NORMALISATION TECHNIQUES IN PROOF THEORY AND CATEGORY THEORY

Taher Tawfik Ahmed Hamza

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

1986

NORMALISATION TECHNIQUES IN PROOF THEORY AND CATEGORY THEORY. AN IMPLEMENTATION AND APPLICATIONS.

A thesis submitted to the university
of St. Andrews for the degree of
Doctor of philosophy

by

Taher Tawfik Ahmed Hamza

Department of Computational Science,
University of St. Andrews. November 1985.

I Taher Tawfik Hamza hereby certify that this thesis has been composed by myself, that it is a record of my own work, and that it has not been accepted in partial or complete fulfillment of any other degree or professional qualification.

Signed . ... Date ..18...iii..1986

I was admitted to the Faculty of Science of the University of St. Andrews under Ordinance No. 12 on April 1982 and as a candidate for the degree of Ph.D. on October 1983.

signed . ... Date ..18...iii..1986

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 <u>bona fide</u> library or research worker.

CERTIFICATE

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate to the degree of Ph.D.

signature of Supervisor Date 18.iii.86

ACKNOWLEDGEMENTS

I would like to express indebtedness and sincere thanks to my supervisor, Dr R. Dyckhoff, for his invaluable assistance in the preparation of this work. He has always been a great help and spared no effort in giving me advice, guidance and supervision of this work. I also thank the Egyptian government for the financial support I have received during this study.

Finally I owe apology to my wife and my children Sulaf and Ahmed for lack of attendance during the preparation of this thesis.

ABSTRACT

The word problem for the free categories with some structure generated by a category X can be solved using proof-theoretical means. These free categories give a semantics in which derivations of GENTZEN's propositional sequent calculus can be interpreted by means of arrows of these categories.

In this thesis we describe, implement and document the cut-elimination and the normalization techniques in proof theory as outlined in SZABO [1978]; we show how these are used in order to solve, mechanically, the word problem for the free categories with structure of : cartesian, bicartesian, distributive bicartesian, cartesian closed, and bicartesian closed. This implementation is extended by a procedure to interpret intuitionistic propositional sequent derivations as arrows of the above categories. Implementation of these techniques has forced us to modify the techniques in various inessential ways.

The description and the representation in the syntax of our implementation of the above categories is contained in chapters 1 – 5, where each chapter describes one theory and concludes with examples of the system in use to represent concepts and solve simple word problems from category theory (of various types). Appendix 1 contains some apparent printing errors we have observed in the work done by SZABO. The algorithms used in the proof of the cut-elimination theorems and normalization through chapters 1 – 5 are collected in appendices 2 – 4. Appendices 5 – 8 concern the implementation and its user manual.

CONTENTS

Chapter 0

INTRODUCTION

The purpose of this thesis is the modification and implementation of the cut-elimination and normalization algorithms described by SZABO in his book, Algebra of Proofs [1978] (hereinafter referred to only as SZABO). This implementation is for the assistance both of category theorists and logicians; for the first, as a step towards the automation of reasoning about morphisms in categories, for the second as a practical method for transformation of proofs in GENTZEN's propositional sequent calculus as expounded by SZABO. Our implementation is in C-Prolog on a VAX 11/750 computer under Unix BSD 4.2.

The connection between category theory and proof theory behind our implementation was used by LAMBEK [1968 , 1969 , 1972], MANN [1975] and SZABO. This connection shows that, on the one hand, category theory gives rise to non-trivial semantics for derivations in the intuitionistic Gentzen systems,. by constructing the free category of a certain type, having as objects the formulae of the relevant deductive system and as arrows, the equivalence classes of proofs; this free category is an algebraic model of the class of proofs in this deductive system. On the other hand proof theory provides both a suitable syntax in which arrows of a free category can be represented, and also techniques, such as cut-elimination and normalization, for transforming such representations. These techniques can be used to solve the word problem in, for example, free cartesian (-closed, ...) categories: equality of arrows is checked by checking their normal representations for syntactic equality.

(The word problem for a left adjoint F : Cat -> C to a forgetful functor U : C -> Cat is defined as the problem of finding an algorithm which decides the equality of arrows of the category F(X) relative to the equality of arrows of the category X)

In the following, I introduce the two main techniques behind our implementation.

a) Cut-elimination

When Gentzen [1969] examined the specific properties of the 'natural calculus ' , he was led to the conjecture that it should be possible to bring purely logical proofs into a certain ' normal form ' in which all concepts required for the proof would in some sense appear in the conclusion of the proof. In order to enunciate and prove his theorem which he called the 'Hauptsatz', however, Gentzen had to abandon his natural 'assumption' calculi and formulate logistic 'sequent' calculi in which the logical rules can be divided into ' structural ' and ' operational' rules. The Hauptsatz then refers to the fact that one of the 'structural' inference rules of the calculus, the 'cut' rule, can be climinated from purely logical proofs. The cut rule as stated by Gentzen is :

$$\frac{G \rightarrow H\ ,\ a \qquad a\ ,\ D \rightarrow L}{G\ ,\ D \rightarrow H\ ,\ L}$$

SZABO gives a detailed algorithm for the elimination of cuts from proofs in a propositional sequent calculus, with a slightly different ' cut ' rule:

$$\frac{G \rightarrow F\ a\ E \qquad D\ a\ L \rightarrow H}{D\ G\ L \rightarrow F\ H\ E}$$

b) Normalization

SZABO defined a monotone partial ordering relation (>=) on the class of GENTZEN derivations. A derivation p is defined to be normal if it is cut-free and if p >= q implies that p = q. A derivation p , say, reduces to q if there exists a finite sequence < p1, ..., pn > of derivations such that p = p1, pn = q and p1 >= p2 >= ...... >= pn and p , say, reduces immediately to q, if p = q or if p >= q by virtue of precisely one of the defining conditions of ' >= ' . Every derivation p which represents an arrow of one of the categories in chapters 1 - 5 , reduces to a cut-free derivation q , and an induction on the number of violations of the conditions , in appendices 3 and 4, occurring in q shows that p reduces to a normal, and hence unique normal, derivation r.

It is worth mentioning the relation between this approach of SZABO (which we have implemented), and other approaches to such problems. KREISEL [1971] and PRAWITZ [1971] have developed Gentzen's approach of 'natural deduction '; PRAWITZ defined an equivalence relation on proofs in the natural deduction systems and proved some theorems for the normalization of these proofs. MANN [1975] showed that the equivalence relation defined by PRAWITZ for the natural deduction proofs is the same as the equivalence relation defined by SZABO for the sequent calculus proofs, in the following ways : firstly, the equivalence classes (under PRAWITZ's definition) can be used to form the maps of a cartesian closed category, and secondly, under the usual translation from the sequent calculus systems to the natural deduction systems, two equivalent proofs (under the SZABO definition) translate to two equivalent proofs (under the PRAWITZ definition). Although it is acknowledged that the systems of natural deduction are in many ways much easier to work with, and in most respects more ' natural ' , than the calculus of sequents, we have nevertheless chosen the sequent calculus approach because there are very explicit algorithms for cut-elimination, normalization, representation of arrows by proofs and interpretations of sequent calculus proofs by means of arrows : these are explicit enough for convenient implementation.

LAMBEK and SCOTT [1984] gave an equational presentation for the cartesian closed categories, by defining " cartesian closed category " as a deductive system with an equivalence relation between proofs, in order to represent the arrows of the cartesian closed categories by equivalence classes of these proofs and satisfy a set of equations. They showed how to construct a cartesian closed category freely generated by a graph X using the deductive system technique. They constructed the free cartesian closed category with weak natural numbers object generated by a typed lambda-calculus and showed how to solve the validity problem (i.e to prove the equality between arrows) in the cartesian closed categories. A detailed proof was given for the case of the free cartesian closed category with weak natural numbers object generated by pure typed lambda-calculus (by a pure typed lambda-calculus is meant a typed lambda-calculus with only two basic types 1 and N). The problem of deciding equality between two arrows reduces to deciding the equality of their corresponding terms in the pure typed lambda-calculus. They gave a solution for the latter problem.

CURIEN [1985] has an equational axiomatic system for cartesian closed categories, and showed how to construct a free cartesian closed category CCC(K) generated by a set K of basic types. He proved some equivalence theorems between CCC(K) and the typed lambda - calculus generated by the set K of basic types. Finally he pointed out that this translation to typed lambda- calculus can be used to solve the validity problem for cartesian closed categories using the confluence of the typed lambda-calculus (c.f POTTINGER [1979]) and the noetherian property as shown for the similar case of the free cartesian closed category over the empty graph by LAMBEK and SCOTT .

Most of the notations, definitions and theorems used for category theory in this work are as in MAcLANE [1971]; the others for logic as in GENTZEN [1969]. For lack of the convenient printer characters for Greek letters, we use the following notations, especially in appendices 2, 3, 4, corresponding to the Greek letters used by SZABO :

a	for	α	||	b	for	β
c	for	γ	||	d	for	δ
G	for	Γ	||	D	for	Δ
L	for	Λ	||	Z	for	Ξ
P	for	Π	||	F	for	Φ
E	for	Ψ	||	H	for	Θ
O	for	Ω	||			

The thesis is divided into sex chapters and eight appendices. The first five chapters correspond to five classes of category : cartesian, bicartesian, distributive bicartesian, cartesian closed and bicartesian closed categories respectively. We follow SZABO in discussing each theory while tidying up some trivial printing errors and other obscurities. In the theory of cartesian categories, for example, we extend the unlabelled deductive system of SZABO by a new rule of inference (interchange in the antecedent) which is required for the elimination of cuts during the normalization process. All the extensions we have done, and the changes in the definition of the normal form, in each theory, will be discused in more detail later, in the first five chapters. We end each chapter with examples and illustrations thereof arising in the theory being studied, in order to demonstrate the use of the system. Chapter six contains notes on the implementation..

In appendix 1, we list some apparent errors in the printing of SZABO; appendix 2 contains the parts of SZABO's cut-elimination algorithm which have been implemented. Appendix 3 consists of new algorithms, some of them to eliminate contractions and interchanges, both in the succedent and the antecedent, in the cases of : cartesian, bicartesian, and distributive bicartesian categories; and others to transform proofs into proofs with restricted applications of the thinning rule. These algorithms correspond to the changes in the definition of the normal form, as well as to prove some lemmas and corollaries (stated without proof in SZABO). Appendix 4 collects together those parts of SZABO's normalization algorithm which we have implemented, in order to complete the description of the normalization. Appendix 5 is a user manual for the system; appendix 6 is an index to the predicates used in the program. In appendix 7, we give some information on how to maintain the system and in appendix 8, we list the implementation program itself.

Prolog is a high-level programming language with powerful pattern-matching facilities. Since our system consists of many transformation rules, to be chosen according to the LHS of a rule matching a fragment of a proof, the pattern-matching facilities of Prolog suggested Prolog as a natural candidate for use as the language of the system. Prolog has a convenient interactive user interface, which allowed a speedy development of the earlier phases of the system.

In addition to the substantial space that ' Prolog ' requires to run, our system requires much space. The following table shows the space used by Prolog system itself and the run time since Prolog started, the space used by our implementation and the time taken by Prolog to restore the saved implementation to its top level in order to be ready for use, and the space used and the run time taken in the computation of some simple and complicated examples.

(all examples obtained at one randomly chosen session)

	space used	run time
Prolog	58100 bytes	1.70 sec
the system	398702 bytes	12.23 sec
example 1 in (1.6)	518 bytes	4.57 sec
(3.6)	45112 bytes	250.36 sec
example 2 in (4.6)	182680 bytes	688.04 sec

In chapter 3, we cannot give examples to test the equality of arrows in the theory of distributive bicartesian categories. This is because the representations of the arrows of this theory are too complicated, and need too much space to be computed; the system responds that there is not enough space to complete the computation. The most troublesome part of the whole technique appears to lie in the complexity of the cut-elimination algorithm. In some of the transformation rules (like (C.3),...... in appendix 2), the movement of an instance of 'cut' up to the leaves of the tree introduces two more 'cuts' and these, by turn, may need to introduce more instances of 'cut' until all the instances of 'cut' reach the leaves of the tree (whence they can be removed, finally).

A modified version of our implementation has been already done, as a step towards overcoming the complexity related to the space and the run time of the computation with our system. All the examples related to category theory mentioned herein have been redone in this new version, without any logical differences being apparent. The following results show that the new version indeed reduces the space and the run time used in computation with our system.

(Note these examples are the same mentioned above and the values between brackets are the old ones)

	space used	run time
example 1 in (1.6)	424 (518) bytes	3.70 (4.57) sec
(3.6)	44585 (45112) bytes	149.77 (250.36) sec
example 2 in (4.6)	157196 (182680) bytes	171.03 (688.04) sec

(Note : the program listed in Appendix 8 is the old version)

Since the purpose of the present thesis is not the presentation of a substantially new algorithm but the implementation, modification and application of SZABO's algorithms, some of the material in the following chapters deliberately bears a close resemblance to SZABO's descriptive and theoretical material. It would have been unnecessarily confusing either to introduce an entirely different description of the algorithms or to omit such description altogether.

Chapter 1

CARTESIAN CATEGORY THEORY

This chapter deals with the connection between the class of cartesian categories and the class of derivations in a particular Gentzen deductive system. By a cartesian category we mean a category with a terminal object T and a binary product function † (but not a category with finite limits).

We follow SZABO in discussing this theory, and tidy up some obscurities. These lie essentially in the definition of the unlabelled deductive system relevant to the theory of cartesian categories. The need for the " interchange in the antecedent " rule (R4) in order to perform large contractions will be discussed in more detail later in (1.5.1). We add a lemma (1.5.3); this is required by the definition of normal form in SZABO, but not there stated. The algorithmic proof of the lemma is implemented in our system. As a result of adding rule (R4) to the definition of the class of unlabelled ' cartesian-derivations ', we have restated, and constructed an algorithmic proof of lemma (1.5.4) which was stated without proof in SZABO. This proof is implemented in our system to complete the the definition of the normal form. We have added some clauses to the proof of the cut-elimination theorem to handle the addition of rule (R4).

This chapter is divided into six sections as follows. Section (1.1) is a definition of what is meant by a cartesian category. The construction of the free cartesian category Fc(X) over a category X, is described through subsections of the section (1.2); in (1.2.1) we define the cartesian formulae over a category X, in section (1.2.2) the class LDc(X) of (cartesian) labelled derivations over X is defined, in (1.2.3) we define an equivalence relation on the class LDc(X), in (1.2.4) we define the equivalence classes and finally in (1.2.5) Fc(X) is defined via this equivalence relation. Section (1.3) is divided into two subsections; (1.3.1) contains the definition of the class Dc(X) of (cartesian) unlabelled derivations, in order to represent the arrows of Fc(X) by means of unlabelled derivations with cuts as an alternative to using labelled derivations. (1.3.2) describes how derivations of Dc(X) are represented in our Prolog implementation. We conclude this section with some examples of derivations in Dc(X) and of how to use the system to check the correctness of these derivations.

Section (1.4) is concerned with the semantics of the class Dc(X) and is divided into two subsections : (1.4.1) contains an interpretation function from Dc(X) to ArFc(X) followed by two examples from the system to illustrate this interpretation function; (1.4.2) contains a proof to show that Dc(X) is adequate, in the sense that every arrow in Fc(X) is representable by means of some derivations in Dc(X) , and also some examples from the system to illustrate this representation. Section (1.5) is divided into five subsections; (1.5.1) is a justification for introducing the interchange rule in the definition of Dc(X); in (1.5.2), (1.5.3), (1.5.4) and (1.5.5) we prove some theorems for eliminating cuts, contractions, and interchanges and getting normal forms for derivations of Dc(X) . We end every subsection with some examples in operation with the system to illustrate the processes being studied in each case. We conclude, in section (1.6), with some examples from category theory in operation with the system.

(1.1) Definition

A *cartesian category* is a category $\underline{\underline{C}}$ with the following structure:

(1) A bifunctor $(-)\uparrow(-) : \underline{\underline{C}} \times \underline{\underline{C}} \longrightarrow \underline{\underline{C}}$.

(2) A distinguished object $T \in \mathrm{Ob}\,\underline{\underline{C}}$.

(3) Two adjunctions α_π and α_τ , where

$$\alpha_\pi = \{\alpha_\pi(A,B,C) : \underline{\underline{C}}(A, B\uparrow C) \longrightarrow \underline{\underline{C}}(A,B) \times \underline{\underline{C}}(A,C) \in \mathrm{Ar\ Ens} \mid A,B,C \in \mathrm{Ob}\,\underline{\underline{C}}\},$$

and

$$\alpha_\tau = \{\alpha_\tau(A) : \underline{\underline{C}}(A,T) \longrightarrow \{e\} \in \mathrm{Ar\ Ens} \mid A \in \mathrm{Ob}\,\underline{\underline{C}}\}.$$

(1.2) The Free Cartesian Category Fc(X)

Let X be a fixed but arbitrary small category. The *free cartesian category* Fc(X) is constructed as follows :-

(1.2.1) Definition

The class cL(X) of *(cartesian) formulae* over X is defined by

(i) Every object of X is a formula ;

(ii) T is a formula ;

(iii) If A,B are formulae , then $A \uparrow B$ is ;

(iv) The class cL(X) is generated by (i) to (iii) .

(1.2.2) The Labelled Derivations LDc(X)

The class of (cartesian) labelled derivations over X , and their conclusions, are simultaneously defined by

i) for objects A,B of X and f in X(A,B) ,
LA1(f) is a derivation
with conclusion f : A ----> B ;

ii) for A in cL(X) , not an object of X ,
LA2(A) is a derivation ,
with conclusion 1(A) : A -------> A ;

iii) for A in cL(X) ,
LA3(A) is a derivation ,
with conclusion tau(A) : A -------> T ;

iv) for A,B in cL(X) ,
LA4(A,B) is a derivation ,
with conclusion pi_left(A,B): A ↑ B --> A;

v) for A,B in cL(X) ,
LA5(A,B) is a derivation ,
with conclusion pi_right(A,B).: A ↑ B --> B;

vi) for derivations F , G with conclusions
f : A -> B , g : B -> C respectively ,
COMP(F,G,B) is a derivation ,
with conclusion h : A -> C , where h is the composition of f and g ;

vii) for derivations F , G with conclusions
f : A -> B , g : A -> C respectively ,
ANGLE(F,G,B ↑ C) is a derivation ,
with conclusion <f,g> : A -> B ↑ C .

viii) The class of the derivations LDc(X) is generated by i to vii.

NB

In clauses vi) & vii) above, it is necessary to mention the active formulae in the derivation, for a technical reason in the clauses vi) and vii) of the proof of completeness theorem .

(1.2.3) Definition

The relation == is defined as the smallest equivalence relation on LDc(X) satisfying

i) If F == G and H == K with conclusions f,g : A -> B ,
h,k : B -> C ,
then COMP(F,H,B) == COMP(G,K,B) ;

ii) If F == G and H == K with conclusions f,g : A -> B ,
h,k : A -> C ,
then ANGLE(F,H,B ↑ C) == ANGLE(G,K,B ↑ C) ;

iii) For F with conclusion f : A -> B , both
COMP(LA2(A), F, A) == F and
COMP(F, LA2(B), B) == F ;

iv) For F,G,H with conclusions f : A -> B , g : B -> C ,
h : C -> D respectively ,
COMP(F, COMP(G,H,C), B) == COMP(COMP(F,G,B), H, C) ;

v) For F,G with conclusions f : A -> B, g : A -> C respectively,
COMP(ANGLE(F,G,B ↑ C), LA4(B,C), B ↑ C) == F and
COMP(ANGLE(F,G,B ↑ C), LA5(B,C), B ↑ C) == G ;

vi) For F with conclusion f : A -> B ↑ C ,
ANGLE(COMP(F, LA4(B,C), B ↑ C) ,
COMP(F, LA5(B,C), B ↑ C) , B ↑ C) == F ;

vii) For F with conclusion f : A -> T ,
LA3(A) == F .

(1.2.4) <u>Definition</u>

[F] : denotes the ==- class of F .

(1.2.5) <u>Definition</u>

We define Fc(X) to be the category having :

i) as objects , the cartesian formulae over X ;

ii) as arrows , the ==-classes of LDc(X) ;

iii) as dom , the function defined by
dom([F]) = A where F has conclusion f : A -> B ;

iv) as cod , the function defined by
cod([F]) = B where F has conclusion f : A -> B ;

v) as comp , the function defined by
comp([G],[F]) = [COMP(F, G, B)] ,
where F has conclusion f : A -> B ,
G has conclusion g : B -> C ;

vi) as id , the function defined by
id(A) = [LA1(id(A))] for A in Ob(X) , and
[LA2(A)] for A not in Ob(X) ;

vii) as T , the formula T ;

viii) as tau , the function defined by

tau(A) = [LA3(A)] ;

ix) as pi_left , the function defined by

pi_left(A,B) = [LA4(A,B)] ;

x) as pi_right, the function defined by

pi_right(A,B) = [LA5(A,B)] ;

xi) as <-,-> , the function defined by

< [F],[G] > = [ANGLE(F, G, cod([F]) ↑ cod([G]))]

when dom([F]) = dom([G]) .

It is now routine to check that, with this structure, Fc(X) is a category, is cartesian, and is free cartesian over X. There is the obvious embedding X -> Fc(X) defined by A |--> A; f |--> [LA1(f)].

(1.3) The Unlabelled Derivations Dc(X)

Instead of using labelled derivations LDc(X), an alternative approach is to use unlabelled (cartesian) derivations Dc(X) to represent the arrows of Fc(X) .

(1.3.1) Definition

We define below the class Dc(X) of unlabelled (cartesian) derivations. In the following, we use

a,b,c,d,e ... for formulae over X ;

D,G,L ... for lists thereof ;

G -> D ... for sequents with antecedent G and succedent D ;

p,p1,p2 ... for derivations .

Derivations are really trees of sequents. We represent these inadequately using terms, called unlabelled (cartesian) derivations ; at the same time, we define the relationship between terms and particular sequents called conclusions then, from a derivation and a conclusion the tree of sequents can be recovered. The point is that the structural rules operate on formulae which may occur in various places, and our derivation terms only indicate the formula and not its occurrences. The conclusion however makes the occurrence clear. Thus, a derivation tree has a unique conclusion; our derivation terms do not necessarily have unique conclusions, but satisfy a relationship with sequents defined inductively as follows :

i) if a,b are objects of X , and if f is in X(a,b) , then

A1(f) is a derivation , with conclusion $a \xrightarrow{f} b$;

ii) A2 is a derivation , with conclusion [] -> T ;

iii) (cut)

if p1,p2 are derivations, with conclusions G -> a, DaL -> c then R1(p1,p2,a) is a derivation with conclusion DGL -> c ;

iv) (thinning in the antecedent)

if p is a derivation with conclusion GD -> b , then R2(p,a) is a derivation with conclusion GaD -> b ;

v) (contraction in the antecedent)

if p is a derivation with conclusion GaaD -> b , then R3(p,a) is a derivation with conclusion GaD -> b ;

vi) (interchange in the antecedent)

if p is a derivation with conclusion GabD -> c , then R4(p,b,a) is a derivation with conclusion GbaD -> c ;

vii) (and-introduction in the succedent)

if p1,p2 are derivations with conclusions G -> a, G -> b, then R10(p1,p2, a ↑ b) is a derivation with conclusion G -> (a ↑ b);

viii) (and-introduction in the antecedent)

if p is a derivation with conclusion GabD -> c , then R11(p, a ↑ b) is a derivation with conclusion G(a ↑ b)D -> c ;

ix) The class of the unlabelled derivations Dc(X) is generated by (i) to (viii) .

(1.3.2) Representation of Dc(X) in Prolog

We now give representations of the unlabelled derivations in Prolog, where a derivation p is represented by a step-collection of steps; these steps represent the conclusions of all subderivations of the derivation p. Each step is given a name followed by a list of two elements e.g.

g0 = [cut(g1,g2,a), [x,y] => [z]]

The first is a predicate which is the name of the operation being applied to the conclusion(s) of the subderivation(s), with arguments representing the name(s) given to the conclusion(s) of the subderivation(s) and the name of the active formula(e) . [By active formula is meant the cut, thinning, contraction, interchange, or and-introduction formula] . The second is composed of two lists, for the antecedent and succedent of the sequent, separated by the symbol =>.

We use the following abbreviations :-

&	for ↑
tr	for T in A2 (Prolog uses upper-case for variables)

true   for the name of the constant in A2
cut   for the name of the operation in R1(p1,p2,a)
th   for the name of the operation in R2(p,a)
con   for the name of the operation in R3(p,a)
inc   for the name of the operation in R4(p,b,a)
ais   for the name of the operation in R10(p1,p2,a ↑ b)
aia   for the name of the operation in R11(p,a ↑ b) .

We use the meta-variables g1, g2, g3 ... for the names of the steps.

i) A1(f) is represented by

g1 = [f, [a] => [b]]

ii) A2 is represented by

g1 = [true, [] => [tr]]

iii) R1(p1,p2,a) is represented by a step-collection with the last step,

g3 = [cut(g1,g2,a), DGL => [c]]

where g1, g2 are names of the last steps of the step-collections for the derivations p1, p2 respectively .

iv) R2(p,a) is represented by a step-collection with the last step,

g2 = [th(g1,a), GaD => [b]]

where g1 is the name of the last step in the step-collection for the derivation p .

v) R3(p,a) is represented by a step-collection with the last step,

g2 = [con(g1,a), GaD => [b]]

where g1 is the name of the last step in the step-collection for the derivation p .

vi) R4(p,b,a) is represented by a step-collection with the last step,

g2 = [inc(g1,b,a), GbaD => [c]]

where g1 is the name of the last step in the step-collection for the derivation p .

vii) R10(p1,p2,a ↑ b) is represented by a step-collection with the last step ,

g3 = [ais(g1,g2,a & b), G => [a & b]]

where g1,g2 are names of the last steps of the step-collections for the derivations p1, p2 respectively .

viii) R11(p,a ↑ b) is represented by a step-collection with the last step,

g2 = [aia(g1,a & b), Ga&bD => [c]]

where g1 is the name of the last step of the step-collection for the derivation p .

We give some examples of unlabelled derivations, their representations in the implementation language and how to check the correctness of a proof using our implementation (proof checker part) .

Note : we write the derivations in tree form .

Example 1 The derivation

```
          f               g                      h
        a -> b         a -> c               e -> d
        ------------------------          -------------
           a -> b ↑ c                     b ↑ c, e -> d
           ------------------------------------------------
                          a,e -> d
```

is represented by

l=[f,[a]=>[b]]

m=[g,[a]=>[c]]

n=[ais(l,m,b&c),[a]=>[b&c]]

o=[h,[e]=>[d]]

p=[th(o,b&c),[b&c,e]=>[d]]

q=[cut(n,p,b&c),[a,e]=>[d]]

The following is input and output session with our system to check the correctness of the above proof. The command "theory(cart)" is used to inform the system that the following proof is constructed according to the definition of Dc(X) in the theory of cartesian categories. The command " prove(Z) ", where Z represents a proof step, is required to prove the correctness of each step in the proof.

```
TH>szabo
 Cut Elimination and Normal Form Program under Prolog
 For help use the help command

yes
| ?- theory(cart).
yes
| ?- prove(l = [f, [a] => [b]]).
yes
| ?- prove(m = [g, [a] => [c]]).
yes
| ?- prove(n = [ais(l,m,b & c), [a] => [b & c]]).
yes
| ?- prove(o = [h, [e] => [d]]).
yes
| ?- prove(p = [th(o,b & c), [b & c,e] => [d]]).
yes
| ?- prove(q = [cut(n,p,b & c), [a,e] => [d]]).
yes
| ?-
```

Example 2 The derivation

```
                                                  h
                                               a -> e
                          g                   --------
       a -> a           b -> c                a,c -> e
      --------         --------              ----------
      a,b -> a         a,b -> c              a ↑ c -> e
      --------------------------          --------------
          a,b -> a ↑ c                    d, a ↑ c -> e
          ---------------------------------------------
                         d,a,b -> e
                        -----------
                         a,d,b -> e
```

is represented by

```
l=[id(a),[a]=>[a]]
m = [th(l,b),[a,b]=>[a]]
n = [g,[b]=>[c]]
o = [th(n,a),[a,b]=>[c]]
p = [als(m,o,a&c),[a,b]=>[a&c]]
q = [h,[c]=>[e]]
r = [th(q,a),[a,c]=>[e]]
s = [ala(r,a&c),[a&c]=>[e]]
t = [th(s,d),[d,a&c]=>[e]]
u = [cut(p,t,a&c),[d,a,b]=>[e]]
v = [inc(u,a,d),[a,d,b]=>[e]]
```

```
yes
| ?- theory(cart).
yes
| ?- prove(l = [id(a), [a] => [a]]).
yes
| ?- prove(m = [th(l,b), [a,b] => [a]]).
yes
| ?- prove(n = [g, [b] => [c]]).
yes
| ?- prove(o = [th(n,a), [a,b] => [c]]).
yes
| ?- prove(p = [als(m,o,a & c), [a,b] => [a & c]]).
yes
| ?- prove(q = [h, [c] => [e]]).
yes
| ?- prove(r = [th(q,a), [a,c] => [e]]).
yes
| ?- prove(s = [ala(r,a & c), [a & c] => [e]]).
yes
| ?- prove(t = [th(s,d), [d,a & c] => [e]]).
yes
| ?- prove(u = [cut(p,t,a & c), [d,a,b] => [e]]).
yes
| ?- prove(v = [inc(u,a,d), [a,d,b] => [e]]).
yes
| ?-
```

(1.4) The Semantics of Dc(X)

In this section, we interpret the derivations of the class Dc(X) as arrows of Fc(X) and prove the completeness theorem. In order to perform this interpretation, the following canonical arrows of Fc(X) are required (By a canonical arrow in Fc(X) is meant, any arrow which can be represented canonically by the adjunctions α_π and α_π . This arrow does not depend on the arrows of the underlying category X, but on its objects.) :

i) delta(A) : A -> A ↑ A for all A in ObFc(X), where

$$\alpha_\pi^{-1}(\langle id(A), id(A)\rangle) = delta(A) .$$

ii) alpha(A,B,C) : A ↑ (B ↑ C) -> (A ↑ B) ↑ C for all A,B,C in ObFc(X), where

((pi_left(A,B ↑ C),comp(pi_left(B,C),pi_right(A,B ↑ C))),
comp(pi_right(B,C), pi_right(A,B ↑ C))) = alpha(A,B,C) .

iii) sigma(A,B) : A ↑ B -> B ↑ A for all A,B in Ob Fc(X), where
< pi_right(A,B), pi_left(A,B) > = sigma(A,B) .

(1.4.1) Definition

We shall shortly define a function S from unlabelled derivations Dc(X) to ArFc(X) .

First, note that if p has conclusion G -> a, then S(p) is to be an arrow with codomain a and domain the product Pi(G) defined by

```
Pi(G) = if G = nil
            then T
            else if  G = [a]  then  a
            else ( Pi(fst(G)) ↑ (lst(G)))
```

where

fst(G) is the sublist of G consisting of all except the last element,
lst(G) is the last element of the list G .

Second, if D and G are lists of formulae, then Pi(DG) is canonically identified to Pi(D) ↑ Pi(G), using induction on the length of G :

i) G = nil

Pi(D nil) = Pi(D) =~ Pi(D) ↑ T = Pi(D) ↑ Pi(nil)

ii) G = [a]

Let D' be the list resulting from appending [a] to the list D

Pi(D[a]) = Pi(D') = Pi(D) ↑ a = Pi(D) ↑ Pi([a])

where the proof of the middle step is from definition of Pi.

iii) Suppose that :

Pi(DG) =~ Pi(D) ↑ Pi(G)

and let G' be the list resulting from prepending the list G to [b]

then

```
Pi(DG') = Pi(DG) ↑ b
        =~ (Pi(D) ↑ Pi(G)) ↑ Pi([b])
        =~ Pi(D) ↑ (Pi(G) ↑ Pi([b]))  [using alpha_star]
        =~ Pi(D) ↑ Pi(G[b])
        =  Pi(D) ↑ Pi(G')
```

and this complets the proof by induction on the length of G.

Now, S is defined as follows :

1) S(A1(f)) = [LA1(f)] for f in X(A,B) ;

2) S(A2) = [LA2(T)] ;

3) S(R1(p1,p2,a)) = comp(S(p2), ((I1 ↑ S(p1)) ↑ I2)) , where I1 is the identity on Pi(D) , I2 is the identity on Pi(L), and we make the canonical identifications of
Pi(DGL) with (Pi(D) ↑ Pi(G)) ↑ Pi(L) and of
Pi(DaL) with (Pi(D) ↑ a) ↑ Pi(L) ,
D,G,a,L being as in the definition of R1(p1,p2,a) ;

4) S(R2(p,a)) = comp(S(p), (pi_left(Pi(G), a) ↑ I)), where I is the identity on Pi(D) , and we make the canonical identifications of
Pi(GaD) with (Pi(G) ↑ a) ↑ Pi(D) and of
Pi(GD) with Pi(G) ↑ Pi(D) ,
G,a,D being as in the definition of R2(p,a) ;

5) S(R3(p,a)) = comp(comp(S(p), (alpha(Pi(G),a,a) ↑ I2)), ((I1 ↑ delta(a)) ↑ I2)) , where I1 is the identity on Pi(G) , I2 is the identity on Pi(D) and we make the canonical identifications of
Pi(Ga ↑ aD) with (Pi(G) ↑ (a ↑ a)) ↑ Pi(D),
Pi(GaD) with (Pi(G) ↑ a) ↑ Pi(D) and of
Pi(GaaD) with ((Pi(G) ↑ a) ↑ a) ↑ Pi(D),
G,a,D being as in the definition of R3(p,a) ;

6) S(R4(p,b,a)) = comp(comp(comp(S(p), (alpha(Pi(G),a,b) ↑ I2)), ((I1 ↑ sigma_inverse(a,b)) ↑ I2)), (alpha_inverse(Pi(G),b,a) ↑ I2)) , where I1 is the identity on Pi(G) , I2 is the identity on Pi(D), sigma_inverse is the inverse arrow for sigma, alpha_inverse is the inverse arrow for alpha and we make the canonical identifications of
Pi(GbaD) with ((Pi(G) ↑ b) ↑ a) ↑ Pi(D),
Pi(Gb ↑ aD) with (Pi(G) ↑ (b ↑ a)) ↑ Pi(D),
Pi(Ga ↑ bD) with (Pi(G)↑(a ↑ b)) ↑ Pi(D) and of
Pi(GabD) with ((Pi(G) ↑ a) ↑ b) ↑ Pi(D) ,
G,b,a,D being as in the definition of R4(p,b,a) ;

7) S(R10(p1,p2,a ↑ b))= comp((S(p1) ↑ S(p2)) , delta(Pi(G))), where we make the canonical identifications of
Pi(GG) with Pi(G) ↑ Pi(G) ,
G being as in the definition of R10(p1,p2,a ↑ b) ;

8) S(R11(p,a ↑ b)) = comp(S(p), (alpha(Pi(G),a,b) ↑ I)),where
I is the identity on Pi(D) and we make the canonical
identifications of
Pi(GabD) with ((Pi(G) ↑ a) ↑ b) ↑ Pi(D) and of
Pi(Ga ↑ bD) with (Pi(G) ↑ (a ↑ b)) ↑ Pi(D)
G,a,b,D being as in the definition of R11(p,a ↑ b) .

We give some examples for derivations of Dc(X) and their interpretations in ArFc(X) using the implementation .

<u>Example 1</u>

Let a,b,c,d be objects of the category X, f in X(a,b), g in X(c,d); then the derivation

```
               f                    g
             a -> b               c -> d
            --------             --------
            c,a -> b             b,c -> d
            -------------------------------
                    c,a,c -> d
                    ----------
                    c,c,a -> d
                    ----------
                      c,a -> d
                    ----------
                    c ↑ a -> d
```

is interpreted in ArFc(X) as follows :

We insert the above proof into the system as described in (1.3.2) Example 1, then using the command " interpr(Z,H) ", where Z represents the name of the last step of the proof and H will be instantiated as the arrow computed to interpret the derivation .

```
yes
| ?- theory(cart).
yes
| ?- prove(l = [f, [a] => [b]]).
yes
| ?- prove(l1 = [th(l,c), [c,a] => [b]]).
yes
| ?- prove(m = [g, [c] => [d]]).
yes
| ?- prove(m1 = [th(m,b), [b,c] => [d]]).
yes
| ?- prove(n = [cut(l1,m1,b), [c,a,c] => [d]]).
yes
| ?- prove(o = [inc(n,a,c), [a,c,c] => [d]]).
yes
| ?- prove(p = [con(o,c), [a,c] => [d]]).
yes
| ?- prove(q = [ala(p,a & c), [a & c] => [d]]).
yes
| ?- interpr(q,H).

H = comp(comp(comp(comp(comp(g,pi_right(b,c)),
                              product(comp(f,pi_right(c,a)),
                                      id(c)) ),
                         product(sigma_inverse(c,a),id(c))),
                    alpha(a,c,c)),
               product(id(a),delta(c)))

yes
| ?-
```

Example 2

Suppose a,b are objects of the category X. Then the derivation

```
      b -> b                 a -> a
    ----------            ----------
     a,b -> b              a,b -> a
   ------------          ------------
   a ↑ b -> b            a ↑ b -> a
   ------------------------------------
              a ↑ b -> b ↑ a
```

is interpreted as follows :

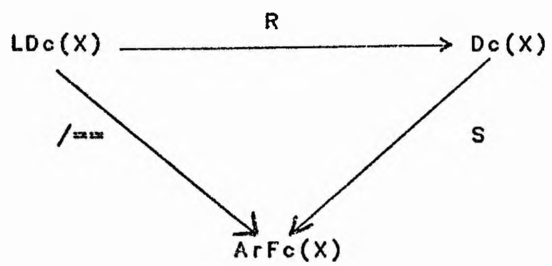
```
yes
| ?- theory(cart).
yes
| ?- prove(l = [id(b), [b] => [b]]).
yes
| ?- prove(l1 = [th(l,a), [a,b] => [b]]).
yes
| ?- prove(l2 = [ala(l1,a & b), [a & b] => [b]]).
yes
| ?- prove(m = [id(a), [a] => [a]]).
yes
| ?- prove(m1 = [th(m,b), [a,b] => [a]]).
yes
| ?- prove(m2 = [ala(m1,a & b), [a & b] => [a]]).
yes
| ?- prove(n = [ais(l2,m2,b & a), [a & b] => [b & a]]).
yes
| ?- interpr(n,H).

H = comp(product(pi_right(a,b),pi_left(a,b)),delta(a&b))
```

The following theorem asserts that the class Dc(X) of unlabelled derivations as defined in (1.3.1) is adequate in the sense that every arrow in the free cartesian category Fc(X) is representable by means of some derivation in Dc(X) :

(1.4.2) <u>The Completeness Theorem for Dc(X)</u>

There is a function

R : LDc(X) ------------> Dc(X) , so that



commutes.

<u>Proof</u> by induction on structure of F in LDc(X), define R(F) by : note that I(A) below is defined by

I(A) = <u>if</u> A is in Ob(X)
<u>then</u> LA1(id(A))
<u>else</u> LA2(A)

i) when F = LA1(f) for f in X(A,B), A,B in Ob(X) ,
define R(F) = A1(f)
then S(R(F)) = S(A1(f)) = [LA1(f)] = [F]

ii) when F = LA2(A) for A in cL(X), not an object of X, we use induction on structure of A :
A is <u>either</u> T : put R(F) = A2
then S(R(F)) = S(A2) = [LA2(T)] = [F]
<u>or</u> is B ↑ C : put
R(F) = R11(R10(R2(R(I(B)), C),
R2(R(I(C)), B), B ↑ C), B ↑ C)

iii) when F is LA3(A) for A in cL(X) , put
R(F) = R2(A2, A)

iv) when F is LA4(A,B) for A,B in cL(X) , put
R(F) = R11(R2(R(LA2(A)), B), A ↑ B)

v) when F is LA5(A,B) for A,B in cL(X) , put
R(F) = R11(R2(R(LA2(B)), A), A ↑ B)

vi) when F is COMP(G,H,A) , put
R(F) = R1(R(G), R(H), A)

vii) when F is ANGLE(G,H,A ↑ B) , put
R(F) = R10(R(G), R(H), A ↑ B)

We now illustrate the algorithm in this proof by showing how some canonical arrows of Fc(X) are represented as unlabelled derivations. these have been implemented in the system .

<u>Example 1</u> For the derivation F = ANGLE(I(A), I(A), A ↑ A)
representing the arrow
delta(A) : A -> A ↑ A
the representation R(F) is
R10(R(LA2(A)), R(LA2(A)),A ↑ A)
which can only be further simplified if we know the structure of A.

<u>Example 2</u> For a derivation
F = ANGLE(ANGLE(LA4(A,B ↑ C),
COMP(LA5(A,B ↑ C), LA4(B, C), B ↑ C) ,
A ↑ B) ,
COMP(LA5(A,B ↑ C), LA5(B,C), B ↑ C),
(A ↑ B) ↑ C))
representing the arrow

```
alpha(A,B,C) : A ↑ (B ↑ C)  ->  (A ↑ B) ↑ C
```

the representation R(F) is

```
R10(R10(R11(R2(R( LA2(A) ), B ↑ C ), A ↑ (B ↑ C)),
        R1(R11(R2(R( LA2(B ↑ C) ), A), A ↑ (B ↑ C)),
           R11(R2(R( LA2(B) ), C), B ↑ C),
           B ↑ C ),
        A ↑ B ),
    R1(R11(R2(R( LA2(B ↑ C) ), A ), A ↑ (B ↑ C) ),
       R11(R2(R( LA2(C) ), B), B ↑ C),
       B ↑ C ),
    (A ↑ B) ↑ C ).
```

Example 3

For derivations F,G representing the arrows

```
f : A -> B , g : C -> D  respectively
```

then the derivation

```
PRODUCT( F,G ) = ANGLE( COMP( LA4(A,C), F, A ),
                        COMP( LA5(A,C), G, C ),
                        B ↑ D )
```

representing the arrow

```
f ↑ g : A ↑ C  ->  B ↑ D
```

the representation R(PRODUCT(F,G)) is

```
R10(R1(R11(R2(R( LA2(A) ), C), A & C), R( F ), A),
    R1(R11(R2(R( LA2(C) ), A), A & C), R( G ), C),
    B ↑ D)
```

Example 4

For the derivation

```
F = ANGLE( LA5(A,B), LA4(A,B), B ↑ A )
```

representing the arrow

```
sigma(A,B) :  A ↑ B  ->  B ↑ A
```

the representation R(F) is

```
R10( R11( R2( R( LA2(B) ), A), A ↑ B),
     R11( R2( R( LA2(A) ), B), A ↑ B),
     B ↑ A)
```

For the derivation

```
F = ANGLE( LA5(B,A), LA4(B,A), A ↑ B )
```

representing the arrow

```
sigma_inverse(A,B) : B ↑ A  ->  A ↑ B
```

the representation R(F) is

```
R10( R11( R2( R( LA2(A) ), B), B ↑ A),
     R11( R2( R( LA2(B) ), A), B ↑ A),
     A ↑ B)
```

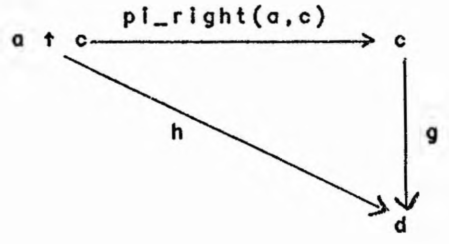
The following examples are sessions of input and output with the implementation to explain how to use the system to produce unlabelled derivations as representations for arrows of Fc(X) .
First, the command " arrow(a,f,b) ", where f in X(a,b) is used in the construction of the arrow in question, will store this arrow for later use. After inserting all arrows of the category X used in the construction of the arrow in question, we use the command "rep_of(Y)" where Y represents the arrow to be represented. Then it will respond with the representation.

