

$$\frac{}{\Sigma: \Psi;\emptyset\overset{A}{\Longrightarrow}A;\Upsilon}\ \mathit{initial}\qquad\Rightarrow\qquad\frac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{A}\\\hline \emptyset & A\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{initial}$$

$$\frac{}{\Sigma: \Psi;\emptyset\overset{A}{\Longrightarrow}\emptyset;A,\Upsilon}\ \mathit{initial?}\qquad\Rightarrow\qquad\frac{}{\begin{array}{|c||c|}\hline \Psi & A,\Upsilon\\\hline \multicolumn{2}{|c|}{A}\\\hline \emptyset & \emptyset\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{initial?}$$

$$\frac{}{\Sigma: \Psi;\emptyset\overset{\bot}{\Longrightarrow}\emptyset;\Upsilon}\ \bot\text{-S}\qquad\Rightarrow\qquad\frac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{\bot}\\\hline \emptyset & \emptyset\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-S}$$

$$\frac{\Sigma: \Psi;\Delta\overset{B_i}{\Longrightarrow}\mathcal{A};\Upsilon}{\Sigma: \Psi;\Delta\overset{B_1\& B_2}{\Longrightarrow}\mathcal{A};\Upsilon}\ \&\text{-S}_i\qquad\Rightarrow\qquad\frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B_i}\\\hline \Delta & \mathcal{A}\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B_1\& B_2}\\\hline \Delta & \mathcal{A}\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-S}_i$$

$$\frac{\Sigma : \Psi; B \Longrightarrow \emptyset; \emptyset; \Upsilon}{\Sigma : \Psi; \emptyset \overset{?B}{\Longrightarrow} \emptyset; \Upsilon}\ ?\text{-S} \qquad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline B & \emptyset & \emptyset \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{?B} \\ \hline \emptyset & \emptyset \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ ?\text{-S}$$

$$\frac{\Sigma : \Psi; \Delta \overset{B[x \mapsto t]}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma : \Psi; \Delta \overset{\forall_\tau x B}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \forall\text{-S} \qquad \Rightarrow \quad \frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B[x \mapsto t]} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\forall_\tau x B} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \forall\text{-S}$$

$$\frac{\Sigma : \Psi; \emptyset \Longrightarrow \emptyset; B; \Upsilon \quad \Sigma : \Psi; \Delta \overset{C}{\Longrightarrow} \mathcal{A}; \Upsilon}{\Sigma : \Psi; \Delta \overset{B \supset C}{\Longrightarrow} \mathcal{A}; \Upsilon}\ \supset\text{-S} \quad \Rightarrow \quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \emptyset & \emptyset & B \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \supset C} \\ \hline \Delta & \mathcal{A} \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \supset\text{-S}$$

A.4 Soundness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$

For each $\mathcal{B}'$ rule we give a $\mathcal{B}$ rule that has an empty output context. Each box on the right-hand side consists of the contexts which are consumed in the corresponding proof branch of the given box on the left-hand side.

$$\frac{}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^\top} & \multicolumn{2}{c|}{\top,\mathcal{B}^0,\mathcal{B}^\top} \\ \hline \emptyset & \mathcal{M}^\top & \emptyset & \mathcal{A}^\top & \emptyset & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R} \quad\Rightarrow\quad \frac{}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & \top,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R}$$

$$\frac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{A,\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{A,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{atomic}\text{-R} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & A,\mathcal{A}^0 & \mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & A,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{atomic}\text{-R}$$

$$\frac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{\bot,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & \mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & \bot,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R}$$

$$\frac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B,C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Sigma} & \multicolumn{4}{c|}{\Psi} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\parr C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-R} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & B,C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Sigma & \multicolumn{2}{c|}{\Psi} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & B\parr C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-R}$$

$$\frac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{B,\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\multimap C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \multimap\text{-R} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,B & \mathcal{A}^0 & C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & B\multimap C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \multimap\text{-R}$$

$$\frac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi,B} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top} & \multicolumn{2}{c|}{B\supset C,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top} \\ \hline \mathcal{M}^1 & \mathcal{M}^\top & \mathcal{A}^1 & \mathcal{A}^\top & \mathcal{B}^1 & \mathcal{B}^\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \supset\text{-R} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi,B & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0 & \mathcal{A}^0 & B\supset C,\mathcal{B}^0 \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \supset\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; B[x\mapsto y],\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \mathcal{B}^\top \\ \hline y:\tau,\Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; \forall_\tau xB,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \mathcal{B}^\top \\ \hline \Sigma \\ \hline \end{array}}\ \forall\text{-R} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; B[x\mapsto y],\mathcal{B}^0 \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline y:\tau,\Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; \forall_\tau xB,\mathcal{B}^0 \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \forall\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; \mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \mathcal{B}^\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; ?B,\mathcal{B}^0,\mathcal{B}^1,\mathcal{B}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \mathcal{B}^\top \\ \hline \Sigma \\ \hline \end{array}}\ ?\text{-R} \quad\Rightarrow\quad \dfrac{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; \mathcal{B}^0 \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; ?B,\mathcal{B}^0 \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ ?\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline B \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline B,\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; \mathcal{B}^1 \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose} \quad\Rightarrow\quad \dfrac{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline B \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \\ \hline \emptyset \;\|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline \mathcal{M}^0,B \;\|\; \mathcal{A}^0 \;|\; \emptyset \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi,B \;\|\; \Upsilon \\ \hline B \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi,B \;\|\; \Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; \mathcal{B}^1 \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose!} \quad\Rightarrow\quad \dfrac{\begin{array}{|c|}\hline \Psi,B \;\|\; \Upsilon \\ \hline B \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \\ \hline \emptyset \;\|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi,B \;\|\; \Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; \emptyset \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose!}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; B \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^\top \;\|\; \mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^\top \;|\; \mathcal{B}^1 \\ \hline \mathcal{M}^1 \;|\; \mathcal{M}^\top \;\|\; \mathcal{A}^1 \;|\; \mathcal{A}^\top \;|\; \mathcal{B}^1 \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose?} \quad\Rightarrow\quad \dfrac{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; B \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \;\|\; B,\Upsilon \\ \hline \mathcal{M}^0 \;\|\; \mathcal{A}^0 \;|\; \emptyset \\ \hline \emptyset \;\|\; \emptyset \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose?}$$