<u>Example 5</u>

For g in X(c,d), the arrow

h = comp(g, pi_right(a,c)) : a ↑ c -> d

in the diagram



is represented by an unlabelled derivation as follows :

```
yes
| ?- arrow(c,g,d).
yes
| ?- rep_of(comp(g, pi_right(a,c))).

The arrow is :
==============

comp(g,pi_right(a,c))

representation :
================

f5=[id(c),[c]=>[c]]
f4=[th(f5,a),[a,c]=>[c]]
f3=[aia(f4,a&c),[a&c]=>[c]]
f2=[g,[c]=>[d]]
f1=[cut(f3,f2,c),[a&c]=>[d]]

yes
| ?-
```

which in tree form is :

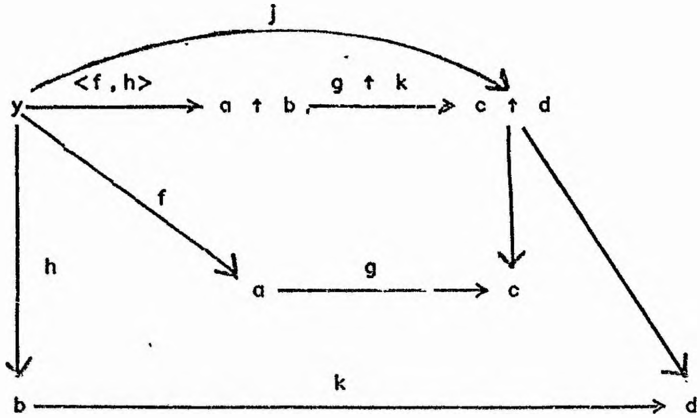
```
                c -> c
              ----------
              a,c -> c
            ------------           g
            a ↑ c -> c           c -> d
            ---------------------------
                  a ↑ c -> d
```

<u>Example 6</u>

For arrows f in X(y,a), g in X(a,c), h in X(y,b) and k in X(b,d), where X is a category. Then the arrow

j = comp(g ↑ k,<f,h>) : y -> c ↑ d ,

in the diagram



is represented by an unlabelled derivation as follows :

```
| ?- theory(cart).
yes
| ?- arrow(y,f,a).
yes
| ?- arrow(a,g,c).
yes
| ?- arrow(y,h,b).
yes
| ?- arrow(b,k,d).
yes
| ?- rep_of(comp(product(g,k), angle(f,h))).

The arrow is :
==============
comp(product(g,k),angle(f,h))

representation :
================
f14=[f,[y]=>[a]]
f15=[h,[y]=>[b]]
f13=[ais(f14,f15,a&b),[y]=>[a&b]]
f7=[id(a),[a]=>[a]]
f6=[th(f7,b),[a,b]=>[a]]
f5=[ala(f6,a&b),[a&b]=>[a]]
f3=[g,[a]=>[c]]
f8=[cut(f5,f3,a),[a&b]=>[c]]
f11=[id(b),[b]=>[b]]
f10=[th(f11,a),[a,b]=>[b]]
f9=[ala(f10,a&b),[a&b]=>[b]]
f4=[k,[b]=>[d]]
f12=[cut(f9,f4,b),[a&b]=>[d]]
f2=[ais(f8,f12,c&d),[a&b]=>[c&d]]
f1=[cut(f13,f2,a&b),[y]=>[c&d]]
```

which in tree form is :

```
                                  a -> a                      b -> b
                                 ---------                   ---------
                                 a,b -> a                    a,b -> b
                                ----------      g           ----------      k
                                a ↑ b -> a    a -> c        a ↑ b -> b    b -> d
      f         h               ----------------------      ----------------------
   y -> a    y -> b                  a ↑ b -> c                  a ↑ b -> d
   -----------------                 --------------------------------------
      y -> a ↑ b                                a ↑ b -> c ↑ d
      ------------------------------------------------------
                          y -> c ↑ d
```

(1.5) The Syntax of Fc(X)

The most important part of the theory of cartesian categories, for our implementation, is the syntax of Fc(X). This is because the arrows in the free cartesian category can be represented by proofs with cuts; then these representations can be manipulated to eliminate cuts and obtain their normal representations.

(1.5.1) Interchange rule and its justification :

We observe in SZABO's algorithm for cut-elimination that in some clauses like [(C.3),(C.14),...], there is a step which has a contraction for say, G . However G may be of length greater than one and in this case, we must perform interchange between the objects of the sequence GG, before carrying out contraction to get the sequence G . This explains the inclusion of the inference rule ' interchange in the antecedent ' in the unlabelled (cartesian) derivations class, and this additional rule leads to changes in the definition of the normalization .
The following example is to justify the need for the rule of interchange in the antecedent :-

alpha(A,B,C) : A ↑ (B ↑ C) ----------------> (A ↑ B) ↑ C

and this arrow is canonical in the theory of cartesian categories .
By the completeness theorem we get its equivalent unlabelled proof in the tree representation as :

```
                    b -> b     c -> c
                  -------    --------
                  b,c -> b   b,c -> c              b -> b     c -> c
                  -------------------             -------    -------
                     b,c -> b ↑ c                 b,c -> b   b,c -> c
                  -----------------               -------------------
                   b ↑ c -> b ↑ c      b -> b        b,c -> b ↑ c
        a -> a    ----------------   ---------    -----------------    c -> c
       --------   a,b ↑ c -> b ↑ c   b,c -> b     b ↑ c -> b ↑ c      --------
       a,b↑c -> a |a↑(b↑c) -> b↑c    b↑c -> b|   a,b ↑ c -> b ↑ c    b,c -> c
      ----------- |--------------------------|   ----------------    --------
      a↑(b↑c) -> a|      a↑(b↑c) -> b        |   a↑(b↑c) -> b↑c      b↑c -> c
      ---------------------------------------    -----------------------------
              a ↑ (b ↑ c) -> (a ↑ b)                  a ↑ (b ↑ c) -> c
              --------------------------------------------------------
                          a ↑ (b ↑ c) -> (a ↑ b) ↑ c
```

Applying clause (C.42) from the cut-elimination algorithm described in Appendix (3) to the cut step in the above tree marked in a box, we get

```
                    b -> b     c -> c
                  -------    --------
                  b,c -> b   b,c -> c                  b -> b     c -> c
                  -------------------                 -------    -------
                     b,c -> b ↑ c       b -> b        b,c -> b   b,c -> c
                  -----------------   ---------       -------------------
                   b ↑ c -> b ↑ c      b,c -> b          b,c -> b ↑ c
        a -> a    ----------------   ---------        -----------------     c -> c
       --------  |a,b ↑ c -> b ↑ c    b↑c -> b|       b ↑ c -> b ↑ c       --------
       a,b↑c -> a|-----------------------------|    -------------------   b,c -> c
      -----------|        a,( b↑ c) -> b        |    a,b ↑ c -> b↑c       --------
      a↑(b↑c) -> a|-----------------------------|   ---------------------  b↑c -> c
      -----------      a ↑ (b ↑ c) -> b            a ↑ (b ↑ c) -> b↑c
      ------------------------------------------   -------------------------------
              a ↑ (b ↑ c) -> (a ↑ b)                    a ↑ (b ↑ c) -> c
              ---------------------------------------------------------
                          a ↑ (b ↑ c) -> (a ↑ b) ↑ c
```

Applying clause (C.34) from the cut_elimination algorithm to the cut step marked in a box above, we get

```
                   b -> b     c -> c
                  --------   --------
                  b,c -> b   b,c -> c    b -> b     b -> b      c -> c
                  -------------------   --------   --------   --------
                     b,c -> b ↑ c       b,c -> b   b,c -> b   b,c -> c
                   ---------------------------     -------------------
                  | b ↑ c -> b ↑ c    b↑c -> b |      b,c -> b ↑ c
                  |----------------------------|   -------------------
        a -> a    |        b ↑ c -> b          |   b ↑ c -> b ↑ c        c -> c
      ----------  |----------------------------|   ---------------    --------
      a,b↑c -> a         a,( b↑ c) -> b            a,b ↑ c -> b↑c     b,c -> c
    ------------     ------------------------    ------------------   --------
    a↑(b↑c) -> a       a ↑ (b ↑ c) -> b         a ↑ (b ↑ c) -> b↑c    b↑c -> c
    ---------------------------------------     ----------------------------
         a ↑ (b ↑ c) -> (a ↑ b)                        a ↑ (b ↑ c) -> c
         -------------------------------------------------------------
                          a ↑ (b ↑ c) -> (a ↑ b) ↑ c
```

Applying clause (C.42) we get

```
                   b -> b     c -> c     b -> b
                  --------   --------   --------
                  b,c -> b   b,c -> c   b,c -> b    b -> b      c -> c
                  -------------------   --------   --------   --------
                 |   b,c -> b ↑ c       b↑c -> b | b,c -> b   b,c -> c
                 |-------------------------------| -------------------
                 |          b,c -> b             |   b,c -> b ↑ c
                 | ----------------------------- | -------------------
        a -> a              b ↑ c -> b             b ↑ c -> b ↑ c        c -> c
      ----------          --------------           ---------------    --------
      a,b↑c -> a         a,( b↑ c) -> b            a,b ↑ c -> b↑c     b,c -> c
    ------------        ------------------       ------------------   --------
    a↑(b↑c) -> a        a ↑ (b ↑ c) -> b        a ↑ (b ↑ c) -> b↑c    b↑c -> c
    --------------------------------------      ----------------------------
         a ↑ (b ↑ c) -> (a ↑ b)                        a ↑ (b ↑ c) -> c
         -------------------------------------------------------------
                          a ↑ (b ↑ c) -> (a ↑ b) ↑ c
```

Applying clause (C.14) to the cut step marked in a box in the above tree we get the following equivalent tree . This uses the rule ' interchange in the antecedent ' for the objects b,c at the step indicated below.

```
                         c -> c     b -> b
                        --------   --------
               b -> b   b,c -> c   b,c -> b
              --------  -------------------
              b,c -> b       b,b,c -> b
              ----------------------------
                  || b,c,b,c -> b ||
                  ||--------------||
                     b,b,c,c -> b                 b -> b      c -> c
                    --------------              --------    --------
                       b,c,c -> b               b,c -> b    b,c -> c
                       ----------               --------------------
                        b,c -> b                    b,c -> b ↑ c
                       ----------                 ----------------
        a -> a         b ↑ c -> b                b ↑ c -> b ↑ c        c -> c
      ----------     --------------            ------------------   --------
      a,b↑c -> a     a,( b↑ c) -> b             a,b ↑ c -> b ↑ c     b,c -> c
    ------------    ------------------        --------------------  --------
    a↑(b↑c) -> a    a ↑ (b ↑ c) -> b          a ↑ (b ↑ c) -> b ↑ c   b↑c -> c
    ----------------------------------       -----------------------------
        a ↑ (b ↑ c) -> (a ↑ b)                       a ↑ (b ↑ c) -> c
        ---------------------------------------------------------------
                         a ↑ (b ↑ c) -> (a ↑ b) ↑ c
```

We continue the processes of the cut-elimination algorithm until, we get the following cut-free equivalent proof :

```
                                 b -> b                   c -> c
                               ----------               ----------
                               b,c -> b                 b,c -> c
                             ------------             ------------
                             b,c,c -> b               b,c,c -> c
                           --------------           --------------
                           b,c,b,c -> b             b,c,b,c -> c
                           --------------           --------------
                           b,b,c,c -> b             b,b,c,c -> c
                           --------------           --------------
                             b,c,c -> b               b,c,c -> c
                           --------------           --------------
                               b,c -> b                 b,c -> c
                           --------------           --------------
            a -> a           b ↑ c -> b               b ↑ c -> c
      ----------------     --------------           --------------
       a,b ↑  c -> a       a,b ↑ c -> b             a,b ↑ c -> c
   -------------------  ------------------     -------------------
   a ↑ (b ↑ c) -> a     a ↑ (b ↑ c) -> b       a ↑ (b ↑ c) -> c
   ---------------------------------------
          a ↑ (b ↑ c) -> a ↑ b
          ---------------------------------------------------------
                        a ↑ (b ↑ c)  -> (a ↑ b) ↑ c
```

(1.5.2) <u>The Cut-Elimination Theorem for Dc(X)</u>

Every p in Dc(X) is equivalent to a cut-free q in Dc(X) .

<u>proof</u>

By the clauses (C.1), (C.2.1) , (C.2.3) , (C.3) , (C.14) , (C.18), (C.19), (C.26), (C.27), (C.34), (C.35) and (C.42) as in SZABO [1978] , in addition to the clauses (C.20) and (C.36) needed because of the additional rule (R4) in the definition of Dc(X) , of the algorithm described in appendix (2), every derivation of Dc(X) containing a cut reduces to a cut-free one .

###

The following example is to illustrate the cut-elimination theorem and to explain how to use the system to eliminate cuts from derivations of Dc(X). As in (1.3.2) Example 1 we insert the proof, using the command " composite(-,-,-) " to insert all possible compositions for the arrows of the category X used in the construction of the derivation, where the third argument is the composition arrow of the arrows represented by the first and the second arguments of the command. Finally the command " cut_free(Z) ", where Z represents the name of the last step in the inserted proof, will produce the equivalent cut-free one .

<u>Example</u>

Suppose a,b,c,e are objects of the category X; and that f in X(b,c), h in X(b,e) with comp(h,g) = k. Then the derivation

```
                                                        h
                                                      c -> e
                        g                           ---------
    a -> a            b -> c                        a,c -> e
  ---------          ---------                     ----------
  a,b -> a           a,b -> c                      a t c -> e
  ---------------------------                   -------------
         a,b -> a t c                            d,a t c -> e
        -----------------------------------------------------
                       d,a,b -> e
```

is transformed to a cut-free one as follows :

```
yes
| ?- theory(cart).
yes
| ?- composite(h,g,k).
yes
| ?- prove(l = [id(a), [a] => [a]]).
yes
| ?- prove(m = [th(l,b), [a,b] => [a]]).
yes
| ?- prove(n = [g, [b] => [c]]).
yes
| ?- prove(o = [th(n,a), [a,b] => [c]]).
yes
| ?- prove(p = [ais(m,o,a & c), [a,b] => [a & c]]).
yes
| ?- prove(q = [h, [c] => [e]]).
yes
| ?- prove(r = [th(q,a), [a,c] => [e]]).
yes
| ?- prove(s = [ala(r,a & c), [a & c] => [e]]).
yes
| ?- prove(t = [th(s,d), [d,a & c] => [e]]).
yes
| ?- prove(u = [cut(p,t,a & c), [d,a,b] => [e]]).
yes
| ?- cut_free(u).

Proof is as follows :
=====================

l=[id(a),[a]=>[a]]
m=[th(l,b),[a,b]=>[a]]
n=[g,[b]=>[c]]
o=[th(n,a),[a,b]=>[c]]
p=[ais(m,o,a&c),[a,b]=>[a&c]]
q=[h,[c]=>[e]]
r=[th(q,a),[a,c]=>[e]]
s=[ala(r,a&c),[a&c]=>[e]]
t=[th(s,d),[d,a&c]=>[e]]
u=[cut(p,t,a&c),[d,a,b]=>[e]]

cut_free Proof is :
===================

f26=[k,[b]=>[e]]
f25=[th(f26,a),[a,b]=>[e]]
f22=[th(f25,b),[a,b,b]=>[e]]
f8=[th(f22,a),[a,b,a,b]=>[e]]
f11=[inc(f8,a,b),[a,a,b,b]=>[e]]
f12=[con(f11,a),[a,b,b]=>[e]]
f3=[con(f12,b),[a,b]=>[e]]
u=[th(f3,d),[d,a,b]=>[e]]

yes
| ?-
```

which in tree form is :

```
              k
          b -> e
        ----------
        a,b -> e
      ------------
      a,b,b -> e
    --------------
    a,b,a,b -> e
    --------------
    a,a,b,b -> e
    --------------
      a,b,b -> e
      ------------
        a,b -> e
      ------------
      d,a,b -> e
```

(1.5.3) Lemma

If p derives G -> a and T is the only atomic subformula of a, then p can be reduced to the form

```
      g
      -> a
    --------   (K)
    G  -> a
```

where (K) denotes the instances of (R2) required to derive G -> a from -> a.

Proof :

The proof follows by induction on the construction of a :

i) a = T

```
     p                 -> T
   G -> T    >      --------
                     G  -> T
```

ii) a = c ↑ d

```
                          p1          p2
                          --> c       --> d
                       ---------------------
     p                        -> c ↑ d
   G -> c ↑ d    >          --------------
                          G   -> c ↑ d
```

###

(1.5.4) Lemma

For every cut-free derivation p in Dc(X) there exists an equivalent cut-free q in Dc(X) containing no subderivations constructed with (R3) or (R4) and where the only subderivations constructed with (R2), if any, have atomic active formulae .

Proof

By the clauses (E.1) , (E.2) , (E.3) , (E.6) , (E.10) , (E.18) and (E.37) and the special cases of the clauses (E.9) and (E.17) in which F and E are empty sequences, of the algorithm described in appendix(3), every derivation of Dc(X) containing subderivations constructed with

(R3) or (R4) reduces to a contraction and interchange - free one . And by the clause (E.41.1) of the same appendix, every derivation of Dc(X) containing subderivations constructed with R2 with active formulae of the form a ↑ b reduces to a derivation where the active formulae of subderivations constructed with (R2) are atomic . These clauses preserve the equivalence relation between proofs as proved in SZABO.

##

The following examples are to illustrate the algorithm in appendix3 for the above lemma. Insert a derivation of Dc(X) to the system in the same way as in (1.3.2) example 1, then use the command "con_inc_free(Z)", where Z represents the name of the last step of the inserted proof. The system responds with an equivalent contraction and interchange-free one where in each instance of the 'thinning in the antecedent' rule, if any, the active formula is atomic .

Example 1

The derivation (from Example of 1.5.2)

```
                 k
        b ------------> e
     --------------------
      a,b ------------> e
    ----------------------
    a,b,b ------------> e
  ------------------------
  a,b,a,b ------------> e
  ------------------------
  a,a,b,b ------------> e
  ------------------------
    a,b,b ------------> e
    ----------------------
      a,b ------------> e
    ----------------------
    d,a,b ------------> e
```

is transformed to a contraction and interchange free one as follows :-

```
yes
| ?- theory(cart).
yes
| ?- prove(l = [k, [b] => [e]]).
yes
| ?- prove(m = [th(l,a), [a,b] => [e]]).
yes
| ?- prove(n = [th(m,b), [a,b,b] => [e]]).
yes
| ?- prove(o = [th(n,a), [a,b,a,b] => [e]]).
yes
| ?- prove(p = [inc(o,a,b), [a,a,b,b] => [e]]).
yes
| ?- prove(q = [con(p,a), [a,b,b] => [e]]).
yes
| ?- prove(r = [con(q,b), [a,b] => [e]]).
yes
| ?- prove(u = [th(r,d), [d,a,b] => [e]]).
yes
| ?- con_inc_free(u).

Proof is as follows :
====================

l=[k,[b]=>[e]]
```

```
m=[th(l,a),[a,b]=>[e]]
n=[th(m,b),[a,b,b]=>[e]]
o=[th(n,a),[a,b,a,b]=>[e]]
p=[inc(o,a,b),[a,a,b,b]=>[e]]
q=[con(p,a),[a,b,b]=>[e]]
r=[con(q,b),[a,b]=>[e]]
u=[th(r,d),[d,a,b]=>[e]]

contraction and interchange free proof is :
============================================

l=[k,[b]=>[e]]
r=[th(l,a),[a,b]=>[e]]
u=[th(r,d),[d,a,b]=>[e]]

yes
| ?-
```

which in tree form is :

```
            k
      b --------> e
   ----------------
    a,b --------> e
  -----------------
  d,a,b --------> e
```

Example 2

Let a be an object of the category X; then the derivation

```
              a -> a
       ---------------
       a, b ↑ c -> a
       ---------------
       b ↑ c ,a -> a
    ------------------
    (b ↑ c) ↑ a -> a
```

uses the ' interchange ' rule and has an occurrence of the ' thinning in the antecedent ' rule with non-atomic active formula b ↑ c . This can be transformed as follows :

```
yes
| ?- theory(cart).
yes
| ?- prove(l = [id(a), [a] => [a]]).
yes
| ?- prove(m = [th(l,b & c), [a,b & c] => [a]]).
yes
| ?- prove(n = [inc(m,b & c,a), [b & c,a] => [a]]).
yes
| ?- prove(o = [aia(n,(b & c) & a), [(b & c) & a] => [a]]).
yes
| ?- con_inc_free(o).

Proof is as follows :
=====================

l=[id(a),[a]=>[a]]
m=[th(l,b&c),[a,b&c]=>[a]]
n=[inc(m,b&c,a),[b&c,a]=>[a]]
o=[aia(n,(b&c)&a),[(b&c)&a]=>[a]]
```

```
•contraction and interchange free proof is :
==========================================

l=[id(a),[a]=>[a]]
f1=[th(l,c),[c,a]=>[a]]
m=[th(f1,b),[b,c,a]=>[a]]
n=[aia(m,b&c),[b&c,a]=>[a]]
o=[aia(n,(b&c)&a),[(b&c)&a]=>[a]]
```

which in tree form is :

```
                a -> a
             ----------
              c,a -> a
            -----------
            b,c,a -> a
          -------------
          b ↑ c ,a -> a
        ---------------
        (b ↑ c) ↑ a -> a
```

(1.5.5) <u>Definition</u>

A derivation p in Dc(X) is <u>normal</u> if it satisfies the following conditions :-

(1) p contains no subderivations constructed with (R1),(R3) or (R4).

(2) If p derives G -> a and T is the only atomic subformula of a, then p is of the form

```
        q
        -> a
   ---------  (K)
   G    -> a
```

where (K) denotes the instances of (R2) required to derive G -> a from -> a.

(3) p contains only subderivations constructed with (R2), if any, in which the active formulae are atomic .

(4) Using =< to denote ' priority ' , and saying Ri =< Rj iff whenever a choice exists as to whether 'rule' Ri or Rj is used , Ri is used first :
for normality it is required that

(R10) =< (R11) =< (R2) .

(1.5.6) <u>The Normalization Theorem for Dc(X)</u>

Every p in Dc(X) is equivalent to a unique normal q in Dc(X).

<u>Proof</u>

By combining the proof of the Cut-Elimination theorem(1.5.2), lemma (1.5.3), lemma (1.5.4) and the following conditions of appendix (4) : (D.1), (D.8), (D.53) and the special cases of (D.7) and (D.49) with F and E are empty, as in SZABO, every p in Dc(X) reduces to a unique equivalent normal derivation q in Dc(X) .

###

Given a derivation p in Dc(X), its normal form can be obtained by the use of our system as follows :
Insert the derivation p to the system in the same way as done in (1.5.2); instead of using the command " cut_free(Z) " use the general command " normal(Z) " where Z represents the name of the last step in the inserted proof. The system starts by eliminating cuts and produces a cut-free one; then it continues to eliminate interchanges and contr_ actions from the cut-free one and satisfy the condition on the active formulae of thinnings and produce an equivalent proof which is cut-free, interchange-free, contraction-free and with atomic thinning formulae. Finally the proof resulting from the above processes is transformed by the part of the implementation of the algorithm in appendix (4), to a normal proof.

Example :-

For objects a,b,c of the category X; the derivation

```
   a -> a          b -> b
 ---------       ---------
 a,b -> a        a,b -> b
 -------------------------
       a,b -> a ↑ b
    ---------------
     a ↑ b -> a ↑ b                    a -> a
   ----------------                   -------
   a ↑ b,c -> a ↑ b                   a,b -> a                      c -> c
   ----------------                 ----------                 ------------
 (a ↑ b) ↑ c -> a ↑ b               a ↑ b -> a                 a ↑ b,c -> c
 ------------------------------------------------            ---------------
              (a ↑ b) ↑ c -> a                               (a ↑ b) ↑ c -> c
              ------------------------------------------------------------------
                                   (a ↑ b) ↑ c -> a ↑ c
```

is transformed to its equivalent normal form as follows :

```
yes
| ?- theory(cart).
yes
| ?- prove(l = [id(a), [a] => [a]]).
yes
| ?- prove(l1 = [th(l,b), [a,b] => [a]]).
yes
| ?- prove(m = [id(b), [b] => [b]]).
yes
| ?- prove(m1 = [th(m,a), [a,b] => [b]]).
yes
| ?- prove(n = [ais(l1,m1,a & b), [a,b] => [a & b]]).
yes
| ?- prove(o = [aia(n,a & b), [a & b] => [a & b]]).
yes
| ?- prove(p = [th(o,c), [a & b,c] => [a & b]]).
yes
| ?- prove(q = [aia(p,(a&b) & c), [(a&b) & c] => [a&b]]).
yes
| ?- prove(r = [id(a), [a] => [a]]).
yes
| ?- prove(r1 = [th(r,b), [a,b] => [a]]).
yes
| ?- prove(r2 = [aia(r1,a & b), [a & b] => [a]]).
yes
| ?- prove(s = [cut(q,r2,a & b), [(a & b) & c] => [a]]).
yes
| ?- prove(t = [id(c), [c] => [c]]).
```

```
yes
| ?- prove(t1 = [th(t,a & b), [a & b,c] => [c]]).
yes
| ?- prove(t2 = [aia(t1,(a&b) & c), [(a&b) & c] => [c]]).
yes
| ?- prove(u = [ais(s,t2,a & c), [(a & b) & c] => [a & c]]).
yes
| ?- normal(u).

Proof is as follows :
=====================

l=[id(a),[a]=>[a]]
l1=[th(l,b),[a,b]=>[a]]
m=[id(b),[b]=>[b]]
m1=[th(m,a),[a,b]=>[b]]
n=[ais(l1,m1,a&b),[a,b]=>[a&b]]
o=[aia(n,a&b),[a&b]=>[a&b]]
p=[th(o,c),[a&b,c]=>[a&b]]
q=[aia(p,(a&b)&c),[(a&b)&c]=>[a&b]]
r=[id(a),[a]=>[a]]
r1=[th(r,b),[a,b]=>[a]]
r2=[aia(r1,a&b),[a&b]=>[a]]
s=[cut(q,r2,a&b),[(a&b)&c]=>[a]]
t=[id(c),[c]=>[c]]
t1=[th(t,a&b),[a&b,c]=>[c]]
t2=[aia(t1,(a&b)&c),[(a&b)&c]=>[c]]
u=[ais(s,t2,a&c),[(a&b)&c]=>[a&c]]

cut_free Proof is :
===================

f32=[id(a),[a]=>[a]]
f29=[th(f32,b),[a,b]=>[a]]
f26=[th(f29,b),[a,b,b]=>[a]]
f14=[th(f26,a),[a,b,a,b]=>[a]]
f17=[inc(f14,a,b),[a,a,b,b]=>[a]]
f18=[con(f17,a),[a,b,b]=>[a]]
f9=[con(f18,b),[a,b]=>[a]]
f6=[aia(f9,a&b),[a&b]=>[a]]
f3=[th(f6,c),[a&b,c]=>[a]]
s=[aia(f3,(a&b)&c),[(a&b)&c]=>[a]]
t=[id(c),[c]=>[c]]
t1=[th(t,a&b),[a&b,c]=>[c]]
t2=[aia(t1,(a&b)&c),[(a&b)&c]=>[c]]
u=[ais(s,t2,a&c),[(a&b)&c]=>[a&c]]

contraction and interchange free proof is :
===========================================

f32=[id(a),[a]=>[a]]
f9=[th(f32,b),[a,b]=>[a]]
f6=[aia(f9,a&b),[a&b]=>[a]]
f3=[th(f6,c),[a&b,c]=>[a]]
s=[aia(f3,(a&b)&c),[(a&b)&c]=>[a]]
t=[id(c),[c]=>[c]]
f34=[th(t,b),[b,c]=>[c]]
f33=[th(f34,a),[a,b,c]=>[c]]
t1=[aia(f33,a&b),[a&b,c]=>[c]]
t2=[aia(t1,(a&b)&c),[(a&b)&c]=>[c]]
u=[ais(s,t2,a&c),[(a&b)&c]=>[a&c]]

 Normal proof is :
==================

f32=[id(a),[a]=>[a]]
f9=[th(f32,b),[a,b]=>[a]]
f6=[aia(f9,a&b),[a&b]=>[a]]
f3=[th(f6,c),[a&b,c]=>[a]]
t=[id(c),[c]=>[c]]
f34=[th(t,b),[b,c]=>[c]]
f33=[th(f34,a),[a,b,c]=>[c]]
t1=[aia(f33,a&b),[a&b,c]=>[c]]
s=[ais(f3,t1,a&c),[a&b,c]=>[a&c]]
u=[aia(s,(a&b)&c),[(a&b)&c]=>[a&c]]

yes
| ?-
```

which in tree form is :

```
                 a -> a               c -> c
                --------             --------
                a,b -> a             b,c -> c
              ----------           ----------
              a ↑ b -> a           a,b,c -> c
           -------------        -------------
           a ↑ b ,c -> a        a ↑ b ,c -> c
           ----------------------------------
                (a ↑ b) ↑ c -> a ↑ c
```

(1.5.7) The CHURCH-ROSSER Theorem for Dc(X)

If p == q, then there exists a normal r in Dc(X) such that p >= r and q >= r.

(1.6) Application to Category Theory

In this section we give some examples of the operation of our implementation based on arrows in the theory of cartesian categories . Proofs in Dc(X) correspond to these arrows; their cut-free equivalents, contraction and interchange-free equivalents and their normal forms are produced. We also give some examples to prove the commutativity of diagrams in the theory of cartesian categories using the implementation.

The command " norm_rep(X) " where X represents arrow from Fc(X), is used to find the normal representation of this arrow. It will start with constructing the representing proof for this arrow. Finally it works on this proof and produces the normal proof representing the given arrow, as follows :

Example 1

The normal representation of the arrow

h = comp(g, pi_right(a,c))

in the digram

```
        pi_right(A,C)             g
A ↑ C  --------------->  C  ---------> D
```

where g in X(c,d), can be obtained as follows :

```
yes
| ?- theory(cart).
yes
| ?- arrow(c,g,d).
yes
| ?- norm_rep(comp(g, pi_right(a,c))).
```

```
The arrow is :
==============

comp(g,pi_right(a,c))

Proof is as follows :
=====================

f5=[id(c),[c]=>[c]]
f4=[th(f5,a),[a,c]=>[c]]
f3=[ala(f4,a&c),[a&c]=>[c]]
f2=[g,[c]=>[d]]
f1=[cut(f3,f2,c),[a&c]=>[d]]

cut_free Proof is :
===================

f11=[g,[c]=>[d]]
f8=[th(f11,a),[a,c]=>[d]]
f1=[ala(f8,a&c),[a&c]=>[d]]

contraction and interchange free proof is :
===========================================

f11=[g,[c]=>[d]]
f8=[th(f11,a),[a,c]=>[d]]
f1=[ala(f8,a&c),[a&c]=>[d]]

 Normal proof is :
==================
f11=[g,[c]=>[d]]
f8=[th(f11,a),[a,c]=>[d]]
f1=[ala(f8,a&c),[a&c]=>[d]]

yes
| ?-
```

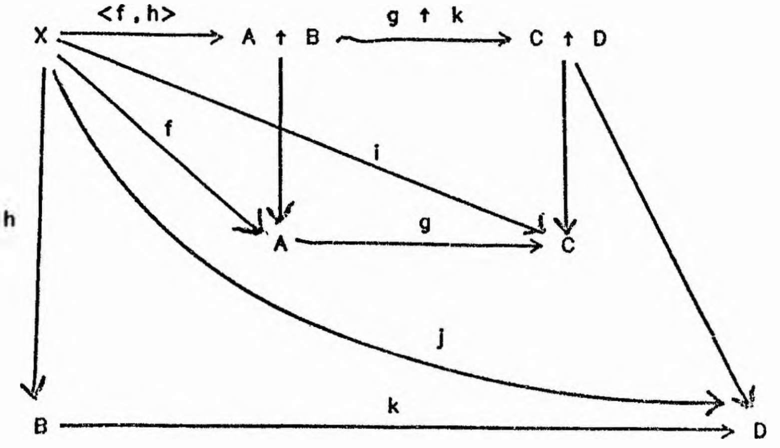
which in tree form is :

```
          g
        c -> d
      ---------
      a,c -> d
    -----------
    a ↑ c -> d
```

The following is a session with the system to prove commutativity of diagrams in the theory of cartesian categories, in other words to prove the equality of arrows by proving the equality of their normal representations. The command " commutative " is to be used for this purpose; after typing this command the system waits to read the LHS arrow of the equality in question and then responds by computing its normal representation as in the above example. The system waits again to read the RHS arrow of the equality and responds by computing its normal representation. Finally it compares these two normal representations and responds with the conclusion as follows :

Example 2

In the following diagram



$< gf, kh > = (g \uparrow k) . < f,h >$

and this can be proved by the use of the system as follows :

```
yes
| ?- theory(cart).
yes
| ?- arrow(x,f,a).
yes
| ?- arrow(x,h,b).
yes
| ?- arrow(x,i,c).
yes
| ?- arrow(x,j,d).
yes
| ?- arrow(a,g,c).
yes
| ?- arrow(b,k,d).
yes
| ?- composite(g,f,i).
yes
| ?- composite(k,h,j).
yes
| ?- commutative.
Insert the first arrow :angle(comp(g,f), comp(k,h)).

The arrow is :
==============

angle(comp(g,f),comp(k,h))

Representation :
================

f4=[f,[x]=>[a]]
f3=[g,[a]=>[c]]
f2=[cut(f4,f3,a),[x]=>[c]]
f7=[h,[x]=>[b]]
f6=[k,[b]=>[d]]
f5=[cut(f7,f6,b),[x]=>[d]]
f1=[ais(f2,f5,c&d),[x]=>[c&d]]

Cut_free proof is :
===================

f2=[i,[x]=>[c]]
f5=[j,[x]=>[d]]
f1=[ais(f2,f5,c&d),[x]=>[c&d]]


Contraction and interchange free proof is :
===========================================

f2=[i,[x]=>[c]]
```

```
f5=[j,[x]=>[d]]
f1=[ais(f2,f5,c&d),[x]=>[c&d]]


 Normal proof is :
================

f2=[i,[x]=>[c]]
f5=[j,[x]=>[d]]
f1=[ais(f2,f5,c&d),[x]=>[c&d]]


Insert the second arrow :comp(product(g,k), angle(f,h)).


The arrow is :
============

comp(product(g,k),angle(f,h))


Representation :
===============

f21=[f,[x]=>[a]]
f22=[h,[x]=>[b]]
f20=[ais(f21,f22,a&b),[x]=>[a&b]]
f14=[id(a),[a]=>[a]]
f13=[th(f14,b),[a,b]=>[a]]
f12=[ala(f13,a&b),[a&b]=>[a]]
f10=[g,[a]=>[c]]
f15=[cut(f12,f10,a),[a&b]=>[c]]
f18=[id(b),[b]=>[b]]
f17=[th(f18,a),[a,b]=>[b]]
f16=[ala(f17,a&b),[a&b]=>[b]]
f11=[k,[b]=>[d]]
f19=[cut(f16,f11,b),[a&b]=>[d]]
f9=[ais(f15,f19,c&d),[a&b]=>[c&d]]
f8=[cut(f20,f9,a&b),[x]=>[c&d]]


Cut_free proof is :
=================

f51=[i,[x]=>[c]]
f45=[th(f51,x),[x,x]=>[c]]
f37=[con(f45,x),[x]=>[c]]
f62=[j,[x]=>[d]]
f56=[th(f62,x),[x,x]=>[d]]
f40=[con(f56,x),[x]=>[d]]
f8=[ais(f37,f40,c&d),[x]=>[c&d]]


contraction and interchange free proof is :
=========================================

f37=[i,[x]=>[c]]
f40=[j,[x]=>[d]]
f8=[ais(f37,f40,c&d),[x]=>[c&d]]


Normal proof is :
===============

f37=[i,[x]=>[c]]
f40=[j,[x]=>[d]]
f8=[ais(f37,f40,c&d),[x]=>[c&d]]


The answer is :
=============

angle(comp(g,f),comp(k,h)) =
comp(product(g,k),angle(f,h))

because they have the same normal representation .

yes
| ?-
```

CHAPTER 2

BICARTESIAN CATEGORY THEORY

In this chapter, we extend the representation of the class of small cartesian categories in chapter 1 to represent the class of small bicartesian categories. We extend the implementation of the theory of cartesian categories, in order to handle the extension of the algorithms for representing arrows of the free bicartesian category Fbc(X) over X by means of derivations from the class Dbc(X) of unlabelled ' bicartesian-derivations ' over X ; interpreting derivations of this class as arrows of Fbc(X) , and the others for cut-eliminations and normal form (2.5).

This chapter is divided in a similar way to that of the previous chapter. We tidy up some obscurities; these lie essentially in the definition of the unlabelled deductive system relevant to the theory of bicartesian categories. The need for the "interchange in the succedent" rule in order to perform large contractions will be discussed in more detail in (2.5.1).

As a consequence of adding rule (R7) "interchange in the succedent", we add more clauses to the proof of the cut-elimination theorem (2.5.2). Corollaries (2.5.6) and (2.5.7), concerning the normal representation for the terminal and the initial arrows in Fbc(X), were stated without proof in SZABO; we have constructed and implemented their proofs in the system. As in the previous chapter, we modified lemma (2.5.8), which was stated without proof in SZABO, to transform all the subderivations constructed by (R4) and (R7) to their equivalent interchange-free ones; the proof has been constructed and implemented in the system.

(2.1) Definition

A bicartesian category is a cartesian category $\underline{C}$ with the following additional structure :-

(4) A bifunctor (-) v (-) : $\underline{C} \times \underline{C} \to \underline{C}$.

(5) A distinguished object $\perp \in \text{Ob}\ \underline{C}$.

(6) Two adjunction α_σ and α_ι , where

. $\alpha_\sigma = \{\alpha_\sigma\ (A,B,C) : \underline{C}(A \vee B,\ C) \rightarrow \underline{C}(A,C) \times \underline{C}(B,C) \in ArEns \mid A,B,C \in Ob\ \underline{C}\}$,

and

$\alpha_\iota = \{\alpha_\iota\ (A) : \underline{C}(\perp, A) \rightarrow \{*\} \in Ar\ Ens \mid A \in Ob\ \underline{C}\}$.

(2.2) The Free Bicartesian Category Fbc(X)

Let X be a fixed but arbitrary small category. The free bicartesian category Fbc(X) is constructed as follows :-

(2.2.1) Definition

The class bcL(X) of (bicartesian) formulae over X is the smallest class satisfying the conditions of Definition (1.2.1) and the following additional conditions :-

v) $\perp$ is a formula ;

vi) A v B is a formula when A,B are ;

vii) The class bcL(X) is generated by i) to vi) .

(2.2.2) The Labelled Derivations LDbc(X)

The class of (bicartesian) labelled derivations LDbc(X) over X is defined inductively by (i) - (viii) of (1.2.2) and the following:

(ix) For A in bcL(X) ,
LA6(A) is a derivation ,
with conclusion tau_star(A) : $\perp$ -> A ;

(x) For A,B in bcL(X) ,
LA7(A,B) is a derivation ,
with conclusion pi_left_star(A,B) : A -> A v B;

(xi) For A,B in bcL(X) ,
LA8(A,B) is a derivation ,
with conclusion pi_right_star(A,B) : B -> A v B;

(xii) For derivations F,G with conclusions f : A -> C and g : B -> C respectively
SQUARE(F, G, A v B) is a derivation
with conclusion [f,g] : A v B -> C .

(2.2.3) Definition

The relation == is defined as the smallest equivalence relation on LDbc(X) satisfying the conditions from (i) to (viii) of definition (1.2.3) and the following additional conditions :-

ix) If F == G and H == K with conclusions f,g : A -> C ,
h,k : B -> C
then SQUARE(F, H, A v B) == SQUARE(G, K, A v B) ;

x) For F,G with conclusions f : A -> C, g : B -> C respectively
COMP(LA7(A,B), SQUARE(F, G, A v B), A v B) == F and
COMP(LA8(A,B), SQUARE(F, G, A v B), A v B) == G ;

xi) For F with conclusion f : A v B -> C ,
SQUARE(COMP(LA7(A,B), F, A v B),
COMP(LA8(A,B), F, A v B),
A v B) == F ;

xii) For F with conclusion f : ⊥ -> A ,
F == LA6(A) .

(2.2.4) <u>Definition</u>

We define Fbc(X) to be the category having :

i) as objects, the bicartesian formulae over X ;
ii) as arrows , the ==- classes of LDbc(X) ;
iii) to xi) as in the Definition (1.2.5) of Fc(X) ;
xii) as ⊥ , the formula ⊥ ;
xiii) as tau_star , the function defined by
tau_star(A) = [LA6(A)] ;
xiv) as pi_left_star, the function defined by
pi_left_star(A,B) = [LA7(A,B)] ;
xv) as pi_right_star, the function defined by
pi_right_star(A,B) = [LA8(A,B)] ;
xvi) as [-,-] , the function defined by
[[F],[G]] = [SQUARE(F,G,dom([F]) v dom([G]))]
when cod([F]) = cod([G]) .

It is now routine to check that, with this structure, Fbc(X) is a category, is bicartesian, and is free bicartesian over X. There is the obvious embedding X |-> Fbc(X) defined by A |-> A; f |-> [LA1(f)].