$$\dfrac{}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline A \\ \hline \mathcal{M}^1 \;\|\; A,\mathcal{A}^1 \\ \hline \mathcal{M}^1 \;|\; \emptyset \;\|\; \mathcal{A}^1 \;|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{initial} \quad\Rightarrow\quad \dfrac{}{\begin{array}{|c|}\hline \Psi \;\|\; \Upsilon \\ \hline A \\ \hline \emptyset \;\|\; A \\ \hline \emptyset \;\|\; \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{initial}$$

$$\frac{}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{A,\Upsilon}\\\hline \multicolumn{4}{|c|}{A}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^1} & \multicolumn{2}{c|}{\mathcal{A}^1}\\\hline \mathcal{M}^1 & \emptyset & \mathcal{A}^1 & \emptyset\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{initial?} \quad\Rightarrow\quad \frac{}{\begin{array}{|c||c|}\hline \Psi & A,\Upsilon\\\hline \multicolumn{2}{|c|}{A}\\\hline \emptyset & \emptyset\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \mathit{initial?}$$

$$\frac{}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{\bot}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^1} & \multicolumn{2}{c|}{\mathcal{A}^1}\\\hline \mathcal{M}^1 & \emptyset & \mathcal{A}^1 & \emptyset\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-S} \quad\Rightarrow\quad \frac{}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{\bot}\\\hline \emptyset & \emptyset\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \bot\text{-S}$$

$$\frac{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{B_i}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{B_1\&B_2}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-S}_i \quad\Rightarrow\quad \frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B_i}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B_1\&B_2}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \&\text{-S}_i$$

$$\frac{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{B} & \multicolumn{2}{c|}{\emptyset} & \multicolumn{2}{c|}{\emptyset}\\\hline \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{?B}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^1} & \multicolumn{2}{c|}{\mathcal{A}^1}\\\hline \mathcal{M}^1 & \emptyset & \mathcal{A}^1 & \emptyset\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ ?\text{-S} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline B & \emptyset & \emptyset\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{?B}\\\hline \emptyset & \emptyset\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ ?\text{-S}$$

$$\frac{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{B[x\mapsto t]}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{\forall_{\tau}xB}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ \forall\text{-S} \quad\Rightarrow\quad \frac{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B[x\mapsto t]}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{\forall_{\tau}xB}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \forall\text{-S}$$

$$\frac{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon}\\\hline \multicolumn{2}{|c||}{\emptyset} & \multicolumn{2}{c|}{\emptyset} & \multicolumn{2}{c|}{B}\\\hline \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset\\\hline \multicolumn{6}{|c|}{\Sigma}\\\hline\end{array} \quad \begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{C}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon}\\\hline \multicolumn{4}{|c|}{B\supset C}\\\hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^1,\mathcal{M}^{\top}} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^1,\mathcal{A}^{\top}}\\\hline \mathcal{M}^1 & \mathcal{M}^{\top} & \mathcal{A}^1 & \mathcal{A}^{\top}\\\hline \multicolumn{4}{|c|}{\Sigma}\\\hline\end{array}}\ \supset\text{-S}\Rightarrow \frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon}\\\hline \emptyset & \emptyset & B\\\hline \emptyset & \emptyset & \emptyset\\\hline \multicolumn{3}{|c|}{\Sigma}\\\hline\end{array} \quad \begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{C}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon\\\hline \multicolumn{2}{|c|}{B\supset C}\\\hline \mathcal{M}^0 & \mathcal{A}^0\\\hline \emptyset & \emptyset\\\hline \multicolumn{2}{|c|}{\Sigma}\\\hline\end{array}}\ \supset\text{-S}$$

A.5 Completeness of $\mathcal{B}'$ w.r.t. $\mathcal{B}$

A translation is given converting a $\mathcal{B}$ rule (which has a trivial translation) into a $\mathcal{B}'$ rule. The output context of the $\mathcal{B}$ rules are empty. The multisets with and without a subscript $\top$ correspond to multisets of formulae which are consumed by a '$\top$-R' rule or by another inference rule, respectively.

$$
\frac{}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0_\top & \mathcal{A}^0_\top & \top,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R}
\quad\Rightarrow\quad
\frac{}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\top,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \top\text{-R}
$$

$$
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & A,\mathcal{A}^0,\mathcal{A}^0_\top & \mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & A,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \mathit{atomic}\text{-R}
\Rightarrow
\frac{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{A,\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{A,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \mathit{atomic}\text{-R}
$$

$$
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & \mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & \bot,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R}
\Rightarrow
\frac{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\bot,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-R}
$$

$$
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & B,C,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Sigma & \multicolumn{2}{c|}{\Psi} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & B\parr C,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-R}
\Rightarrow
\frac{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{B,C,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Sigma} & \multicolumn{4}{c|}{\Psi} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{B\parr C,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \parr\text{-R}
$$

$$
\frac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline B,\mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & C,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & B\multimap C,\mathcal{B}^0,\mathcal{B}^0_\top \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \multimap\text{-R}
\Rightarrow
\frac{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{B,\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{C,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|cc||cc|cc|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{B\multimap C,\mathcal{B}^0,\mathcal{B}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \mathcal{B}^0_\top \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \multimap\text{-R}
$$

$$\dfrac{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid C, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid B \supset C, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \supset\text{-R} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid C, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid B \supset C, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline \Sigma \\ \hline \end{array}}\ \supset\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid B[x \mapsto y], \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline y : \tau, \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \forall_\tau x B, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \forall\text{-R} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid B[x \mapsto y], \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline y : \tau, \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \forall_\tau x B, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline \Sigma \\ \hline \end{array}}\ \forall\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \parallel B, \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid ?B, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ ?\text{-R} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi \parallel B, \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid ?B, \mathcal{B}^0, \mathcal{B}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \mathcal{B}^0_\top \\ \hline \Sigma \\ \hline \end{array}}\ ?\text{-R}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline B \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset \parallel \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline B, \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \emptyset \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline B \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi \parallel \Upsilon \\ \hline B, \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \emptyset \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose}$$

$$\dfrac{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline B \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset \parallel \emptyset \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \emptyset \\ \hline \emptyset \parallel \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose!} \Rightarrow \dfrac{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline B \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \\ \hline \Sigma \\ \hline \end{array}}{\begin{array}{|c|}\hline \Psi, B \parallel \Upsilon \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top \parallel \mathcal{A}^0, \mathcal{A}^0_\top \mid \emptyset \\ \hline \emptyset \mid \mathcal{M}^0_\top \parallel \emptyset \mid \mathcal{A}^0_\top \mid \emptyset \mid \emptyset \\ \hline \Sigma \\ \hline \end{array}}\ \mathit{choose!}$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & B \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|}\hline \Psi & \multicolumn{2}{c|}{B,\Upsilon} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top & \emptyset \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{choose?} \Rightarrow \dfrac{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{4}{c|}{B,\Upsilon} \\ \hline \multicolumn{2}{|c|}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{B} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{4}{c|}{B,\Upsilon} \\ \hline \multicolumn{2}{|c|}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} & \multicolumn{2}{c|}{\emptyset} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{choose?}$$

$$\dfrac{}{\begin{array}{|c|c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \emptyset & A \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial} \quad \Rightarrow \quad \dfrac{}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{A} \\ \hline \multicolumn{2}{|c|}{\emptyset} & \multicolumn{2}{c|}{A} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial}$$

$$\dfrac{}{\begin{array}{|c|c|}\hline \Psi & A,\Upsilon \\ \hline \multicolumn{2}{|c|}{A} \\ \hline \emptyset & \emptyset \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial?} \quad \Rightarrow \quad \dfrac{}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{A,\Upsilon} \\ \hline \multicolumn{4}{|c|}{A} \\ \hline \multicolumn{2}{|c|}{\emptyset} & \multicolumn{2}{c|}{\emptyset} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}\ \textit{initial?}$$

$$\dfrac{}{\begin{array}{|c|c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\bot} \\ \hline \emptyset & \emptyset \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-S} \quad \Rightarrow \quad \dfrac{}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{\bot} \\ \hline \multicolumn{2}{|c|}{\emptyset} & \multicolumn{2}{c|}{\emptyset} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}\ \bot\text{-S}$$

$$\dfrac{\begin{array}{|c|c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_i} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B_1 \& B_2} \\ \hline \mathcal{M}^0,\mathcal{M}^0_\top & \mathcal{A}^0,\mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-S}_i \Rightarrow \dfrac{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B_i} \\ \hline \multicolumn{2}{|c|}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B_1 \& B_2} \\ \hline \multicolumn{2}{|c|}{\mathcal{M}^0,\mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0,\mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}\ \&\text{-S}_i$$

$$\dfrac{\begin{array}{|c|c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline B & \emptyset & \emptyset \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{?B} \\ \hline \emptyset & \emptyset \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}\ ?\text{-S} \Rightarrow \dfrac{\begin{array}{|c|c|c|c|c|c|}\hline \multicolumn{3}{|c|}{\Psi} & \multicolumn{3}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c|}{B} & \multicolumn{2}{c|}{\emptyset} & \multicolumn{2}{c|}{\emptyset} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{?B} \\ \hline \multicolumn{2}{|c|}{\emptyset} & \multicolumn{2}{c|}{\emptyset} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}\ ?\text{-S}$$

$$\dfrac{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B[x \mapsto t]} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{\forall_\tau x B} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \ \forall\text{-S} \Rightarrow \dfrac{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B[x \mapsto t]} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{\forall_\tau x B} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} \ \forall\text{-S}$$

$$\dfrac{\begin{array}{|c||c|c|}\hline \Psi & \multicolumn{2}{c|}{\Upsilon} \\ \hline \emptyset & \emptyset & B \\ \hline \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{3}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{C} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c||c|}\hline \Psi & \Upsilon \\ \hline \multicolumn{2}{|c|}{B \supset C} \\ \hline \mathcal{M}^0, \mathcal{M}^0_\top & \mathcal{A}^0, \mathcal{A}^0_\top \\ \hline \emptyset & \emptyset \\ \hline \multicolumn{2}{|c|}{\Sigma} \\ \hline \end{array}} \ \supset\text{-S} \Rightarrow \dfrac{\begin{array}{|c|c||c|c|c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{4}{c|}{\Upsilon} \\ \hline \multicolumn{2}{|c||}{\emptyset} & \multicolumn{2}{c|}{\emptyset} & \multicolumn{2}{c|}{B} \\ \hline \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \hline \multicolumn{6}{|c|}{\Sigma} \\ \hline \end{array} \qquad \begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}}{\begin{array}{|c|c||c|c|}\hline \multicolumn{2}{|c||}{\Psi} & \multicolumn{2}{c|}{\Upsilon} \\ \hline \multicolumn{4}{|c|}{B \supset C} \\ \hline \multicolumn{2}{|c||}{\mathcal{M}^0, \mathcal{M}^0_\top} & \multicolumn{2}{c|}{\mathcal{A}^0, \mathcal{A}^0_\top} \\ \hline \emptyset & \mathcal{M}^0_\top & \emptyset & \mathcal{A}^0_\top \\ \hline \multicolumn{4}{|c|}{\Sigma} \\ \hline \end{array}} \ \supset\text{-S}$$

Appendix B

Source Code

B.1 Module aux.mod

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Helper Predicates
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% The source code is based on an earlier version by Dale Miller
% (see http://www.cis.upenn.edu/~dale/forum).
% rewritten by Christian Urban             last modified 07.09.96
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

module aux.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "split L K M" succeeds if  L is the result of appending K to M.
type split     list A -> list A -> list A -> o.