(2.3) The Unlabelled Derivations Dbc(X)

We now extend the class Dc(X) of unlabelled cartesian derivations to the class Dbc(X) of unlabelled bicartesian derivations to represent the arrows of Fbc(X) .

(2.3.1) Definition

The class Dbc(X) of unlabelled (bicartesian) derivations over a category X is defined inductively as follows :-

Note : we use the same notations for formulae , lists , sequents and derivations as in (1.3) .

i) If a,b are objects of X , and if f is in X(a,b) , then A1(f) is a derivation with conclusion $a \xrightarrow{f} b$;

ii) A2 is a derivation with conclusion [] -> T ;

iii) A3 is a derivation with conclusion ⊥ -> [] ;

iv) (cut)

a) if p1, p2 are derivations with conclusions G -> a, DaL -> F, then
R1(p1,p2,a) is a derivation with conclusion DGL -> F

b) if p1, p2 are derivations with conclusions G -> FaE, a -> H, then
R1(p1,p2,a) is a derivation with conclusion G -> FHE ;

v) (thinning in the antecedent)
if p is a derivation with conclusion GD -> F , then
R2(p,a) is a derivation with conclusion GaD -> F ;

vi) (contraction in the antecedent)
if p is a derivation with conclusion GaaD -> F , then
R3(p,a) is a derivation with conclusion GaD -> F ;

vii) (interchange in the antecedent)
if p is a derivation with conclusion GabD -> F , then
R4(p,b,a) is a derivation with conclusion GbaD -> F ;

viii) (thinning in the succedent)
if p is a derivation with conclusion G -> FE , then
R5(p,a) is a derivation with conclusion G -> FaE ;

ix) (contraction in the succedent)
if p is a derivation with conclusion G -> FaaE , then
R6(p,a) is a derivation with conclusion G -> FaE ;

x) (interchange in the succedent)
if p is a derivation with conclusion G -> FabE , then
R7(p,b,a) is a derivation with conclusion G -> FbaE ;

xi) (and introduction in the succedent)

If p1,p2 are derivations with conclusions G -> a, G -> b, then R10(p1,p2,a † b) is a derivation with conclusion G -> (a † b);

xii) (and introduction in the antecedent)

If p is a derivation with conclusion GabD -> F , then R11(p, a † b) is a derivation with conclusion G(a † b)D -> F;

xiii) (or introduction in the antecedent)

If p1,p2 are derivations with conclusions a -> F, b -> F, then R12(p1,p2,a v b) is a derivation with conclusion (a v b) -> F;

xiv) (or introduction in the succedent)

If p is a derivation with conclusion G -> FabE , then R13(p, a v b) is a derivation with conclusion G -> F(a v b)E;

xv) The class of unlabelled derivations Dbc(X) is generated by (i) to (xiv) .

(2.3.2) Representation of Dbc(X) in Prolog

We use the technique mentioned in (1.3.2) to represent the unlabelled derivations Dbc(X) in the Prolog. We extend the abbreviations used in (1.3.2) by the following :-

#	for v
bo	for ⊥ in A3
bottom	for the name of the constant in A3
th	for the name of the operation in R2 and R5 (thinning)
con	for the name of the operation in R3 and R6 (contraction)
inc	for the name of the operation in R4 and R7 (interchange)
ois	for the name of the operation in R12 (or introduction in the succedent)
oia	for the name of the operation in R13 (or introduction in the antecedent)

i) to viii) similar to (1.3.2)

ix) A3 is represented by

g1 = [bottom, [bo] => []]

x) R5(p,a) is represented by a step-collection with the last step,

g2 = [th(g1,a), G => FaE]

where g1 is the name of the last step in the step-collection for the derivation p .

xi) R6(p,a) is represented by a step-collection with the last step,

g2 = [con(g1,a), G => FaE]

where g1 is the name of the last step in the step-collection for the derivation p .

xii) R7(p,b,a) is represented by a step-collection with last step,

g2 = [inc(g1,b,a), G => FbaE]

where g1 is the name of the last step in the step-collection for the derivation p .

xiii) R12(p1,p2,a v b) is represented by a step-collection with the last step ,

g3 = [ola(g1,g2,a # b), [a # b] => F]

where g1 , g2 are the names of the last step of the step-collections for the derivations p1, p2 respectively .

xiv) R13(p,a v b) is represented by a step-collection with the last step ,

g2 = [ols(g1,a # b), G => Fa # bE]

where g1 is the name of the last step in the step-collection for the derivation p .

As an example of a derivation of the class Dbc(X), its representation in Prolog and the use of the implementation to check the correctness of this derivation, consider the following :

<u>Example</u>

The derivation

```
    f
  x -> a
---------
x,y -> a
------------
x,y -> a,b          g                h              i           j
--------------    a -> c          a -> d          b -> c      b -> d
                  --------------------------      --------------------
x,y -> a,b,z            a -> c ↑ d                     b -> c ↑ d
----------------  -----------------------------------------------------
x,y -> a v b,z                      a v b -> c ↑ d
----------------------------------------------------------------
                      x,y -> c ↑ d,z
```

is represented by

g1 = [f, [x] => [a]]

g2 = [th(g1,y), [x,y] => [a]]

g3 = [th(g2,b), [x,y] => [a,b]]

g4 = [th(g3,z), [x,y] => [a,b,z]]

g5 = [ols(g4,a#b), [x,y] => [a#b,z]]

g6 = [g, [a] => [c]]

g7 = [h, [a] => [d]]

```
g8 = [ais(g6,g7,c&d), [a] => [c&d]]
g9 = [i, [b] => [c]]
g10 = [j, [b] => [d]]
g11 = [ais(g9,g10,c&d), [b] => [c&d]]
g12 = [oia(g8,g11,a#b), [a#b] => [c&d]]
g13 = [cut(g5,g12,a#b), [x,y] => [c&d,z]]
```

and the use of the system to check its correctness as follows :

```
yes
| ?- theory(bicart).
yes
| ?- prove(g1 = [f, [x] => [a]]).
yes
| ?- prove(g2 = [th(g1,y), [x,y] => [a]]).
yes
| ?- prove(g3 = [th(g2,b), [x,y] => [a,b]]).
yes
| ?- prove(g4 = [th(g3,z), [x,y] => [a,b,z]]).
yes
| ?- prove(g5 = [ois(g4,a # b), [x,y] => [a # b,z]]).
yes
| ?- prove(g6 = [g, [a] => [c]]).
yes
| ?- prove(g7 = [h, [a] => [d]]).
yes
| ?- prove(g8 = [ais(g6,g7,c & d), [a] => [c & d]]).
yes
| ?- prove(g9 = [i, [b] => [c]]).
yes
| ?- prove(g10 = [j, [b] => [d]]).
yes
| ?- prove(g11 = [ais(g9,g10,c & d), [b] => [c & d]]).
yes
| ?- prove(g12 = [oia(g8,g11,a # b), [a # b] => [c & d]]).
yes
| ?- prove(g13 = [cut(g5,g12,a # b), [x,y] => [c & d,z]]).
yes
| ?-
```

(2.4) The Semantics of Dbc(X)

In this section, we extend the definition of the interpretation function in (1.4) to interpret the derivations of Dbc(X) as arrows of Fbc(X) and prove the completeness theorem. In order to define this interpretation, we extend the canonical arrows in (1.4) by the following ones of Fbc(X), determined by α_σ :-

iv) delta_star(A) : A v A -> A for all A in ObFbc(X) , where
α_σ^{-1}(< id(A) , id(A) >) = delta_star(A) .

v) alpha_star(A,B,C) : A v (B v C) -> (A v B) v C
for all A, B, C in ObFbc(X) , where
α_σ^{-1}(comp(pi_left_star(A v B,C),
pi_left_star(A,B)) ,
α_σ^{-1} (comp(pi_left_star(A v B,C),
pi_right_star(A,B)),
pi_right_star(A v B,C))) = alpha_star(A,B,C)

vi) sigma_star(A,B) : A v B -> B v A for all A,B in ObFbc(X), where
α_σ^{-1}(< pi_right_star(B,A), pi_left_star(B,A) >)
= sigma_star(A,B) .

(2.4.1) <u>Definition</u>

We define a function S from the class Dbc(X) of unlabelled derivations to ArFbc(X) satisfying conditions (1) - (8) of (1.4.1) and the following additional conditions :

Note that if a derivation p has conclusion G -> F, then S(p) is an arrow with domain Pi(G) (where Pi as in (1.4.1)) and codomain Un(F) defined by

Un(F) = <u>if</u> F = nil
<u>then</u> T
<u>else</u> <u>if</u> F = [a] <u>then</u> a
<u>else</u> (Un(fst(F)) v lst(F))

where

fst(F) and lst(F) similar to that in (1.4.1) .

9) S(A3) = [LA2($\perp$)] ;

10) S(R1(p1,p2,a)) = comp((I1 v S(p2)) v I2 , S(p1)) , where
I1 is the identity on Un(F) ,
I2 is the identity on Un(E), and we make the canonical identifications of
Un(FHE) with (Un(F) v Un(H)) v Un(E) and of
Un(FaE) with (Un(F) v a) v Un(E) ,
F,H,a,E being as in part(b) of the definition of R1(p1,p2,a) ;

11) S(R5(p,a)) = comp((pi_left_star(Un(F),a) v I), S(p)) ,
where I is the identity on E , and we make the canonical identifications of
Un(FaE) with (Un(F) v a) v Un(E) and of
Un(FE) with Un(F) v Un(E) ,
F,a,E being as in the definition of R5(p,a) ;

12) S(R6(p,a)) = comp(comp((I1 v delta_star(a)) v I2) ,
alpha_star_inverse(Un(F),a,a) v I2),
S(p)) , where
I1 is the identity on Un(F) ,
I2 is the identity on Un(E) ,
alpha_star_inverse is the inverse of alpha_star
and we make the canonical identifications of

Un(FaE) with (Un(F) v a) v Un(E) ,
Un(Fa aE) with (Un(F) v (a v a)) v Un(E) and of
Un(FaaE) with ((Un(F) v a) v a) v Un(E),
F,a,E being as in the definition of R6(p,a) ;

13) S(R7(p,b,a)) = comp(comp(comp(alpha_star(Un(F),b,a) v I2 ,
(I1 v sigma_star(a,b)) v I2),
alpha_star_inverse(Un(F),a,b) v I2),
S(p)), where
I1 is the identity on Un(F) ,
I2 is the identity on Un(E) and we make
the canonical identifications of
Un(FbaE) with ((Un(F) v b) v a) v Un(E) ,
Un(Fb v aE) with (Un(F) v (b v a)) v Un(E),
Un(Fa v bE) with (Un(F) v (a v b)) v Un(E) and of
Un(FabE) with ((Un(F) v a) v b) v Un(E) ,
F,a,b,E being as in the definition of R7(p,b,a).

14) S(R12(p1,p2,a v b)) = comp(delta_star(Un(F)) , S(p1) v S(p2)),
we make the canonical identifications of
Un(FF) with Un(F) v Un(F) ,
F being as in the definition of R12(p1,p2,a v b);

15) S(R13(p,a v b)) = comp(alpha_star_inverse(Un(F),a,b) v I,S(p)),
where I is the identity on Un(E) and we
make the canonical identifications of
Un(Fa v bE) with (Un(F) v (a v b)) v Un(E) and of
Un(FabE) with ((Un(F) v a) v b) v Un(E) ,
F,a,b,G being as in the definition of R13(p,a v b).

The following is an example for a derivation of Dbc(X), and its interpretation in ArFbc(X) by the use of the implementation .

<u>Example</u>

For arrows f in X(a,c), g in X(c,a), and objects a,b,c of the category X, the derivation

```
                       f                           g
c -> c          a -> c                   c -> a        a -> a
------------------------------          ------------------------------
      c v a -> c                              c v a -> a
      ---------------------------------------------------------
                     c v a -> c ↑ a
                     ----------------
                     c v a -> c ↑ a,c
                     --------------------
                     c v a -> (c ↑ a) v c
```

is interpreted in ArFbc(X) as follows :

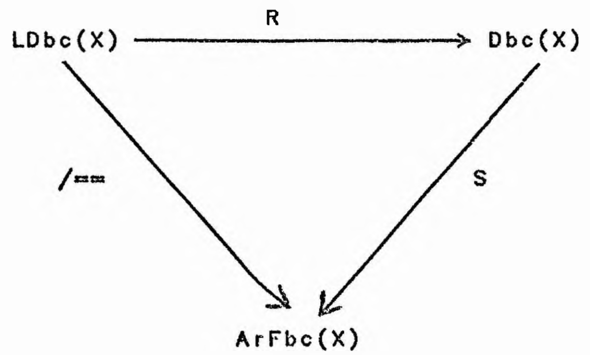
```
yes
| ?- theory(bicart).
yes
| ?- prove(l = [id(c), [c] => [c]]).
yes
| ?- prove(m = [f, [a] => [c]]).
yes
| ?- prove(n = [oia(l,m,c # a), [c # a] => [c]]).
yes
| ?- prove(o = [g, [c] => [a]]).
yes
| ?- prove(p = [id(a), [a] => [a]]).
yes
| ?- prove(q = [oia(o,p,c # a), [c # a] => [a]]).
yes
| ?- prove(r = [ais(n,q,c & a), [c # a] => [c & a]]).
yes
| ?- prove(s = [th(r,c), [c # a] => [c & a,c]]).
yes
| ?- prove(t = [ois(s,(c&a) # c), [c#a] => [(c & a) # c]]).
yes
| ?- interpr(t,H).

H = comp(pi_left_star(c&a,c),
         comp(product(comp(delta_star(c),union(id(c),f)),
                      comp(delta_star(a),union(g,id(a)))),
              delta(c#a)))

yes
| ?-
```

The following theorem proves that the class Dbc(X) constructed for the theory of bicartesian categories is adequate, in the sense that every arrow in the free bicartesian category Fbc(X) can be represented by means of some derivation in Dbc(X) :

(2.4.2) <u>The Completeness Theorem for Dbc(X)</u>

There is a function

R : LDbc(X) ------------> Dbc(X) , so that



commutes.

Proof

We define a function R to satisfy conditions (i) - (vii) of (1.4.2) and the following

Note that I(A) in this proof and the following examples is as defined in (1.4.2) .

viii) when F = LA2(A) for A in bcL(X), not an object of X, we use induction on structure of A :

A is either T
or B ↑ C } as in (ii) of (1.5.2)

or ⊥ : put R(F) = A3
then S(R(F)) = S(A3) = [LA2(⊥)] = [F]

or B v C : put
R(F) = R13(R12(R5(R(I(B)), C),
R5(R(I(C)), B),
B v C),
B v C)

ix) when F is LA6(A) for A in bcL(X) , put
R(F) = R5(A3, A)

x) when F is LA7(A,B) for A,B in bcL(X) , put
R(F) = R13(R5(R(I(A)), B), A v B)

xi) when F is LA8(A,B) for A,B in bcL(X) , put
R(F) = R13(R5(R(I(B)), A), A v B)

xii) when F is SQUARE(G,H,A v B) , put
R(F) = R12(R(G), R(H), A v B)

###

We give some examples to illustrate the algorithm in this proof by showing how some arrows of Fbc(X) are representable by means of unlabelled (bicartesian) derivations. (These particular arrows, and their representations are built into the system for the user's convenience).

Example 1

For the derivation

F = SQUARE(I(A), I(A), A v A)

representing the arrow

delta_star(A) : A v A -> A ,

the representation R(F) is

R12(R(I(A)) , R(I(A)) , A v A)

Example 2

For a derivation

```
F = SQUARE( COMP( LA7(A v B,C), LA7(A,B), A v B ),
            SQUARE( COMP( LA7(A v B,C),LA8(A,B),A v B),
                    LA8(A v B,C),
                    B v C ),
            A v (B v C) ) )
```

representing the arrow

```
alpha_star(A,B,C) : A v (B v C) -> (A v B) v C
```

the representation R(F) is

```
R12(R1(R13(R5(R( I(A v B) ), C ), (A v B) v C ),
       R13(R5(R( I(A) ), B ), A v B),
       A v B ),
    R12(R1(R13(R5(R( I(A v B) ), C), (A v B) v C),
           R13(R5(R( I(B) ), A), A v B),
           A v B),
        R13(R5(R( I(C) ), A v B), (A v B) v C),
        B v C),
    A v (B v C))
```

Example 3

For derivations F,G representing the arrows

f : A -> B , g : C -> D respectively

then for the derivation

```
UNION( F,G ) = SQUARE( COMP( F, LA7(B,D), B ),
                       COMP( G, LA8(B,D), D ),
                       A v C )
```

representing the arrow

```
f v g : A v C -> B v D
```

the representation R(UNION(F,G)) is

```
R12( R1( R( F ),
         R13( R5( R( I(B) ), D), B v D),
         B ),
     R1( R( G ),
         R13( R5( R( I(D) ), B), B v D),
         D ),
     A v C )
```

Example 4

For the derivation

F = SQUARE(LA8(B,A), LA7(B,A), A v B)

representing the arrow

sigma_star(A,B) : A v B -> B v A

the representation R(F) is

```
R12( R13( R5( R( I(A) ), B), B v A),
     R13( R5( R( I(B) ), A), B v A),
     A v B )
```

For the derivation

F = SQUARE(LA8(A,B), LA7(A,B), B v A)

representing the arrow

sigma_star_inverse(A,B) : B v A -> A v B

the representation R(F) is

```
R12( R13( R5( R( I(B) ), A), A v B),
     R13( R5( R( I(A) ), B), A v B),
     B v A )
```

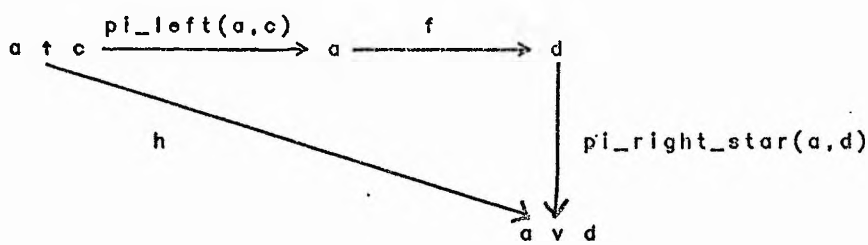
The following example is a session of input and output with the system to represent an arrow of Fbc(X) by means of some unlabelled bicartesian derivation :

Example 5

For f in X(a,d) , the arrow

h = comp(comp(pi_right_star(a,d),f),pi_left(a,c)) : a ↑ c -> a v d

in the diagram



is represented by an unlabelled derivation as follows :

```
yes
| ?- theory(bicart).
yes
| ?- arrow(a,f,d).
yes
| ?- rep_of(comp(comp(pi_right_star(a,d),f),pi_left(a,c))).

The arrow is
============

comp(comp(pi_right_star(a,d),f),pi_left(a,c))
```

```
Representation :-
=================

f9=[id(a),[a]=>[a]]
f8=[th(f9,c),[a,c]=>[a]]
f7=[aia(f8,a&c),[a&c]=>[a]]
f6=[f,[a]=>[d]]
f5=[id(d),[d]=>[d]]
f4=[th(f5,a),[d]=>[a,d]]
f3=[ois(f4,a#d),[d]=>[a#d]]
f2=[cut(f6,f3,d),[a]=>[a#d]]
f1=[cut(f7,f2,a),[a&c]=>[a#d]]

yes
| ?-
```

which in tree form is :

```
                                          d -> d
                                          --------
        a -> a                            d -> a,d
       --------                    f      ----------
      a,c -> a                  a -> d    d -> a v d
     ----------                 ----------------------
     a ↑ c -> a                        a -> a v d
     ---------------------------------------------
                  a ↑ c -> a v d
```

(2.5) The Syntax of Fbc(X)

After representing the arrows of the free bicartesian category Fbc(X) by unlabelled derivations with cuts, we are able to manipulate these derivations to eliminate cuts and obtain their normal representations.

(2.5.1) Interchange in the succedent rule and its justification

Likewise as in (1.5.1), we observe in SZABO'S algorithm for cut-elimination that in some clauses like [(C.4), (C.15),...], there is a big contraction say for, H since H may be of length greater than one. In this case we must perform interchanges first between the objects of the sequence HH and then using the contraction rule (R6) to obtain H. This explains the inclusion of the inference rule 'interchange in the succedent ' in the definition of the class Dbc(X) of unlabelled (bicartesian) derivations. As a result of the inclusion of this rule, there are a few changes in the definition of the normalization .

The following example is to justify the need for the rule ' interchange in the succedent ' :-

alpha_star(A,B,C) : A v (B v C) ----------> (A v B) v C

and this is a canonical arrow of the free bicartesian category Fbc(X). Using the completeness theorem, its unlabelled representation derivation is as follows :

```
                                                 a -> a      b -> b
                                                 --------    --------
                 a -> a      b -> b              a -> a,b    b -> a,b
                 --------    --------            --------    --------
                 a -> a,b    b -> a,b              a v b -> a,b
                 --------------------            ----------------------
                   a v b -> a,b         b ---> b   a v b -> a v b
                 --------------------   --------  ----------------------
a ---> a           a v b -> a v b       b-> a,b   avb -> avb,c      c -> c
--------         --------------------   --------  ----------------  ----------
a -> a,b           a v b -> a v b,c     b->avb    avb -> (avb)v c   c -> avb,c
--------         --------------------   --------  ----------------  ------------
|a -> a v b   a v b -> (a v b) v c|       b -> (a v b) v c          c -> (avb)vc
|---------------------------------|     ----------------------------------------
|       a -> (a v b) v c          |             b v c -> (a v b) v c
|---------------------------------|-----------------------------------------
                    a v (b v c) -> (a v b) v c
```

Applying clause (C.29) from the cut-elimination algorithm in appendix 3 to the cut in the marked box above , we get

```
                                                 a -> a      b -> b
                                                 ------------------
                 a -> a     b -> b               a -> a,b b -> a,b
                 ----------------                ------------------
                 a -> a,b b -> a,b                 a v b -> a,b
                 -----------------                 ---------------
a -> a             a v b -> a,b         b -> b     avb -> avb
--------         --------------------   --------   ---------------
a -> a,b           a v b -> a v b       b-> a,b    avb -> avb,c     c -> c
--------         --------------------   --------   ---------------  ------------
|a -> a v b    a v b -> a v b,c|        b-> avb    avb -> (avb)vc   c -> avb,c
|------------------------------|        ---------------------------  ------------
|          a -> a v b,c        |            b -> (a v b) v c        c -> (avb)vc
|------------------------------|           ----------------------------------------
          a -> (a v b) v c                          b v c -> (a v b) v c
          ------------------------------------------------------------------
                          a v (b v c) -> (a v b) v c
```

Applying clause (C.21), the above tree is transformed to the following:

```
                                                a -> a       b -> b
                                                ---------    ----------
               a -> a        b -> b             a -> a,b      b -> a,b
               ---------     ---------          ---------    ----------
 a -> a        a -> a,b      b -> a,b              a v b -> a,b
 --------      ------------------------            ----------------
 a -> a,b         a v b -> a,b          b ---> b   a v b -> a v b
 --------      ------------------------ --------   ----------------
|a -> a v b      a v b -> a v b  |      b-> a,b    avb -> avb,c     c -> c
|--------------------------------|      --------   ----------------  ----------
|       a -> a v b               |      b->a v b   avb -> (avb)vc   c -> avb,c
|--------------------------------|      ---------------------------  ------------
      a -> a v b,c                        b -> (a v b) v c          c -> (avb)vc
 ----------------------------------      ----------------------------------------
      a -> (a v b) v c                            b v c -> (a v b) v c
      ------------------------------------------------------------------
                          a v (b v c) -> (a v b) v c
```

Applying clause (C.29) to the cut in the above marked box; the above tree is transformed to the following one :

```
                                                      a -> a      b -> b
                                                      --------    --------
a -> a      a -> a      b -> b                        a -> a,b    b -> a,b
--------    --------    --------                      --------    --------
a -> a,b     a -> a,b     b -> a,b                       a v b -> a,b
             ----------------------                   ------------------
|a -> a v b      a v b -> a,b        |  b -> b      a v b -> a v b
|------------------------------------|  -------     ----------------
|          a -> a,b                  |  b-> a,b     avb -> avb,c       c -> c
|------------------------------------|  -------     ----------------   ----------
           a -> a v b                   b->avb      avb -> (avb)vc     c -> avb,c
--------------------------------------  --------------------------    ------------
           a -> a v b,c                       b -> (a v b) v c         c -> (avb)vc
--------------------------------------        ----------------------------------
           a -> (a v b) v c                          b v c -> (a v b) v c
           --------------------------------------------------------------
                          a v (b v c) -> (a v b) v c
```

Applying clause (C.15) to the cut in the above marked box , we get the following equivalent tree. This uses the rule ' interchange in the succedent ' for the objects a,b at the step indicated bellow .

```
a -> a      a -> a
--------    --------
a -> a,b    a -> a,b    b -> b
--------------------    --------
    a -> a,b,b          b -> a,b                  a -> a      b -> b
--------------------------------                  --------    --------
||        a -> a,b,a,b         ||                 a -> a,b    b -> a,b
||-----------------------------||                 --------------------
          a -> a,a,b,b                               a v b -> a,b
  ------------------------------                  --------------------
          a -> a,b,b                 b -> b          a v b -> a v b
  ------------------------------     -------      --------------------
          a -> a,b                   b-> a,b      avb -> avb,c           c -> c
  ------------------------------     -------      --------------------   ----------
          a -> a v b                 b->avb       avb -> (avb)vc         c -> avb,c
  ------------------------------     -----------------------------     ------------
          a -> a v b,c                  b -> (a v b) v c                 c -> (avb)vc
  ------------------------------        -------------------------------------------
          a -> (a v b) v c                        b v c -> (a v b) v c
          ---------------------------------------------------------------
                        a v (b v c) -> (a v b) v c
```

We continue the processes of the cut-elimination algorithm until , we get the following equivalent cut-free proof :

```
                        b -> b
                        ------------------
a -> a                  b -> a,b
------------------      ------------------
a -> a,b                b -> a,b,b
------------------      ------------------
a -> a,b,b              b -> a,b,a,b
------------------      ------------------
a -> a,b,a,b            b -> a,a,b,b
------------------      ------------------
a -> a,a,b,b            b -> a,b,b
------------------      ------------------
a -> a,b,b              b -> a,b
------------------      ------------------
a -> a,b                b -> a v b              c -> c
------------------      ------------------      ----------------------
a -> a v b              b -> a v b,c            c -> a v b,c
------------------      ------------------      ----------------------
a -> a v b,c            b -> (a v b) v c        c -> (a v b) v c
------------------      -----------------------------------------------
a --> (a v b) v c                  b v c -> (a v b) v c
-----------------------------------------------------------------------
                   a v (b v c) -> (a v b) v c
```

(2.5.2) <u>The Cut-Elimination Theorem for Dbc(X)</u>

Every p in Dbc(X) is equivalent to a cut-free q in Dbc(X).

<u>Proof</u>

By Theorem (1.5.2) , together with the following clauses of the algorithm in appendix (2) :
(C.4), (C.7), (C.8), (C.9), (C.12), (C.15) , (C.21) , (C.22) , (C.29) , (C.37) , (C.38) , (C.43) and (C.44) as in SZABO [1978] , in addition to clauses (C.23) and (C.39) needed because of the additional rule (R7) in the definition of Dbc(X), there is an equivalent cut-free derivation for every derivation of the class Dbc(X) of unlabelled derivations .

###

The following example is to illustrate the above cut-elimination theorem :

<u>Example :</u>

For f in X(a,c), g in X(d,b), where a,b,c,d are objects of a category X, the derivation

```
                                                                  a -> a    b -> b
                                                                  -------   -------
                                                                  a -> a,b  b -> a,b
                                                                  ------------------
                                                                     avb -> a,b
                                                                     ----------
 a -> a            d -> d                           b -> b           avb -> avb
--------          --------                         --------        ------------
a,d -> a    f     a,d -> d     g         b -> b    b -> b,a        avb -> avb,c
--------          --------             --------    --------       --------------
a↑d -> a  a -> c  a↑d -> d  d -> b     c,b -> b    b -> avb       avb -> (avb)vc
----------------  ----------------     --------    -----------------------------
  a ↑ d -> c        a ↑ d -> b         c↑b -> b           b -> (a v b) v c
----------------------------------     -------------------------------------
         a ↑ d -> c ↑ b                         c ↑ b -> (a v b) v c
         -------------------------------------------------------------
                            a ↑ d -> (a v b) v c
```

is transformed to a cut_free one as follows :

```
yes
| ?- theory(bicart).
yes
| ?- prove(s1 = [id(a), [a] => [a]]).
yes
| ?- prove(s2 = [th(s1,d), [a,d] => [a]]).
yes
| ?- prove(s3 = [ala(s2,a & d), [a & d] => [a]]).
yes
| ?- prove(s4 = [f, [a] => [c]]).
yes
| ?- prove(s5 = [cut(s3,s4,a), [a & d] => [c]]).
yes
| ?- prove(s6 = [id(d), [d] => [d]]).
yes
| ?- prove(s7 = [th(s6,a), [a,d] => [d]]).
yes
| ?- prove(s8 = [ala(s7,a & d), [a & d] => [d]]).
```

```
yes
| ?- prove(s9 = [g, [d] => [b]]).
yes
| ?- prove(s10 = [cut(s8,s9,d), [a & d] => [b]]).
yes
| ?- prove(s11 = [ais(s5,s10,c & b), [a & d] => [c & b]]).
yes
| ?- prove(s12 = [id(b), [b] => [b]]).
yes
| ?- prove(s13 = [th(s12,c), [c,b] => [b]]).
yes
| ?- prove(s14 = [aia(s13,c & b), [c & b] => [b]]).
yes
| ?- prove(s15 = [id(b), [b] => [b]]).
yes
| ?- prove(s16 = [th(s15,a), [b] => [a,b]]).
yes
| ?- prove(s17 = [ois(s16,a # b), [b] => [a # b]]).
yes
| ?- prove(s18 = [d(a), [a] => [a]]).
yes
| ?- prove(s19 = [th(s18,b), [a] => [a,b]]).
yes
| ?- prove(s20 = [id(b), [b] => [b]]).
yes
| ?- prove(s21 = [th(s20,a), [b] => [a,b]]).
yes
| ?- prove(s22 = [oia(s19,s21,a # b), [a # b] => [a,b]]).
yes
| ?- prove(s23 = [ois(s22,a # b), [a # b] => [a # b]]).
yes
| ?- prove(s24 = [th(s23,c), [a # b] => [a # b,c]]).
yes
| ?- prove(s25 = [ois(s24,(a#b)#c), [a#b] => [(a#b)#c]]).
yes
| ?- prove(s26 = [cut(s17,s25,a # b), [b] => [(a#b)# c]]).
yes
| ?- prove(s27 = [cut(s14,s26,b), [c&b] => [(a#b)#c]]).
yes
| ?- prove(s28 = [cut(s11,s27,c & b), [a&d] => [(a#b)#c]]).
yes
| ?- cut_free(s28).

Proof is as follows :
=====================

s1=[id(a),[a]=>[a]]
s2=[th(s1,d),[a,d]=>[a]]
s3=[aia(s2,a&d),[a&d]=>[a]]
s4=[f,[a]=>[c]]
s5=[cut(s3,s4,a),[a&d]=>[c]]
s6=[id(d),[d]=>[d]]
s7=[th(s6,a),[a,d]=>[d]]
s8=[aia(s7,a&d),[a&d]=>[d]]
s9=[g,[d]=>[b]]
s10=[cut(s8,s9,d),[a&d]=>[b]]
s11=[ais(s5,s10,c&b),[a&d]=>[c&b]]
s12=[id(b),[b]=>[b]]
s13=[th(s12,c),[c,b]=>[b]]
s14=[aia(s13,c&b),[c&b]=>[b]]
s15=[id(b),[b]=>[b]]
s16=[th(s15,a),[b]=>[a,b]]
s17=[ois(s16,a#b),[b]=>[a#b]]
s18=[id(a),[a]=>[a]]
s19=[th(s18,b),[a]=>[a,b]]
s20=[id(b),[b]=>[b]]
s21=[th(s20,a),[b]=>[a,b]]
s22=[oia(s19,s21,a#b),[a#b]=>[a,b]]
s23=[ois(s22,a#b),[a#b]=>[a#b]]
s24=[th(s23,c),[a#b]=>[a#b,c]]
s25=[ois(s24,(a#b)#c),[a#b]=>[(a#b)#c]]
s26=[cut(s17,s25,a#b),[b]=>[(a#b)#c]]
s27=[cut(s14,s26,b),[c&b]=>[(a#b)#c]]
s28=[cut(s11,s27,c&b),[a&d]=>[(a#b)#c]]

Cut_free proof is :
===================

f115=[g,[d]=>[b]]
f112=[th(f115,a),[a,d]=>[b]]
```

```
f122=[aia(f112,a&d),[a&d]=>[b]]
f121=[th(f122,a),[a,a&d]=>[b]]
f118=[th(f121,d),[a,d,a&d]=>[b]]
f104=[aia(f118,a&d),[a&d,a&d]=>[b]]
f99=[con(f104,a&d),[a&d]=>[b]]
f96=[th(f99,a),[a&d]=>[a,b]]
f93=[th(f96,b),[a&d]=>[a,b,b]]
f90=[th(f93,a),[a&d]=>[a,b,a,b]]
f87=[inc(f90,a,b),[a&d]=>[a,a,b,b]]
f84=[con(f87,a),[a&d]=>[a,b,b]]
f81=[con(f84,b),[a&d]=>[a,b]]
f78=[ois(f81,a#b),[a&d]=>[a#b]]
f75=[th(f78,c),[a&d]=>[a#b,c]]
s28=[ois(f75,(a#b)#c),[a&d]=>[(a#b)#c]]


yes
| ?-
```

which in tree form is :

```
                  g
                d -> b
              ---------
              a,d -> b
            -----------
            a ↑ d -> b
          -------------
          a,a ↑ d -> b
        ---------------
        a,d,a ↑ d -> b
      -----------------
      a ↑ d,a ↑ d -> b
      -----------------
            a ↑ d -> b
            -------------
            a ↑ d -> a,b
            ---------------
            a ↑ d -> a,b,b
            -----------------
            a ↑ d -> a,b,a,b
            -----------------
            a ↑ d -> a,a,b,b
            -----------------
            a ↑ d -> a,b,b
            ---------------
            a ↑ d -> a,b
            ---------------
            a ↑ d -> a v b
            -----------------
            a ↑ d -> a v b,c
            ---------------------
            a ↑ d -> (a v b) v c
```

Notation : =~ denotes the isomorphic relation between the objects.

(2.5.3) Lemma

$$A \;=\!\sim\; A \vee \perp \;=\!\sim\; A \uparrow T \quad \text{for all } A \text{ in } ObFbc(X).$$

(2.5.4) Lemma

The objects $\perp \uparrow \perp$ and $T \vee T$ are neither initial nor terminal objects of Fbc(X).

(2.5.5) Corollary

The initial and terminal objects of Fbc(X) are characterized by the following properties :

(1) If $a =\sim T$ and $b =\sim \perp$, then T and $\perp$ are the only atomic subformula of a and b.

(2) $a \uparrow b =\sim T$ iff $a =\sim b =\sim T$.

(3) $a \vee b =\sim T$ iff $a(b) =\sim T$ and $b(a) =\sim \perp$.

(4) $a \uparrow b =\sim \perp$ iff $a(b) =\sim T$ and $b(a) =\sim \perp$.

(5) $a \vee b =\sim \perp$ iff $a =\sim b =\sim \perp$.

(2.5.6) <u>Corollary</u>

If $a =\sim T$, there exists a unique derivation p of $\to a$ consisting at most of instances of (A2), (R5), (R10), and (R13).

<u>Proof:</u>

By induction on the construction of a:

i) $a = T$ | as in (1.5.3)

ii) $a = c \uparrow d$ |

iii) $a = c \vee d$

1) $c =\sim T$ and $d =\sim \perp$

$$\cfrac{\cfrac{\overset{q}{\to c}}{\to c, d}}{\to c \vee d}$$

2) $c =\sim \perp$ and $d =\sim T$

$$\cfrac{\cfrac{\overset{r}{\to d}}{\to c, d}}{\to c \vee d}$$

###

(2.5.7) <u>Corollary</u>

If $b =\sim \perp$, there exists a unique derivation p of $b \to$ consisting at most of instances of (A3), (R2), (R11), and (R12).

<u>Proof:</u>

By induction on the construction of b:

i) $b = \perp$

$$\to \perp$$

ii) $b = c \uparrow d$

1) $c =\sim T$ and $d =\sim \perp$

$$\cfrac{\cfrac{d \overset{q}{\to}}{c, d \to}}{c \uparrow d \to}$$

2) $c = \sim \bot$ and $d = \sim T$

$$\frac{\dfrac{c \overset{r}{\rightarrow}}{c, d \rightarrow}}{c \uparrow d \rightarrow}$$

iii) $b = c \vee d$ where $c = \sim d = \sim \bot$

$$\frac{c \overset{q}{\rightarrow} \qquad d \overset{r}{\rightarrow}}{c \vee d \rightarrow}$$

###

(2.5.8) Lemma

For every cut-free p in Dbc(X) there exists an equivalent cut-free q in Dbc(X) containing no subderivations constructed with (R3),(R4),(R6) or (R7), and no subderivations constructed with (R2) or (R5) whose active formulae are of the form $a \uparrow b$ or $a \vee b$ respectively.

Proof

By Lemma (1.5.4) and the following clauses of the algorithm in appendix (3) :
(E.4), (E.5), (E.7), (E.8), (E.12), (E.15) , (E.16) , (E.20) , (E.23), (E.24), (E.25), (E.26), (E.28), (E.30), (E.33), (E.35), (E.38) and the special cases of the clauses (E.29) and (E.34) in which G and D are empty sequences, of the the same algorithm in Appendix (3); every derivation of Dbc(X) containing subderivations constructed with (R3), (R4), (R6) or (R7) reduces to contraction and interchange-free one .
By clauses (E.41.1) and (E.42.2) of Appendix (3), every derivation of Dbc(X) containing subderivation constructed with (R2) or (R5) with active formula of the form $a \uparrow b$ or $a \vee b$ respectively reduces to an equivalent one where there is no instances of (R2) or (R5) with active formula of the form $a \uparrow b$ or $a \vee b$ respectively .

###

The following example for some derivation of Dbc(X) which have the contraction and interchange rules in its construction, is to illustrate the above lemma .

Example

The cut_free derivation resulted from Example of (2.5.2) is transformed to a contraction and interchange free one as follows :

```
yes
| ?- theory(bicart).
yes
| ?- prove(s1 = [g, [d] => [b]]).
yes
| ?- prove(s2 = [th(s1,a), [a,d] => [b]]).
yes
| ?- prove(s3 = [aia(s2,a & d), [a & d] => [b]]).
yes
| ?- prove(s4 = [th(s3,a), [a,a & d] => [b]]).
yes
| ?- prove(s5 = [th(s4,d), [a,d,a & d] => [b]]).
yes
| ?- prove(s6 = [aia(s5,a & d), [a & d,a & d] => [b]]).
yes
| ?- prove(s7 = [con(s6,a & d), [a & d] => [b]]).
yes
| ?- prove(s8 = [th(s7,a), [a & d] => [a,b]]).
yes
| ?- prove(s9 = [th(s8,b), [a & d] => [a,b,b]]).
yes
| ?- prove(s10 = [th(s9,a), [a & d] => [a,b,a,b]]).
yes
| ?- prove(s11 = [inc(s10,a,b), [a & d] => [a,a,b,b]]).
yes
| ?- prove(s12 = [con(s11,a), [a & d] => [a,b,b]]).
yes
| ?- prove(s13 = [con(s12,b), [a & d] => [a,b]]).
yes
| ?- prove(s14 = [ois(s13,a # b), [a & d] => [a # b]]).
yes
| ?- prove(s15 = [th(s14,c), [a & d] => [a # b,c]]).
yes
| ?- prove(s16 = [ois(s15,(a#b)#c), [a&d] => [(a#b)#c]]).
yes
| ?- con_inc_free(s16).


Proof is as follows :
====================

s1=[g,[d]=>[b]]
s2=[th(s1,a),[a,d]=>[b]]
s3=[aia(s2,a&d),[a&d]=>[b]]
s4=[th(s3,a),[a,a&d]=>[b]]
s5=[th(s4,d),[a,d,a&d]=>[b]]
s6=[aia(s5,a&d),[a&d,a&d]=>[b]]
s7=[con(s6,a&d),[a&d]=>[b]]
s8=[th(s7,a),[a&d]=>[a,b]]
s9=[th(s8,b),[a&d]=>[a,b,b]]
s10=[th(s9,a),[a&d]=>[a,b,a,b]]
s11=[inc(s10,a,b),[a&d]=>[a,a,b,b]]
s12=[con(s11,a),[a&d]=>[a,b,b]]
s13=[con(s12,b),[a&d]=>[a,b]]
s14=[ois(s13,a#b),[a&d]=>[a#b]]
s15=[th(s14,c),[a&d]=>[a#b,c]]
s16=[ois(s15,(a#b)#c),[a&d]=>[(a#b)#c]]

Contraction and interchange free proof is :
===========================================

s1=[g,[d]=>[b]]
s2=[th(s1,a),[a,d]=>[b]]
s7=[aia(s2,a&d),[a&d]=>[b]]
s13=[th(s7,a),[a&d]=>[a,b]]
s14=[ois(s13,a#b),[a&d]=>[a#b]]
s15=[th(s14,c),[a&d]=>[a#b,c]]
s16=[ois(s15,(a#b)#c),[a&d]=>[(a#b)#c]]


yes
| ?-
```

which in tree form is :

$$\begin{array}{c} d \xrightarrow{g} b \\ \hline a,d \to b \\ \hline a \uparrow d \to b \\ \hline a \uparrow d \to a,b \\ \hline a \uparrow d \to a \vee b \\ \hline a \uparrow d \to a \vee b,c \\ \hline a \uparrow d \to (a \vee b) \vee c \end{array}$$

(2.5.9) <u>Definition</u>

A derivation p of Dbc(X) is <u>normal</u> if it satisfies the following conditions :-

(1) p contains no subderivations constructed with (R1), (R3), (R6), (R4) or (R7) .

(2) If p derives G -> F , and if one of the disjunctions of the formulae of F is isomorphic to T, then p is of the form

$$\frac{\xrightarrow{q} F}{G \to F}\ (K)$$

where q is the unique derivation of -> F compatible with corollary (2.5.6), and where (K) consists of the instances of (R2) required to derive G -> F from -> F.