split nil nil nil.
split (X::L) (X::K) M :- split L K M.
split (X::L) K (X::M) :- split L K M.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "membNrest X L M" succeeds for every occurrence of X in L,
% where M is the result of removing that occurrence from L.
type membNrest      A -> list A -> list A -> o.

membNrest X (X::Rest) Rest.
membNrest X (Y::Tail) (Y::Rest) :- membNrest X Tail Rest.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "memb X L" succeeds if X is a member of L; this will succeed
% at most once.
type memb      A -> list A -> o.

memb X (X::Rest) :- !.
memb X (Y::Tail) :- memb X Tail.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "member X L" succeeds if X is a member of L; this will succeed
% as often as X unifies with members of L.
```

```
type member     A -> list A -> o.

member X (X::Rest).
member X (Y::Tail) :- member X Tail.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "append L K M" succeeds if M is the result of appending K to L.
type append  list A -> list A -> list A -> o.

append nil K K.
append (X::XS) YS (X::ZS) :- append XS YS ZS.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "removed F X Y" succeeds if Y contains less occurrences
% of F than X
% e.g., removed 2 (1::2::3::nil) (1::3::nil).
type    removed A -> list A -> list A -> o.

removed F CIn COut :-     count F CIn  OccIn,
                          count F COut OccOut,
                          OccIn > OccOut.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "count F X Y" succeeds if Y is the number of occurrences of
% F in X.
type    count   A -> list A -> int    -> o.

count F nil 0.
count F (F::Rest) OccNew :- count F Rest OccOld, OccNew is OccOld + 1.
count F (Y::Rest)    Occ :- not (F = Y), count F Rest Occ.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "deleteone F K L" succeeds if L is the result of removing the
% first occurrence of F in K and in case F does not occur in K,
% it succeeds when K = L.
type    deleteone   A -> list A -> list A -> o.

deleteone X nil nil.
deleteone X (X::Tail) Tail :- !.
deleteone X (Y::Tail) (Y::Rest) :-  deleteone X Tail Rest.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% multiset difference
% "diff X Y Z" succeeds if Z is the result of removing all elements
% of Y from X (Y is a submultiset of X).
% e.g.,  diff (1::2::3::4::nil) (2::4::nil) (1::3::nil)
type    diff   list A -> list A -> list A -> o.

diff nil X nil.
diff X nil X.
diff T1 (X::T2) Rest :- deleteone X T1 TRest , diff TRest T2 Rest.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% multiset intersection
% "inter X Y Z" succeeds if Z consists only on elements which are
% in X and Y
% e.g., inter (1::2::3::4::nil) (2::4::nil) (2::4::nil).
type    inter   list A -> list A -> list A -> o.
```

```
inter nil Y nil.
inter (X::Rest) Y (X::T) :- memb X Y, deleteone X Y U, inter Rest U T, !.
inter (X::Rest) Y T :- inter Rest Y T.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% removes a formula from the slack output when necessary
% "remove F X Y Z Z" succeeds if Y and Z together contain less
% occurrences of F than X;
% "remove F X Y Z W" succeeds if Y and Z together contain the
% same number of occurrences of F as X, then W is the result
% of removing an occurrence of F from Z.
% e.g., remove 1 (1::2::1::1::nil) (1::2::nil) (1::nil) (1::nil).
%       remove 1 (1::2::1::1::nil) (1::2::nil) (1::1::nil) (1::nil).
type    remove  A -> list A -> list A -> list A -> list A -> o.

remove F XI XO XS XSnew :- append XO XS XT,
                           count F XI OccI,
                           count F XT OccO,
                           (( OccI > OccO, copy XS XSnew );
                           ( OccI = OccO, memb F XS, deleteone F XS XSnew )).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "copy X Y" succeeds if list X is equivalent to list Y
type    copy list A -> list A -> o.

copy nil nil.
copy (X::R) (X::T) :- copy R T.
```

B.2 Module forumlog.mod

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Declaration of FORUM's Connectives
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% The source code is based on an earlier version by Dale Miler
% (see http://www.cis.upenn.edu/~dale/forum).
% modified by Christian Urban           last modified 07.09.96
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
module forumlog.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  The logical connectives for Forum.
%% These are divided into four groups.
%% (1) implications

  type ==>              o -> o -> o.    % intuitionistic implication
  type <==              o -> o -> o.    % reverse intuitionistic implication
  type --o              o -> o -> o.    % lollipop
  type o--              o -> o -> o.    % reverse lollipop
  infixr --o 1.
  infixl o-- 1.
  infixr ==> 1.
  infixl <== 1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% (2) unit elements

  type top            o.              % top (additive true)
  type bot            o.              % multiplicative false

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% (3) connectives

  type @                o -> o -> o.   % with (additive or)
  type |                o -> o -> o.   % par (multiplicative or)
  infixr @  3.
  infixr |  2.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% (4) quantifier and exponential

  type forall           (A -> o) -> o.  % universal quantification
  type ?                o -> o.        % why-not modal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% (5) negation and non-primitive connectives

  type neg              o -> o.

  type x                o -> o -> o.        % times
  infixr x 2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

type atomic        o -> o.
type non_atomic    o -> o.


non_atomic (B ==> C).
non_atomic (B <== C).
non_atomic (B --o C).
non_atomic (B o-- C).
non_atomic (top).
non_atomic (bot).
non_atomic (B @ C).
non_atomic (B | C).
non_atomic (? B).
non_atomic (forall B).
non_atomic (neg B).
non_atomic (B x C).

atomic B :- not (non_atomic B).
```

B.3 Module choosetest.mod

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Choose Test
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% written by Christian Urban              last modified 07.09.96
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
module choosetest.
```

```
accumulate forumlog, aux.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "choosetest F X" succeeds if F is 'bot', of the form '? B' or an
% atomic formula which is a member of X;
% in case F is a compound formula it will be decomposed recursively.
type    choosetest   A -> list A -> o.