(3) If p derives G -> F , and if one of the conjunctions of the formulae of G is isomorphic to , and no disjunctions of the formulae of F is isomorphic to T, then p is of the form

$$\frac{G \xrightarrow{q}}{G \to \quad F}\ (K)$$

where q is the unique derivation of G -> compatible with corollary (2.5.7), and where (K) consists of the instances of (R5) required to derive G -> F from G -> .

(4) p contains no subderivations constructed with (R2) or (R5) whose active formulae are of the form $a \uparrow b$ or $a \vee b$ respectively .

(5) Using the relation =< as defined in (1.5.5) ; for normality in Dbc(X) it is required that

(R5) =< (R13) =< (R10) =< (R12) =< (R11) =< (R2)

(2.5.10) <u>The Normalization Theorem for Dbc(X)</u>

For every derivation p in Dbc(X) there is a unique equivalent normal derivation q in Dbc(X) .

<u>Proof</u>

We extend the Normalization theorem (1.5.6) by the Cut-Elimination theorem (2.5.2), the above Lemmas and corollaries and the following additional clauses of Appendix (4) :
(D.4), (D.10), (D.34), (D.36), (D.38), (D.55), (D.62) and the special cases of clauses (D.37) and (D.59) with G and D are empty sequences; this proves the theorem .

###

For a derivation p of Dbc(X), we can use the implementation to get its equivalent normal form as follows :

<u>Example</u>

For f in X(a,c), g in X(b,d), h in X(c,b), k in X(d,c), i1 in X(a,b) = comp(h,f) , i2 in X(c,d) = comp(g,h) , i3 in X(b,c) = comp(k,g) , i4 in X(d,b) = comp(h,k), where a,b,c,d in Ob(X), the derivation

```
                                                                   b -> b          c -> c
                                                                  ---------       ---------
                                                                   b -> b,c        c -> b,c
                                                              h   ---------   k   ---------
  a -> a            b -> b                     c -> c       c->b  b -> bvc  d->c  c -> bvc
----------        ----------                  ----------   ----------------  ----------------
a,b -> a          a,b -> b          c -> c    c -> c,d       c -> b v c        d -> b v c
----------  f     ----------  g    ----------  ----------  ------------------------------------
a↑b -> a   a->c   a↑b -> b   b->d  c,d -> c    c -> cvd            c v d -> b v c
-----------------  ----------------  ----------  -------------------------------------------
  a↑b -> c           a↑b -> d        c↑d -> c              c -> b v c
-------------------------------------  ----------------------------------------------------
        a ↑ b -> c ↑ d                                c ↑ d -> b v c
-------------------------------------------------------------------------------------------
                              a ↑ b -----------> b v c
```

is transformed to its equivalent normal form as follows :

```
yes
| ?- theory(bicart).
yes
| ?- composite(h,f,i1).
yes
| ?- composite(g,h,i2).
yes
| ?- composite(k,g,i3).
yes
| ?- composite(h,k,i4).
yes
| ?- prove(g1 = [id(a), [a] => [a]]).
yes
| ?- prove(g2 = [th(g1,b), [a,b] => [a]]).
yes
| ?- prove(g3 = [ala(g2,a & b), [a & b] => [a]]).
yes
```

```
| ?- prove(g4 = [f, [a] => [c]]).
yes
| ?- prove(g5 = [cut(g3,g4,a), [a & b] => [c]]).
yes
| ?- prove(g6 = [id(b), [b] => [b]]).
yes
| ?- prove(g7 = [th(g6,a), [a,b] => [b]]).
yes
| ?- prove(g8 = [ala(g7,a & b), [a & b] => [b]]).
yes
| ?- prove(g9 = [g, [b] => [d]]).
yes
| ?- prove(g10 = [cut(g8,g9,b), [a & b] => [d]]).
yes
| ?- prove(g11 = [ais(g5,g10,c & d), [a & b] => [c & d]]).
yes
| ?- prove(g12 = [id(c), [c] => [c]]).
yes
| ?- prove(g13 = [th(g12,d), [c,d] => [c]]).
yes
| ?- prove(g14 = [ala(g13,c & d), [c & d] => [c]]).
yes
| ?- prove(g15 = [id(c), [c] => [c]]).
yes
| ?- prove(g16 = [th(g15,d), [c] => [c,d]]).
yes
| ?- prove(g17 = [ois(g16,c # d), [c] => [c # d]]).
yes
| ?- prove(g18 = [h, [c] => [b]]).
yes
| ?- prove(g19 = [id(b), [b] => [b]]).
yes
| ?- prove(g20 = [th(g19,c), [b] => [b,c]]).
yes
| ?- prove(g21 = [ois(g20,b # c), [b] => [b # c]]).
yes
| ?- prove(g22 = [cut(g18,g21,b), [c] => [b # c]]).
yes
| ?- prove(g23 = [k, [d] => [c]]).
yes
| ?- prove(g24 = [id(c), [c] => [c]]).
yes
| ?- prove(g25 = [th(g24,b), [c] => [b,c]]).
yes
| ?- prove(g26 = [ois(g25,b # c), [c] => [b # c]]).
yes
| ?- prove(g27 = [cut(g23,g26,c), [d] => [b # c]]).
yes
| ?- prove(g28 = [oia(g22,g27,c # d), [c # d] => [b # c]]).
yes
| ?- prove(g29 = [cut(g17,g28,c # d), [c] => [b # c]]).
yes
| ?- prove(g30 = [cut(g14,g29,c), [c & d] => [b # c]]).
yes
| ?- prove(g31 = [cut(g11,g30,c & d), [a & b] => [b # c]]).
yes
| ?- normal(g31).

Proof is as follows :
=====================

g1=[id(a),[a]=>[a]]
g2=[th(g1,b),[a,b]=>[a]]
g3=[ala(g2,a&b),[a&b]=>[a]]
g4=[f,[a]=>[c]]
g5=[cut(g3,g4,a),[a&b]=>[c]]
g6=[id(b),[b]=>[b]]
g7=[th(g6,a),[a,b]=>[b]]
g8=[ala(g7,a&b),[a&b]=>[b]]
g9=[g,[b]=>[d]]
g10=[cut(g8,g9,b),[a&b]=>[d]]
g11=[ais(g5,g10,c&d),[a&b]=>[c&d]]
g12=[id(c),[c]=>[c]]
g13=[th(g12,d),[c,d]=>[c]]
g14=[ala(g13,c&d),[c&d]=>[c]]
g15=[id(c),[c]=>[c]]
g16=[th(g15,d),[c]=>[c,d]]
g17=[ois(g16,c#d),[c]=>[c#d]]
g18=[h,[c]=>[b]]
g19=[id(b),[b]=>[b]]
g20=[th(g19,c),[b]=>[b,c]]
```

```
g21=[ois(g20,b#c),[b]=>[b#c]]
g22=[cut(g18,g21,b),[c]=>[b#c]]
g23=[k,[d]=>[c]]
g24=[id(c),[c]=>[c]]
g25=[th(g24,b),[c]=>[b,c]]
g26=[ois(g25,b#c),[c]=>[b#c]]
g27=[cut(g23,g26,c),[d]=>[b#c]]
g28=[oia(g22,g27,c#d),[c#d]=>[b#c]]
g29=[cut(g17,g28,c#d),[c]=>[b#c]]
g30=[cut(g14,g29,c),[c&d]=>[b#c]]
g31=[cut(g11,g30,c&d),[a&b]=>[b#c]]

Cut_free proof is :
==================

f124=[i1,[a]=>[b]]
f121=[th(f124,b),[a,b]=>[b]]
f118=[aia(f121,a&b),[a&b]=>[b]]
f115=[th(f118,b),[a&b,b]=>[b]]
f112=[th(f115,a),[a&b,a,b]=>[b]]
f100=[aia(f112,a&b),[a&b,a&b]=>[b]]
f95=[con(f100,a&b),[a&b]=>[b]]
f92=[th(f95,c),[a&b]=>[b,c]]
f89=[th(f92,c),[a&b]=>[b,c,c]]
f86=[ois(f89,b#c),[a&b]=>[b#c,c]]
f83=[th(f86,b),[a&b]=>[b#c,b,c]]
f80=[ois(f83,b#c),[a&b]=>[b#c,b#c]]
g31=[con(f80,b#c),[a&b]=>[b#c]]

Contraction and interchange free proof is :
============================================

f124=[i1,[a]=>[b]]
f121=[th(f124,b),[a,b]=>[b]]
f95=[aia(f121,a&b),[a&b]=>[b]]
f80=[th(f95,c),[a&b]=>[b,c]]
g31=[ois(f80,b#c),[a&b]=>[b#c]]

Normal proof is :
================

f124=[i1,[a]=>[b]]
f121=[th(f124,c),[a]=>[b,c]]
f95=[ois(f121,b#c),[a]=>[b#c]]
f80=[th(f95,b),[a,b]=>[b#c]]
g31=[aia(f80,a&b),[a&b]=>[b#c]]

yes
| ?-
```

which in tree form is :

```
          i1
       a -> b
      --------
       a -> b,c
      ----------
       a -> b v c
     ------------
     a,b -> b v c
    --------------
    a ↑ b -> b v c
```

(2.5.11) The CHURCH-ROSSER Theorem for Dbc(X)

If p == g, then there exists a normal r in Dbc(X) such that p >= r and q >= r.

(2.6) Application to Category Theory

In this section we illustrate the use of the system to represent some arrows of Fbc(X), to obtain their normal representations, and (where possible) to show the equality of arrows of Fbc(X).

Example 1

The normal representation of the following arrow

```
                           id(a) v tau(b v c)
          a v (b v c) ------------------------> a v T
```

is obtained as follows :

```
yes
| ?- theory(bicart).

yes
| ?- norm_rep(union(id(a),tau(b # c))).

The arrow is :
==============

union(id(a),tau(b#c))

Proof is as follows :
=====================

f2=[id(a),[a]=>[a]]
f7=[id(a),[a]=>[a]]
f6=[th(f7,tr),[a]=>[a,tr]]
f5=[ois(f6,a#tr),[a]=>[a#tr]]
f8=[cut(f2,f5,a),[a]=>[a#tr]]
f4=[true,[]=>[tr]]
f3=[th(f4,b#c),[b#c]=>[tr]]
f11=[id(tr),[tr]=>[tr]]
f10=[th(f11,a),[tr]=>[a,tr]]
f9=[ois(f10,a#tr),[tr]=>[a#tr]]
f12=[cut(f3,f9,tr),[b#c]=>[a#tr]]
f1=[oia(f8,f12,a#b#c),[a#b#c]=>[a#tr]]

Cut_free proof is :
===================

f18=[id(a),[a]=>[a]]
f15=[th(f18,tr),[a]=>[a,tr]]
f8=[ois(f15,a#tr),[a]=>[a#tr]]
f27=[true,[]=>[tr]]
f24=[th(f27,b#c),[b#c]=>[tr]]
f21=[th(f24,a),[b#c]=>[a,tr]]
f12=[ois(f21,a#tr),[b#c]=>[a#tr]]
f1=[oia(f8,f12,a#b#c),[a#b#c]=>[a#tr]]

Contraction and interchange free proof is :
===========================================

f28=[true,[]=>[tr]]
f30=[th(f28,a),[]=>[a,tr]]
f15=[th(f30,a),[a]=>[a,tr]]
f8=[ois(f15,a#tr),[a]=>[a#tr]]
f34=[true,[]=>[tr]]
f36=[th(f34,a),[]=>[a,tr]]
f37=[th(f36,b),[b]=>[a,tr]]
f40=[true,[]=>[tr]]
f39=[th(f40,a),[]=>[a,tr]]
f38=[th(f39,c),[c]=>[a,tr]]
f21=[oia(f37,f38,b#c),[b#c]=>[a,tr]]
f12=[ois(f21,a#tr),[b#c]=>[a#tr]]
f1=[oia(f8,f12,a#b#c),[a#b#c]=>[a#tr]]

Normal proof is :
=================

f28=[true,[]=>[tr]]
f30=[th(f28,a),[]=>[a,tr]]
f15=[ois(f30,a#tr),[]=>[a#tr]]
f8=[th(f15,a),[a]=>[a#tr]]
```

```
f34=[true,[]=>[tr]]
f36=[th(f34,a),[]=>[a,tr]]
f37=[ois(f36,a#tr),[]=>[a#tr]]
f21=[th(f37,b),[b]=>[a#tr]]
f40=[true,[]=>[tr]]
f39=[th(f40,a),[]=>[a,tr]]
f38=[ois(f39,a#tr),[]=>[a#tr]]
f41=[th(f38,c),[c]=>[a#tr]]
f12=[oia(f21,f41,b#c),[b#c]=>[a#tr]]
f1=[oia(f8,f12,a#b#c),[a#b#c]=>[a#tr]]


yes
| ?-
```

which in tree form is :

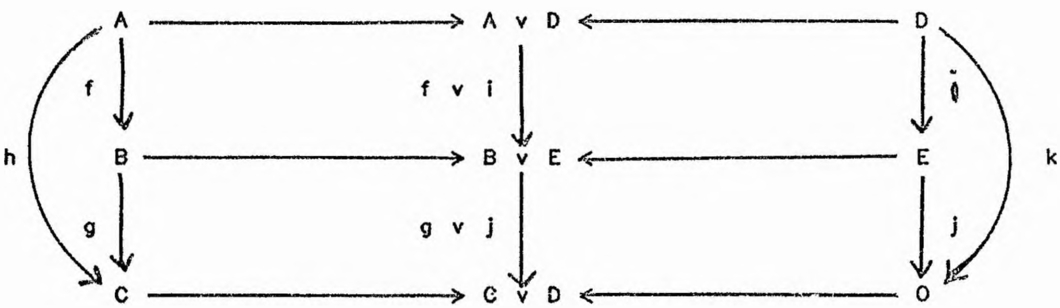
```
                                          -> T            -> T
                                    ----------       ---------
              -> T                    -> a,T          -> a,T
           --------             ----------         ----------
            -> a,T                -> a v T          -> a v T
       ----------               ----------         ----------
          -> a v T              b -> a v T         c -> a v T
       ----------               ------------------------------
        a -> a v T                     b v c -> a v T
       ----------------------------------------------
                     a v (b v c) -> a v T
```

Example 2

For f,g,h,i,j and k of Ar(X), by the use of the system we can prove

$$h \vee k = (g \vee j) \,.\, (f \vee i)$$

in the following diagram



as follows :

```
yes
?- theory(bicart).

yes
| ?- arrow(a,f,b).

yes
| ?- arrow(b,g,c).

yes
| ?- arrow(a,h,c).

yes
| ?- arrow(d,i,e).

yes
| ?- arrow(e,j,o).
```

```
yes
| ?- arrow(d,k,o).

yes
| ?- composite(g,f,h).

yes
| ?- composite(j,i,k).

yes
| ?- commutative.

Insert the first arrow : union(h,k).


The arrow is :
==============

union(h,k)


Proof is as follows :
=====================

f2=[h,[a]=>[c]]
f6=[id(c),[c]=>[c]]
f5=[th(f6,o),[c]=>[c,o]]
f4=[ois(f5,c#o),[c]=>[c#o]]
f7=[cut(f2,f4,c),[a]=>[c#o]]
f3=[k,[d]=>[o]]
f10=[id(o),[o]=>[o]]
f9=[th(f10,c),[o]=>[c,o]]
f8=[ois(f9,c#o),[o]=>[c#o]]
f11=[cut(f3,f8,o),[d]=>[c#o]]
f1=[oia(f7,f11,a#d),[a#d]=>[c#o]]


Cut_free proof is :
===================

f17=[h,[a]=>[c]]
f14=[th(f17,o),[a]=>[c,o]]
f7=[ois(f14,c#o),[a]=>[c#o]]
f23=[k,[d]=>[o]]
f20=[th(f23,c),[d]=>[c,o]]
f11=[ois(f20,c#o),[d]=>[c#o]]
f1=[oia(f7,f11,a#d),[a#d]=>[c#o]]


Contraction and interchange free proof is :
===========================================

f17=[h,[a]=>[c]]
f14=[th(f17,o),[a]=>[c,o]]
f7=[ois(f14,c#o),[a]=>[c#o]]
f23=[k,[d]=>[o]]
f20=[th(f23,c),[d]=>[c,o]]
f11=[ois(f20,c#o),[d]=>[c#o]]
f1=[oia(f7,f11,a#d),[a#d]=>[c#o]]


Normal proof is :
=================

f17=[h,[a]=>[c]]
f14=[th(f17,o),[a]=>[c,o]]
f7=[ois(f14,c#o),[a]=>[c#o]]
f23=[k,[d]=>[o]]
f20=[th(f23,c),[d]=>[c,o]]
f11=[ois(f20,c#o),[d]=>[c#o]]
f1=[oia(f7,f11,a#d),[a#d]=>[c#o]]

Insert the second arrow : comp(union(g,j),union(f,i)).


The arrow is :
==============

comp(union(g,j),union(f,i))
```

```
Proof is as follows :
=====================

f37=[f,[a]=>[b]]
f41=[id(b),[b]=>[b]]
f40=[th(f41,e),[b]=>[b,e]]
f39=[ols(f40,b#e),[b]=>[b#e]]
f42=[cut(f37,f39,b),[a]=>[b#e]]
f38=[i,[d]=>[e]]
f45=[id(e),[e]=>[e]]
f44=[th(f45,b),[e]=>[b,e]]
f43=[ols(f44,b#e),[e]=>[b#e]]
f46=[cut(f38,f43,e),[d]=>[b#e]]
f36=[ola(f42,f46,a#d),[a#d]=>[b#e]]
f26=[g,[b]=>[c]]
f30=[id(c),[c]=>[c]]
f29=[th(f30,o),[c]=>[c,o]]
f28=[ols(f29,c#o),[c]=>[c#o]]
f31=[cut(f26,f28,c),[b]=>[c#o]]
f27=[j,[e]=>[o]]
f34=[id(o),[o]=>[o]]
f33=[th(f34,c),[o]=>[c,o]]
f32=[ols(f33,c#o),[o]=>[c#o]]
f35=[cut(f27,f32,o),[e]=>[c#o]]
f25=[ola(f31,f35,b#e),[b#e]=>[c#o]]
f24=[cut(f36,f25,b#e),[a#d]=>[c#o]]


Cut_free proof is :
===================

f105=[h,[a]=>[c]]
f104=[th(f105,o),[a]=>[c,o]]
f101=[th(f104,o),[a]=>[c,o,o]]
f98=[ols(f101,c#o),[a]=>[c#o,o]]
f95=[th(f98,c),[a]=>[c#o,c,o]]
f81=[ols(f95,c#o),[a]=>[c#o,c#o]]
f82=[con(f81,c#o),[a]=>[c#o]]
f134=[k,[d]=>[o]]
f131=[th(f134,c),[d]=>[c,o]]
f128=[th(f131,o),[d]=>[c,o,o]]
f125=[ols(f128,c#o),[d]=>[c#o,o]]
f122=[th(f125,c),[d]=>[c#o,c,o]]
f110=[ols(f122,c#o),[d]=>[c#o,c#o]]
f111=[con(f110,c#o),[d]=>[c#o]]
f24=[ola(f82,f111,a#d),[a#d]=>[c#o]]

Contraction and interchange free proof is :
===========================================

f105=[h,[a]=>[c]]
f81=[th(f105,o),[a]=>[c,o]]
f82=[ols(f81,c#o),[a]=>[c#o]]
f128=[k,[d]=>[o]]
f110=[th(f128,c),[d]=>[c,o]]
f111=[ols(f110,c#o),[d]=>[c#o]]
f24=[ola(f82,f111,a#d),[a#d]=>[c#o]]

Normal proof is :
=================

f105=[h,[a]=>[c]]
f81=[th(f105,o),[a]=>[c,o]]
f82=[ols(f81,c#o),[a]=>[c#o]]
f128=[k,[d]=>[o]]
f110=[th(f128,c),[d]=>[c,o]]
f111=[ols(f110,c#o),[d]=>[c#o]]
f24=[ola(f82,f111,a#d),[a#d]=>[c#o]]


The answer is :
===============

union(h,k) =
comp(union(g,j),union(f,i))


because they have the same normal representation .

yes
| ?-
```

CHAPTER 3

DISTRIBUTIVE BICARTESIAN CATEGORY THEORY

In this chapter, we study the effect of the distributivity of $\uparrow$ over $\vee$; and define the class of small distributive bicartesian categories. We describe our representation of this class in Prolog, and the extensions to the implementation of bicartesian category theory required. Rules such as (R1) "cut rule", (R10) "and-introduction in the succedent " and (R12) " or-introduction in the antecedent " now become more general, and extra transformation rules are required in the cut-elimination and normalization algorithms. We describe these changes and their implementations.

Corollaries (3.5.5) and (3.5.6), concerning the terminal and initial objects in Fdbc(X), are stated without proof in SZABO; we have constructed and implemented their proofs in the system in order to be able to automate derivations containing terminal or initial objects. Lemma (3.5.7) is a modification of a similar lemma, stated without proof in SZABO, to meet the changes introduced in the previous chapters by adding (R4) and (R7); we have constructed and implemented the proof of this lemma in the system, in order to remove the contractions and interchanges from derivations in the class Ddbc(X) according the definition of the normal form (3.5.8).

(3.1) Definition

A distributive bicartesian category is a cartesian category with the following additional structure :

(4) A bifunctor $(-) \vee (-) : \underline{\underline{C}} \times \underline{\underline{C}} \to \underline{\underline{C}}$.

(5) A distinguished object $\perp \in \mathrm{Ob}\,\underline{\underline{C}}$.

(6) Two adjunctions α_δ and α_i , where

$$\alpha_\delta = \{\alpha_\delta(A,B,C,D) : \underline{\underline{C}}(A \uparrow (B \vee C),D) \to \underline{\underline{C}}(A \uparrow B,D) \times \underline{\underline{C}}(A \uparrow C,D) \in \mathrm{ArEns} \mid A,B,C,D \in \mathrm{Ob}\,\underline{\underline{C}}\}$$

and

$$\alpha_i = \{\alpha_i(A) : \underline{\underline{C}}(\perp,A) \to \{*\} \in \mathrm{ArEns} \mid A \in \mathrm{Ob}\,\underline{\underline{C}}\}.$$

(3.2) The Free Distributive Bicartesian Category Fdbc(X)

Let X be a fixed but arbitrary small category. The free distributive bicartesian category Fdbc(X) is constructed as follows :

(3.2.1) Definition

The class dbcL(X) of (distributive bicartesian) formulae over a category X is the same as the class bcL(X) of (bicartesian) formulae over X as defined in (2.2.1) .

(3.2.2) The labelled Derivations LDdbc(X)

The class of (distributive bicartesian) labelled derivations LDdbc(X) over a category X is an extension to the class LDbc(X) in (2.2.2) by the replacement of (xii) by

xii') for derivations F,G with conclusions f : A ↑ B -> D and g : A ↑ C -> D respectively

ROUND(F, G, A ↑ (B v C)) is a derivation with conclusion (f,g) : A ↑ (B v C) -> D .

(3.2.3) Definition

The relation == is defined as the smallest equivalence relation on LDdbc(X) satisfying the conditions of Definition (1.2.3) and the following additional conditions :

[Note : we use the notation

```
LA(A) = if A is an object of  X
            then  LA1(id(A))
        else  LA2(A) . ]
```

ix) If F == G and H == K with conclusions f,g : A ↑ B -> D and h,k : A ↑ C -> D

then

ROUND(F, H, A ↑ (B v C)) == ROUND(G, K, A ↑ (B v C)) ;

x) For F,G with conclusions f : A ↑ B -> D , g : A ↑ C -> D respectively

then

```
COMP( PRODUCT( LA(A), LA7(B,C) ),
      ROUND( F, G, A ↑ (B v C) ),
      A ↑ (B v C) )   == F
```

and

```
COMP( PRODUCT( LA(A), LA8(B,C) ),
      ROUND( F, G, A ↑ (B v C) ),
      A ↑ (B v C) )   == G          ;
```

xi) For F with conclusion f : A ↑ (B v C) -> D

ROUND(COMP(PRODUCT(LA(A), LA7(B,C)), F, A ↑ (B v C)),
COMP(PRODUCT(LA(A), LA8(B,C)), F, A ↑ (B v C)),
A ↑ (B v C)) == F ;

xii) For F with conclusion f : ⊥ -> A ,

F == LA6(A) .

(3.2.4) <u>Definition</u>

Fdbc(X) is defined to be the category having :

i) as objects , the distributive bicartesian formulae over X ;

ii) as arrows , the ==-classes of LDdbc(X) ;

iii) to xv) as in the definition of Fbc(X) ;

xvi) as (-,-) , the function defined by

([F],[G]) = [ROUND(F, G, A ↑ (B v C))]

where dom([F]) = A ↑ B ,

dom([G]) = A ↑ C and

cod([F]) = cod([G]) .

It is now routine to check that, with this structure, Fdbc(X) is a category, is distributive bicartesian, and is free over X. There is the obvious embedding X |-> Fdbc(X) defined by A |-> A; f |-> [LA1(f)].

(3.3) The Unlabelled Derivations Ddbc(X)

The class Ddbc(X) of unlabelled derivations (for representation of the free distributive bicartesian category over X) is defined as Dbc(X) in (2.3) with extra generality for the constructors (R1), (R10) and (R12) as follows :

iv') (cut)

if p1,p2 are derivations with conclusions G -> FaE, DaL -> H
then
R1(p1,p2,a) is a derivation with conclusion DGL -> FHE ;

xi') (and-introduction in the succedent)

if p1,p2 are derivations with conclusions G -> FaE , G -> FbE
then
R10(p1,p2,a ↑ b) is a derivation with conclusion G -> Fa ↑ bE;

xiii') (or-introduction in the antecedent)

if p1,p2 are derivations with conclusions GaD -> F , GbD -> F
then
R12(p1,p2,a v b) is a derivation with conclusion Ga v bD -> F.

The following is an example of a derivation in Ddbc(X), its representation in Prolog and the use of the implementation to check the correctness of this derivation :

<u>Example</u>

For objects a,b,c in the category X , the derivation

```
                                  c -> c
                                 ---------
        a -> a                  c,a -> c              a -> a
       ---------               ----------            ---------
        a -> a,b              c,a,b -> c              a -> a,b
       ----------            -------------           ----------
        a -> c,a,b           c,a ↑ b -> c             a -> c,a,b
       ---------------------------------             ------------
            a,a ↑ b -> c,a,b                          a,c -> c,a,b
            ----------------------------------------------------
                          a,(a ↑ b) v c -> c,a,b
```

is represented by

```
l  = [id(a), [a] => [a]]
l1 = [th(l,b), [a] => [a,b]]
l2 = [th(l1,c), [a] => [c,a,b]]
m  = [id(c), [c] => [c]]
m1 = [th(m,a), [c,a] => [c]]
m2 = [th(m1,b), [c,a,b] => [c]]
m3 = [ala(m2,a & b), [c,a & b] => [c]]
n  = [cut(l2,m3,c), [a,a & b] => [c,a,b]]
o  = [th(l2,c), [a,c] => [c,a,b]]
p  = [ola(n,o,(a & b) # c), [a,(a & b) # c] => [c,a,b]]
```

and the following is a session with the system to check the correctness of the above derivation :

```
*------------------------------------------------------------------------*
|        yes                                                             |
|        | ?- theory(dbicart).                                           |
|        yes                                                             |
|        | ?- prove(l = [id(a), [a] => [a]]).                            |
|        yes                                                             |
|        | ?- prove(l1 = [th(l,b), [a] => [a,b]]).                       |
|        yes                                                             |
|        | ?- prove(l2 = [th(l,c), [a] => [c,a,b]]).                     |
|        yes                                                             |
|        | ?- prove(m = [id(c), [c] => [c]]).                            |
|        yes                                                             |
|        | ?- prove(m1 = [th(m,a), [c,a] => [c]]).                       |
|        yes                                                             |
|        | ?- prove(m2 = [th(m1,b), [c,a,b] => [c]]).                    |
|        yes                                                             |
|        | ?- prove(m3 = [ala(m2,a & b), [c,a & b] => [c]]).             |
|        yes                                                             |
|        | ?- prove(n = [cut(l2,m3,c), [a,a & b] => [c,a,b]]).           |
|        yes                                                             |
|        | ?- prove(o = [th(l2,c), [a,c] => [c,a,b]]).                   |
|        yes                                                             |
|        | ?- prove(p = [ola(n,o,(a&b)#c), [a,(a&b)#c] => [c,a,b]]).     |
|        yes                                                             |
|        | ?-                                                            |
*------------------------------------------------------------------------*
```

(3.4) The Semantics of Ddbc(X)

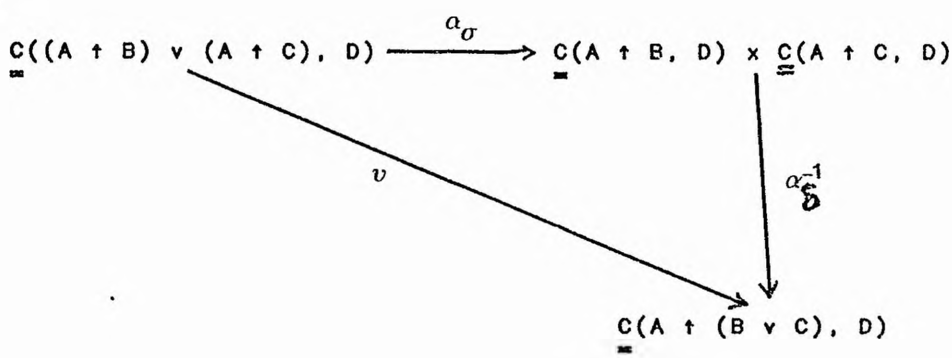
We now define an interpretation function to interpret the derivations of Ddbc(X) in ArFdbc(X) and prove the completeness theorem. For this purpose we extend the canonical arrows in (1.4) and (2.4) by the following canonical arrows of Fdbc(X) :

vii) delta_left(A,B,C) : A † (B v C) -> (A † B) v (A † C)

for all A,B,C in ObFdbc(X) , and where

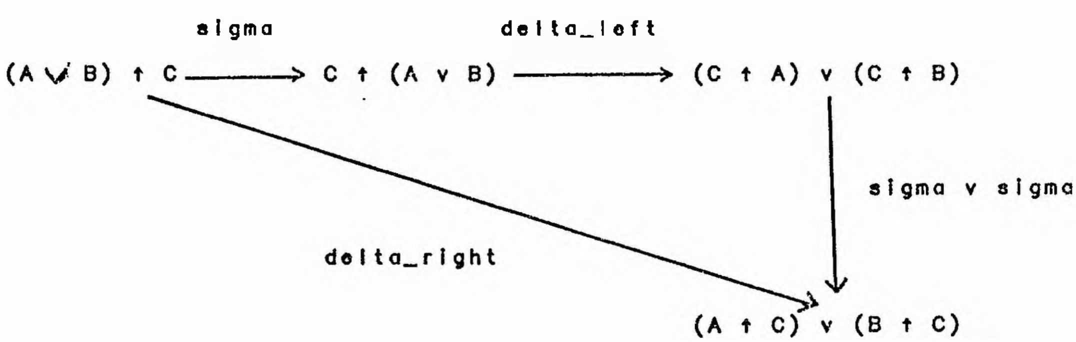
delta_left(A,B,C) = υ((A † B) v (A † C))(id((A † B) v (A † C))),

and where υ is the natural isomorphism determined by the compositions

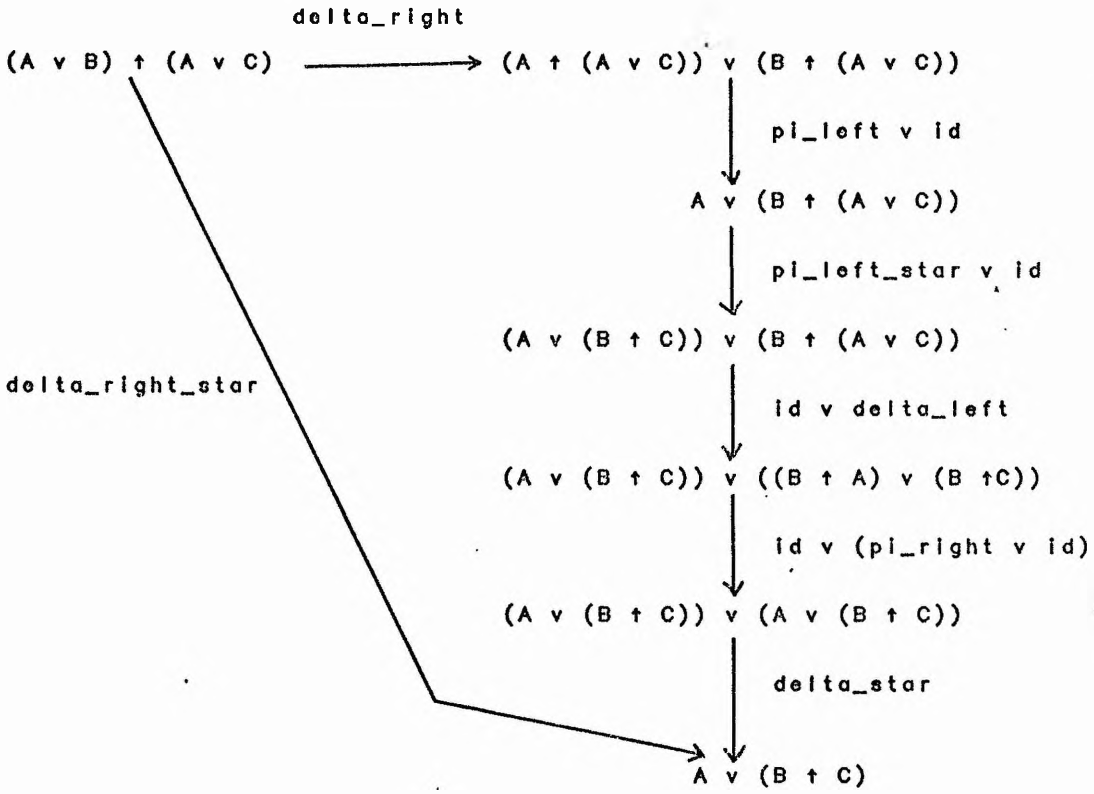


with $\underline{\underline{C}}$ = Fdbc(X) .

viii) delta_right(A,B,C) : (A v B) † C -> (A † C) v (B † C)
for all A,B,C in ObFdbc(X), with delta_right defined by the compositions

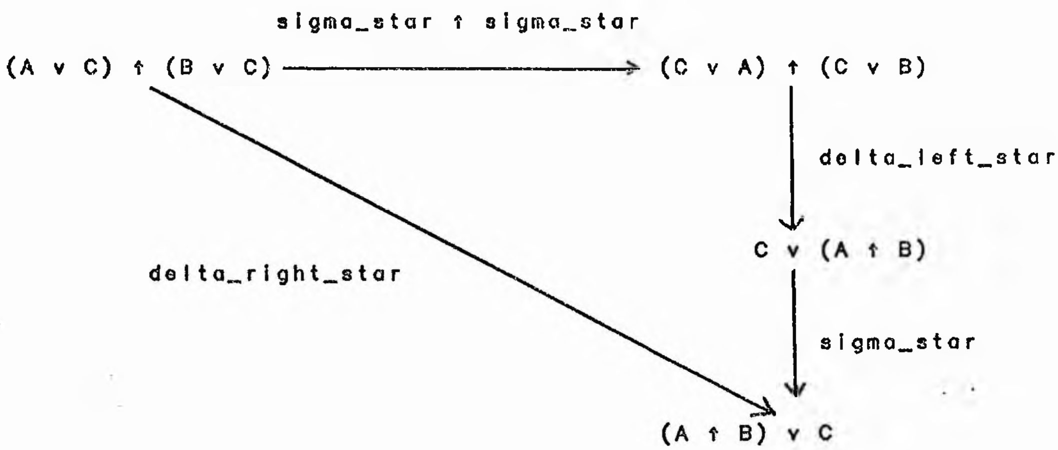


ix) delta_left_star(A,B,C) : (A v B) † (A v C) -> A v (B † C)
for all A,B,C in ObFdbc(X), with delta_left_star defined by the compositions



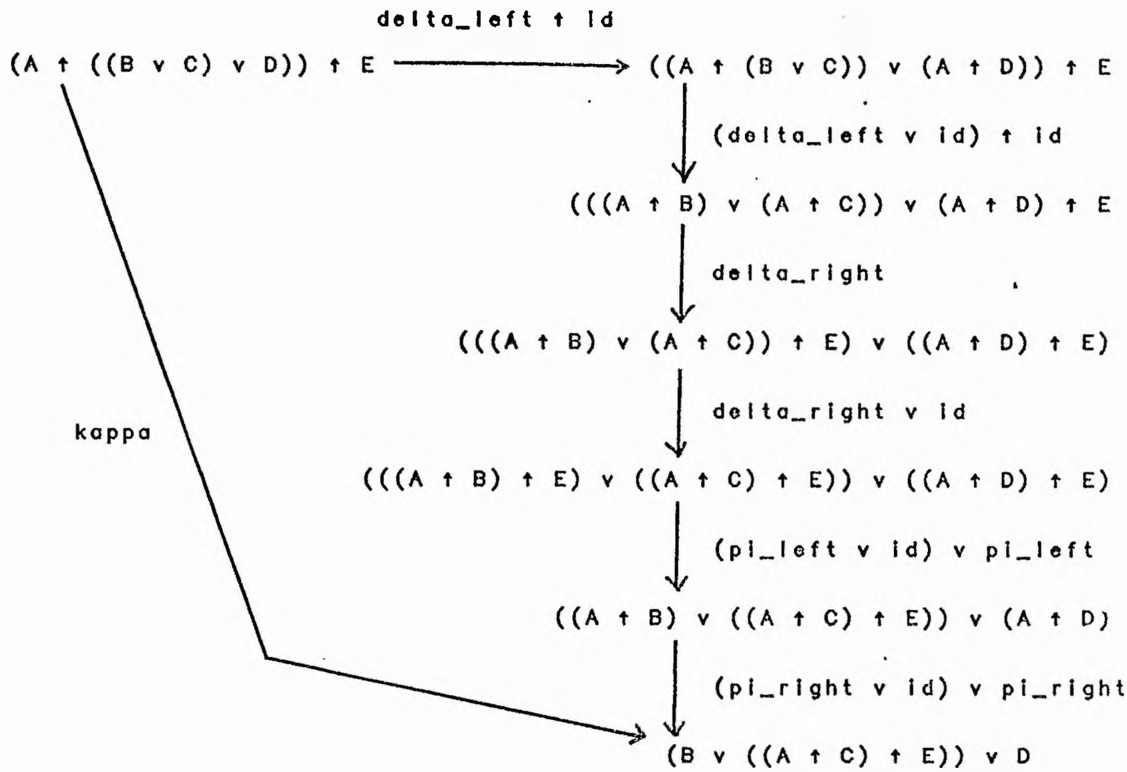
x) delta_right_star(A,B,C) : (A v C) ↑ (B v C) -> (A ↑ B) v C

for all A,B,C in ObFdbc(X), with delta_right_star defined by the compositions



xi) kappa(A,B,C,D,E) : (A ↑ ((B v C) v D)) ↑ E ->
(B v ((A ↑ C) ↑ E)) v D

for all A,B,C,D,E in ObFdbc(X), with kappa defined by the compositions



(3.4.1) <u>Definition</u>

We extend the interpretation function S defined in (2.4.1) to interpret derivations of Ddbc(X) in ArFdbc(X) and satisfy the following generalizations of conditions (3),(10),(7) and (14) mentioned in (1.4.1) and (2.4.1) :

```
10') S(R1(p1,p2,a)     = comp(comp(( I1 v S(p2)) v I2,
                                 kappa(PI(D),Un(F),a,Un(E),PI(L))),
                             (I3 ↑ S(p1)) ↑ I4 )  ,   where
                         I1 is the identity on Un(F) ,
                         I2 is the identity on Un(E) ,
                         I3 is the identity on PI(D) ,
                         I4 is the identity on PI(L) and
                         G,D,L,a,F,H,E are as in (3.3) condition iv').

7') S(R10(p1,p2,a ↑ b) = comp(comp(comp((delta_left_star(Un(F),a,b) v I),
                                    delta_right_star(Un(F) v a,
                                                      Un(F) v b,Un(E))),
                                 S(p1) ↑ S(p2) ),
                             delta(PI(G)) )  ,         where
                         I is the identity on Un(E) and
                         G,F,a,b,E are as in (3.3) condition (xi').
```

```
14') S(R12(p1,p2,a v b)= comp(comp(comp(delta_star(Un(F)),
                                  S(p1) v S(p2) ),
                         delta_right(Pi(G) ↑ a,Pi(G) ↑ b,Pi(D))),
                     (delta_left(Pi(G),a,b) ↑ I) ),    where
                 I is the identity over Pi(D) , and
                 G,a,b,D,F being as in (3.3) condition xiii').
```

The following example is a session with the system to interpret a derivation of the class Ddbc(X) in ArFdbc(X) :

Example

Let a,b be objects of the category X, f in X(b,a); then the derivation

```
a -> a
--------
a -> a,d                                a -> a
----------          f                   --------
a -> b,a,d        b -> a                a -> b,a
------------------------                ----------
      a -> a,a,d                        a -> b,a,d
      ------------                      ------------
      a -> a,a v d                      a -> b,a v d
      --------------------------------------------
                  a -> a ↑ b, a v d
                  ---------------------
                  a -> (a ↑ b) v (a v d)
```

is interpreted in ArFdbc(X) as follows :

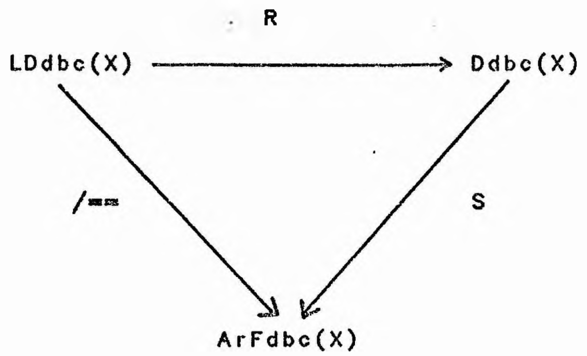
```
yes
| ?- theory(dbicart).

yes
| ?- prove(i = [id(a), [a] => [a]]).

yes
| ?- prove(i1 = [th(i,d), [a] => [a,d]]).

yes
| ?- prove(i2 = [th(i1,b), [a] => [b,a,d]]).

yes
| ?- prove(m = [f, [b] => [a]]).

yes
| ?- prove(n = [cut(i2,m,b), [a] => [a,a,d]]).

yes
| ?- prove(n1 = [ois(n,a # d), [a] => [a,a # d]]).

yes
| ?- prove(o = [th(i,b), [a] => [b,a]]).

yes
| ?- prove(o1 = [th(o,d), [a] => [b,a,d]]).

yes
| ?- prove(o2 = [ois(o1,a # d), [a] => [b,a # d]]).

yes
| ?- prove(p = [ais(n1,o2,a & b), [a] => [a&b,a#d]]).

yes
| ?- prove(q = [ois(p,(a&b)#(a#d)), [a] => [(a&b)#(a#d)]]).
```

```
yes
| ?- interpr(q,H).

H = comp(comp(delta_right_star(a,b,a#d),
              product(comp(alpha_star_inverse(a,a,d),
                           comp(union(f,id(a#d)),
                                comp(pi_right_star(b,a#d),
                                     pi_left_star(a,d)))),
                      comp(alpha_star_inverse(b,a,d),
                           comp(pi_left_star(b#a,d),
                                pi_right_star(b,a))))),
         delta(a))

yes
| ?-
```

(3.4.2) <u>The Completeness Theorem for Ddbc(X)</u>

There is a function

R : LDdbc(X) -----------> Ddbc(X) , so that



commutes .

<u>Proof</u>

The proof is identical to the proof of the completeness theorem for Dbc(X) in (2.4.2) , except case (xii) is replaced by :

xii') when F is ROUND(G, H, A ↑ (B v C)), put

```
R(F) = R11( R12( R1( R10( R2( R( I(A) ), B ),
                          R2( R( I(B) ), A ), A ↑ B ),
                     R( G ) , A ↑ B ),
                 R1( R10( R2( R( I(A) ), C ),
                          R2( R( I(C) ), A ), A ↑ C ),
                     R( H ) , A ↑ C ),
                 B v C ),
            A ↑ (B v C) ) .
```

The following examples of some canonical arrows of Fdbc(X) and their representations (unlabelled derivations) are to illustrate the above theorem; they are also implemented as a primitive arrows in the system.

Example 1

For a derivation F with conclusion

```
delta_left(A,B,C) : A ↑ (B v C) -> (A ↑ B) v (A ↑ C)
```

the representation R(F) is

```
R( ROUND(LA7(A ↑ B, A ↑ C), LA8(A ↑ B, A ↑ C), A ↑ (B v C)))
```

Example 2

Foe a derivation F with conclusion

```
delta_right(A,B,C) : (A v B) ↑ C -> (A ↑ C) v (B ↑ C)
```

the representation R(F) is

```
R( COMP( UNION(F1, F2),
         COMP(F3, F4, C ↑ (A v B)),
         (C ↑ A) v (C ↑ B) ) )
```

where

F1, F2, F3 and F4 are derivations with conclusions

```
sigma(C,A) : C ↑ A -> A ↑ C ,
sigma(C,B) : C ↑ B -> B ↑ C ,
delta_left(C,A,B) : C ↑ (A v B) -> (C ↑ A) v (C ↑ B)      and
sigma(A v B,C) : (A v B) ↑ C -> C ↑ (A v B)     respectively .
```

Example 3

For a derivation F with conclusion

```
delta_left_star(A,B,C) : (A v B) ↑ (A v C) -> A v (B ↑ C)
```

the representation R(F) is

```
R(COMP(COMP(COMP(COMP(COMP(F1,
                           UNION(LA2(A v (B ↑ C)),
                                 UNION(LA5(B,A),LA2(B ↑ C))),
                           (A v (B ↑ C)) v (A v (B ↑ C)) ),
                      UNION( LA2(A v (B ↑ C)), F2 ),
                      (A v (B ↑ C)) v ((B ↑ A) v (B ↑ C)) ),
                 UNION( LA7(A, B ↑ C), LA2(B ↑ (A v C)) ),
                 (A v (B ↑ C)) v (B ↑ (A v C)) ),
            UNION( LA4(A, A v C), LA2(B ↑ (A v C)) ),
            A v (B ↑ (A v C)) ),
       F2,
       (A ↑ (A v C)) v (B ↑ (A v C)) ) )
```

where

F1 and F2 are derivations with conclusions

```
delta_star(Av(B↑C)) : (A v (B ↑ C)) v (A v (B ↑ C)) -> A v (B ↑ C)
```

and

```
delta_right(A,B,A v C) : (A v B) ↑ (A v C) -> (A ↑ (A v C)) v
                                              (B ↑ (A v C))
```

respectively .

Example 4

For a derivation F with conclusion

```
delta_right_star(A,B,C) : (A v B) ↑ (B v C) -> (A ↑ B) v C
```

the representation R(F) is

```
R( COMP( COMP( F1, F2, C v (A ↑ B) ),
               PRODUCT( F3, F4),
               (C v A) ↑ (C v B) ) )
```

where

F1, F2, F3 and F4 are derivations with conclusions

```
sigma_star(C,A ↑ B) : C v (A ↑ B) -> (A ↑ B) v C ,
delta_left_star(C,A,B) : (C v A) ↑ (C v B) -> C v (A ↑ B) ,
sigma_star(A,C) : A v C -> C v A              and
sigma_star(B,C) : B v C -> C v B
```

respectively .

Example 5

For a derivation F with conclusion

```
kappa(A,B,C,D,E) : (A ↑ ((B v C) v D)) ↑ E -> (B v ((A ↑ C) ↑ E)) v D
```

the representation R(F) is

```
R(COMP(COMP(COMP(COMP(COMP(UNION(UNION(LA5(A,B),
                                       LA2((A ↑ C) ↑ E)),
                                 LA5(A,D)),
                           UNION( UNION( LA4(A ↑ B, E)
                                         LA2((A ↑ C) ↑ D)),
                                  LA4(A ↑ D,E)),
                           ((A ↑ B) v ((A ↑ C) ↑ E)) v (A ↑ D)),
                      UNION( F1, LA2((A ↑ D) ↑ E)),
                      (((A ↑ B) ↑ E) v ((A ↑ C) ↑ E)) v ((A ↑ D) ↑ E)),
                  F2,
                  (((A ↑ B) v (A ↑ C)) ↑ E) v ((A ↑ D) ↑ E) ),
             PRODUCT( UNION( F3, LA2(A ↑ D)), LA2(E) ),
             (((A ↑ B) v (A ↑ C)) v (A ↑ D)) ↑ E ),
        PRODUCT( F4, LA2(E) ),
        ((A ↑ (B v C)) v (A ↑ D)) ↑ E ) )
```

where

F1, F2, F3 and F4 are derivations with conclusions

delta_right(A ↑ B,A ↑ C,E) : ((A ↑ B) v (A ↑ C) ↑ E ->
((A ↑ B) ↑ E) v ((A ↑ C) ↑ E) ,

delta_right((A ↑ B) v (A ↑ C), A ↑ D, E) :
(((A ↑ B) v (A ↑ C)) v (A ↑ D)) ↑ E ->
(((A ↑ B) v (A ↑ C)) ↑ E) v ((A ↑ D)'↑ E) ,

delta_left(A,B,C) : A ↑ (B v C) -> (A ↑ B) v (A ↑ C) and

delta_left(A,B v C,D) : A ↑ ((B v C) v D) -> (A ↑ (B v C)) v (A ↑ D)

respectively .

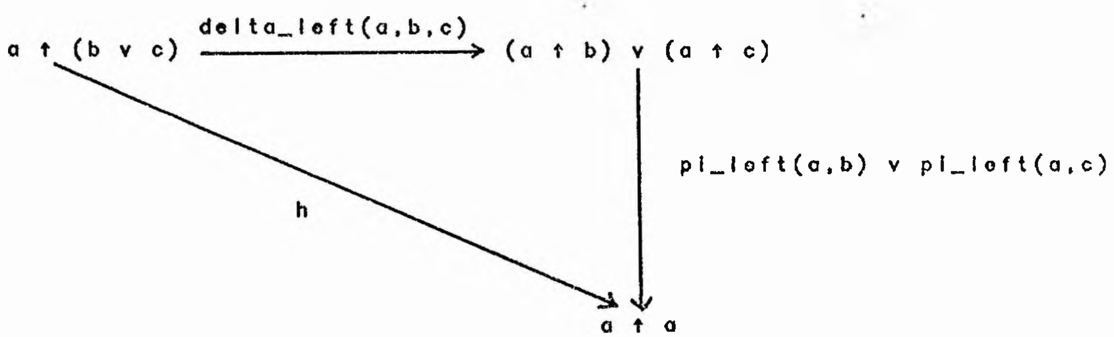
The following example is a session of input and output with the system to obtain an unlabelled distributive bicartesian representation for some arrow of Fdbc(X) :

Example 6

For objects a,b,c of the category X , the arrow

h = comp(union(pi_left(a,b),pi_left(a,c)),delta_left(a,b,c))

in the diagram



is represented by an unlabelled derivation as follows :

```
yes
| ?- theory(dbicart).

yes
| ?- rep_of(comp(union(pi_left(a,b),pi_left(a,c)),
                 delta_left(a,b,c))).


The arrow is :
==============

comp(union(pi_left(a,b),pi_left(a,c)),delta_left(a,b,c))


Representation :-
=================

f43=[id(a),[a]=>[a]]
f39=[th(f43,b),[a,b]=>[a]]
f44=[id(b),[b]=>[b]]
f40=[th(f44,a),[a,b]=>[b]]
```

```
f37=[ais(f39,f40,a&b),[a,b]=>[a&b]]
f24=[id(a),[a]=>[a]]
f22=[th(f24,b),[a,b]=>[a]]
f25=[id(b),[b]=>[b]]
f23=[th(f25,a),[a,b]=>[b]]
f21=[ais(f22,f23,a&b),[a,b]=>[a&b]]
f20=[aia(f21,a&b),[a&b]=>[a&b]]
f19=[th(f20,a&c),[a&b]=>[a&b,a&c]]
f18=[ois(f19,(a&b)#a&c),[a&b]=>[(a&b)#a&c]]
f35=[cut(f37,f18,a&b),[a,b]=>[(a&b)#a&c]]
f45=[id(a),[a]=>[a]]
f41=[th(f45,c),[a,c]=>[a]]
f46=[id(c),[c]=>[c]]
f42=[th(f46,a),[a,c]=>[c]]
f38=[ais(f41,f42,a&c),[a,c]=>[a&c]]
f32=[id(a),[a]=>[a]]
f30=[th(f32,c),[a,c]=>[a]]
f33=[id(c),[c]=>[c]]
f31=[th(f33,a),[a,c]=>[c]]
f29=[ais(f30,f31,a&c),[a,c]=>[a&c]]
f28=[aia(f29,a&c),[a&c]=>[a&c]]
f27=[th(f28,a&b),[a&c]=>[a&b,a&c]]
f26=[ois(f27,(a&b)#a&c),[a&c]=>[(a&b)#a&c]]
f36=[cut(f38,f26,a&c),[a,c]=>[(a&b)#a&c]]
f34=[oia(f35,f36,b#c),[a,b#c]=>[(a&b)#a&c]]
f17=[aia(f34,a&b#c),[a&b#c]=>[(a&b)#a&c]]
f5=[id(a),[a]=>[a]]
f4=[th(f5,b),[a,b]=>[a]]
f3=[aia(f4,a&b),[a&b]=>[a]]
f11=[id(a),[a]=>[a]]
f10=[th(f11,a),[a]=>[a,a]]
f9=[ois(f10,a#a),[a]=>[a#a]]
f12=[cut(f3,f9,a),[a&b]=>[a#a]]
f8=[id(a),[a]=>[a]]
f7=[th(f8,c),[a,c]=>[a]]
f6=[aia(f7,a&c),[a&c]=>[a]]
f15=[id(a),[a]=>[a]]
f14=[th(f15,a),[a]=>[a,a]]
f13=[ois(f14,a#a),[a]=>[a#a]]
f16=[cut(f6,f13,a),[a&c]=>[a#a]]
f2=[oia(f12,f16,(a&b)#a&c),[(a&b)#a&c]=>[a#a]]
f1=[cut(f17,f2,(a&b)#a&c),[a&b#c]=>[a#a]]

yes
| ?-
```

(3.5) The Syntax of Fdbc(X)

The advantages of Ddbc(X) over LDdbc(X) lie in the fact that the derivations of Ddbc(X) code their own labels and provide cut-free representations of ArFdbc(X) as follows .

(3.5.1) The Cut-Elimination Theorem for Ddbc(X)

Every derivation in Ddbc(X) is equivalent to a cut-free in Ddbc(X).

Proof

Because of the generality of rules (R1), (R10) and (R12) in Ddbc(X) we must review cases (C.3), (C.4), (C.8-9), (C.12), (C.14), (C.15), (C.18), (C.19), (C.20), (C.21), (C.22), (C.26), (C.27), (C.29), (C.34), (C.35), (C.36), (C.37), (C.38), (C.42), (C.43) and (C.44) mentioned in (1.5.2) and (2.5.2) to become general cases; the above clauses and the following additional ones of the same algorithm of appendix (2) : (C.28) and (C.41) prove the theorem .

###

(3.5.2) Lemma

$A \cong \sim A \vee \bot \cong \sim A \wedge T$ for all A in ObFdbc(X).

(3.5.3) Lemma

The object $T \vee T$ is neither an initial nor a terminal object of Fdbc(X).

(3.5.4) Corollary

The initial and terminal objects of Fdbc(X) are characterized by the following isomorphism conditions :

(1) $a \wedge b \cong\sim T$ iff $a \cong\sim b \cong\sim T$.

(2) $a \vee b \cong\sim T$ iff $a \cong\sim T$ and $b \cong\sim \bot$

or $b \cong\sim T$ and $a \cong\sim \bot$

(3) $a \wedge b \cong\sim \bot$ iff $a \cong\sim \bot$ or $b \cong\sim \bot$.

(4) $a \vee b \cong\sim \bot$ iff $a \cong\sim b \cong\sim \bot$.

(3.5.5) Corollary

If $a \cong\sim T$, there exists a unique derivation of $\rightarrow a$ consisting of instances of (A2), (R5), (R10), and (R13).

Proof:

Identical to the proof of corollary (2.5.6).

(3.5.6) Corollary

If $b \cong\sim \bot$, there exists a unique derivation of $b \rightarrow$ consisting at most of instances of (A3), (R2), (R11), and (R12).

Proof:

Identical to the proof of corollary (2.5.7) except in ii) c or $d \cong\sim \bot$ only.

(3.5.7) Lemma

For every cut-free derivation p in Ddbc(X) there exists an equivalent cut-free q in Ddbc(X) containing no subderivations constructed with (R3),(R4),(R6) or (R7); the subderivations constructed with instance of (R2) or (R5) , if any , must have atomic active formulae .

Proof

By (1.5.4) and (2.5.8) in the cartesian and bicartesian categories theories, after considering the general forms of the clauses (E.9), (E.17),(E.29) and (E.34), in order to meet the generality of rules (R10) and (R12) and the following additional clauses (E.11), (E.19), (E.27), (E.32), (E.41.2) and (E.42.1) in Appendix (3), we can prove the lemma .

###

(3.5.8) <u>Definition</u>

A derivation p in Ddbc(X) is defined to be <u>normal</u> if it is normal in the sense of definition (2.5.9) after replacing condition (4) by the more general form

4') p contains no subderivations constructed with (R2) or (R5) whose active formulae are of the form a ↑ b or a v b .

(3.5.9) <u>The Normalization Theorem for Ddbc(X)</u>

Every derivation p in Ddbc(X) is equivalent to a unique normal derivation q in Ddbc(X) .

<u>Proof</u>

The theorem follows from theorem (2.5.10) with clauses (D.7), (D.37), (D.49) and (D.59) generalized to allow non-empty sequences F,E, G and D, together with the following conditions of Appendix (4) : (D.9) , (D.35) , (D.48) , (D.50) , (D.51) , (D.54) and (D.58) .

###

The following examples are sessions of input and output with the system for derivations of Ddbc(X) and their equivalents of cut-free, contraction & interchange-free and normal form derivations; in order to illustrate the above processes and to satisfy the definition of normal form :

<u>Example 1</u>

For an arrow f in X(a,d), where a,b,c,d are objects of X , the derivation

```
                          f
    a -> a              a -> d
  ----------          ----------
  a,b -> a            a,c -> d                d -> d
  ------------        ------------          ---------
  a,b -> a,d          a,c -> a,d            d,b -> d                          b -> b
  ----------------------------------      -----------                   ------------
        a,b v c -> a,d                      c,d,b -> d                  c ↑ a,b -> b
  ----------------------------------------------------           ------------------------
              c,a,b v c,b -> a,d                                 c ↑ a,b v c,b -> b
        ------------------------------                           ------------------------
        c ↑ a,b v c,b -> a,d                                     c ↑ a,b v c,b -> b,d
        ---------------------------------------------------------------------------------
                                 c ↑ a, b v c, b -> a ↑ b , d
                          ------------------------------------------
                          (c ↑ a) ↑ (b v c), b -> a ↑ b , d
                          ------------------------------------------
                          (c ↑ a) ↑ (b v c), b -> (a ↑ b) v d
```

is transformed to its normal form as follows :

```
yes
| ?- theory(dblcart).
yes
| ?- prove(s1 = [id(a), [a] => [a]]).
yes
| ?- prove(s2 = [th(s1,b), [a,b] => [a]]).
yes
| ?- prove(s3 = [th(s2,d), [a,b] => [a,d]]).
yes
| ?- prove(s4 = [f, [a] => [d]]).
yes
| ?- prove(s5 = [th(s4,c), [a,c] => [d]]).
yes
| ?- prove(s6 = [th(s5,a), [a,c] => [a,d]]).
yes
| ?- prove(s7 = [ois(s3,s6,b # c), [a,b # c] => [a,d]]).
yes
| ?- prove(s8 = [id(d), [d] => [d]]).
yes
| ?- prove(s9 = [th(s8,b), [d,b] => [d]]).
yes
| ?- prove(s10 = [th(s9,c), [c,d,b] => [d]]).
yes
| ?- prove(s11 = [cut(s7,s10,d), [c,a,b#c,b] => [a,d]]).
yes
| ?- prove(s12 = [aia(s11,c&a), [c&a,b#c,b] => [a,d]]).
yes
| ?- prove(s13 = [id(b), [b] => [b]]).
yes
| ?- prove(s14 = [th(s13,c&a),[c&a,b] => [b]]).
yes
| ?- prove(s15 = [th(s14,b#c), [c&a,b#c,b] => [b]]).
yes
| ?- prove(s16 = [th(s15,d), [c&a,b#c,b] => [b,d]]).
yes
| ?- prove(s17 = [ais(s12,s16,a&b),[c&a,b#c,b] => [a&b,d]]).
yes
| ?- prove(s18=[aia(s17,(c&a)&(b#c)),[(c&a)&(b#c),b]=>[a&b,d]]).
yes
| ?- prove(s19 = [ois(s18,(a&b)#d),[(c&a)&(b#c),b]=>[(a&b)#d]]).
yes
| ?- normal(s19).

Proof is as follows :
=====================

s1=[id(a),[a]=>[a]]
s2=[th(s1,b),[a,b]=>[a]]
s3=[th(s2,d),[a,b]=>[a,d]]
s4=[f,[a]=>[d]]
s5=[th(s4,c),[a,c]=>[d]]
s6=[th(s5,a),[a,c]=>[a,d]]
s7=[oia(s3,s6,b#c),[a,b#c]=>[a,d]]
s8=[id(d),[d]=>[d]]
s9=[th(s8,b),[d,b]=>[d]]
s10=[th(s9,c),[c,d,b]=>[d]]
s11=[cut(s7,s10,d),[c,a,b#c,b]=>[a,d]]
s12=[aia(s11,c&a),[c&a,b#c,b]=>[a,d]]
s13=[id(b),[b]=>[b]]
s14=[th(s13,c&a),[c&a,b]=>[b]]
s15=[th(s14,b#c),[c&a,b#c,b]=>[b]]
s16=[th(s15,d),[c&a,b#c,b]=>[b,d]]
s17=[ais(s12,s16,a&b),[c&a,b#c,b]=>[a&b,d]]
s18=[aia(s17,(c&a)&b#c),[(c&a)&b#c,b]=>[a&b,d]]
s19=[ois(s18,(a&b)#d),[(c&a)&b#c,b]=>[(a&b)#d]]


Cut_free proof is :
===================

s1=[id(a),[a]=>[a]]
f13=[th(s1,b),[a,b]=>[a]]
f9=[th(f13,d),[a,b]=>[a,d]]
f19=[f,[a]=>[d]]
f16=[th(f19,c),[a,c]=>[d]]
```

```
f12=[th(f16,a),[a,c]=>[a,d]]
f6=[oia(f9,f12,b#c),[a,b#c]=>[a,d]]
f3=[th(f6,b),[a,b#c,b]=>[a,d]]
s11=[th(f3,c),[c,a,b#c,b]=>[a,d]]
s12=[aia(s11,c&a),[c&a,b#c,b]=>[a,d]]
s13=[id(b),[b]=>[b]]
s14=[th(s13,c&a),[c&a,b]=>[b]]
s15=[th(s14,b#c),[c&a,b#c,b]=>[b]]
s16=[th(s15,d),[c&a,b#c,b]=>[b,d]]
s17=[ais(s12,s16,a&b),[c&a,b#c,b]=>[a&b,d]]
s18=[aia(s17,(c&a)&b#c),[(c&a)&b#c,b]=>[a&b,d]]
s19=[ois(s18,(a&b)#d),[(c&a)&b#c,b]=>[(a&b)#d]]


Contraction and Interchange free proof is :
===========================================

s1=[id(a),[a]=>[a]]
f13=[th(s1,b),[a,b]=>[a]]
f9=[th(f13,d),[a,b]=>[a,d]]
f19=[f,[a]=>[d]]
f16=[th(f19,c),[a,c]=>[d]]
f12=[th(f16,a),[a,c]=>[a,d]]
f6=[oia(f9,f12,b#c),[a,b#c]=>[a,d]]
f3=[th(f6,b),[a,b#c,b]=>[a,d]]
s11=[th(f3,c),[c,a,b#c,b]=>[a,d]]
s12=[aia(s11,c&a),[c&a,b#c,b]=>[a,d]]
s13=[id(b),[b]=>[b]]
f21=[th(s13,a),[a,b]=>[b]]
f20=[th(f21,c),[c,a,b]=>[b]]
s14=[aia(f20,c&a),[c&a,b]=>[b]]
f22=[th(s14,b),[c&a,b,b]=>[b]]
s13=[id(b),[b]=>[b]]
f21=[th(s13,a),[a,b]=>[b]]
f20=[th(f21,c),[c,a,b]=>[b]]
f24=[aia(f20,c&a),[c&a,b]=>[b]]
f23=[th(f24,c),[c&a,c,b]=>[b]]
s15=[oia(f22,f23,b#c),[c&a,b#c,b]=>[b]]
s16=[th(s15,d),[c&a,b#c,b]=>[b,d]]
s17=[ais(s12,s16,a&b),[c&a,b#c,b]=>[a&b,d]]
s18=[aia(s17,(c&a)&b#c),[(c&a)&b#c,b]=>[a&b,d]]
s19=[ois(s18,(a&b)#d),[(c&a)&b#c,b]=>[(a&b)#d]]


Normal proof is :
=================

s1=[id(a),[a]=>[a]]
f13=[th(s1,d),[a]=>[a,d]]
f9=[th(f13,b),[a,b]=>[a,d]]
f19=[f,[a]=>[d]]
f16=[th(f19,a),[a]=>[a,d]]
f12=[th(f16,c),[a,c]=>[a,d]]
f6=[oia(f9,f12,b#c),[a,b#c]=>[a,d]]
f3=[th(f6,b),[a,b#c,b]=>[a,d]]
s11=[th(f3,c),[c,a,b#c,b]=>[a,d]]
s12=[aia(s11,c&a),[c&a,b#c,b]=>[a,d]]
s13=[id(b),[b]=>[b]]
f21=[th(s13,d),[b]=>[b,d]]
f20=[th(f21,a),[a,b]=>[b,d]]
s14=[th(f20,c),[c,a,b]=>[b,d]]
f22=[aia(s14,c&a),[c&a,b]=>[b,d]]
s15=[th(f22,b),[c&a,b,b]=>[b,d]]
s13=[id(b),[b]=>[b]]
f21=[th(s13,d),[b]=>[b,d]]
f20=[th(f21,a),[a,b]=>[b,d]]
f24=[th(f20,c),[c,a,b]=>[b,d]]
f23=[aia(f24,c&a),[c&a,b]=>[b,d]]
f25=[th(f23,c),[c&a,c,b]=>[b,d]]
s16=[oia(s15,f25,b#c),[c&a,b#c,b]=>[b,d]]
s17=[ais(s12,s16,a&b),[c&a,b#c,b]=>[a&b,d]]
s18=[ois(s17,(a&b)#d),[c&a,b#c,b]=>[(a&b)#d]]
s19=[aia(s18,(c&a)&b#c),[(c&a)&b#c,b]=>[(a&b)#d]]


yes
| ?-
```

which in tree form is :

```
                      f
  a -> a           a -> d              b -> b           b -> b
 ---------        ---------           ---------        ---------
  a -> a,d         a -> a,d            b -> b,d         b -> b,d
-----------      ----------          ----------       ----------
a,b -> a,d       a,c -> a,d          a,b -> b,d       a,b -> b,d
-----------------------------       ------------     ------------
     a,b v c -> a,d                 c,a,b -> b,d     c,a,b -> b,d
   -----------------               ------------     ------------
    a,b v c,b -> a,d               c ↑ a,b -> b,d   c ↑ a,b -> b,d
  --------------------            ---------------  ---------------
  c,a,b v c,b -> a,d             c ↑ a,b,b -> b,d  c ↑ a,c,b -> b,d
----------------------           -----------------------------------
c ↑ a,b v c,b -> a,d                  c ↑ a, b v c, b -> b,d
--------------------------------------------------------------
              c ↑ a, b v c, b -> a ↑ b, d
             -------------------------------
              c ↑ a, b v c, b -> (a ↑ b) v d
       ---------------------------------------
       (c ↑ a) ↑ (b v c), b -> (a ↑ b) v d
```

Example 2

For objects a,b,c,d in the category X , the derivation

```
                                                            a -> a    b -> b
                                                           -------   -------
                                                           a,b -> a  a,b -> b
                                                           ------------------
                                                             a,b -> a ↑ b
                                                            --------------
                  a -> a      b -> b        d -> d          a ↑ b -> a ↑ b
                 --------    --------    -----------      -----------------
                 a,b -> a    a,b -> b    d,a ↑ b -> d     d,a ↑ b -> a ↑ b
                 ---------------------   ----------------------------------
                    a,b -> a ↑ b         d,a ↑ b -> d ↑ (a ↑ b)
                   --------------        -----------------------
   d -> d          a ↑ b -> a ↑ b        d ↑ (a ↑ b) -> d ↑ (a ↑ b)
  --------        ----------------       ---------------------------
  d,a↑b -> d      d,a ↑ b -> a ↑ b       d ↑ (a ↑ b) -> d ↑ (a ↑ b),d ↑ c
  --------------------------------       -----------------------------------
     d,a ↑ b -> d ↑ (a ↑ b)              d ↑ (a ↑ b) -> (d ↑ (a ↑ b)) v (d ↑c)
     ------------------------------------------------------------------------
                  d,a ↑ b -> (d ↑ (a ↑ b)) v (d ↑ c)
```

is transformed to a normal form as follows :

```
yes
| ?- theory(dbicart).
yes
| ?- prove(s1 = [id(d), [d] => [d]]).
yes
| ?- prove(s2 = [th(s1,(a&b)#c), [(a&b)#c,d] => [d]]).
yes
| ?- prove(s3 = [ala(s2,((a&b)#c)&d), [((a&b)#c)&d] => [d]]).
yes
| ?- prove(s4 = [id(a), [a] => [a]]).
yes
| ?- prove(s5 = [th(s4,b), [a,b] => [a]]).
yes
| ?- prove(s6 = [id(b), [b] => [b]]).
yes
| ?- prove(s7 = [th(s6,a), [a,b] => [b]]).
yes
| ?- prove(s8 = [ais(s5,s7;a&b), [a,b] => [a&b]]).
yes
| ?- prove(s9 = [ala(s8,a&b), [a&b] => [a&b]]).
yes
| ?- prove(s10 = [th(s9,c), [a&b] => [a&b,c]]).
yes
| ?- prove(s11 = [id(c), [c] => [c]]).
yes
| ?- prove(s12 = [th(s11,a&b), [c] => [a&b,c]]).
yes
```

```
| ?- prove(s13 = [ola(s10,s12,(a&b)#c), [(a&b)#c] => [a&b,c]]).
yes
| ?- prove(s14 = [ois(s13,(a&b)#c), [(a&b)#c] => [(a&b)#c]]).
yes
| ?- prove(s15 = [th(s14,d), [(a&b)#c,d] => [(a&b)#c]]).
yes
| ?- prove(s16 = [ala(s15,((a&b)#c)&d), [((a&b)#c)&d] =>
|                                        [(a&b)#c]]).
yes
| ?- prove(s17 = [ais(s3,s16,d&((a&b)#c)), [((a&b)#c)&d]=>
|                                           [d&((a&b)#c)]]).
yes
| ?- prove(s18 = [th(s1,a&b), [d,a&b] => [d]]).
yes
| ?- prove(s19 = [th(s9,d), [d,a&b] => [a&b]]).
yes
| ?- prove(s20 = [ais(s18,s19,d&(a&b)), [d,a&b] => [d&(a&b)]]).
yes
| ?- prove(s21 = [ala(s20,d&(a&b)), [d&(a&b)] => [d&(a&b)]]).
yes
| ?- prove(s22 = [th(s21,d&c), [d&(a&b)] => [d&(a&b),d&c]]).
yes
| ?- prove(s23 = [ois(s22,(d&(a&b))#(d&c)), [d&(a&b)]=>
|                                            [(d&(a&b))#(d&c)]]).
yes
| ?- prove(s24 = [cut(s20,s23,d&(a&b)), [d,a&b] =>
|                                        [(d&(a&b))#(d&c)]]).
yes
| ?- normal(s24).

Proof is as follows :
=====================

f15=[id(d),[d]=>[d]]
s18=[th(f15,a&b),[d,a&b]=>[d]]
f30=[id(a),[a]=>[a]]
f27=[th(f30,b),[a,b]=>[a]]
s6=[id(b),[b]=>[b]]
f32=[th(s6,a),[a,b]=>[b]]
s8=[ais(f27,f32,a&b),[a,b]=>[a&b]]
f24=[ala(s8,a&b),[a&b]=>[a&b]]
f21=[th(f24,d),[d,a&b]=>[a&b]]
f12=[ais(s18,f21,d&a&b),[d,a&b]=>[d&a&b]]
f2=[id(d),[d]=>[d]]
f14=[th(f2,a&b),[d,a&b]=>[d]]
f29=[id(a),[a]=>[a]]
f7=[th(f29,b),[a,b]=>[a]]
f34=[id(b),[b]=>[b]]
f31=[th(f34,a),[a,b]=>[b]]
f26=[ais(f7,f31,a&b),[a,b]=>[a&b]]
f23=[ala(f26,a&b),[a&b]=>[a&b]]
s19=[th(f23,d),[d,a&b]=>[a&b]]
f11=[ais(f14,s19,d&a&b),[d,a&b]=>[d&a&b]]
s21=[ala(f11,d&a&b),[d&a&b]=>[d&a&b]]
s22=[th(s21,d&c),[d&a&b]=>[d&a&b,d&c]]
s23=[ois(s22,(d&a&b)#d&c),[d&a&b]=>[(d&a&b)#d&c]]
s24=[cut(f12,s23,d&a&b),[d,a&b]=>[(d&a&b)#d&c]]

Cut_free proof is :
===================

f181=[id(d),[d]=>[d]]
f178=[th(f181,a&b),[d,a&b]=>[d]]
f175=[th(f178,b),[d,a&b,b]=>[d]]
f172=[th(f175,a),[d,a&b,a,b]=>[d]]
f169=[th(f172,b),[d,a&b,a,b,b]=>[d]]
f166=[th(f169,a),[d,a&b,a,b,a,b]=>[d]]
f163=[inc(f166,a,b),[d,a&b,a,a,b,b]=>[d]]
f160=[con(f163,a),[d,a&b,a,b,b]=>[d]]
f157=[con(f160,b),[d,a&b,a,b]=>[d]]
f154=[ala(f157,a&b),[d,a&b,a&b]=>[d]]
f148=[th(f154,d),[d,a&b,d,a&b]=>[d]]
f138=[id(a),[a]=>[a]]
f135=[th(f138,b),[a,b]=>[a]]
f132=[th(f135,b),[a,b,b]=>[a]]
f126=[th(f132,a),[a,b,a,b]=>[a]]
f145=[id(b),[b]=>[b]]
f144=[th(f145,a),[a,b]=>[b]]
f141=[th(f144,b),[a,b,b]=>[b]]
f129=[th(f141,a),[a,b,a,b]=>[b]]
```

```
f102=[ais(f126,f129,a&b),[a,b,a,b]=>[a&b]]
f105=[inc(f102,a,b),[a,a,b,b]=>[a&b]]
f106=[con(f105,a),[a,b,b]=>[a&b]]
f97=[con(f106,b),[a,b]=>[a&b]]
f188=[aia(f97,a&b),[a&b]=>[a&b]]
f187=[th(f188,d),[d,a&b]=>[a&b]]
f184=[th(f187,a&b),[d,a&b,a&b]=>[a&b]]
f151=[th(f184,d),[d,a&b,d,a&b]=>[a&b]]
f45=[ais(f148,f151,d&a&b),[d,a&b,d,a&b]=>[d&a&b]]
f48=[inc(f45,d,a&b),[d,d,a&b,a&b]=>[d&a&b]]
f49=[con(f48,d),[d,a&b,a&b]=>[d&a&b]]
f40=[con(f49,a&b),[d,a&b]=>[d&a&b]]
f37=[th(f40,d&c),[d,a&b]=>[d&a&b,d&c]]
s24=[ois(f37,(d&a&b)#d&c),[d,a&b]=>[(d&a&b)#d&c]]
```

Contraction and interchange free proof is :

```
f181=[id(d),[d]=>[d]]
f190=[th(f181,b),[d,b]=>[d]]
f189=[th(f190,a),[d,a,b]=>[d]]
f49=[aia(f189,a&b),[d,a&b]=>[d]]
f138=[id(a),[a]=>[a]]
f106=[th(f138,b),[a,b]=>[a]]
f145=[id(b),[b]=>[b]]
f193=[th(f145,a),[a,b]=>[b]]
f97=[ais(f106,f193,a&b),[a,b]=>[a&b]]
f195=[aia(f97,a&b),[a&b]=>[a&b]]
f196=[th(f195,d),[d,a&b]=>[a&b]]
f40=[ais(f49,f196,d&a&b),[d,a&b]=>[d&a&b]]
f197=[th(f40,d),[d,a&b]=>[d&a&b,d]]
f215=[id(d),[d]=>[d]]
f214=[th(f215,b),[d,b]=>[d]]
f213=[th(f214,a),[d,a,b]=>[d]]
f201=[aia(f213,a&b),[d,a&b]=>[d]]
f231=[id(a),[a]=>[a]]
f224=[th(f231,b),[a,b]=>[a]]
f237=[id(b),[b]=>[b]]
f225=[th(f237,a),[a,b]=>[b]]
f222=[ais(f224,f225,a&b),[a,b]=>[a&b]]
f216=[aia(f222,a&b),[a&b]=>[a&b]]
f202=[th(f216,d),[d,a&b]=>[a&b]]
f199=[ais(f201,f202,d&a&b),[d,a&b]=>[d&a&b]]
f198=[th(f199,c),[d,a&b]=>[d&a&b,c]]
f37=[ais(f197,f198,d&c),[d,a&b]=>[d&a&b,d&c]]
s24=[ois(f37,(d&a&b)#d&c),[d,a&b]=>[(d&a&b)#d&c]]
```

Normal proof is :

```
f181=[id(d),[d]=>[d]]
f190=[th(f181,d),[d]=>[d,d]]
f189=[th(f190,b),[d,b]=>[d,d]]
f49=[th(f189,a),[d,a,b]=>[d,d]]
f40=[aia(f49,a&b),[d,a&b]=>[d,d]]
f138=[id(a),[a]=>[a]]
f106=[th(f138,d),[a]=>[a,d]]
f97=[th(f106,b),[a,b]=>[a,d]]
f145=[id(b),[b]=>[b]]
f193=[th(f145,d),[b]=>[b,d]]
f245=[th(f193,a),[a,b]=>[b,d]]
f195=[ais(f97,f245,a&b),[a,b]=>[a&b,d]]
f196=[aia(f195,a&b),[a&b]=>[a&b,d]]
f243=[th(f196,d),[d,a&b]=>[a&b,d]]
f197=[ais(f40,f243,d&a&b),[d,a&b]=>[d&a&b,d]]
f215=[id(d),[d]=>[d]]
f214=[th(f215,c),[d]=>[d,c]]
f213=[th(f214,b),[d,b]=>[d,c]]
f201=[th(f213,a),[d,a,b]=>[d,c]]
f199=[aia(f201,a&b),[d,a&b]=>[d,c]]
f231=[id(a),[a]=>[a]]
f224=[th(f231,c),[a]=>[a,c]]
f222=[th(f224,b),[a,b]=>[a,c]]
f237=[id(b),[b]=>[b]]
f225=[th(f237,c),[b]=>[b,c]]
f244=[th(f225,a),[a,b]=>[b,c]]
f216=[ais(f222,f244,a&b),[a,b]=>[a&b,c]]
f202=[aia(f216,a&b),[a&b]=>[a&b,c]]
f242=[th(f202,d),[d,a&b]=>[a&b,c]]
f198=[ais(f199,f242,d&a&b),[d,a&b]=>[d&a&b,c]]
```

```
| f37=[ais(f197,f198,d&c),[d,a&b]=>[d&a&b,d&c]]                    |
| s24=[ois(f37,(d&a&b)#d&c),[d,a&b]=>[(d&a&b)#d&c]]                |
*------------------------------------------------------------------*
```

which in tree form is :

```
                    a -> a      b -> b                           a -> a       b -> b
                   --------    --------                         --------     --------
    d -> d         a -> a,d    b -> b,d          d -> d         a -> a,c     b -> b,c
   --------       ----------  ----------        --------       ----------   ----------
    d -> d,d      a,b -> a,d  a,b -> b,d         d -> d,c      a,b -> a,c   a,b -> b,c
  -----------     ----------------------      -----------      -----------------------
  d,b -> d,d          a,b -> a ↑ b,d          d,b -> d,c          a,b -> a ↑ b,c
 ------------      ------------------        ------------      ------------------
 d,a,b -> d,d      a ↑ b -> a ↑ b,d          d,a,b -> d,c      a ↑ b -> a ↑ b,c
 ------------    --------------------        ------------    --------------------
 d,atb -> d,d    d,a ↑ b -> a ↑ b,d          d,atb -> d,c    d,a ↑ b -> a ↑ b,c
 ------------------------------------        --------------------------------------
     d,a ↑ b -> d ↑ (a ↑ b),d                        d,a ↑ b -> d ↑ (a ↑ b),c
     ------------------------------------------------------------------------
                       d,a ↑ b -> (d ↑ (a ↑ b)) v (d ↑ c)
```

(3.5.10) <u>The CHURCH-ROSSER Theorem for Ddbc(X)</u>

If p == q, then there exists a normal r in Ddbc(X) such that p >= r and q >= r.

(3.6) Application to Category Theory

In this section , we use the system to represent some arrows of Fdbc(X) and obtain their normal representations .