choosetest (bot) X.
choosetest (? B) X.
choosetest A X :- atomic A, memb A X.
choosetest (B @ C) X :- choosetest B X ; choosetest C X.
choosetest (B | C) X :- choosetest B X , choosetest C X.
choosetest (B --o C) X :- choosetest C X.
choosetest (C o-- B) X :- choosetest C X.
choosetest (B ==> C) X :- choosetest C X.
choosetest (C <== B) X :- choosetest C X.
choosetest (forall B) X :- choosetest (B T) X.
choosetest (neg B) X :- choosetest (B --o bot) X.
choosetest (B x C) X :- choosetest (neg ((neg B) | (neg C))) X.
```

B.4 Module forum.mod (Calculus $\mathcal{F}'$)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Proof system F'
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% The source code is based on an earlier version by Dale Miler
% (see http://www.cis.upenn.edu/~dale/forum).
% rewritten by Christian Urban              last modified 07.09.96
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The right predicate represents the provability of a non-stoup sequent;
% the stoup predicate represents the provability of a stoup sequent.
module forum.

accumulate forumlog, aux, choosetest.

type right    list o ->           % Psi            (classical)
              list o ->           % Delta formulae (linear)
              list o ->           % AC formulae    (linear)
              list o ->           % BC formulae    (linear)
              list o -> o.        % Upsilon    (classical)

type stoup    list o ->           % Psi            (classical)
              list o ->           % Delt formulae (linear)
              o ->                % Stoup
              list o ->           % AC formulae    (linear)
              list o -> o.        % Upsilon        (classical)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% atomic-R rule
right Psi Delta AC (A::BC) Upsilon :-
  atomic A,
  right Psi Delta (A::AC) BC Upsilon.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-R rule (@-R rule)
right Psi Delta AC ((B @ C)::BC) Upsilon :-
  right Psi Delta AC (B::BC) Upsilon ,
  right Psi Delta AC (C::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-R rule
right Psi Delta AC (bot::BC) Upsilon :-
  right Psi Delta AC BC Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-R rule (|-R rule)
right Psi Delta AC ((B | C)::BC) Upsilon :-
  right Psi Delta AC (B::C::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-R rule (--o-R rule)
right Psi Delta AC ((B --o C)::BC) Upsilon :-
  right Psi (B::Delta) AC (C::BC) Upsilon.

right Psi Delta AC ((C o-- B)::BC) Upsilon :-
  right Psi (B::Delta) AC (C::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hook-R rule (==>-R rule)

right Psi Delta AC ((B ==> C)::BC) Upsilon :-
  right (B::Psi) Delta AC (C::BC) Upsilon.

right Psi Delta AC ((C <== B)::BC) Upsilon :-
  right (B::Psi) Delta AC (C::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-R rule

right Psi Delta AC ((forall B)::BC) Upsilon :-
  pi x\ (right Psi Delta AC ((B x)::BC) Upsilon).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-R rule

right Psi Delta AC ((? B)::BC) Upsilon :-
  right Psi Delta AC BC (B::Upsilon).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-R rule (x-R)

right Psi Delta AC ((B x C)::BC) Upsilon :-
  right Psi Delta AC ((neg ((neg B) | (neg C)))::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-R rule (neg-R)

right Psi Delta AC ((neg B)::BC) Upsilon :-
  right Psi Delta AC ((B --o bot)::BC) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initial rule
```

```
stoup Psi nil A (A::nil) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inititial! rule

stoup Psi nil A nil Upsilon :-
  atomic A,
  memb A Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-S rule

stoup Psi nil bot nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-S rule (@-S rule)

stoup Psi Delta (B @ C) AC Upsilon :-
  stoup Psi Delta B AC Upsilon;
  stoup Psi Delta C AC Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-S rule

stoup Psi nil (? B) nil Upsilon :-
  right Psi (B::nil) nil nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-S rule (|-S rule)

stoup Psi Delta (B | C) AC Upsilon :-
  split Delta Delta1 Delta2,
  split AC AC1 AC2,
  stoup Psi Delta1 B AC1 Upsilon ,
  stoup Psi Delta2 C AC2 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-S rule

stoup Psi Delta (forall B) AC Upsilon :-
  stoup Psi Delta (B T) AC Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-S rule (--o-S rule)

stoup Psi Delta (B --o C) AC Upsilon :-
  split Delta Delta1 Delta2,
  split AC AC1 AC2,
  stoup Psi Delta1 C AC1 Upsilon,
  right Psi Delta2 AC2 (B::nil) Upsilon.

stoup Psi Delta (C o-- B) AC Upsilon :-
  split Delta Delta1 Delta2,
  split AC AC1 AC2,
  stoup Psi Delta1 C AC1 Upsilon,
  right Psi Delta2 AC2 (B::nil) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Hook-S rule (==>-S rule)

stoup Psi Delta (B ==> C) AC Upsilon :-
  stoup Psi Delta C AC Upsilon ,
  right Psi nil nil (B::nil) Upsilon.

stoup Psi Delta (C <== B) AC Upsilon :-
  stoup Psi Delta C AC Upsilon,
  right Psi nil nil (B::nil) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-S rule (x-S)

stoup Psi Delta (B x C) AC Upsilon :-
  stoup Psi Delta (neg ((neg B) |(neg C))) AC Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-S rule (neg-S)

stoup Psi Delta (neg B) AC Upsilon :-
  stoup Psi Delta (B --o bot) AC Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% choose  rule
% choose! rule
% choose? rule

right Psi BDelta AC nil Upsilon :-
  membNrest B BDelta Delta,
   append AC Upsilon Rightside,
   choosetest B Rightside,
  stoup  Psi Delta B AC Upsilon.

right Psi Delta AC nil Upsilon :-
  member C Psi,
   append AC Upsilon Rightside,
   choosetest C Rightside,
  stoup  Psi Delta C AC Upsilon.

right Psi Delta AC nil Upsilon :-
  member D Upsilon,
  right Psi Delta AC (D::nil) Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% top-R rule

right Psi Delta AC (top::BC) Upsilon.
```