<u>Example</u> The normal representation of the following arrow

$$a \uparrow (b \vee c) \xrightarrow{\text{delta\_left}(a,b,c)} (a \uparrow b) \vee (a \uparrow c)$$

is obtained as follows :

```
*------------------------------------------------------------------*
|     yes                                                          |
|     | ?- theory(dblcart).                                        |
|     yes                                                          |
|     | ?- norm_rep(delta_left(a,b,c)).                            |
|                                                                  |
|     The arrow is :                                               |
|     ==============                                               |
|                                                                  |
|     delta_left(a,b,c)                                            |
|                                                                  |
|                                                                  |
|     Proof is as follows :                                        |
|     =====================                                        |
|                                                                  |
|     f27=[id(a),[a]=>[a]]                                         |
|     f23=[th(f27,b),[a,b]=>[a]]                                   |
|     f28=[id(b),[b]=>[b]]                                         |
|     f24=[th(f28,a),[a,b]=>[b]]                                   |
|     f21=[ais(f23,f24,a&b),[a,b]=>[a&b]]                          |
|     f8=[id(a),[a]=>[a]]                                          |
|     f6=[th(f8,b),[a,b]=>[a]]                                     |
|     f9=[id(b),[b]=>[b]]                                          |
|     f7=[th(f9,a),[a,b]=>[b]]                                     |
|     f5=[ais(f6,f7,a&b),[a,b]=>[a&b]]                             |
|     f4=[aia(f5,a&b),[a&b]=>[a&b]]                                |
|     f3=[th(f4,a&c),[a&b]=>[a&b,a&c]]                             |
|     f2=[ois(f3,(a&b)#a&c),[a&b]=>[(a&b)#a&c]]                    |
|     f19=[cut(f21,f2,a&b),[a,b]=>[(a&b)#a&c]]                     |
|     f29=[id(a),[a]=>[a]]                                         |
|     f25=[th(f29,c),[a,c]=>[a]]                                   |
|     f30=[id(c),[c]=>[c]]                                         |
|     f26=[th(f30,a),[a,c]=>[c]]                                   |
```

```
f22=[ais(f25,f26,a&c),[a,c]=>[a&c]]
f16=[id(a),[a]=>[a]]
f14=[th(f16,c),[a,c]=>[a]]
f17=[id(c),[c]=>[c]]
f15=[th(f17,a),[a,c]=>[c]]
f13=[ais(f14,f15,a&c),[a,c]=>[a&c]]
f12=[aia(f13,a&c),[a&c]=>[a&c]]
f11=[th(f12,a&b),[a&c]=>[a&b,a&c]]
f10=[ois(f11,(a&b)#a&c),[a&c]=>[(a&b)#a&c]]
f20=[cut(f22,f10,a&c),[a,c]=>[(a&b)#a&c]]
f18=[oia(f19,f20,b#c),[a,b#c]=>[(a&b)#a&c]]
f1=[aia(f18,a&b#c),[a&b#c]=>[(a&b)#a&c]]
```

Cut_free proof is :

```
f77=[id(a),[a]=>[a]]
f74=[th(f77,b),[a,b]=>[a]]
f71=[th(f74,b),[a,b,b]=>[a]]
f65=[th(f71,a),[a,b,a,b]=>[a]]
f84=[id(b),[b]=>[b]]
f83=[th(f84,a),[a,b]=>[b]]
f80=[th(f83,b),[a,b,b]=>[b]]
f68=[th(f80,a),[a,b,a,b]=>[b]]
f41=[ais(f65,f68,a&b),[a,b,a,b]=>[a&b]]
f44=[inc(f41,a,b),[a,a,b,b]=>[a&b]]
f45=[con(f44,a),[a,b,b]=>[a&b]]
f36=[con(f45,b),[a,b]=>[a&b]]
f33=[th(f36,a&c),[a,b]=>[a&b,a&c]]
f19=[ois(f33,(a&b)#a&c),[a,b]=>[(a&b)#a&c]]
f131=[id(a),[a]=>[a]]
f128=[th(f131,c),[a,c]=>[a]]
f125=[th(f128,c),[a,c,c]=>[a]]
f119=[th(f125,a),[a,c,a,c]=>[a]]
f138=[id(c),[c]=>[c]]
f137=[th(f138,a),[a,c]=>[c]]
f134=[th(f137,c),[a,c,c]=>[c]]
f122=[th(f134,a),[a,c,a,c]=>[c]]
f95=[ais(f119,f122,a&c),[a,c,a,c]=>[a&c]]
f98=[inc(f95,a,c),[a,a,c,c]=>[a&c]]
f99=[con(f98,a),[a,c,c]=>[a&c]]
f90=[con(f99,c),[a,c]=>[a&c]]
f87=[th(f90,a&b),[a,c]=>[a&b,a&c]]
f20=[ois(f87,(a&b)#a&c),[a,c]=>[(a&b)#a&c]]
f18=[oia(f19,f20,b#c),[a,b#c]=>[(a&b)#a&c]]
f1=[aia(f18,a&b#c),[a&b#c]=>[(a&b)#a&c]]
```

Contraction and interchange free proof is :

```
f77=[id(a),[a]=>[a]]
f45=[th(f77,b),[a,b]=>[a]]
f84=[id(b),[b]=>[b]]
f141=[th(f84,a),[a,b]=>[b]]
f36=[ais(f45,f141,a&b),[a,b]=>[a&b]]
f142=[th(f36,a),[a,b]=>[a&b,a]]
f153=[id(a),[a]=>[a]]
f146=[th(f153,b),[a,b]=>[a]]
f159=[id(b),[b]=>[b]]
f147=[th(f159,a),[a,b]=>[b]]
f144=[ais(f146,f147,a&b),[a,b]=>[a&b]]
f143=[th(f144,c),[a,b]=>[a&b,c]]
f33=[ais(f142,f143,a&c),[a,b]=>[a&b,a&c]]
f19=[ois(f33,(a&b)#a&c),[a,b]=>[(a&b)#a&c]]
f131=[id(a),[a]=>[a]]
f99=[th(f131,c),[a,c]=>[a]]
f138=[id(c),[c]=>[c]]
f162=[th(f138,a),[a,c]=>[c]]
f90=[ais(f99,f162,a&c),[a,c]=>[a&c]]
f163=[th(f90,a),[a,c]=>[a,a&c]]
f174=[id(a),[a]=>[a]]
f167=[th(f174,c),[a,c]=>[a]]
f180=[id(c),[c]=>[c]]
f168=[th(f180,a),[a,c]=>[c]]
f165=[ais(f167,f168,a&c),[a,c]=>[a&c]]
f164=[th(f165,b),[a,c]=>[b,a&c]]
f87=[ais(f163,f164,a&b),[a,c]=>[a&b,a&c]]
f20=[ois(f87,(a&b)#a&c),[a,c]=>[(a&b)#a&c]]
f18=[oia(f19,f20,b#c),[a,b#c]=>[(a&b)#a&c]]
f1=[aia(f18,a&b#c),[a&b#c]=>[(a&b)#a&c]]
```

```
Normal proof is :
=================

f77=[id(a),[a]=>[a]]
f45=[th(f77,a),[a]=>[a,a]]
f36=[th(f45,b),[a,b]=>[a,a]]
f84=[id(b),[b]=>[b]]
f141=[th(f84,a),[b]=>[b,a]]
f184=[th(f141,a),[a,b]=>[b,a]]
f142=[ais(f36,f184,a&b),[a,b]=>[a&b,a]]
f153=[id(a),[a]=>[a]]
f146=[th(f153,c),[a]=>[a,c]]
f144=[th(f146,b),[a,b]=>[a,c]]
f159=[id(b),[b]=>[b]]
f147=[th(f159,c),[b]=>[b,c]]
f183=[th(f147,a),[a,b]=>[b,c]]
f143=[ais(f144,f183,a&b),[a,b]=>[a&b,c]]
f33=[ais(f142,f143,a&c),[a,b]=>[a&b,a&c]]
f19=[ois(f33,(a&b)#a&c),[a,b]=>[(a&b)#a&c]]
f131=[id(a),[a]=>[a]]
f99=[th(f131,a),[a]=>[a,a]]
f90=[th(f99,c),[a,c]=>[a,a]]
f138=[id(c),[c]=>[c]]
f162=[th(f138,a),[c]=>[a,c]]
f182=[th(f162,a),[a,c]=>[a,c]]
f163=[ais(f90,f182,a&c),[a,c]=>[a,a&c]]
f174=[id(a),[a]=>[a]]
f167=[th(f174,b),[a]=>[b,a]]
f165=[th(f167,c),[a,c]=>[b,a]]
f180=[id(c),[c]=>[c]]
f168=[th(f180,b),[c]=>[b,c]]
f181=[th(f168,a),[a,c]=>[b,c]]
f164=[ais(f165,f181,a&c),[a,c]=>[b,a&c]]
f87=[ais(f163,f164,a&b),[a,c]=>[a&b,a&c]]
f20=[ois(f87,(a&b)#a&c),[a,c]=>[(a&b)#a&c]]
f18=[oia(f19,f20,b#c),[a,b#c]=>[(a&b)#a&c]]
f1=[aia(f18,a&b#c),[a&b#c]=>[(a&b)#a&c]]

yes
| ?-
```

which in tree form is :

```
 a -> a    b -> b     a -> a    b -> b      a -> a    c -> c     a -> a    c -> c
 ------    ------     ------    ------      ------    ------     ------    ------
 a->a,a    b->b,a     a->a,c    b->b,c      a->a,a    c->a,c     a->b,a    c->b,c
-------  --------   -------- --------    --------  --------   --------  --------
a,b->a,a a,b->b,a   a,b->a,c a,b->b,c    a,c->a,a  a,c->a,c   a,c->b,a  a,c->b,c
-----------------   -----------------    ------------------   ------------------
 a,b -> a ↑ b,a      a,b -> a ↑ b,c       a,c -> a,a ↑ c       a,c -> b,a ↑ c
 ------------------------------------     -----------------------------------
        a,b -> a ↑ b , a ↑ c                        a,c -> a ↑ b , a ↑ c
        ----------------------------                ------------------------
        a,b -> (a ↑ b) v (a ↑ c)                    a,c -> (a ↑ b) v (a ↑ c)
        ----------------------------------------------------------------------
                       a , b v c -> (a ↑ b) v (a ↑ c)
                       ------------------------------
                       a ↑ (b v c) -> (a ↑ b) v (a ↑ c)
```

Chapter 4

CARTESIAN CLOSED CATEGORY THEORY

In this chapter, we represent the class of small cartesian closed categories and implement SZABO's algorithms for representing arrows from the free cartesian closed category Fccl(X) over X by means of derivations from the class Dccl(X) of unlabelled 'cartesian-closed-derivations' over X; for interpreting derivations of this class as arrows of Fccl(X); for elimination of cuts in, and normalization of, such derivations; and, finally, for giving the normal representation of an arrow of Fccl(X).

Several lemmas concerning the cartesian closed category theory are stated without proof in SZABO :

i) Lemma (4.5.4), concerning derivations with terminal arrows of the category Fccl(X);

ii) Lemma (4.5.5), concerning derivations which have instances of (R2) " thinning in the antecedent ", (R3) "contraction in the antecedent " and (R4) " interchange in the antecedent " with active formulae of the form a † b;

iii) Lemma (4.5.6), concerning derivations of Dccl(X) which have instances of (R14) "hook introduction in the antecedent" with the property that the active formulae in their right premisses are terminal objects;

we have constructed and implemented their proofs in the system to satisfy the definition of the normal form (4.5.7).

(4.1) Definition

A cartesian closed category is a cartesian category $\underline{C}$ with the following additional structure :

(4) A bifunctor (-) => (-) : op($\underline{C}$) x $\underline{C}$ -> $\underline{C}$.

(5) An adjunction α_λ , where

$\alpha_\lambda = \{\alpha_\lambda(A,B,C) : \underline{C}(A \dagger B,\ C) \to \underline{C}(B,\ A \Rightarrow C) \in ArEns \mid A,B,C \in Ob\underline{C}\}$.

(4.2) The Free Cartesian Closed Category Fccl(X)

Let X be a fixed but arbitrary small category. The free cartesian closed category Fccl(X) is constructed as follows :

(4.2.1) Definition

The class cclL(X) of (cartesian closed) formulae over a category X is defined to contain the class cL(X) of (cartesian) formulae and

v) If A,B are formulae , then A => B is .

(4.2.2) The Labelled Derivations LDccl(X)

The class LDccl(X) of (cartesian closed) labelled derivations over X and their conclusions, are simultaneously defined by conditions (i) - (viii) of (1.2.2) and the following additional conditions :

ix) for derivations F,G with conclusions

f : A -> B , g : C -> D respectively,

EPSILON(F,G) is a derivation ,

with conclusion epsilon_left(f,g) : A ↑ (B => C) -> D ;

x) for a derivation F with conclusion f : A ↑ B -> C

ALPHA(F) is a derivation ,

with conclusion alpha_right(f) : B -> (A => C) .

(4.2.3) Definition

The relation == is defined to be the smallest equivalence relation on LDccl(X) satisfying conditions (i) - (viii) of (1.2.3) and the following additional conditions

ix) If F == G and H == K , with conclusions f,g : A -> B ,
h,k : C -> D ,

then

EPSILON (F,H) == EPSILON(G,K) ;

x) IF F == G , with conclusions f,g : A ↑ B -> C

then

ALPHA(F) == ALPHA(G) ;

xi) For derivations F, G, H, K with conclusions

f : A -> B , g : C -> D ,

h : B -> E , k : J -> C respectively

[Note for derivations F,G as above we define the derivation HOOK(F,G) = ALPHA(EPSILON(F,G)), with conclusion f => g = alpha_right(epsilon_left(f,g)) : (B => C) -> (A => D)] then after introducing the definition of the derivation HOOK we have :

COMP(HOOK(H,K), HOOK(F,G), B => C) ==
HOOK(COMP(F, H, B), COMP(K, G, C)) ;

xii) For F with conclusion f : A ↑ B -> C ,

```
COMP( PRODUCT( I(A), ALPHA( F ) ),
      EPSILON( I(A), I(C) ),
      A ↑ (A => C) )  ==  F
```

where I(A) is as defined in (1.4.2) ;

xiii) For G with conclusion g : B -> (A => C) ,

```
ALPHA( COMP( PRODUCT( I(A), G ),
             EPSILON( I(A), I(C) ),
             A ↑ (A => C) ) )  ==  G        ;
```

xiv) For F,G with conclusions f : D -> C and g : A ↑ B -> D respectively

```
COMP( ALPHA( G ), HOOK( I(A), F ), A => D )  ==
ALPHA( COMP( G, F, D ) )                     ;
```

xv) For F,G,H with conclusions f : A -> D , g : D ↑ E -> C , h : B -> E respectively

```
COMP( COMP( H, ALPHA( G ), E ),
      HOOK( F, I(C) ),
      D => C )  ==
ALPHA( COMP( PRODUCT( F, H ), G, D ↑ E ) )    ;
```

xvi) For F,G,H with conclusions f : E -> (C => D), g : A -> C ; h : D -> B respectively

```
COMP( COMP( EPSILON( I(C), I(D) ),
            PRODUCT( G, F ),
            C ↑ (C => D) ),
      H,
      D )  ==
COMP( PRODUCT( I(A), COMP( F, HOOK( G, H ), C => D ) ),
      EPSILON( I(A), I(B) ),
      A ↑ (A => B) )  .
```

(4.2.4) Definition

The free cartesian closed category Fccl(X) is defined to be the category having :

i) as objects , the cartesian closed formulae over X ;

ii) as arrows , the ==-class of LDccl(X) ;

iii) - xi) as in the definition (1.2.5) ;

xii) as epsilon_left(-,-), the function defined by
epsilon_left([F],[G]) = [EPSILON(F, G)];

xiii) as alpha_right(-,-) , the function defined by

alpha_right([F]) = [ALPHA(F)] .

It is now routine to check that, with this structure, Fccl(X) is a category, is cartesian closed, and is free cartesian closed over X. There is the obvious embedding X -> Fccl(X) defined by A |-> A; f |-> [LA1(f)].

(4.3) The Unlabelled Derivations Dccl(X)

We define the class Dccl(X) of unlabelled (cartesian closed) derivations as another way to represent these arrows of Fccl(X) by composition free derivations .

(4.3.1) Definition

The class Dccl(X) is defined inductively by the conditions in the definition (1.3.1) and the following additional rules :

x) (hook introduction in the antecedent)
If p1,p2 are derivations with conclusions G -> a, DbL -> F, then R14(p1,p2,a => b) is a derivation with conclusion DG(a => b)L -> F;

xi) (hook introduction in the succedent)
If p is a derivation with conclusion GaD -> b , then R15(p, a => b) is a derivation with conclusion GD -> (a => b).

(4.3.2) Representation of Dccl(X) in Prolog

In order to represent derivations of Dccl(X) in Prolog we extend the representation in (1.3.2) by the following :

we extend the abbreviations in (1.3.2) by

->	for =>
hia	for the name of the operation in R14(p1,p2,a => b)
his	for the name of the operation in R15(p, a => b) .

ix) R14(p1,p2,a => b) is represented by a step-collection with the last step ,

g3 = [hia(g1,g2,a -> b), DGa->bL => F]

where g1,g2 are names of the last steps of the step-collections for the derivations p1,p2 respectively .

x) R15(p,a => b) is represented by a step-collection with the last step.

g2 = [his(g1,a -> b), GD => [a -> b]]

where g1 is the name of the last step of the step-collection for the derivation p .

As an example of a derivation of the class Dccl(X) and the use of the system to check its correctness, consider the following :

Example

The derivation

```
                              c -> c
                             --------
                             b,c -> c
                           ----------
                           b,e,c -> c
                           ----------------
a -> a                       b,c -> (e => c)
----------------------------------------------
       a,a => b,c -> (e => c)
----------------------------------------------
  a,(a => b) & c -> (e => c)
----------------------------------------------
  (a => b) & c,a -> (e => c)
```

can be checked for correctness correct using the system as follows :

```
yes
| ?- theory(cartcl).

yes
| ?- prove(l = [id(a), [a] => [a]]).

yes
| ?- prove(m = [id(c), [c] => [c]]).

yes
| ?- prove(m1 = [th(m,b), [b,c] => [c]]).

yes
| ?- prove(m2 = [th(m1,e), [b,e,c] => [c]]).

yes
| ?- prove(m3 = [his(m2,e -> c), [b,c] => [e -> c]]).

yes
| ?- prove(n = [hia(l,m3,a -> b),[a,a -> b,c] => [e -> c]]).

yes
| ?- prove(o = [ala(n,(a->b)&c),[a,(a->b)&c] => [e -> c]]).

yes
| ?- prove(p = [inc(o,(a->b)&c,a),[(a->b)&c,a] => [e->c]]).

yes
| ?-
```

(4.4) The Semantics of Dccl(X)

In this section, we define a function to interpret derivations of Dccl(X) as arrows of Fccl(X) and prove the completeness theorem in the sense that every arrow of Fccl(X) is representable by means of some derivation of Dccl(X). For the sake of this interpretation we extend the canonical arrows in (1.4) by the following canonical arrows of Fccl(X), determined by the adjunction α_λ :

iv) epsilon(A,B) : A ↑ (A => B) -> B for all A,B of ObFccl(X), where
epsilon(A,B) = α_λ^{-1} (A, A => B, B)(id(A => B))

v) ita(A,B) : B -> A => (A ↑ B) for all A,B of ObFccl(X), where
ita(A,B) = α_λ (A, B, A ↑ B)(id(A ↑ B)) .

(4.4.1) Definition

We extend the definition of the function S in (1.4.1) to interpret derivations of Dccl(X) as arrows of Fccl(X) as follows :

```
(9) S(R14(p1,p2,a => b) = comp(comp(comp(S(p2),(I1 ↑ epsilon(a,b)) ↑ I2),
                                 ((I1 ↑ S(p1)) ↑ I3) ↑ I2 ),
                            alpha_inverse(PI(D), PI(G), a => b) ↑ I2)
                          where
                            I1  is the identity on  PI(D) ,
                            I2  is the identity on  PI(L)  and
                            I3  is the identity on  a => b     ;

(10) S(R15(p,a => b)     = comp(comp(comp(I1 => S(p),
                                   I1 => (sigma(a,PI(G)) ↑ I2) ),
                                I1 => alpha( a, PI(G), PI(D) ) ),
                            ita( a, PI(G) ↑ PI(D) ) )      , where
                            I1  is the identity on  a  and
                            I2  is the identity on  PI(D) .
```

The following is an example for a session with the system; as input is a derivation of Dccl(X) and as output is an arrow of Fccl(X) as interpretation for that derivation .

Example

For the arrows f in X(a,c), g in X(b,c) and e in Ob(X), the derivation

```
              e -> e
            ----------
            a,e -> e                                    g
         -------------                               b -> c
         a,e,e -> e                    f           ----------
         -------------------------   a -> c        e,b -> c
            a,e -> (c => e)          -------------------------
                                        a,c => e,b -> c
            --------------------------------------------------
                             a,a,e,b -> c
                            --------------
                               a,e,b -> c
                            --------------
                             a,e ↑ b -> c
```

is interpreted in ArFccl(X) as follows :

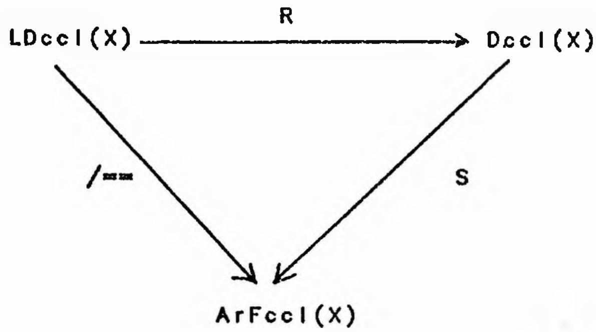
```
yes
| ?- theory(cartcl).

yes
| ?- prove(l = [id(e), [e] => [e]]).

yes
| ?- prove(l1 = [th(l,a), [a,e] => [e]]).

yes
| ?- prove(l2 = [th(l1,c), [a,e,c] => [e]]).

yes
| ?- prove(l3 = [his(l2,c -> e), [a,e] => [c -> e]]).

yes
| ?- prove(m = [f, [a] => [c]]).

yes
| ?- prove(n = [g, [b] => [c]]).

yes
| ?- prove(n1 = [th(n,e), [e,b] => [c]]).

yes
| ?- prove(o = [hia(m,n1,c -> e), [a,c -> e,b] => [c]]).

yes
| ?- prove(p = [cut(l3,o,c -> e), [a,a,e,b] => [c]]).

yes
| ?- prove(q = [con(p,a), [a,e,b] => [c]]).

yes
| ?- prove(r = [aia(q,e & b), [a,e & b] => [c]]).

yes
| ?- interpr(r,H).

H = comp(comp(comp(comp(comp(comp(g,pi_right(e,b)),
                          product(epsilon(c,e),id(b))),
                     product(product(f,id(c->e)),id(b))),
                product(product(id(a),
                                comp(comp(hook(id(c),
                                               comp(pi_right(a,e),
                                                    pi_left(a&e,c))),
                                          hook(id(c),sigma(c,a&e))),
                                     lta(c,a&e))),
                        id(b))),
           product(delta(a),id(e&b))),
      alpha(a,e,b))


yes
| ?-
```

In order to prove that the class Dccl(X) of the unlabelled cartesian closed derivations is adequate in the sense that every arrow in Fccl(X) is representable by means of derivation in this class, we prove the following theorem :

(4.4.2) <u>The Completeness Theorem for Dccl(X)</u>

There is a function

R : LDccl(X) -----------------> Dccl(X) , so that



commutes .

Proof :

We extend the definition of R(F) for F of LDccl(X) in (1.4.2) by the following :

We extend iii) of (1.4.2) by

```
iii) or is B => C : put
          R( F ) = R15( R14( R( I(B) ),
                             R( I(C) ), B => C ),
                   B => C )

viii) when F is EPSILON( G,H ) , put
          R( F ) = R11( R14( R( G ), R( H ), B => C ),
                   A ↑ (B => C) )

ix)   when F is ALPHA( G ) , put
          R( F ) = R15( R1( R10( R2( R( I(A) ), B ),
                                 R2( R( I(B) ), A ),
                                 A ↑ B ),
                            R( G ),
                            A ↑ B ),
                   A => C )
```

We extend the examples in (1.4.2) by the following ones for some arrows of Fccl(X) in order to illustrate the above algorithm. These arrows and their representations are built in to the system, for convenience.

Example 1

For the derivation F = EPSILON(I(A), I(B)) representing the arrow

epsilon(A,B) : A ↑ (A => B) -> B

the representation R(F) is

```
R11( R14( R( I(A) ), R( I(B) ), A => B ),
     A ↑ (A => B) )
```

Example 2

For the derivation F = ALPHA(LA2(A † B)) representing the arrow

ita(A,B) : B -> A => (A † B)

the representation R(F) is

```
R15( R10( R2( R( I(A) ), B ),
          R2( R( I(B) ), A ),
          A † B ),
     A => (A † B) ) .
```

Example 3 For derivations F, G representing the arrows

f : A -> B , g : C -> D respectively ,

with the derivation

HOOK(F, G) = ALPHA(EPSILON(F, G))

representing the arrow

f => g : (B => C) -> (A => D)

then the representation R(HOOK(F, G)) is

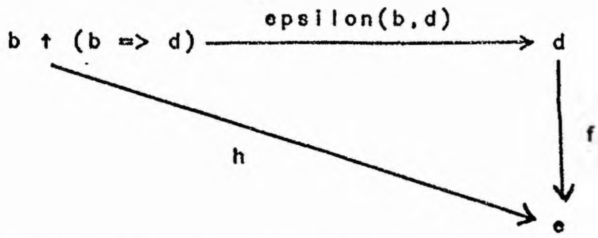
```
R( ALPHA( EPSILON( F, G ) ) ) .
R15( R1( R10( R2( R( I(A) ), B ),
              R2( R( I(B) ), A ),
              A † B ),
         R11( R14( R( G ), R( H ), B => C ),
              A † (B => C) )
         A † B ),
     A => C )
```

The following is a representation of an arrow of Fccl(X) by means of a derivation of the class Dccl(X) :

Example 4 for f in X(d,e) , the arrow

h = comp(f, epsilon(b,d)) : b † (b => d) -> e

in the diagram



is represented by a derivation of Dccl(X) as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- arrow(d,f,e).
yes
| ?- rop_of(comp(f,epsilon(b,d))).

The arrow is :
==============
comp(f,epsilon(b,d))
```

```
Representation :-
================

f5=[id(b),[b]=>[b]]
f6=[id(d),[d]=>[d]]
f4=[hia(f5,f6,b->d),[b,b->d]=>[d]]
f3=[aia(f4,b,b->d),[b&(b->d)]=>[d]]
f2=[f,[d]=>[e]]
f1=[cut(f3,f2,d),[b&(b->d)]=>[e]]

yes
| ?-
```

which in tree form is :

```
          b -> b        d -> d
          ------------------------
             b,b => d -> d
       ------------------------                 f
       b ↑ (b => d) -> d                     d -> e
       -------------------------------------------------
                  b ↑ (b => d) -> e
```

(4.5) The Syntax of Fccl(X)

After representing the arrows of Fccl(X) by means of derivations of the class Dccl(X), we are able to manipulate these derivations to eliminate cuts and obtain their normal representations.

(4.5.1) The Cut-Elimination Theorem for Dccl(X)

There exists for every derivation p in Dccl(X) an equivalent cut-free derivation q in Dccl(X) .

Proof

By the proof of (1.5.2) and the following additional clauses of Appendix (2) :

(C.5), (C.16), (C.30), C.31) and (C.45), every derivation in Dccl(X) has an equivalent cut-free one .

###

(4.5.2) Definition

Let L(T) be the subclass of the class of (cartesian closed) formulae cclL(X) generated by the conditions :

1) T is in L(T).
2) If a,b are in L(T), then a ↑ b is in L(T).
3) If a is in L(T) and b is in cclL(X), then (b => a) is in L(T).

(4.5.3) Lemma

a ≅ T in Fccl(X) iff a is in L(T).

(4.5.4) Lemma

If p derives G -> a and a = ~ T, then p is of the form

```
        q
         -> a
---------  (K)
G        -> a
```

Proof:

The proof is by induction on the construction of the formula a :

i) a = T
ii) a = c ↑ d — as in (1.5.3)

iii) a = c => d where d = ~ T and b is in ObFccl(X),

```
      g
       -> d
----------
c      -> d
----------------
       -> (c => d)
----------------
G      -> (c => d)
```

This completes the induction proof.

###

(4.5.5) Lemma

For every cut-free derivation p in Dccl(X), there exists an equivalent cut-free q in Dccl(X), containing no instances of (R2), (R3) and (R4) whose active formulae are of the form a ↑ b.

Proof:

The proof in case of (R2) follows from clause (E.41.1) in Appendix3. In caes of (R3) and (R4) the proof follows from the proof of lemma (1.5.4) and the clauses (E.13), (E.14), (E.21) and (E.22) in Appendix(3). There are some cases in which an instance of (R14) followed by an instance of (R3) or (R4) with active formulae of the form a ↑ b and these active formulae are not in the antecedent of one of the premisses of (R14) but each one in the antecedent of each premiss like

```
a ↑ b,G ------> c          D,a ↑ b,d,L -------> e
------------------------------------------------------
          D,a ↑ b,a ↑ b,G,c => d,L ------> e
          --------------------------------------
             D,a ↑ b,G,c => d,L -------> e
```

In these cases we use the priority described in the clauses mentioned below in (4.5.8) to move the instance of (R11) applied to construct a ↑ b below the application of the instance of (R14) and then using the clauses of lemma (1.5.4) again to remove the instances of (R3) and (R4) with active formulae of the form a ↑ b.

###

(4.5.6) Lemma

Every cut-free p in Dccl(X) is equivalent to a cut-free q in Dccl(X) containing no instance of (R14) with the property that the active formulae in their right premisses are isomorphic to T.

Proof:

Let p is a derivation of the form

$$\frac{G \to a \qquad DbL \xrightarrow{p1} c}{DG\ a \Rightarrow b\ L \to c}$$

where b is isomorphic to T, then it is equivalent to

$$\frac{G \to a \qquad \dfrac{DL \xrightarrow{p2} c}{DbL \to c}}{DG\ a \Rightarrow b\ L \to c}$$

where p2 is the unique cut-free derivation obtained by applying the cut elimination algorithm of appendix (2) to the derivation

$$\frac{\xrightarrow{m} b \qquad DbL \xrightarrow{p1} c}{DL \to c}$$

where m is the unique derivation of -> b by lemma (4.5.4), then using clause (D.11.4) in appendix (4) to remove the instance of (R14). ###

(4.5.7) Definition

A derivation p in Dccl(X) is normal if it satisfies the following conditions :

(1) All subderivations of p not containing instances of (R14) and (R15) are normal in the sense of the definition (1.5.4) with the condition that the active formulae of instances of (R2) are atomic replace by the condition that the active formulae of instances of (R2) are either atomic or of the form a => b.

(2) p contains no instance of (R14) whose active formula in the right-hand premiss is isomorphic to T.

(3) If p derives G -> a and a =~ T, then p is of the form

$$\frac{\to a}{G \quad \to a}\ (K)$$

where q is the unique derivation of -> a, and where (K) consists of the instances of (R2) required to derive G -> a from -> a.

(4) p contains no instances of (R2), (R3) and (R4) whose active formulae are of the form a ↑ b.

(5) Using the relation =< as defined in (1.5.5); for normality in Dccl(X) it is required that

(R10) =< (R15) =< (R14) =< (R3) =< (R11) =< (R2) =< (R4).

(4.5.8) <u>The Normalization Theorem For Dccl(X)</u>

For every derivation p in the class Dccl(X) there is a unique equivalent normal derivation q in Dccl(X).

<u>Proof</u>

By combining the results of the Cut-Elimination theorem (4.5.1), the theorem (1.5.6), the above lemmas and the following additional clauses of the algorithm in appendix (4) :
(D.2), (D.3), (D.11), (D.12), (D.15), (D.16), (D.18), (D.19), (D.22), (D.23), (D.28), (D.29), (D.32), (D.33), (D.52), (D.56), (D.57), (D.64), (D.65), (D.73), (D.75), (D.77), (D.79), (D.81.3), (D.82.3) and (D.82.4), there is a unique equivalent normal form for every derivation in Dccl(X).

###

The following example is a session with the system to construct the normal equivalent proof for a derivation of the class Dccl(X), illustrating above theorems for cut-elimination and normalization :

<u>Example 1</u>

Let a,b,c,d be objects of the category X, f in X(a,b), g in X(c,d): then the following derivation which has an instance of (R14) whose active formula in the right-hand premise is (T ↑ T) which is isomorphic to T :

```
                                      f
                                    a -> b
                                  ----------
                                  a,T -> b
                                -------------
                                a,e,T -> b
                              ---------------
                              a,e,T,j -> b
                            -----------------
                            a ↑ e,T,j -> b
                          -------------------
                          a ↑ e,T,T,j -> b
          g             ---------------------
        c -> d          a ↑ e,T ↑ T,j -> b
        ------------------------------------
          a ↑ e,c,d => (T ↑ T),j -> b
```

is transformed to its equivalent normal form, as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- prove(s1 = [g, [c] => [d]]).
yes
| ?- prove(s2 = [f, [a] => [b]]).
yes
| ?- prove(s3 = [th(s2,tr), [a,tr] => [b]]).
yes
| ?- prove(s4 = [th(s3,e), [a,e,tr] => [b]]).
yes
| ?- prove(s5 = [th(s4,j), [a,e,tr,j] => [b]]).
yes
| ?- prove(s6 = [aia(s5,a & e), [a & e,tr,j] => [b]]).
yes
| ?- prove(s7 = [th(s6,tr), [a & e,tr,tr,j] => [b]]).
yes
| ?- prove(s8 = [aia(s7,tr & tr), [a & e,tr & tr,j] => [b]]).
yes
| ?- prove(s=[hia(s1,s8,d->(tr&tr)),[a&e,c,d->(tr&tr),j] => [b]]).
yes
| ?- normal(s).

Proof is as follows :
====================

s1=[g,[c]=>[d]]
s2=[f,[a]=>[b]]
s3=[th(s2,tr),[a,tr]=>[b]]
s4=[th(s3,e),[a,e,tr]=>[b]]
s5=[th(s4,j),[a,e,tr,j]=>[b]]
s6=[aia(s5,a&e),[a&e,tr,j]=>[b]]
s7=[th(s6,tr),[a&e,tr,tr,j]=>[b]]
s8=[aia(s7,tr&tr),[a&e,tr&tr,j]=>[b]]
s=[hia(s1,s8,d->tr&tr),[a&e,c,d->tr&tr,j]=>[b]]


Cut_free proof is :
==================

s1=[g,[c]=>[d]]
s2=[f,[a]=>[b]]
s3=[th(s2,tr),[a,tr]=>[b]]
s4=[th(s3,e),[a,e,tr]=>[b]]
s5=[th(s4,j),[a,e,tr,j]=>[b]]
s6=[aia(s5,a&e),[a&e,tr,j]=>[b]]
s7=[th(s6,tr),[a&e,tr,tr,j]=>[b]]
s8=[aia(s7,tr&tr),[a&e,tr&tr,j]=>[b]]
s=[hia(s1,s8,d->tr&tr),[a&e,c,d->tr&tr,j]=>[b]]


Normal proof is :
================

f21=[f,[a]=>[b]]
f18=[th(f21,j),[a,j]=>[b]]
f15=[th(f18,e),[a,e,j]=>[b]]
f10=[aia(f15,a&e),[a&e,j]=>[b]]
f5=[th(f10,d->tr&tr),[a&e,d->tr&tr,j]=>[b]]
s=[th(f5,c),[a&e,c,d->tr&tr,j]=>[b]]

yes
| ?-
```

which in tree form is

```
                                  f
                                a -> b
                              ----------
                              a,j -> b
                            ------------
                            a,e,j -> b
                          --------------
                          a ↑ e,j -> b
          ----------------------------
          a ↑ e,d => (T ↑ T),j -> b
        ------------------------------
        a ↑ e,c,d => (T ↑ T),j -> b
```

Example 2

Let a,b,c,d be objects of the category X, f in X(a,b), g in X(c,d); then the following derivation

```
                                        f             g
                                      a -> b        c -> d
                                      ---------------------
                                         a,b => c -> d
                                    ---------------------
                                     a ↑ (b => c) -> d
             f                    -----------------------
           a -> b                 c,a ↑ (b => c) -> d
          -------------------------------------------
                 a,b => c,a ↑ (b => c) -> d
           -----------------------------------
           a ↑ (b => c),a ↑ (b => c) -> d
           -----------------------------------
                   a ↑ (b => c) -> d
```

is represented, and then transformed to its normal form, as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- prove(s1 = [f, [a] => [b]]).
yes
| ?- prove(s2 = [g, [c] => [d]]).
yes
| ?- prove(s3 = [hia(s1,s2,b -> c), [a,b -> c] => [d]]).
yes
| ?- prove(s4 = [aia(s3,a&(b->c)), [a&(b->c)] => [d]]).
yes
| ?- prove(s5 = [th(s4,c), [c,a&(b->c)] => [d]]).
yes
| ?- prove(s6 = [hia(s1,s5,b->c),[a,b->c,a&(b->c)] => [d]]).
yes
| ?- prove(s7 =[aia(s6,a&(b->c)),[a&(b->c),a&(b->c)]=>[d]]).
yes
| ?- prove(s = [con(s7,a&(b->c)), [a&(b->c)] => [d]]).
yes
| ?- normal(s).

Proof is as follows :
======================

f2=[f,[a]=>[b]]
f1=[f,[a]=>[b]]
s2=[g,[c]=>[d]]
s3=[hia(f1,s2,b->c),[a,b->c]=>[d]]
s4=[aia(s3,a&(b->c)),[a&(b->c)]=>[d]]
s5=[th(s4,c),[c,a&(b->c)]=>[d]]
s6=[hia(f2,s5,b->c),[a,b->c,a&(b->c)]=>[d]]
s7=[aia(s6,a&(b->c)),[a&(b->c),a&(b->c)]=>[d]]
s=[con(s7,a&(b->c)),[a&(b->c)]=>[d]]

Cut_free proof is :
===================

f2=[f,[a]=>[b]]
f1=[f,[a]=>[b]]
s2=[g,[c]=>[d]]
s3=[hia(f1,s2,b->c),[a,b->c]=>[d]]
s4=[aia(s3,a&(b->c)),[a&(b->c)]=>[d]]
s5=[th(s4,c),[c,a&(b->c)]=>[d]]
s6=[hia(f2,s5,b->c),[a,b->c,a&(b->c)]=>[d]]
s7=[aia(s6,a&(b->c)),[a&(b->c),a&(b->c)]=>[d]]
s=[con(s7,a&(b->c)),[a&(b->c)]=>[d]]

Normal proof is :
=================

f1=[f,[a]=>[b]]
s2=[g,[c]=>[d]]
s3=[hia(f1,s2,b->c),[a,b->c]=>[d]]
s=[aia(s3,a&(b->c)),[a&(b->c)]=>[d]]
```

which in tree form is

```
      f           g
   a -> b      c -> d
   ----------------------
      a,b => c -> d
   -------------------
   a ↑ (b => c) -> d
```

Example 3

Let a,b,c,d be objects of the category X; then the following derivation :

```
                                          a -> a
                                         --------
                                          a,b -> a
                                        ----------
        b -> b                          a ↑ b -> a
      --------                        ------------
       c,b -> b                        a ↑ b,c -> a
     ----------                      --------------
     c ↑ b -> b                      a ↑ b,d,c -> a
     ------------------------------------------------
            a ↑ b,c ↑ b,b => d,c -> a
            ---------------------------
            c ↑ b,a ↑ b,b => d,c -> a
```

is transformed to its equivalent normal form, as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- prove(s1 = [id(b), [b] => [b]]).
yes
| ?- prove(s2 = [th(s1,c), [c,b] => [b]]).
yes
| ?- prove(s3 = [ala(s2,c & b), [c & b] => [b]]).
yes
| ?- prove(s4 = [id(a), [a] => [a]]).

| ?- prove(s5 = [th(s4,b), [a,b] => [a]]).
yes
| ?- prove(s6 = [ala(s5,a & b), [a & b] => [a]]).
yes
| ?- prove(s7 = [th(s6,c), [a & b,c] => [a]]).
yes
| ?- prove(s8 = [th(s7,d), [a&b,d,c] => [a]]).
yes
| ?- prove(s9 = [hia(s3,s8,b->d), [a&b,c&b,b->d,c] => [a]]).
yes
| ?- prove(s = [inc(s9,c&b,a&b), [c&b,a&b,b->d,c] => [a]]).
yes
| ?- normal(s).

Proof is as follows :
=====================

s1=[id(b),[b]=>[b]]
s2=[th(s1,c),[c,b]=>[b]]
s3=[ala(s2,c&b),[c&b]=>[b]]
s4=[id(a),[a]=>[a]]
s5=[th(s4,b),[a,b]=>[a]]
s6=[ala(s5,a&b),[a&b]=>[a]]
s7=[th(s6,c),[a&b,c]=>[a]]
s8=[th(s7,d),[a&b,d,c]=>[a]]
s9=[hia(s3,s8,b->d),[a&b,c&b,b->d,c]=>[a]]
s=[inc(s9,c&b,a&b),[c&b,a&b,b->d,c]=>[a]]

Cut_free proof is :
===================

s1=[id(b),[b]=>[b]]
s2=[th(s1,c),[c,b]=>[b]]
s3=[ala(s2,c&b),[c&b]=>[b]]
s4=[id(a),[a]=>[a]]
```

```
s5=[th(s4,b),[a,b]=>[a]]
s6=[ala(s5,a&b),[a&b]=>[a]]
s7=[th(s6,c),[a&b,c]=>[a]]
s8=[th(s7,d),[a&b,d,c]=>[a]]
s9=[hla(s3,s8,b->d),[a&b,c&b,b->d,c]=>[a]]
s=[inc(s9,c&b,a&b),[c&b,a&b,b->d,c]=>[a]]

Normal proof is :
=================

s4=[id(a),[a]=>[a]]
s5=[th(s4,b),[a,b]=>[a]]
s6=[ala(s5,a&b),[a&b]=>[a]]
s7=[th(s6,c),[a&b,c]=>[a]]
s3=[th(s7,b->d),[a&b,b->d,c]=>[a]]
f2=[th(s3,b),[b,a&b,b->d,c]=>[a]]
f1=[th(f2,c),[c,b,a&b,b->d,c]=>[a]]
s=[ala(f1,c&b),[c&b,a&b,b->d,c]=>[a]]

yes
| ?-
```

which in tree form is

```
                              a -> a
                           ----------
                            a,b -> a
                         ------------
                         a ↑ b -> a
                       ------------
                       a ↑ b,c -> a
              ---------------------
              a ↑ b,b => d,c -> a
           ------------------------
           b,a ↑ b,b => d,c -> a
         --------------------------
         c,b,a ↑ b,b => d,c -> a
      -----------------------------
      c ↑ b,a ↑ b,b => d,c -> a
```

Example 4

For f in X(a,b), g in X(c,d) and c,d,e are objects of the category X; the derivation

```
                  c -> c                         e -> e
                ----------                     ----------
                c,e -> c                       c,e -> e
              ------------        g          ------------
              c ↑ e -> c       c -> d        c ↑ e -> e     e -> e
              ----------------------        -------------------------
                  c ↑ e -> d                      c ↑ e -> e
    f             ------------------------------------------
 a -> b                       c ↑ e -> d ↑ e                          d -> d
 ----------------------------------------------------               ----------
        a,b => (c ↑ e) -> d ↑ e                                      d,e -> d
   -----------------------------------                             -----------
   a ↑ (b => (c ↑ e)) -> d ↑ e                                     d ↑ e -> d
   ---------------------------------------------------------------------------
                         a ↑ (b => (c ↑ e)) -> d
```

is transformed to its equivalent normal form, as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- prove(s1 = [f, [a] => [b]]).
yes
| ?- prove(s2 = [id(c), [c] => [c]]).
yes
| ?- prove(s3 = [th(s2,e), [c,e] => [c]]).
yes
| ?- prove(s4 = [ala(s3,c & e), [c & e] => [c]]).
```

```
yes.
| ?- prove(s5 = [g, [c] => [d]]).
yes
| ?- prove(s6 = [cut(s4,s5,c), [c & e] => [d]]).
yes
| ?- prove(s7 = [id(e), [e] => [e]]).
yes
| ?- prove(s8 = [th(s7,c), [c,e] => [e]]).
yes
| ?- prove(s9 = [ala(s8,c & e), [c & e] => [e]]).
yes
| ?- prove(s10 = [id(e), [e] => [e]]).
yes
| ?- prove(s11 = [cut(s9,s10,e), [c & e] => [e]]).
yes
| ?- prove(s12 = [ais(s6,s11,d & e), [c & e] => [d & e]]).
yes
| ?- prove(s13 = [hia(s1,s12,b->(c&e)), [a,b -> (c&e)] => [d&e]]).
yes
| ?- prove(s14 = [ala(s13,a&(b->(c&e))),[a&(b->(c&e))] => [d&e]]).
yes
| ?- prove(s15 = [id(d), [d] => [d]]).
yes
| ?- prove(s16 = [th(s15,e), [d,e] => [d]]).
yes
| ?- prove(s17 = [ala(s16,d & e), [d & e] => [d]]).
yes
| ?- prove(s = [cut(s14,s17,d&e), [a&(b->(c&e))] => [d]]).
yes
| ?- normal(s).