B.5 Module forum.mod (Box Calculus $\mathcal{B}$)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Proof System B
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% The source code is based on an earlier version by Dale Miler
% (see http://www.cis.upenn.edu/~dale/forum).
% rewritten by Christian Urban              last modified 07.09.96
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The right predicate represents the provability of a non-stoup box;
% the stoup predicate represents the provability of a stoup box.
module forum.

accumulate forumlog, aux, choosetest.

type right    list o ->               % Psi        (classical)
              list o -> list o ->     % M formulae (linear)
              list o -> list o ->     % A formulae (linear)
              list o -> list o ->     % B formulae (linear)
              list o -> o.            % Upsilon    (classical)

type stoup    list o ->               % Psi        (classical)
              list o -> list o ->     % M formulae (linear)
              o ->                    % Stoup
              list o -> list o ->     % A formulae (linear)
              list o -> o.            % Upsilon    (classical)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% atomic-R rule
right Psi MOM1 M1 AOA1 A1 (A::BOB1) B1 Upsilon :-
  atomic A,
  right Psi MOM1 M1 (A::AOA1) A1 BOB1 B1 Upsilon ,
  removed A (A::AOA1) A1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-R rule (@-R rule)
right Psi MOM1 M1 AOA1 A1 ((B @ C)::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 (B::BOB1) B1 Upsilon,
  removed B (B::BOB1) B1,
  diff MOM1 M1 MO,
  diff AOA1 A1 AO,
  diff BOB1 B1 BO,
  right Psi MO nil AO nil (C::BO) nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-R rule
right Psi MOM1 M1 AOA1 A1 (bot::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 BOB1 B1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-R rule (|-R rule)
right Psi MOM1 M1 AOA1 A1 ((B | C)::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 (B::C::BOB1) B1 Upsilon,
  removed B (B::BOB1) B1,
  removed C (C::BOB1) B1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-R rule (--o-R rule)
right Psi MOM1 M1 AOA1 A1 ((B --o C)::BOB1) B1 Upsilon :-
  right Psi (B::MOM1) M1 AOA1 A1 (C::BOB1) B1 Upsilon,
  removed C (C::BOB1) B1,
  removed B (B::MOM1) M1.

right Psi MOM1 M1 AOA1 A1 ((C o-- B)::BOB1) B1 Upsilon :-
  right Psi (B::MOM1) M1 AOA1 A1 (C::BOB1) B1 Upsilon,
  removed C (C::BOB1) B1,
  removed B (B::MOM1) M1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hook-R rule (==>-R rule)

right Psi MOM1 M1 AOA1 A1 ((B ==> C)::BOB1) B1 Upsilon :-
  right (B::Psi) MOM1 M1 AOA1 A1 (C::BOB1) B1 Upsilon ,
  removed C BOB1 B1.

right Psi MOM1 M1 AOA1 A1 ((C <== B)::BOB1) B1 Upsilon :-
  right (B::Psi) MOM1 M1 AOA1 A1 (C::BOB1) B1 Upsilon ,
  removed C (C::BOB1) B1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-R rule

right Psi MOM1 M1 AOA1 A1 ((forall B)::BOB1) B1 Upsilon :-
  pi x\ (right Psi MOM1 M1 AOA1 A1 ((B x)::BOB1) B1 Upsilon,
         removed (B x) ((B x)::BOB1) B1 ).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-R rule

right Psi MOM1 M1 AOA1 A1 ((? B)::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 BOB1 B1 (B::Upsilon).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-R rule (x-R)

right Psi MOM1 M1 AOA1 A1 ((B x C)::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 ((neg ((neg B) | (neg C)))::BOB1) B1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-R rule (neg-R)

right Psi MOM1 M1 AOA1 A1 ((neg B)::BOB1) B1 Upsilon :-
  right Psi MOM1 M1 AOA1 A1 ((B --o bot)::BOB1) B1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initial rule

stoup Psi M1 M1 A AA1 A1 Upsilon :-
  atomic A,
  membNrest A AA1 A1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inititial! rule

stoup Psi M1 M1 A A1 A1 Upsilon :-
  atomic A,
  memb A Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-S rule

stoup Psi M1 M1 bot A1 A1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-S rule (@-S rule)
```

```
stoup Psi M0M1 M1 (B @ C) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 B A0A1 A1 Upsilon;
  stoup Psi M0M1 M1 C A0A1 A1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-S rule

stoup Psi M1 M1 (? B) A1 A1 Upsilon :-
  right Psi (B::nil) nil nil nil nil nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-S rule (|-S rule)

stoup Psi M0M1M2 M2 (B | C) A0A1A2 A2 Upsilon :-
  stoup Psi M0M1M2 M1M2 B A0A1A2 A1A2 Upsilon,
  stoup Psi M1M2 M2 C A1A2 A2 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-S rule

stoup Psi M0M1 M1 (forall B) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 (B T) A0A1 A1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-S rule (--o-S rule)

stoup Psi M0M1M2 M2 (B --o C) A0A1A2 A2 Upsilon :-
  stoup Psi M0M1M2 M1M2 C A0A1A2 A1A2 Upsilon,
  right Psi M1M2 M2 A1A2 A2 (B::nil) nil Upsilon.

stoup Psi M0M1M2 M2 (C o-- B) A0A1A2 A2 Upsilon :-
  stoup Psi M0M1M2 M1M2 C A0A1A2 A1A2 Upsilon,
  right Psi M1M2 M2 A1A2 A2 (B::nil) nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hook-S rule (==>-S rule)

stoup Psi M0M1 M1 (B ==> C) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 C A0A1 A1 Upsilon ,
  right Psi nil nil nil nil (B::nil) nil Upsilon.

stoup Psi M0M1 M1 (C <== B) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 C A0A1 A1 Upsilon ,
  right Psi nil nil nil nil (B::nil) nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-S rule (x-S)

stoup Psi M0M1 M1 (B x C) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 (neg ((neg B) |(neg C))) A0A1 A1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-S rule (neg-S)

stoup Psi M0M1 M1 (neg B) A0A1 A1 Upsilon :-
  stoup Psi M0M1 M1 (B --o bot) A0A1 A1 Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% choose  rule
% choose! rule
% choose? rule

right Psi BMOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  membNrest B BMOM1 MOM1,
   append AOA1 Upsilon Rightside,
   choosetest B Rightside,
  stoup Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Psi,
   append AOA1 Upsilon Rightside,
   choosetest B Rightside,
  stoup  Psi MOM1 M1 B AOA1 A1 Upsilon.

right Psi MOM1 M1 AOA1 A1 B1 B1 Upsilon :-
  member B Upsilon,
  right Psi MOM1 M1 AOA1 A1 (B::nil) nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% top-R rule

right Psi MOM1 M1 AOA1 A1 (top::BOB1) B1 Upsilon :-
  split MOM1 MO M1,
  split AOA1 AO A1,
  split BOB1 BO B1.
```

B.6 Module forum.mod (Box Calculus $\mathcal{B}'$)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Proof System B'
% (written in Terzo lambda-Prolog, Version 1.0b)
%
% The source code is based on an earlier version by Dale Miler
% (see http://www.cis.upenn.edu/~dale/forum).
% rewritten by Christian Urban                last modified 07.09.96
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The right predicate represents the provability of a non-stoup sequent;
% the stoup predicate represents the provability of a stoup sequent.

module forum.

accumulate forumlog, aux, choosetest.

type right   list o ->                          % Psi        (classical)
             list o -> list o -> list o ->      % M formulae (linear)
             list o -> list o -> list o ->      % A formulae (linear)
             list o -> list o -> list o ->      % B formulae (linear)
             list o -> o.                       % Upsilon    (classical)

type stoup   list o ->                          % Psi        (classical)
             list o -> list o -> list o ->      % M formulae (linear)
             o ->                               % Stoup
             list o -> list o -> list o ->      % A formulae (linear)
             list o -> o.                       % Upsilon    (classical)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% atomic-R rule
right Psi M0M1MT M1 MT A0A1AT A1 AT (A::B0B1BT) B1 BT Upsilon :-
  atomic A,
  right Psi M0M1MT M1 MT (A::A0A1AT) A1 AT' B0B1BT B1 BT Upsilon,
  remove A (A::A0A1AT) A1 AT' AT.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-R rule (@-R rule)
right Psi M0M1MT M1M1* MT^MT* A0A1AT A1A1* AT^AT*
                                   ((B @ C)::B0B1BT) B1B1* BT^BT* Upsilon :-
  right Psi M0M1MT M1 MT  A0A1AT A1 AT (B::B0B1BT) B1 BT' Upsilon,
  remove B (B::B0B1BT) B1 BT' BT,
  diff M0M1MT M1 M0,
  diff A0A1AT A1 A0,
  diff B0B1BT B1 B0,
  append M0 MT M0*M1*MT*,
  append A0 AT A0*A1*AT*,
  append B0 BT B0*B1*BT*,
  right Psi M0*M1*MT* M1* MT* A0*A1*AT* A1* AT* (C::B0*B1*BT*) B1* BT* Upsilon,
  inter M0 M1* nil,
  inter A0 A1* nil,
  inter B0 B1* nil,
  append M1 M1* M1M1*,
  append A1 A1* A1A1*,
  append B1 B1* B1B1*,
  inter MT MT* MT^MT*,
  inter AT AT* AT^AT*,
  inter BT BT* BT^BT*.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-R rule
right Psi M0M1MT M1 MT A0A1AT A1 AT (bot::B0B1BT) B1 BT Upsilon :-
  right Psi M0M1MT M1 MT A0A1AT A1 AT B0B1BT B1 BT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-R rule (|-R rule)
right Psi  M0M1MT M1 MT A0A1AT A1 AT ((B | C)::B0B1BT) B1 BT Upsilon :-
  right Psi  M0M1MT M1 MT A0A1AT A1 AT (B::C::B0B1BT) B1 BT' Upsilon,
  remove B (B::B0B1BT) B1 BT' BT'',
  remove C (C::B0B1BT) B1 BT'' BT.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-R rule (--o-R rule)
right Psi M0M1MT M1 MT A0A1AT A1 AT ((B --o C)::B0B1BT) B1 BT Upsilon :-
  right Psi (B::M0M1MT) M1 MT' A0A1AT A1 AT (C::B0B1BT) B1 BT' Upsilon,
  remove C (C::B0B1BT) B1 BT' BT,
  remove B (B::M0M1MT) M1 MT' MT.

right Psi M0M1MT M1 MT A0A1AT A1 AT ((C o-- B)::B0B1BT) B1 BT Upsilon :-
  right Psi (B::M0M1MT) M1 MT' A0A1AT A1 AT (C::B0B1BT) B1 BT' Upsilon,
  remove C (C::B0B1BT) B1 BT' BT,
  remove B (B::M0M1MT) M1 MT' MT.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hook-R rule (==>-R rule)

right Psi M0M1MT M1 MT A0A1AT A1 AT ((B ==> C)::B0B1BT) B1 BT Upsilon :-
```

```
  right (B::Psi) M0M1MT M1 MT A0A1AT A1 AT (C::B0B1BT) B1 BT' Upsilon,
  remove C (C::B0B1BT) B1 BT' BT.

right Psi M0M1MT M1 MT A0A1AT A1 AT ((C <== B)::B0B1BT) B1 BT Upsilon :-
  right (B::Psi) M0M1MT M1 MT A0A1AT A1 AT (C::B0B1BT) B1 BT' Upsilon,
  remove C (C::B0B1BT) B1 BT' BT.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-R rule

right Psi M0M1MT M1 MT A0A1AT A1 AT ((forall B)::B0B1BT) B1 BT Upsilon :-
  pi x\ (right Psi M0M1MT M1 MT A0A1AT A1 AT ((B x)::B0B1BT) B1 BT' Upsilon,
         remove (B x) ((B x)::B0B1BT) B1 BT' BT).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-R rule

right Psi M0M1MT M1 MT A0A1AT A1 AT ((? B)::B0B1BT) B1 BT Upsilon :-