Proof is as follows :
=====================

s1=[f,[a]=>[b]]
s2=[id(c),[c]=>[c]]
s3=[th(s2,e),[c,e]=>[c]]
s4=[ala(s3,c&e),[c&e]=>[c]]
s5=[g,[c]=>[d]]
s6=[cut(s4,s5,c),[c&e]=>[d]]
s7=[id(e),[e]=>[e]]
s8=[th(s7,c),[c,e]=>[e]]
s9=[ala(s8,c&e),[c&e]=>[e]]
s10=[id(e),[e]=>[e]]
s11=[cut(s9,s10,e),[c&e]=>[e]]
s12=[ais(s6,s11,d&e),[c&e]=>[d&e]]
s13=[hia(s1,s12,b->c&e),[a,b->c&e]=>[d&e]]
s14=[ala(s13,a&(b->c&e)),[a&(b->c&e)]=>[d&e]]
s15=[id(d),[d]=>[d]]
s16=[th(s15,e),[d,e]=>[d]]
s17=[ala(s16,d&e),[d&e]=>[d]]
s=[cut(s14,s17,d&e),[a&(b->c&e)]=>[d]]

Cut_free proof is :
===================

f19=[f,[a]=>[b]]
f48=[g,[c]=>[d]]
f45=[th(f48,e),[c,e]=>[d]]
f42=[ala(f45,c&e),[c&e]=>[d]]
f39=[th(f42,e),[c&e,e]=>[d]]
f36=[th(f39,c),[c&e,c,e]=>[d]]
f24=[ala(f36,c&e),[c&e,c&e]=>[d]]
f18=[con(f24,c&e),[c&e]=>[d]]
f15=[hia(f19,f18,b->c&e),[a,b->c&e]=>[d]]
s=[ala(f15,a&(b->c&e)),[a&(b->c&e)]=>[d]]

Normal proof is :
=================

f19=[f,[a]=>[b]]
f48=[g,[c]=>[d]]
f45=[th(f48,e),[c,e]=>[d]]
f18=[ala(f45,c&e),[c&e]=>[d]]
f15=[hia(f19,f18,b->c&e),[a,b->c&e]=>[d]]
s=[ala(f15,a&(b->c&e)),[a&(b->c&e)]=>[d]]
```

which in tree form is :

```
                                  g
                                c -> d
                              ----------
                              c,e -> d
           f                 -----------
         a -> b              c ↑ e -> d
         ---------------------------------
              a,b => (c ↑ e) -> d
         ---------------------------------
         a ↑ (b => (c ↑ e)) -> d
```

(4.5.9) The CHURCH-ROSSER Theorem for Dccl(X)

If p == q, then there exists a normal r in Dccl(X) such that p >= r and q >= r.

(4.6) Applications to Category theory

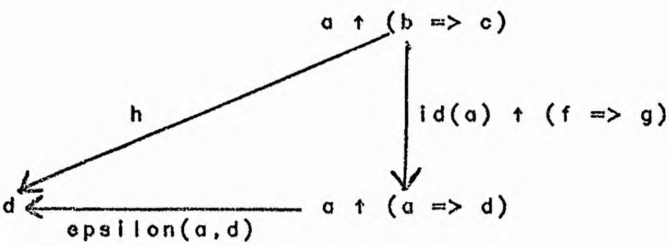
In this section, we illustrate the use of the system to represent some arrows of Fccl(X), to obtain their normal representations and (where possible) to show the equality of arrows of Fccl(X).

Example 1

For arrows f in X(a,b) and g in X(c,d), the normal representation of the arrow

h = comp(epsilon(a,d),product(id(a),hook(f,g))) : a ↑ (b => c) -> d

in the diagram



is obtained, as follows :

```
yes
| ?- theory(cartcl).
yes
| ?- arrow(a,f,b).
yes
| ?- arrow(c,g,d).
yes
| ?- norm_rep(comp(epsilon(a,d),product(id(a),hook(f,g)))).
The arrow is :
============

comp(epsilon(a,d),product(id(a),hook(f,g)))

Proof is as follows :
===================

f24=[id(a),[a]=>[a]]
f23=[th(f24,b->c),[a,b->c]=>[a]]
f22=[ala(f23,a&(b->c)),[a&(b->c)]=>[a]]
f7=[id(a),[a]=>[a]]
f25=[cut(f22,f7,a),[a&(b->c)]=>[a]]
f30=[id(b),[b]=>[b]]
f31=[id(c),[c]=>[c]]
f29=[hia(f30,f31,b->c),[b,b->c]=>[c]]
f28=[his(f29,b->c),[b->c]=>[b->c]]
f27=[th(f28,a),[a,b->c]=>[b->c]]
f26=[ala(f27,a&(b->c)),[a&(b->c)]=>[b->c]]
f17=[id(a),[a]=>[a]]
```

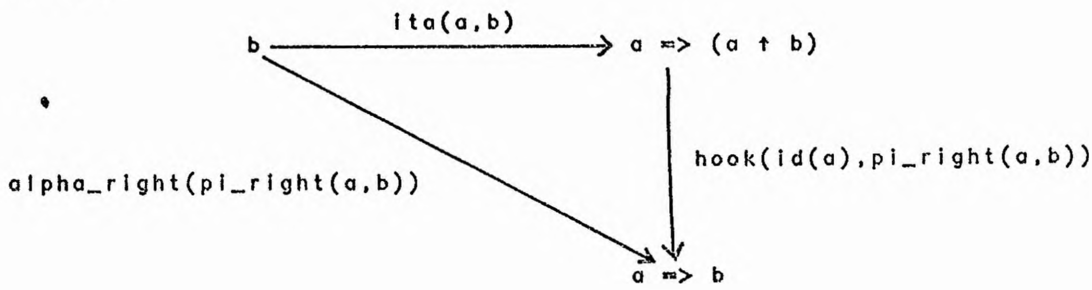
```
f15=[th(f17,b->c),[a,b->c]=>[a]]
f20=[id(b),[b]=>[b]]
f21=[id(c),[c]=>[c]]
f19=[hia(f20,f21,b->c),[b,b->c]=>[c]]
f18=[his(f19,b->c),[b->c]=>[b->c]]
f16=[th(f18,a),[a,b->c]=>[b->c]]
f14=[ais(f15,f16,a&(b->c)),[a,b->c]=>[a&(b->c)]]
f10=[f,[a]=>[b]]
f11=[g,[c]=>[d]]
f12=[hia(f10,f11,b->c),[a,b->c]=>[d]]
f9=[aia(f12,a&(b->c)),[a&(b->c)]=>[d]]
f13=[cut(f14,f9,a&(b->c)),[a,b->c]=>[d]]
f8=[his(f13,a->d),[b->c]=>[a->d]]
f32=[cut(f26,f8,b->c),[a&(b->c)]=>[a->d]]
f6=[ais(f25,f32,a&(a->d)),[a&(b->c)]=>[a&(a->d)]]
f4=[id(a),[a]=>[a]]
f5=[id(d),[d]=>[d]]
f3=[hia(f4,f5,a->d),[a,a->d]=>[d]]
f2=[aia(f3,a&(a->d)),[a&(a->d)]=>[d]]
f1=[cut(f6,f2,a&(a->d)),[a&(b->c)]=>[d]]

Cut_free proof is :
===================

f251=[f,[a]=>[b]]
f248=[th(f251,b->c),[a,b->c]=>[b]]
f245=[aia(f248,a&(b->c)),[a&(b->c)]=>[b]]
f242=[th(f245,b->c),[a&(b->c),b->c]=>[b]]
f239=[th(f242,a),[a&(b->c),a,b->c]=>[b]]
f235=[aia(f239,a&(b->c)),[a&(b->c),a&(b->c)]=>[b]]
f236=[g,[c]=>[d]]
f232=[hia(f235,f236,b->c),[a&(b->c),a&(b->c),b->c]=>[d]]
f229=[th(f232,a),[a&(b->c),a&(b->c),a,b->c]=>[d]]
f226=[aia(f229,a&(b->c)),[a&(b->c),a&(b->c),a&(b->c)]=>[d]]
f223=[th(f226,a),[a&(b->c),a&(b->c),a,a&(b->c)]=>[d]]
f220=[inc(f223,a,a&(b->c)),[a&(b->c),a,a&(b->c),a&(b->c)]=>[d]]
f217=[th(f220,b->c),[a&(b->c),a,b->c,a&(b->c),a&(b->c)]=>[d]]
f202=[aia(f217,a&(b->c)),[a&(b->c),a&(b->c),a&(b->c),a&(b->c)]=>[d]]
f197=[con(f202,a&(b->c)),[a&(b->c),a&(b->c),a&(b->c)]=>[d]]
f147=[con(f197,a&(b->c)),[a&(b->c),a&(b->c)]=>[d]]
f1=[con(f147,a&(b->c)),[a&(b->c)]=>[d]]

Normal proof is :
=================

f251=[f,[a]=>[b]]
f236=[g,[c]=>[d]]
f248=[hia(f251,f236,b->c),[a,b->c]=>[d]]
f1=[aia(f248,a&(b->c)),[a&(b->c)]=>[d]]
```

which in tree form is :

```
   f            g
a -> b        c --> d
-------------------
      a,b => c -> d
------------------
a ↑ (b => c) -> d
```

Example 2

Let a,b be objects of the category X; then the following diagram in Fcl(X)



commutes .

i.e by the use of the system we prove that

alpha_right(pi_right(a,b)) = comp(hook(id(a),pi_right(a,b)),ita(a,b))

as follows :

```
yes
| ?- theory(cartcl).

yes
| ?- commutative.

Insert the first arrow : alpha_right(pi_right(a,b)).


The arrow is :
==============

alpha_right(pi_right(a,b))


Proof is as follows :
=====================

f9=[id(a),[a]=>[a]]
f7=[th(f9,b),[a,b]=>[a]]
f10=[id(b),[b]=>[b]]
f8=[th(f10,a),[a,b]=>[b]]
f6=[ais(f7,f8,a&b),[a,b]=>[a&b]]
f4=[id(b),[b]=>[b]]
f3=[th(f4,a),[a,b]=>[b]]
f2=[aia(f3,a&b),[a&b]=>[b]]
f5=[cut(f6,f2,a&b),[a,b]=>[b]]
f1=[his(f5,a->b),[b]=>[a->b]]


Cut_free proof is :
===================

f33=[id(b),[b]=>[b]]
f32=[th(f33,a),[a,b]=>[b]]
f29=[th(f32,b),[a,b,b]=>[b]]
f15=[th(f29,a),[a,b,a,b]=>[b]]
f18=[inc(f15,a,b),[a,a,b,b]=>[b]]
f19=[con(f18,a),[a,b,b]=>[b]]
f5=[con(f19,b),[a,b]=>[b]]
f1=[his(f5,a->b),[b]=>[a->b]]


Normal proof is :
=================

f33=[id(b),[b]=>[b]]
f5=[th(f33,a),[a,b]=>[b]]
f1=[his(f5,a->b),[b]=>[a->b]]


Insert the second arrow : comp(hook(id(a),pi_right(a,b)),
|:                              ita(a,b)).


The arrow is :
==============

comp(hook(id(a),pi_right(a,b)),ita(a,b))


Proof is as follows :
=====================

f60=[id(a),[a]=>[a]]
f58=[th(f60,b),[a,b]=>[a]]
f61=[id(b),[b]=>[b]]
f59=[th(f61,a),[a,b]=>[b]]
```

```
f57=[als(f58,f59,a&b),[a,b]=>[a&b]]
f56=[his(f57,a->a&b),[b]=>[a->a&b]]
f46=[id(a),[a]=>[a]]
f44=[th(f46,a->a&b),[a,a->a&b]=>[a]]
f49=[id(a),[a]=>[a]]
f54=[id(a),[a]=>[a]]
f52=[th(f54,b),[a,b]=>[a]]
f55=[id(b),[b]=>[b]]
f53=[th(f55,a),[a,b]=>[b]]
f51=[ais(f52,f53,a&b),[a,b]=>[a&b]]
f50=[aia(f51,a&b),[a&b]=>[a&b]]
f48=[hia(f49,f50,a->a&b),[a,a->a&b]=>[a&b]]
f47=[his(f48,a->a&b),[a->a&b]=>[a->a&b]]
f45=[th(f47,a),[a,a->a&b]=>[a->a&b]]
f43=[ais(f44,f45,a&(a->a&b)),[a,a->a&b]=>[a&(a->a&b)]]
f37=[id(a),[a]=>[a]]
f40=[id(b),[b]=>[b]]
f39=[th(f40,a),[a,b]=>[b]]
f38=[aia(f39,a&b),[a&b]=>[b]]
f41=[hia(f37,f38,a->a&b),[a,a->a&b]=>[b]]
f36=[aia(f41,a&(a->a&b)),[a&(a->a&b)]=>[b]]
f42=[cut(f43,f36,a&(a->a&b)),[a,a->a&b]=>[b]]
f35=[his(f42,a->b),[a->a&b]=>[a->b]]
f34=[cut(f56,f35,a->a&b),[b]=>[a->b]]
```

Cut_free proof is :
======================

```
f281=[id(b),[b]=>[b]]
f438=[th(f281,a),[a,b]=>[b]]
f437=[th(f438,b),[a,b,b]=>[b]]
f434=[th(f437,b),[a,b,b,b]=>[b]]
f431=[th(f434,b),[a,b,b,b,b]=>[b]]
f428=[th(f431,a),[a,b,b,a,b,b]=>[b]]
f425=[th(f428,b),[a,b,b,b,a,b,b]=>[b]]
f422=[th(f425,a),[a,b,b,a,b,a,b,b]=>[b]]
f419=[th(f422,b),[a,b,b,a,b,b,a,b,b]=>[b]]
f416=[th(f419,a),[a,b,b,a,b,b,a,a,b,b]=>[b]]
f413=[inc(f416,a,b),[a,b,b,a,b,a,b,a,b,b]=>[b]]
f410=[inc(f413,a,b),[a,b,b,a,a,b,b,a,b,b]=>[b]]
f407=[con(f410,a),[a,b,b,a,b,b,a,b,b]=>[b]]
f404=[inc(f407,a,b),[a,b,b,a,b,a,b,b,b]=>[b]]
f401=[inc(f404,a,b),[a,b,b,a,a,b,b,b,b]=>[b]]
f398=[con(f401,a),[a,b,b,a,b,b,b,b]=>[b]]
f395=[con(f398,b),[a,b,b,a,b,b,b]=>[b]]
f392=[con(f395,b),[a,b,b,a,b,b]=>[b]]
f389=[inc(f392,a,b),[a,b,a,b,b,b]=>[b]]
f386=[inc(f389,a,b),[a,a,b,b,b,b]=>[b]]
f298=[con(f386,a),[a,b,b,b,b]=>[b]]
f293=[con(f298,b),[a,b,b,b]=>[b]]
f290=[con(f293,b),[a,b,b]=>[b]]
f287=[th(f290,a),[a,b,a,b]=>[b]]
f284=[inc(f287,a,b),[a,a,b,b]=>[b]]
f131=[con(f284,a),[a,b,b]=>[b]]
f126=[con(f131,b),[a,b]=>[b]]
f34=[his(f126,a->b),[b]=>[a->b]]
```

Normal proof is :
=================

```
f131=[id(b),[b]=>[b]]
f126=[th(f131,a),[a,b]=>[b]]
f34=[his(f126,a->b),[b]=>[a->b]]
```

The answer is :
================

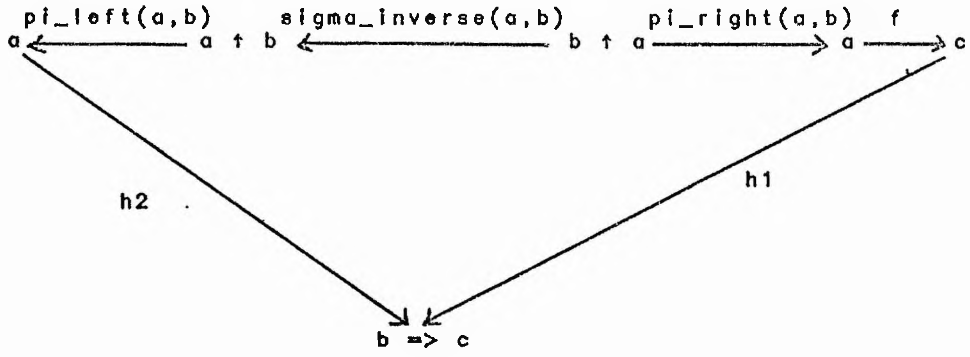
```
alpha_right(pi_right(a,b)) =
comp(hook(id(a),pi_right(a,b)),ita(a,b))
```

because they have the same normal representation .

```
yes
| ?-
```

Example 3

Suppose a,b,c are objects of the category X; and that f in X(a,c). Then the following diagram of the free cartesian closed category Fccl(X)



where

h1 = alpha_right(comp(f,pi_right(b,a))),

h2 = alpha_right(comp(f,comp(pi_left(a,b),sigma_inverse(a,b))))

commutes.

i.e. by the use of the system we can prove

```
    comp(alpha_right(comp(f,pi_right(b,a))),pi_right(b,a))
  = comp(comp(alpha_right(comp(comp(f,pi_left(a,b)),
                                      sigma_inverse(a,b))),
              pi_left(a,b)),
          sigma_inverse(a,b))
```

as follows :

```
yes
| ?- theory(cartcl).

yes
| ?- arrow(a,f,c).

yes
| ?- commutative.
Insert the first arrow : comp(alpha_right(comp(f,pi_right(b,a))),
|:                                 pi_right(b,a)).


The arrow is :
==============

comp(alpha_right(comp(f,pi_right(b,a))),pi_right(b,a))


Proof is as follows :
=====================

f16=[id(a),[a]=>[a]]
f15=[th(f16,b),[b,a]=>[a]]
f14=[aia(f15,b&a),[b&a]=>[a]]
f12=[id(b),[b]=>[b]]
f10=[th(f12,a),[b,a]=>[b]]
f13=[id(a),[a]=>[a]]
f11=[th(f13,b),[b,a]=>[a]]
f9=[ais(f10,f11,b&a),[b,a]=>[b&a]]
f7=[id(a),[a]=>[a]]
```

```
f6=[th(f7,b),[b,a]=>[a]]
f5=[aia(f6,b&a),[b&a]=>[a]]
f4=[f,[a]=>[c]]
f3=[cut(f5,f4,a),[b&a]=>[c]]
f8=[cut(f9,f3,b&a),[b,a]=>[c]]
f2=[his(f8,b->c),[a]=>[b->c]]
f1=[cut(f14,f2,a),[b&a]=>[b->c]]
```

```
Cut_free proof is :
===================

f76=[f,[a]=>[c]]
f73=[th(f76,b),[b,a]=>[c]]
f70=[aia(f73,b&a),[b&a]=>[c]]
f70=[aia(f73,b&a),[b&a]=>[c]]
f91=[th(f92,a),[b,a,b&a]=>[c]]
f88=[th(f91,b),[b,b,a,b&a]=>[c]]
f85=[aia(f88,b&a),[b,b&a,b&a]=>[c]]
f82=[th(f85,b),[b,b&a,b,b&a]=>[c]]
f79=[inc(f82,b,b&a),[b,b,b&a,b&a]=>[c]]
f53=[con(f79,b),[b,b&a,b&a]=>[c]]
f48=[con(f53,b&a),[b,b&a]=>[c]]
f1=[his(f48,b->c),[b&a]=>[b->c]]
```

```
Normal proof is :
=================

f76=[f,[a]=>[c]]
f73=[th(f76,b),[b,a]=>[c]]
f53=[aia(f73,b&a),[b&a]=>[c]]
f48=[th(f53,b),[b,b&a]=>[c]]
f1=[his(f48,b->c),[b&a]=>[b->c]]
```

```
Insert the second arrow :
|:          comp(comp(alpha_right(comp(comp(f,pi_left(a,b)),
|:                                              sigma_inverse(a,b))),
|:                    pi_left(a,b)),
|:               sigma_inverse(a,b)).
```

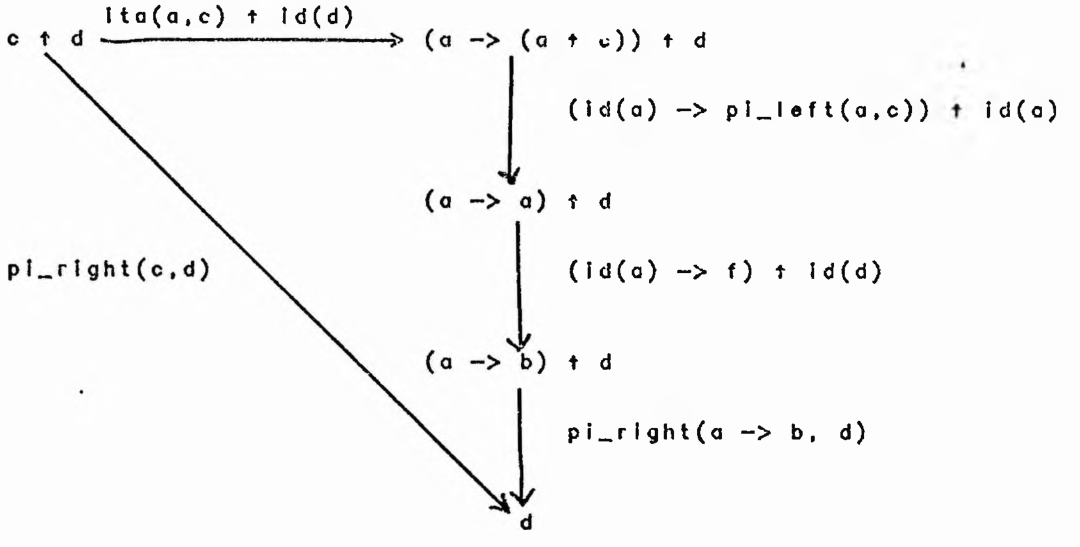
```
The arrow is :
==============

comp(comp(alpha_right(comp(comp(f,pi_left(a,b)),
                                  sigma_inverse(a,b))),
          pi_left(a,b)),
     sigma_inverse(a,b))
```

```
Proof is as follows :
=====================

f121=[id(a),[a]=>[a]]
f120=[th(f121,b),[b,a]=>[a]]
f119=[aia(f120,b&a),[b&a]=>[a]]
f124=[id(b),[b]=>[b]]
f123=[th(f124,a),[b,a]=>[b]]
f122=[aia(f123,b&a),[b&a]=>[b]]
f118=[ais(f119,f122,a&b),[b&a]=>[a&b]]
f117=[id(a),[a]=>[a]]
f116=[th(f117,b),[a,b]=>[a]]
f115=[aia(f116,a&b),[a&b]=>[a]]
f113=[id(b),[b]=>[b]]
f111=[th(f113,a),[b,a]=>[b]]
f114=[id(a),[a]=>[a]]
f112=[th(f114,b),[b,a]=>[a]]
f110=[ais(f111,f112,b&a),[b,a]=>[b&a]]
f105=[id(a),[a]=>[a]]
f104=[th(f105,b),[b,a]=>[a]]
f103=[aia(f104,b&a),[b&a]=>[a]]
f108=[id(b),[b]=>[b]]
f107=[th(f108,a),[b,a]=>[b]]
f106=[aia(f107,b&a),[b&a]=>[b]]
f102=[ais(f103,f106,a&b),[b&a]=>[a&b]]
f101=[id(a),[a]=>[a]]
f100=[th(f101,b),[a,b]=>[a]]
f99=[aia(f100,a&b),[a&b]=>[a]]
f98=[f,[a]=>[c]]
f97=[cut(f99,f98,a),[a&b]=>[c]]
```

```
f96=[cut(f102,f97,a&b),[b&a]=>[c]]
f109=[cut(f110,f96,b&a),[b,a]=>[c]]
f95=[his(f109,b->c),[a]=>[b->c]]
f94=[cut(f115,f95,a),[a&b]=>[b->c]]
f93=[cut(f118,f94,a&b),[b&a]=>[b->c]]


Cut_free proof is :
===================

f608=[f,[a]=>[c]]
f605=[th(f608,b),[b,a]=>[c]]
f602=[aia(f605,b&a),[b&a]=>[c]]
f599=[th(f602,b),[b&a,b]=>[c]]
f596=[th(f599,a),[b&a,b,a]=>[c]]
f584=[aia(f596,b&a),[b&a,b&a]=>[c]]
f579=[con(f584,b&a),[b&a]=>[c]]
f647=[th(f579,b),[b,b&a]=>[c]]
f646=[th(f647,a),[b,a,b&a]=>[c]]
f643=[th(f646,b),[b,b,a,b&a]=>[c]]
f640=[aia(f643,b&a),[b,b&a,b&a]=>[c]]
f637=[th(f640,b),[b,b&a,b,b&a]=>[c]]
f634=[th(f637,a),[b,b&a,b,a,b&a]=>[c]]
f622=[aia(f634,b&a),[b,b&a,b&a,b&a]=>[c]]
f617=[con(f622,b&a),[b,b&a,b&a]=>[c]]
f614=[th(f617,b),[b,b&a,b,b&a]=>[c]]
f611=[inc(f614,b,b&a),[b,b,b&a,b&a]=>[c]]
f556=[con(f611,b),[b,b&a,b&a]=>[c]]
f551=[con(f556,b&a),[b,b&a]=>[c]]
f548=[th(f551,b),[b,b&a,b]=>[c]]
f545=[th(f548,a),[b,b&a,b,a]=>[c]]
f542=[th(f545,b),[b,b&a,b,b,a]=>[c]]
f539=[aia(f542,b&a),[b,b&a,b,b&a]=>[c]]
f536=[th(f539,b),[b,b&a,b,b&a,b]=>[c]]
f533=[th(f536,a),[b,b&a,b,b&a,b,a]=>[c]]
f530=[aia(f533,b&a),[b,b&a,b,b&a,b&a]=>[c]]
f527=[con(f530,b&a),[b,b&a,b,b&a]=>[c]]
f524=[th(f527,a),[b,b&a,b,a,b&a]=>[c]]
f521=[th(f524,b),[b,b&a,b,b,a,b&a]=>[c]]
f518=[aia(f521,b&a),[b,b&a,b,b&a,b&a]=>[c]]
f515=[th(f518,b),[b,b&a,b,b&a,b,b&a]=>[c]]
f512=[th(f515,a),[b,b&a,b,b&a,b,a,b&a]=>[c]]
f509=[aia(f512,b&a),[b,b&a,b,b&a,b&a,b&a]=>[c]]
f506=[con(f509,b&a),[b,b&a,b,b&a,b&a]=>[c]]
f503=[th(f506,b),[b,b&a,b,b&a,b,b&a]=>[c]]
f500=[inc(f503,b,b&a),[b,b&a,b,b,b&a,b&a]=>[c]]
f497=[con(f500,b),[b,b&a,b,b&a,b&a]=>[c]]
f494=[con(f497,b&a),[b,b&a,b,b&a]=>[c]]
f491=[inc(f494,b,b&a),[b,b,b&a,b&a]=>[c]]
f386=[con(f491,b),[b,b&a,b&a]=>[c]]
f381=[con(f386,b&a),[b,b&a]=>[c]]
f93=[his(f381,b->c),[b&a]=>[b->c]]


Normal proof is :
=================

f608=[f,[a]=>[c]]
f605=[th(f608,b),[b,a]=>[c]]
f556=[aia(f605,b&a),[b&a]=>[c]]
f381=[th(f556,b),[b,b&a]=>[c]]
f93=[his(f381,b->c),[b&a]=>[b->c]]


The answer is :
===============

comp(alpha_right(comp(f,pi_right(b,a))),pi_right(b,a)) =
comp(comp(alpha_right(comp(comp(f,pi_left(a,b)),
                                sigma_inverse(a,b))),
          pi_left(a,b)),
     sigma_inverse(a,b))

because they have the same normal representation .

yes
| ?-
```

Example 4

For f in X(a,b) and a,b,c,d are objects of the category X; then the following diagram of Fccl(X)



commutes.

```
yes
| ?- theory(cartcl).

yes
| ?- arrow(a,f,b).

yes
| ?- commutative.
Insert the first arrow : pi_right(c,d).


The arrow is :
==============

pi_right(c,d)

Proof is as follows :
=====================


f3=[id(d),[d]=>[d]]
f2=[th(f3,c),[c,d]=>[d]]
f1=[ala(f2,c&d),[c&d]=>[d]]

Cut_free proof is :
===================


f3=[id(d),[d]=>[d]]
f2=[th(f3,c),[c,d]=>[d]]
f1=[ala(f2,c&d),[c&d]=>[d]]

Normal proof is :
=================


f3=[id(d),[d]=>[d]]
f2=[th(f3,c),[c,d]=>[d]]
f1=[ala(f2,c&d),[c&d]=>[d]]
```

```
Insert the second arrow :
|:          comp(comp(comp(pi_right(a->b,d),
|:                             product(hook(id(a),f),id(d))),
|:                      product(hook(id(a),pi_left(a,c)),id(d))),
|:             product(ita(a,c),id(d))).
```

The arrow is :

```
comp(comp(comp(pi_right(a->b,d),product(hook(id(a),f),id(d))),
          product(hook(id(a),pi_left(a,c)),id(d))),
      product(ita(a,c),id(d)))
```

Proof is as follows :

```
f86=[id(c),[c]=>[c]]
f85=[th(f86,d),[c,d]=>[c]]
f84=[aia(f85,c&d),[c&d]=>[c]]
f81=[id(a),[a]=>[a]]
f79=[th(f81,c),[a,c]=>[a]]
f82=[id(c),[c]=>[c]]
f80=[th(f82,a),[a,c]=>[c]]
f78=[ais(f79,f80,a&c),[a,c]=>[a&c]]
f77=[his(f78,a->a&c),[c]=>[a->a&c]]
f87=[cut(f84,f77,c),[c&d]=>[a->a&c]]
f90=[id(d),[d]=>[d]]
f89=[th(f90,c),[c,d]=>[d]]
f88=[aia(f89,c&d),[c&d]=>[d]]
f83=[id(d),[d]=>[d]]
f91=[cut(f88,f83,d),[c&d]=>[d]]
f76=[ais(f87,f91,(a->a&c)&d),[c&d]=>[(a->a&c)&d]]
f64=[id(a),[a]=>[a]]
f69=[id(a),[a]=>[a]]
f67=[th(f69,c),[a,c]=>[a]]
f70=[id(c),[c]=>[c]]
f68=[th(f70,a);[a,c]=>[c]]
f66=[ais(f67,f68,a&c),[a,c]=>[a&c]]
f65=[aia(f66,a&c),[a&c]=>[a&c]]
f63=[hia(f64,f65,a->a&c),[a,a->a&c]=>[a&c]]
f62=[his(f63,a->a&c),[a->a&c]=>[a->a&c]]
f61=[th(f62,d),[a->a&c,d]=>[a->a&c]]
f60=[aia(f61,(a->a&c)&d),[(a->a&c)&d]=>[a->a&c]]
f49=[id(a),[a]=>[a]]
f47=[th(f49,a->a&c),[a,a->a&c]=>[a]]
f52=[id(a),[a]=>[a]]
f57=[id(a),[a]=>[a]]
f55=[th(f57,c),[a,c]=>[a]]
f58=[id(c),[c]=>[c]]
f56=[th(f58,a),[a,c]=>[c]]
f54=[ais(f55,f56,a&c),[a,c]=>[a&c]]
f53=[aia(f54,a&c),[a&c]=>[a&c]]
f51=[hia(f52,f53,a->a&c),[a,a->a&c]=>[a&c]]
f50=[his(f51,a->a&c),[a->a&c]=>[a->a&c]]
f48=[th(f50,a),[a,a->a&c]=>[a->a&c]]
f46=[ais(f47,f48,a&(a->a&c)),[a,a->a&c]=>[a&(a->a&c)]]
f40=[id(a),[a]=>[a]]
f43=[id(a),[a]=>[a]]
f42=[th(f43,c),[a,c]=>[a]]
f41=[aia(f42,a&c),[a&c]=>[a]]
f44=[hia(f40,f41,a->a&c),[a,a->a&c]=>[a]]
f39=[aia(f44,a&(a->a&c)),[a&(a->a&c)]=>[a]]
f45=[cut(f46,f39,a&(a->a&c)),[a,a->a&c]=>[a]]
f38=[his(f45,a->a),[a->a&c]=>[a->a]]
f71=[cut(f60,f38,a->a&c),[(a->a&c)&d]=>[a->a]]
f74=[id(d),[d]=>[d]]
f73=[th(f74,a->a&c),[a->a&c,d]=>[d]]
f72=[aia(f73,(a->a&c)&d),[(a->a&c)&d]=>[d]]
f59=[id(d),[d]=>[d]]
f75=[cut(f72,f59,d),[(a->a&c)&d]=>[d]]
f37=[ais(f71,f75,(a->a)&d),[(a->a&c)&d]=>[(a->a)&d]]
f30=[id(a),[a]=>[a]]
f31=[id(a),[a]=>[a]]
f29=[hia(f30,f31,a->a),[a,a->a]=>[a]]
f28=[his(f29,a->a),[a->a]=>[a->a]]
f27=[th(f28,d),[a->a,d]=>[a->a]]
f26=[aia(f27,(a->a)&d),[(a->a)&d]=>[a->a]]
f20=[id(a),[a]=>[a]]
f18=[th(f20,a->a),[a,a->a]=>[a]]
```

```
f23=[id(a),[a]=>[a]]
f24=[id(a),[a]=>[a]]
f22=[hia(f23,f24,a->a),[a,a->a]=>[a]]
f21=[his(f22,a->a),[a->a]=>[a->a]]
f19=[th(f21,a),[a,a->a]=>[a->a]]
f17=[ais(f18,f19,a&(a->a)),[a,a->a]=>[a&(a->a)]]
f13=[id(a),[a]=>[a]]
f14=[f,[a]=>[b]]
f15=[hia(f13,f14,a->a),[a,a->a]=>[b]]
f12=[aia(f15,a&(a->a)),[a&(a->a)]=>[b]]
f16=[cut(f17,f12,a&(a->a)),[a,a->a]=>[b]]
f11=[his(f16,a->b),[a->a]=>[a->b]]
f32=[cut(f26,f11,a->a),[(a->a)&d]=>[a->b]]
f35=[id(d),[d]=>[d]]
f34=[th(f35,a->a),[a->a,d]=>[d]]
f33=[aia(f34,(a->a)&d),[(a->a)&d]=>[d]]
f25=[id(d),[d]=>[d]]
f36=[cut(f33,f25,d),[(a->a)&d]=>[d]]
f10=[ais(f32,f36,(a->b)&d),[(a->a)&d]=>[(a->b)&d]]
f9=[id(d),[d]=>[d]]
f8=[th(f9,a->b),[a->b,d]=>[d]]
f7=[aia(f8,(a->b)&d),[(a->b)&d]=>[d]]
f6=[cut(f10,f7,(a->b)&d),[(a->a)&d]=>[d]]
f5=[cut(f37,f6,(a->a)&d),[(a->a&c)&d]=>[d]]
f4=[cut(f76,f5,(a->a&c)&d),[c&d]=>[d]]
```

Cut_free proof is :
==================

```
f695=[id(d),[d]=>[d]]
f692=[th(f695,c),[c,d]=>[d]]
f696=[aia(f692,c&d),[c&d]=>[d]]
f684=[th(f696,c&d),[c&d,c&d]=>[d]]
f679=[con(f684,c&d),[c&d]=>[d]]
f676=[th(f679,d),[c&d,d]=>[d]]
f673=[th(f676,c),[c&d,c,d]=>[d]]
f670=[aia(f673,c&d),[c&d,c&d]=>[d]]
f667=[th(f670,c&d),[c&d,c&d,c&d]=>[d]]
f641=[con(f667,c&d),[c&d,c&d]=>[d]]
f636=[con(f641,c&d),[c&d]=>[d]]
f633=[th(f636,d),[c&d,d]=>[d]]
f630=[th(f633,c),[c&d,c,d]=>[d]]
f627=[aia(f630,c&d),[c&d,c&d]=>[d]]
f624=[th(f627,c&d),[c&d,c&d,c&d]=>[d]]
f621=[con(f624,c&d),[c&d,c&d]=>[d]]
f618=[th(f621,d),[c&d,c&d,d]=>[d]]
f615=[th(f618,c),[c&d,c&d,c,d]=>[d]]
f612=[aia(f615,c&d),[c&d,c&d,c&d]=>[d]]
f609=[th(f612,c&d),[c&d,c&d,c&d,c&d]=>[d]]
f570=[con(f609,c&d),[c&d,c&d,c&d]=>[d]]
f570=[con(f609,c&d),[c&d,c&d,c&d]=>[d]]
f4=[con(f539,c&d),[c&d]=>[d]]
```

Normal proof is :
================

```
f695=[id(d),[d]=>[d]]
f692=[th(f695,c),[c,d]=>[d]]
f4=[aia(f692,c&d),[c&d]=>[d]]
```

The answer is :
=============

```
pi_right(c,d) =
comp(comp(comp(pi_right(a->b,d),product(hook(id(a),f),id(d))),
              product(hook(id(a),pi_left(a,c)),id(d))),
      product(ita(a,c),id(d)))
```

because they have the same normal representation .