  right Psi M0M1MT M1 MT A0A1AT A1 AT B0B1BT B1 BT (B::Upsilon).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-R rule (x-R)

right Psi M0M1MT M1 MT A0A1AT A1 AT ((B x C)::B0B1BT) B1 BT Upsilon :-
  right Psi M0M1MT M1 MT A0A1AT A1 AT
                        ((neg ((neg B) | (neg C)))::B0B1BT) B1 BT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-R rule (neg-R)

right Psi M0M1MT M1 MT A0A1AT A1 AT ((neg B)::B0B1BT) B1 BT Upsilon :-
  right Psi M0M1MT M1 MT A0A1AT A1 AT ((B --o bot)::B0B1BT) B1 BT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% top-R rule

right Psi MT nil MT AT nil AT (top::BT) nil BT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initial rule

stoup Psi M1 M1 nil A AA1 A1 nil Upsilon :-
  atomic A,
  membNrest A AA1 A1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inititial! rule

stoup Psi M1 M1 nil  A A1 A1 nil Upsilon :-
  atomic A,
  memb A Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bot-S rule

stoup Psi M1 M1 nil bot A1 A1 nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% With-S rule (@-S rule)
```

```
stoup Psi M0M1MT M1 MT (B @ C) A0A1AT A1 AT Upsilon :-
  stoup Psi M0M1MT M1 MT B A0A1AT A1 AT Upsilon;
  stoup Psi M0M1MT M1 MT C A0A1AT A1 AT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ?-S rule

stoup Psi M1 M1 nil (? B) A1 A1 nil Upsilon :-
  right Psi (B::nil) nil nil nil nil nil nil nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Par-S rule (|-S rule)

stoup Psi M0M1MT M1^M1* MT^M1*MT* (B | C) A0A1AT A1^A1* AT^A1*AT* Upsilon :-
  stoup  Psi M0M1MT M1 MT B A0A1AT A1 AT Upsilon,
  append M1 MT M0*M1*MT*,
  append A1 AT A0*A1*AT*,
  stoup Psi M0*M1*MT* M1* MT* C A0*A1*AT* A1* AT* Upsilon,
  inter M1 M1* M1^M1*,
  inter A1 A1* A1^A1*,
  inter MT M1* MT^M1*,
  inter AT A1* AT^A1*,
  append MT^M1* MT* MT^M1*MT*,
  append AT^A1* AT* AT^A1*AT*.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forall-S rule

stoup Psi M0M1MT M1 MT (forall B) A0A1AT A1 AT Upsilon :-
  stoup Psi M0M1MT M1 MT (B T) A0A1AT A1 AT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lolli-S rule (--o-S rule)

stoup Psi M0M1MT M1^M1* MT^M1*MT* (B --o C) A0A1AT A1^A1* AT^A1*AT* Upsilon :-
  stoup  Psi M0M1MT M1 MT C A0A1AT A1 AT Upsilon,
  append M1 MT M0*M1*MT*,
  append A1 AT A0*A1*AT*,
  right Psi M0*M1*MT* M1* MT* A0*A1*AT* A1* AT* (B::nil) nil nil Upsilon,
  inter M1 M1* M1^M1*,
  inter A1 A1* A1^A1*,
  inter MT M1* MT^M1*,
  inter AT A1* AT^A1*,
  append MT^M1* MT* MT^M1*MT*,
  append AT^A1* AT* AT^A1*AT*.

stoup Psi M0M1MT M1^M1* MT^M1*MT* (C o-- B) A0A1AT A1^A1* AT^A1*AT* Upsilon :-
  stoup  Psi M0M1MT M1 MT C A0A1AT A1 AT Upsilon,
  append M1 MT M0*M1*MT*,
  append A1 AT A0*A1*AT*,
  right Psi M0*M1*MT* M1* MT* A0*A1*AT* A1* AT* (B::nil) nil nil Upsilon,
  inter M1 M1* M1^M1*,
  inter A1 A1* A1^A1*,
  inter MT M1* MT^M1*,
  inter AT A1* AT^A1*,
  append MT^M1* MT* MT^M1*MT*,
  append AT^A1* AT* AT^A1*AT*.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Hook-S rule (==>-S rule)

stoup Psi MOM1MT M1 MT (B ==> C) AOA1AT A1 AT Upsilon :-
  stoup  Psi MOM1MT M1 MT C AOA1AT A1 AT Upsilon,
  right Psi nil nil MT* nil nil AT* (B::nil) nil nil Upsilon.

stoup Psi MOM1MT M1 MT (C <== B) AOA1AT A1 AT Upsilon :-
  stoup  Psi MOM1MT M1 MT C AOA1AT A1 AT Upsilon,
  right Psi nil nil MT* nil nil AT* (B::nil) nil nil Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Times-S rule (x-S)

stoup Psi MOM1MT M1 MT (B x C) AOA1AT A1 AT Upsilon :-
  stoup Psi MOM1MT M1 MT (neg ((neg B) |(neg C))) AOA1AT A1 AT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% neg-S rule (neg-S)

stoup Psi MOM1MT M1 MT (neg B) AOA1AT A1 AT Upsilon :-
  stoup Psi MOM1MT M1 MT (B --o bot) AOA1AT A1 AT Upsilon.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% choose  rule
% choose! rule
% choose? rule

right Psi BMOM1MT M1 MT AOA1AT A1 AT B1 B1 nil Upsilon :-
  membNrest B BMOM1MT MOM1MT,
   append AOA1AT Upsilon Rightside,
   choosetest B Rightside,
  stoup  Psi MOM1MT M1 MT B AOA1AT A1 AT  Upsilon.

right Psi MOM1MT M1 MT AOA1AT A1 AT B1 B1 nil Upsilon :-
  member B Psi,
   append AOA1AT Upsilon Rightside,
   choosetest B Rightside,
  stoup  Psi MOM1MT M1 MT B AOA1AT A1 AT  Upsilon.

right Psi MOM1MT M1 MT AOA1AT A1 AT B1 B1 nil Upsilon :-
  member B Upsilon,
  right Psi MOM1MT M1 MT AOA1AT A1 AT (B::nil) nil nil Upsilon.
```