```
yes
| ?-
```

Chapter 5

BICARTESIAN CLOSED CATEGORY THEORY

In this chapter, we combine all results and implementations of cartesian, bicartesian, distributive bicartesian and cartesian closed categories in order to complete the system to handle bicartesian closed categories. By a bicartesian category is meant a category with an initial and object $\perp$, a terminal object T, and binary operators $\dagger$, v, and $\Rightarrow$. The class of small bicartesian closed categories serves as a model for the proof theory of the full intuitionist propositional calculus.

We have constructed and implemented a proof for corollary (5.5.4), concerning derivations of the form $\rightarrow a$ where $a \approx \sim T$ (the terminal object of the free bicartesian closed category generated by X, Fbcc(X) (cf 5.2.4)). To complete the algorithm for computation of normal form, we have constructed and implemented a proof of lemma (5.5.6), stated without proof in SZABO .

(5.1) Definition

A bicartesian closed category is a bicartesian category $\underline{C}$ with the following additional structure :

(7) A bifunctor $(-) \Rightarrow (-) : op(\underline{C}) \times \underline{C} \rightarrow \underline{C}$.

(8) An adjunction α_λ , where

$$\alpha_\lambda = \{\alpha_\lambda(A,B,C) : \underline{C}(A \dagger B,\ C) \rightarrow \underline{C}(B,\ A \Rightarrow C) \in ArEns \mid A,B,C \in Ob\underline{C}\}.$$

(5.2) The Free Bicartesian Closed Category Fbccl(X)

Let X be a fixed but arbitrary small category. The free bicartesian closed category Fbccl(X) is constructed as follows :-

(5.2.1) Definition

The class bcclL(X) of (bicartesian closed) formulae over X is defined to contain all the (bicartesian) formulae of the class bcL(X) and the (cartesian closed) formulae of the class cclL(X) .

(5.2.2) The Labelled Derivations LDbccl(X)

The class LDbccl(X) of (bicartesian closed) labelled derivations over X, and their conclusions, is defined by conditions (i) - (viii) of (1.2.2), (ix) - (xii) of (2.2.2) and (ix) - (x) of (4.2.2) .

(5.2.3) Definition

The relation == is defined as the the smallest equivalence relation on LDbccl(X) satisfying conditions (i) - (viii) of (1.2.3), conditions (ix) - (xii) of (2.2.3) and conditions (ix) - (xvi) of (4.2.3).

(5.2.4) Definition

The free bicartesian closed category is defined to be the category having :

i) as objects , the bicartesian closed formulae over X ;

ii) as arrows , the ==- classes of LDbccl(X) ;

iii) - xi) as in the definition (1.2.5) of Fc(X) ;

xii) - xvi) as in the definition (2.2.4) of Fbc(X) ;

xii) - xiii) as in the definition (4.2.4) of Fccl(X) .

(5.3) The Unlabelled Derivations Dbccl(X)

The class Dbccl(X) of unlabelled (bicartesian closed) derivations is defined inductively by the conditions (i) - (xiii'), except (x) [we omit (x), since R7 is an admissible rule : if p is a derivation with conclusion G -> FabE, then

R7(p,b,a) = R1(R13(p,a v b),
R12(R5 (A1(id(a)), b),
R5 (A1(id(b)), a), a v b), a v b)]

of (3.3) and the conditions (x) - (xi) of (4.3.1) .

(5.4) The Semantics of Dbccl(X)

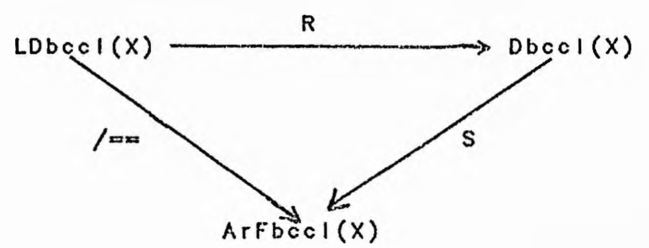
In order to interpret the derivations of Dbccl(X) in ArFbccl(X), we combine the interpretations of Dc(X), Dbc(X), Ddbc(X) and Dccl(X) to an interpretation S : Dbccl(X) -> ArFbccl(X) .

In the following theorem we prove that the class Dbccl(X) is adequate, in the sense that every arrow of Fbccl(X) is representable by means of some derivation of the class Dbccl(X) :

(5.4.1) The Completeness Theorem For Dbccl(X)

There is a function

R : LDbccl(X) -----------------> Dbccl(X) , so that



commutes .

Proof :

We combine the proofs of theorems (1.4.2), (2.4.2), (3.4.2) and (4.4.2) .

###

(5.5) The Syntax of Fbccl(X)

After obtaining a composition-free description of ArFbccl(X), in this section we describe the algorithm, to construct cut-free representations for ArFbccl(X) and to characterize the arrows of Fbccl(X) by constructing their normal representations.

(5.5.1) The Cut-Elimination Theorem for Dbccl(X)

For every derivation p in Dbccl(X), there exists an equivalent cut-free q in Dbccl(X) .

Proof

The proof follows by combining the proofs of theorems (1.5.2), (2.5.2), (3.5.1) and (4.5.1) in addition to the two clauses (C.10) and (C.45) of appendix (2) .

###

(5.5.2) Definition

Let L(⊥) be the subclass of the class of (bicartesian closed) formula bcclL(X) generated by the conditions :

1) ⊥ is in L(⊥).
2) If a,b are in L(⊥), then (a v b) .
3) If a is in L(⊥) and b is in bcclL(X), then (a † b), (b † a) are in L(⊥).

with the help of L(⊥), the subclass L(T) of the class bcclL(X) is is generated by the conditions :

1) T is in L(T).
2) If a,b are in L(T), then (a † b).
3) If a is in L(T) and b is in bcclL(X), then (b => a) is in L(T).
4) If a is in L(⊥) and b is in bcclL(X), then (a => b) is in L(T).
5) If a is in L(T) and b is in L(⊥), then (a v b), (b v a) are in L(T).

(5.5.3) Lemma

a = ~ T and b = ~ ⊥ in Fbccl(X) iff a is in L(T) and b is in L(⊥).

(5.5.4) Corollary

If b = ~⊥, there exists a unique derivation p of b -> consisting at most of instances of (A3), (R2), (R11), and (R12).

Proof:

Identical to the proof of corollary (3.5.6).

(5.5.5) Corollary

If a = ~ T, there exists a unique derivation p of -> a consisting at most of instances of (A2), (R2), (R5), (R10), (R13), and (R15).

Proof:

Combining the proofs of corollary (3.5.5) and lemma (4.5.4) and the following :

a = (c => d), c is in L(⊥) and d is in bcclL(X)

```
      h
    c ->
    -------
    c ->   d
    --------------
      -> (c => d)
```

where h is the unique derivation of c -> compatible with the above corollary.

(5.5.6) Lemma

For every cut-free p in Dbccl(X) there exists an equivalent cut-free q in Dbccl(X) containing no instances of (R6) and (R7), and containing no instances of (R2),(R3),(R4) and (R5) whose active formulae are of the form a † b or a v b.

Proof:

In case of (R6) and (R7) the proof follows from lemma (3.5.7) and the additional clauses (E.31) and (E.36) of Appendix(3).
The proof in case (R2) and (R5) follows from the proof of lemma (3.5.7).
The proof in case (R3) and (R4) is the same as in (4.5.5) but using the more generalized lemma (3.5.7) instead of lemma (1.5.4) and using the priority included in the proof of (5.5.9) below as an extension to that in (4.5.8) and the additional clauses (E.39) and (E.40) of Appendix (3).

###

(5.5.7) Lemma

Every cut-free p in Dbccl(X) is equivalent to a cut-free q in Dbccl(X) containing no instances of (R14) with the property that the active formulae in their right premisses are isomorphic to T.

Proof:

Identical to the proof of lemma (4.5.6).

(5.5.8) Definition

A derivation p in Dbccl(X) is normal if it satisfies the following conditions :

(1) All subderivations of p not containing instances of (R14) and (R15) are normal in the sense of (3.5.3), with the condition that the active formulae of instances of (R2) and (R5) are atomic replaced by the condition that the active formulae of instances of (R2) and (R5) are either atomic or of the form a => b.

(2) p contains no instances of (R2), (R3), (R4) or (R5) whose active formulae are of the form $a \uparrow b$ or $a \vee b$.

(3) If p derives G -> F, and if one of the disjunctions of the formulae of F is isomorphic to T, then p is of the form

```
      q
     -> F
  ---------  (K)
  G   -> F
```

where q is the unique derivation of -> F compatible with corollary (5.5.5), and where (K) consists of the unique steps required to derive G -> F from -> F.

(4) If p derives G -> F , and if one of the conjunctions of the formulae of G is isomorphic to , and if no disjunction of the formulae of F is isomorphic to T, then p is of the form

```
     q
  G ->
  ---------  (K)
  G ->    F
```

where q is the unique derivation of G -> compatible with corollary (5.5.4), and where (K) consists of the unique steps required to derive G -> F from G -> .

(5) p contains no instance of (R14) whose active formula in the right-hand premiss is isomorphic to T.

(6) Using the relation =< as defined in (1.5.4); for normality in Dbccl(X) it is required that

(R5) =< (R13) =< (R10) =< (R12) =< (R14) =< (R3) =< (R11) =< (R2) =< (R4),

(R15) =< (R12) =< (R14) =< (R3) =< (R11) =< (R2) =< (R4).

(5.5.9) The Normalization Theorem For Dbccl(X)

For every p in Dbccl(X), there exists an equivalent normal q in Dbccl(X).

Proof :

The theorem follows from theorems (3.5.9) and (4.5.8) and the following additional clauses of Appendix (4) :
(D.17), (D.20), (D.21), (D.25), (D.30), (D.31), (D.39), (D.60), (D.61), (D.63), (D.74), (D.76), and (D.78).

###

The following examples are sessions of input and output with the system for derivations of Dbccl(X) and their equivalents of cut-free and normal form , in order to illustrate the algorithms in (5.5.1), (5.5.6) and (5.5.9) and satisfy the definition in (5.5.8).

Example 1

Suppose a,b,c,d are objects of the category X; and that f in X(a,c), g in X(d,a), h in X(d,b), i in X(d,c) with comp(f, g) = i. Then the derivation

```
                                                                      h
                                                                   d -> b
                                                                 ----------
                                                                  a,d -> b
                                          g        h            ------------
  a -> a      b -> b                   d -> a   d -> b           a,b,d -> b          f
 --------    --------            f    -----------------        --------------      a -> c
a,b -> a    a,b -> b          a -> c     d -> a ↑ b             a ↑ b,d -> b      --------
--------------------          --------------------------      --------------      a,b -> c
   a,b -> a ↑ b                 a,c => d -> a ↑ b            a ↑ b -> d=>b      ----------
 -----------------             ----------------------        --------------     a ↑ b -> c
 a ↑ b -> a ↑ b                a,c => d -> c,a ↑ b            ---------------------------
------------------           ---------------------------        a ↑ b -> (d => b) ↑ c
a ↑ b -> c,a ↑ b             a ↑ (c => d) -> c,a ↑ b          ------------------------
-----------------------------------------------------         a ↑ b,d -> (d => b) ↑ c
     (a ↑ b) v (a ↑ (c => d)) -> c,a ↑ b                    ---------------------------
                                                              a ↑ b,d -> (d => b) ↑ c,e
    -----------------------------------------------------------------------------------
                (a ↑ b) v (a ↑ (c => d)),d -> c,(d => b) ↑ c,e
```

is represented, and then transformed to its normal form, as follows

```
*------------------------------------------------------------------------*
| yes
| ?- theory(bicartcl).
| yes
| ?- composite(f,g,i).
| yes
| ?- prove(s1 = [id(a), [a] => [a]]).
| yes
| ?- prove(s2 = [th(s1,b), [a,b] => [a]]).
| yes
| ?- prove(s3 = [id(b), [b] => [b]]).
| yes
| ?- prove(s4 = [th(s3,a), [a,b] => [b]]).
| yes
| ?- prove(s5 = [ais(s2,s4,a & b), [a,b] => [a & b]]).
| yes
| ?- prove(s6 = [aia(s5 ,a&b), [a&b] => [a&b]]).
| yes
| ?- prove(s7 = [th(s6,c), [a&b] => [c,a&b]]).
| yes
| ?- prove(s8 = [f, [a] => [c]]).
| yes
| ?- prove(s9 = [g, [d] => [a]]).
| yes
| ?- prove(s10 = [h, [d] => [b]]).
| yes
```

```
| ?- prove(s11 = [ais(s9,s10,a&b), [d] => [a&b]]).
yes
| ?- prove(s12 = [hia(s8,s11,c->d), [a,c->d] => [a&b]]).
yes
| ?- prove(s13 = [th(s12,c), [a,c->d] => [c,a&b]]).
yes
| ?- prove(s14 = [aia(s13,a&(c->d)), [a&(c->d)] => [c,a&b]]).
yes
| ?- prove(s15 = [oia(s7,s14,(a&b)#(a&(c->d))),
|                                  [(a&b)#(a&(c->d))] => [c,a&b]]).
yes
| ?- prove(s16 = [th(s10,a), [a,d] => [b]]).
yes
| ?- prove(s17 = [th(s16,b), [a,b,d] => [b]]).
yes
| ?- prove(s18 = [aia(s17,a&b), [a&b,d] => [b]]).
yes
| ?- prove(s19 = [his(s18,d -> b), [a&b] => [d -> b]]).
yes
| ?- prove(s20 = [th(s8,b), [a,b] => [c]]).
yes
| ?- prove(s21 = [aia(s20,a&b), [a&b] => [c]]).
yes
| ?- prove(s22 = [ais(s19,s21,(d->b)&c), [a&b] => [(d->b)&c]]).
yes
| ?- prove(s23 = [th(s22,d), [a&b,d] => [(d->b)&c]]).
yes
| ?- prove(s24 = [th(s23,e), [a&b,d] => [(d->b)&c,e]]).
yes
| ?- prove(s = [cut(s15,s24,a&b), [(a&b)#(a&(c->d)),d]=>
|                                          [c,(d->b)&c,e]]).
yes
| ?- normal(s).

Proof is as follows :
=====================

s1=[id(a),[a]=>[a]]
s2=[th(s1,b),[a,b]=>[a]]
s3=[id(b),[b]=>[b]]
s4=[th(s3,a),[a,b]=>[b]]
s5=[ais(s2,s4,a&b),[a,b]=>[a&b]]
s6=[aia(s5,a&b),[a&b]=>[a&b]]
s7=[th(s6,c),[a&b]=>[c,a&b]]
f6=[f,[a]=>[c]]
s9=[g,[d]=>[a]]
f4=[h,[d]=>[b]]
s11=[ais(s9,f4,a&b),[d]=>[a&b]]
s12=[hia(f6,s11,c->d),[a,c->d]=>[a&b]]
s13=[th(s12,c),[a,c->d]=>[c,a&b]]
s14=[aia(s13,a&(c->d)),[a&(c->d)]=>[c,a&b]]
s15=[oia(s7,s14,(a&b)#a&(c->d)),[(a&b)#a&(c->d)]=>[c,a&b]]
f5=[h,[d]=>[b]]
s16=[th(f5,a),[a,d]=>[b]]
s17=[th(s16,b),[a,b,d]=>[b]]
s18=[aia(s17,a&b),[a&b,d]=>[b]]
s19=[his(s18,d->b),[a&b]=>[d->b]]
f7=[f,[a]=>[c]]
s20=[th(f7,b),[a,b]=>[c]]
s21=[aia(s20,a&b),[a&b]=>[c]]
s22=[ais(s19,s21,(d->b)&c),[a&b]=>[(d->b)&c]]
s23=[th(s22,d),[a&b,d]=>[(d->b)&c]]
s24=[th(s23,e),[a&b,d]=>[(d->b)&c,e]]
s=[cut(s15,s24,a&b),[(a&b)#a&(c->d),d]=>[c,(d->b)&c,e]]

Cut_free proof is :
===================

f58=[h,[d]=>[b]]
f57=[th(f58,a),[a,d]=>[b]]
f54=[th(f57,b),[a,b,d]=>[b]]
f51=[th(f54,b),[a,b,b,d]=>[b]]
f39=[th(f51,a),[a,b,a,b,d]=>[b]]
f42=[inc(f39,a,b),[a,a,b,b,d]=>[b]]
f43=[con(f42,a),[a,b,b,d]=>[b]]
f34=[con(f43,b),[a,b,d]=>[b]]
f31=[aia(f34,a&b),[a&b,d]=>[b]]
f28=[his(f31,d->b),[a&b]=>[d->b]]
f22=[th(f28,c),[a&b]=>[c,d->b]]
f71=[f,[a]=>[c]]
f83=[h,[d]=>[b]]
f82=[th(f83,d),[d,d]=>[b]]
```

```
f76=[th(f82,d),[d,d,d]=>[b]]
f70=[con(f76,d),[d,d]=>[b]]
f67=[hia(f71,f70,c->d),[a,c->d,d]=>[b]]
f64=[his(f67,d->b),[a,c->d]=>[d->b]]
f61=[th(f64,c),[a,c->d]=>[c,d->b]]
f25=[aia(f61,a&(c->d)),[a&(c->d)]=>[c,d->b]]
f16=[oia(f22,f25,(a&b)#a&(c->d)),[(a&b)#a&(c->d)]=>[c,d->b]]
f118=[f,[a]=>[c]]
f115=[th(f118,b),[a,b]=>[c]]
f112=[th(f115,b),[a,b,b]=>[c]]
f100=[th(f112,a),[a,b,a,b]=>[c]]
f103=[inc(f100,a,b),[a,a,b,b]=>[c]]
f104=[con(f103,a),[a,b,b]=>[c]]
f95=[con(f104,b),[a,b]=>[c]]
f92=[aia(f95,a&b),[a&b]=>[c]]
f86=[th(f92,c),[a&b]=>[c,c]]
f128=[f,[a]=>[c]]
f139=[i,[d]=>[c]]
f133=[th(f139,d),[d,d]=>[c]]
f127=[con(f133,d),[d]=>[c]]
f124=[hia(f128,f127,c->d),[a,c->d]=>[c]]
f121=[th(f124,c),[a,c->d]=>[c,c]]
f89=[aia(f121,a&(c->d)),[a&(c->d)]=>[c,c]]
f19=[oia(f86,f89,(a&b)#a&(c->d)),[(a&b)#a&(c->d)]=>[c,c]]
f13=[ais(f16,f19,(d->b)&c),[(a&b)#a&(c->d)]=>[c,(d->b)&c]]
f10=[th(f13,d),[(a&b)#a&(c->d),d]=>[c,(d->b)&c]]
s=[th(f10,e),[(a&b)#a&(c->d),d]=>[c,(d->b)&c,e]]
```

Normal proof is :

```
f58=[h,[d]=>[b]]
f57=[his(f58,d->b),[]=>[d->b]]
f34=[th(f57,e),[]=>[d->b,e]]
f31=[th(f34,c),[]=>[c,d->b,e]]
f28=[th(f31,b),[b]=>[c,d->b,e]]
f22=[th(f28,a),[a,b]=>[c,d->b,e]]
f118=[f,[a]=>[c]]
f95=[th(f118,e),[a]=>[c,e]]
f92=[th(f95,c),[a]=>[c,c,e]]
f142=[th(f92,b),[a,b]=>[c,c,e]]
f16=[ais(f22,f142,(d->b)&c),[a,b]=>[c,(d->b)&c,e]]
f13=[aia(f16,a&b),[a&b]=>[c,(d->b)&c,e]]
f83=[h,[d]=>[b]]
f70=[his(f83,d->b),[]=>[d->b]]
f67=[th(f70,e),[]=>[d->b,e]]
f71=[th(f67,c),[]=>[c,d->b,e]]
f61=[th(f71,c->d),[c->d]=>[c,d->b,e]]
f25=[th(f61,a),[a,c->d]=>[c,d->b,e]]
f128=[f,[a]=>[c]]
f127=[i,[d]=>[c]]
f124=[th(f127,e),[d]=>[c,e]]
f121=[th(f124,c),[d]=>[c,c,e]]
f141=[hia(f128,f121,c->d),[a,c->d]=>[c,c,e]]
f19=[ais(f25,f141,(d->b)&c),[a,c->d]=>[c,(d->b)&c,e]]
f140=[aia(f19,a&(c->d)),[a&(c->d)]=>[c,(d->b)&c,e]]
f10=[oia(f13,f140,(a&b)#a&(c->d)),[(a&b)#a&(c->d)]=>[c,(d->b)&c,e]]
s=[th(f10,d),[(a&b)#a&(c->d),d]=>[c,(d->b)&c,e]]
```

which in tree form is

```
         h                                                     h
      d -> b                                                d -> b
   -------------                                         -------------
    -> d => b                                              -> d => b
  ---------------          f                           ----------------                       i
   -> d => b,e          a -> c                           -> d => b,e                        d -> c
 -----------------     ---------                     ------------------                   ---------
  -> c,d => b,e         a -> c,e                       -> c,d => b,e                      d -> c,e
-------------------   ------------              ---------------------------       f     -----------
 b -> c,d => b,e       a -> c,c,e                c => d -> c,d => b,e        a -> c    d -> c,c,e
---------------------  -------------            ----------------------------  --------------------
a,b -> c,d => b,e      a,b -> c,c,e             a,c => d -> c,d => b,e        a,c => d -> c,c,e
---------------------------------------        ------------------------------------------------
    a,b -> c,(d => b) ↑ c,e                          a,c => d -> c,(d => b) ↑ c,e
 ---------------------------------                 -------------------------------------
 a ↑ b -> c,(d => b) ↑ c,e                         a ↑ (c => d) -> c,(d => b) ↑ c,e
 -------------------------------------------------------------------------------------------
              (a ↑ b) v (a ↑ (c => d)) -> c,(d => b) ↑ c,e
             ----------------------------------------------
              (a ↑ b) v (a ↑ (c => d)),d -> c,(d => b) ↑ c,e
```

Example 2

Let a,b,c be objects of the category X, f in X(a,b), g in X(c,b); then the derivation

```
                                             f             g
                                           a -> b        c -> b
            f                             --------      --------
          a -> b        b -> b            a,a -> b      c,a -> b
         --------      --------           ----------------------
         a,c -> b      c,b -> b                a v c,a -> b
       ----------    ----------               ---------------
       a,b,c -> b    c,b,c -> b                a v b,d,a -> b
       ------------------------           ---------------------
          a v c,b,c -> b                  a v c,e,d,a -> b
          ----------------------------------------------------
                 a v c,a v c,b,c,b => e,d,a -> b
                 -------------------------------
                    a v c,b,c,b => e,d,a -> b
```

is transformed to its equivalent normal form, as follows :

```
yes
| ?- theory(bicartcl).
yes
| ?- prove(s1 = [f, [a] => [b]]).
yes
| ?- prove(s2 = [th(s1,c), [a,c] => [b]]).
yes
| ?- prove(s3 = [th(s2,b), [a,b,c] => [b]]).
yes
| ?- prove(s4 = [id(b), [b] => [b]]).
yes
| ?- prove(s5 = [th(s4,c), [c,b] => [b]]).
yes
| ?- prove(s6 = [th(s5,c), [c,b,c] => [b]]).
yes
| ?- prove(s7 = [ola(s3,s6,a # c), [a # c,b,c] => [b]]).
yes
| ?- prove(s8 = [th(s1,a), [a,a] => [b]]).
yes
| ?- prove(s9 = [g, [c] => [b]]).
yes
| ?- prove(s10 = [th(s9,a), [c,a] => [b]]).
yes
| ?- prove(s11 = [ola(s8,s10,a # c), [a # c,a] => [b]]).
yes
| ?- prove(s12 = [th(s11,d), [a#c,d,a] => [b]]).
yes
| ?- prove(s13 = [th(s12,e), [a # c,e,d,a] => [b]]).
yes
| ?- prove(s14=[hla(s7,s13,b->e),[a#c,a#c,b,c,b->e,d,a]=>[b]]).
yes
| ?- prove(s = [con(s14,a#c), [a#c,b,c,b->e,d,a] => [b]]).
yes
| ?- normal(s).

Proof is as follows :
====================

f1=[f,[a]=>[b]]
s2=[th(f1,c),[a,c]=>[b]]
s3=[th(s2,b),[a,b,c]=>[b]]
s4=[id(b),[b]=>[b]]
s5=[th(s4,c),[c,b]=>[b]]
s6=[th(s5,c),[c,b,c]=>[b]]
s7=[ola(s3,s6,a#c),[a#c,b,c]=>[b]]
f2=[f,[a]=>[b]]
s8=[th(f2,a),[a,a]=>[b]]
s9=[g,[c]=>[b]]
s10=[th(s9,a),[c,a]=>[b]]
s11=[ola(s8,s10,a#c),[a#c,a]=>[b]]
s12=[th(s11,d),[a#c,d,a]=>[b]]
s13=[th(s12,e),[a#c,e,d,a]=>[b]]
s14=[hla(s7,s13,b->e),[a#c,a#c,b,c,b->e,d,a]=>[b]]
s=[con(s14,a#c),[a#c,b,c,b->e,d,a]=>[b]]
```

```
Cut_free proof is :
===================

f1=[f,[a]=>[b]]
s2=[th(f1,c),[a,c]=>[b]]
s3=[th(s2,b),[a,b,c]=>[b]]
s4=[id(b),[b]=>[b]]
s5=[th(s4,c),[c,b]=>[b]]
s6=[th(s5,c),[c,b,c]=>[b]]
s7=[oia(s3,s6,a#c),[a#c,b,c]=>[b]]
f2=[f,[a]=>[b]]
s8=[th(f2,a),[a,a]=>[b]]
s9=[g,[c]=>[b]]
s10=[th(s9,a),[c,a]=>[b]]
s11=[oia(s8,s10,a#c),[a#c,a]=>[b]]
s12=[th(s11,d),[a#c,d,a]=>[b]]
s13=[th(s12,e),[a#c,e,d,a]=>[b]]
s14=[hia(s7,s13,b->e),[a#c,a#c,b,c,b->e,d,a]=>[b]]
s=[con(s14,a#c),[a#c,b,c,b->e,d,a]=>[b]]


Normal proof is :
=================

f2=[f,[a]=>[b]]
s9=[g,[c]=>[b]]
s8=[oia(f2,s9,a#c),[a#c]=>[b]]
s11=[th(s8,a),[a#c,a]=>[b]]
s12=[th(s11,d),[a#c,d,a]=>[b]]
f4=[th(s12,b->e),[a#c,b->e,d,a]=>[b]]
s14=[th(f4,c),[a#c,c,b->e,d,a]=>[b]]
s=[th(s14,b),[a#c,b,c,b->e,d,a]=>[b]]

yes
| ?-
```

which in tree form is

```
                  f             g
                a -> b        c -> b
                --------------------
                    a # c -> b
                   -------------
                   a v c,a -> b
                 ---------------
                 a v c,d,a -> b
          ------------------------
          a v c,b => e,d,a -> b
        --------------------------
        a v c,c,b => e,d,a -> b
      ----------------------------
      a v c,b,c,b => e,d,a -> b
```

(5.5.10) <u>The CHURCH-ROSSER Theorem for Dbccl(X)</u>

If p == q, then there exists a normal r in Dbccl(X) such that p >= r and q >= r.

(5.6) Application To Category Theory

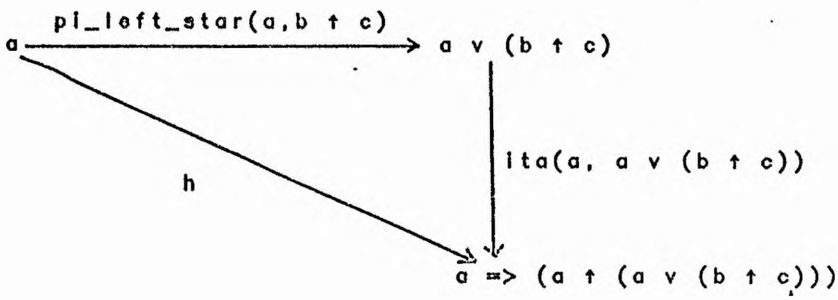
In this section, we use the system to produce the normal representations for some arrows of Fbccl(X) and test the commutativity of some diagrams in Fbccl(X).

<u>Example 1</u>

For objects a,b,c of the category X, the normal representation of the arrow

h = comp(ita(a, a v (b ↑ c)), pi_left_star(a, b ↑ c))

in the diagram



is obtained as follows :

```
yes
| ?- theory(bicartcl).

yes
| ?- norm_rep(comp(ita(a,a#(b&c)), pi_left_star(a,b&c))).

The arrow is :
==============

comp(ita(a,a#b&c),pi_left_star(a,b&c))

Proof is as follows :
=====================

f40=[id(a),[a]=>[a]]
f39=[th(f40,b&c),[a]=>[a,b&c]]
f38=[ois(f39,a#b&c),[a]=>[a#b&c]]
f26=[id(a),[a]=>[a]]
f24=[th(f26,a#b&c),[a,a#b&c]=>[a]]
f31=[id(a),[a]=>[a]]
f29=[th(f31,b&c),[a]=>[a,b&c]]
f36=[id(b),[b]=>[b]]
f34=[th(f36,c),[b,c]=>[b]]
f37=[id(c),[c]=>[c]]
f35=[th(f37,b),[b,c]=>[c]]
f33=[ais(f34,f35,b&c),[b,c]=>[b&c]]
f32=[ala(f33,b&c),[b&c]=>[b&c]]
f30=[th(f32,a),[b&c]=>[a,b&c]]
f28=[ola(f29,f30,a#b&c),[a#b&c]=>[a,b&c]]
f27=[ois(f28,a#b&c),[a#b&c]=>[a#b&c]]
f25=[th(f27,a),[a,a#b&c]=>[a#b&c]]
f23=[ais(f24,f25,a&a#b&c),[a,a#b&c]=>[a&a#b&c]]
f22=[his(f23,a->a&a#b&c),[a#b&c]=>[a->a&a#b&c]]
f21=[cut(f38,f22,a#b&c),[a]=>[a->a&a#b&c]]


Cut_free proof is :
===================

f68=[id(a),[a]=>[a]]
f67=[th(f68,a),[a]=>[a,a]]
f64=[th(f67,a),[a,a]=>[a,a]]
f56=[th(f64,a),[a,a,a]=>[a,a]]
f57=[con(f56,a),[a,a,a]=>[a]]
f46=[con(f57,a),[a,a]=>[a]]
f90=[id(a),[a]=>[a]]
f122=[th(f90,b&c),[a]=>[a,b&c]]
f125=[th(f122,a),[a]=>[a,a,b&c]]
f126=[th(f125,b),[a]=>[a,a,b,b&c]]
f123=[th(f126,b&c),[a]=>[a,a,b,b&c,b&c]]
f124=[th(f123,a),[a,a]=>[a,a,b,b&c,b&c]]
f90=[id(a),[a]=>[a]]
f128=[th(f90,b&c),[a]=>[a,b&c]]
f131=[th(f128,a),[a]=>[a,a,b&c]]
f132=[th(f131,c),[a]=>[a,a,c,b&c]]
f129=[th(f132,b&c),[a]=>[a,a,c,b&c,b&c]]
f130=[th(f129,a),[a,a]=>[a,a,c,b&c,b&c]]
f100=[ais(f124,f130,b&c),[a,a]=>[a,a,b&c,b&c,b&c]]
f101=[con(f100,b&c),[a,a]=>[a,a,b&c,b&c]]
f102=[con(f101,a),[a,a]=>[a,b&c,b&c]]
f93=[con(f102,a),[a]=>[a,b&c,b&c]]
f79=[th(f93,a),[a]=>[a,b&c,a,b&c]]
f82=[inc(f79,a,b&c),[a]=>[a,a,b&c,b&c]]
```

```
f83=[con(f82,a),[a]=>[a,b&c,b&c]]
f80=[con(f83,b&c),[a]=>[a,b&c]]
f71=[ois(f80,a#b&c),[a]=>[a#b&c]]
f49=[th(f71,a),[a,a]=>[a#b&c]]
f43=[ais(f46,f49,a&a#b&c),[a,a]=>[a&a#b&c]]
f21=[his(f43,a->a&a#b&c),[a]=>[a->a&a#b&c]]

Normal proof is :
================

f64=[id(a),[a]=>[a]]
f90=[id(a),[a]=>[a]]
f134=[th(f90,b),[a]=>[a,b]]
f136=[id(a),[a]=>[a]]
f135=[th(f136,c),[a]=>[a,c]]
f80=[ais(f134,f135,b&c),[a]=>[a,b&c]]
f71=[ois(f80,a#b&c),[a]=>[a#b&c]]
f46=[ais(f64,f71,a&a#b&c),[a]=>[a&a#b&c]]
f43=[his(f46,a->a&a#b&c),[]=>[a->a&a#b&c]]
f21=[th(f43,a),[a]=>[a->a&a#b&c]]


yes
| ?-
```

which in tree form is

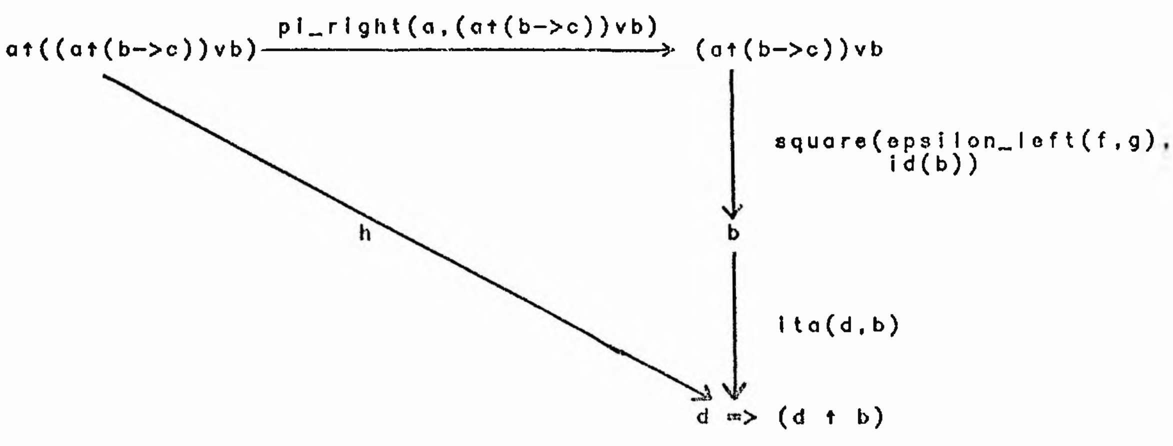
```
                         a -> a      a -> a
                         --------    --------
                         a -> a,b    a -> a,c
                         --------------------
                           a -> a, b † c
                           ---------------
       a -> a              a -> a v (b † c)
       ------------------------------------
           a -> a † (a v (b † c))
        ---------------------------
           -> a => (a † (a v (b † c)))
        ---------------------------
        a -> a => (a † (a v (b † c)))
```

Example 2

For arrows f in X(a,b) and g in X(c,b) for some category X, the normal representation of the arrow

```
h = comp(comp(ita(d,b), square(epsilon_left(f,g),id(b))),
         pi_right(a, (a † (b -> c)) v b))
```

in the diagram



is obtained as follows :

```
yes
| ?- theory(bicartcl).

yes
| ?- arrow(a,f,b).

yes
| ?- arrow(c,g,b).

yes
| ?- norm_rep(comp(comp(ita(d,b), square(epsilon_left(f,g),
|                                          id(b))),
|                   pi_right(a,(a&(b->c))#b))).


The arrow is :
=============

comp(comp(ita(d,b),square(epsilon_left(f,g),id(b))),
     pi_right(a,(a&(b->c))#b))

Proof is as follows :
=====================

f26=[id(a),[a]=>[a]]
f24=[th(f26,b->c),[a,b->c]=>[a]]
f29=[id(b),[b]=>[b]]
f30=[id(c),[c]=>[c]]
f28=[hia(f29,f30,b->c),[b,b->c]=>[c]]
f27=[his(f28,b->c),[b->c]=>[b->c]]
f25=[th(f27,a),[a,b->c]=>[b->c]]
f23=[ais(f24,f25,a&(b->c)),[a,b->c]=>[a&(b->c)]]
f21=[aia(f23,a&(b->c)),[a&(b->c)]=>[a&(b->c)]]
f19=[th(f21,b),[a&(b->c)]=>[a&(b->c),b]]
f22=[id(b),[b]=>[b]]
f20=[th(f22,a&(b->c)),[b]=>[a&(b->c),b]]
f18=[oia(f19,f20,(a&(b->c))#b),[(a&(b->c))#b]=>[a&(b->c),b]]
f17=[ois(f18,(a&(b->c))#b),[(a&(b->c))#b]=>[(a&(b->c))#b]]
f16=[th(f17,a),[a,(a&(b->c))#b]=>[(a&(b->c))#b]]
f15=[aia(f16,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[(a&(b->c))#b]]
f11=[f,[a]=>[b]]
f12=[g,[c]=>[b]]
f13=[hia(f11,f12,b->c),[a,b->c]=>[b]]
f10=[aia(f13,a&(b->c)),[a&(b->c)]=>[b]]
f14=[id(b),[b]=>[b]]
f9=[oia(f10,f14,(a&(b->c))#b),[(a&(b->c))#b]=>[b]]
f7=[id(d),[d]=>[d]]
f5=[th(f7,b),[d,b]=>[d]]
f8=[id(b),[b]=>[b]]
f6=[th(f8,d),[d,b]=>[b]]
f4=[ais(f5,f6,d&b),[d,b]=>[d&b]]
f3=[his(f4,d->d&b),[b]=>[d->d&b]]
f2=[cut(f9,f3,b),[(a&(b->c))#b]=>[d->d&b]]
f1=[cut(f15,f2,(a&(b->c))#b),[a&(a&(b->c))#b]=>[d->d&b]]


Cut_free proof is :
===================

f141=[f,[a]=>[b]]
f137=[th(f141,b->c),[a,b->c]=>[b]]
f131=[id(d),[d]=>[d]]
f138=[th(f131,c),[d,c]=>[d]]
f134=[hia(f137,f138,b->c),[d,a,b->c,b->c]=>[d]]
f109=[th(f134,a),[d,a,b->c,a,b->c]=>[d]]
f112=[inc(f109,a,b->c),[d,a,a,b->c,b->c]=>[d]]
f113=[con(f112,a),[d,a,b->c,b->c]=>[d]]
f104=[con(f113,b->c),[d,a,b->c]=>[d]]
f176=[aia(f104,a&(b->c)),[d,a&(b->c)]=>[d]]
f177=[th(f176,d),[d,a&(b->c)]=>[d,d]]
f172=[th(f177,d),[d,d,a&(b->c)]=>[d,d]]
f216=[id(d),[d]=>[d]]
f215=[th(f216,b),[d,b]=>[d]]
f212=[th(f215,d),[d,b]=>[d,d]]
f209=[th(f212,d),[d,d,b]=>[d,d]]
f206=[th(f209,b),[d,d,b,b]=>[d,d]]
f203=[th(f206,d),[d,d,b,b]=>[d,d,d]]
f186=[th(f203,d),[d,d,d,b,b]=>[d,d,d]]
f187=[con(f186,d),[d,d,d,b,b]=>[d,d]]
```

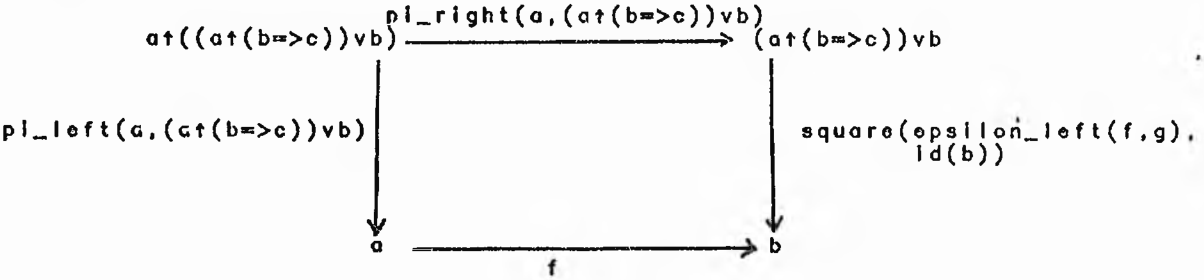
```
f181=[con(f187,d),[d,d,b,b]=>[d,d]]
f175=[con(f181,b),[d,d,b]=>[d,d]]
f90=[ola(f172,f175,(a&(b->c))#b),[d,d,(a&(b->c))#b]=>[d,d]]
f91=[con(f90,d),[d,d,(a&(b->c))#b]=>[d]]
f85=[con(f91,d),[d,(a&(b->c))#b]=>[d]]
f82=[th(f85,a),[d,a,(a&(b->c))#b]=>[d]]
f76=[ala(f82,a&(a&(b->c))#b),[d,a&(a&(b->c))#b]=>[d]]
f280=[f,[a]=>[b]]
f276=[th(f280,b->c),[a,b->c]=>[b]]
f277=[g,[c]=>[b]]
f273=[hla(f276,f277,b->c),[a,b->c,b->c]=>[b]]
f249=[th(f273,a),[a,b->c,a,b->c]=>[b]]
f252=[inc(f249,a,b->c),[a,a,b->c,b->c]=>[b]]
f253=[con(f252,a),[a,b->c,b->c]=>[b]]
f244=[con(f253,b->c),[a,b->c]=>[b]]
f310=[ala(f244,a&(b->c)),[a&(b->c)]=>[b]]
f306=[th(f310,b),[a&(b->c)]=>[b,b]]
f301=[id(b),[b]=>[b]]
f302=[th(f301,b),[b]=>[b,b]]
f317=[th(f302,b),[b,b]=>[b,b]]
f316=[th(f317,b),[b,b]=>[b,b,b]]
f313=[con(f316,b),[b,b]=>[b,b]]
f309=[con(f313,b),[b]=>[b,b]]
f230=[ola(f306,f309,(a&(b->c))#b),[(a&(b->c))#b]=>[b,b]]
f231=[con(f230,b),[(a&(b->c))#b]=>[b]]
f222=[th(f231,a),[a,(a&(b->c))#b]=>[b]]
f219=[ala(f222,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]
f79=[th(f219,d),[d,a&(a&(b->c))#b]=>[b]]
f73=[ais(f76,f79,d&b),[d,a&(a&(b->c))#b]=>[d&b]]
f1=[his(f73,d->d&b),[a&(a&(b->c))#b]=>[d->d&b]]


Normal proof is :
================

f131=[id(d),[d]=>[d]]
f141=[th(f131,b->c),[d,b->c]=>[d]]
f104=[th(f141,a),[d,a,b->c]=>[d]]
f91=[ala(f104,a&(b->c)),[d,a&(b->c)]=>[d]]
f216=[id(d),[d]=>[d]]
f319=[th(f216,b),[d,b]=>[d]]
f85=[ola(f91,f319,(a&(b->c))#b),[d,(a&(b->c))#b]=>[d]]
f82=[th(f85,a),[d,a,(a&(b->c))#b]=>[d]]
f76=[ala(f82,a&(a&(b->c))#b),[d,a&(a&(b->c))#b]=>[d]]
f280=[f,[a]=>[b]]
f277=[g,[c]=>[b]]
f244=[hla(f280,f277,b->c),[a,b->c]=>[b]]
f230=[ala(f244,a&(b->c)),[a&(b->c)]=>[b]]
f320=[id(b),[b]=>[b]]
f231=[ola(f230,f320,(a&(b->c))#b),[(a&(b->c))#b]=>[b]]
f222=[th(f231,a),[a,(a&(b->c))#b]=>[b]]
f219=[ala(f222,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]
f79=[th(f219,d),[d,a&(a&(b->c))#b]=>[b]]
f73=[ais(f76,f79,d&b),[d,a&(a&(b->c))#b]=>[d&b]]
f1=[his(f73,d->d&b),[a&(a&(b->c))#b]=>[d->d&b]]

yes
| ?-
```

which in tree form is

```
                                                           f           g
        d -> d                                           a -> b       c -> b
    ----------------                                    ---------------------
     d,b => c -> d                                          a,b => c -> b
    ------------------                                    ------------------
   d,a,b => c -> d          d -> d                      a ↑ (b => c) -> b     b -> b
 -----------------------   ----------                  ------------------------------
 d,a ↑ (b => c) -> d        d,b -> d                     (a ↑ (b => c)) v b -> b
 -------------------------------------                 -------------------------------
       d,(a ↑ (b => c)) v b -> d                        a,(a ↑ (b => c)) v b -> b
     --------------------------------                ---------------------------------
     d,a,(a ↑ (b => c)) v b -> d                      a ↑ ((a ↑ (b => c)) v b) -> b
 -------------------------------------             ------------------------------------
 d,a ↑ ((a ↑ (b => c)) v b) -> d                    d,a ↑ ((a ↑ (b => c)) v b) -> b
 ----------------------------------------------------------------------------------------
                      d,a ↑ ((a ↑ (b => c)) v b) -> d ↑ b
                    ------------------------------------------
                      a ↑ ((a ↑ (b => c)) v b) -> d => (d ↑ b)
```

Example 3

For f in X(a,b) and g in X(c,b) for some category X, the following diagram in Fbccl(X)



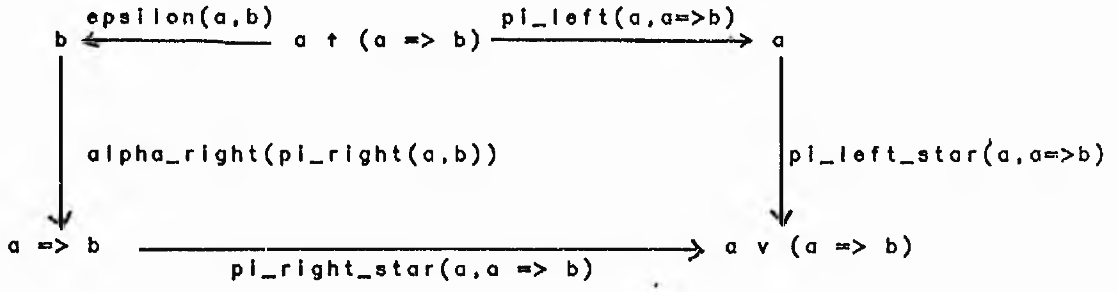
does not commute.

```
yes
| ?- theory(bicartcl).

yes
| ?- arrow(a,f,b).

yes
| ?- arrow(c,g,b).

yes
| ?- commutative.

Insert the first arrow : comp(square(epsilon_left(f,g),id(b)),
|:                          pi_right(a,(a&(b->c))#b)).

The arrow is :
==============

comp(square(epsilon_left(f,g),id(b)),pi_right(a,(a&(b->c))#b))

Proof is as follows :
=====================

f19=[id(a),[a]=>[a]]
f17=[th(f19,b->c),[a,b->c]=>[a]]
f22=[id(b),[b]=>[b]]
f23=[id(c),[c]=>[c]]
f21=[hia(f22,f23,b->c),[b,b->c]=>[c]]
f20=[his(f21,b->c),[b->c]=>[b->c]]
f18=[th(f20,a),[a,b->c]=>[b->c]]
f16=[ais(f17,f18,a&(b->c)),[a,b->c]=>[a&(b->c)]]
f14=[aia(f16,a&(b->c)),[a&(b->c)]=>[a&(b->c)]]
f12=[th(f14,b),[a&(b->c)]=>[a&(b->c),b]]
f15=[id(b),[b]=>[b]]
f13=[th(f15,a&(b->c)),[b]=>[a&(b->c),b]]
f11=[oia(f12,f13,(a&(b->c))#b),[(a&(b->c))#b]=>[a&(b->c),b]]
f10=[ois(f11,(a&(b->c))#b),[(a&(b->c))#b]=>[(a&(b->c))#b]]
f9=[th(f10,a),[a,(a&(b->c))#b]=>[(a&(b->c))#b]]
f8=[aia(f9,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[(a&(b->c))#b]]
f4=[f,[a]=>[b]]
f5=[g,[c]=>[b]]
f6=[hia(f4,f5,b->c),[a,b->c]=>[b]]
f3=[aia(f6,a&(b->c)),[a&(b->c)]=>[b]]
f7=[id(b),[b]=>[b]]
f2=[oia(f3,f7,(a&(b->c))#b),[(a&(b->c))#b]=>[b]]
f1=[cut(f8,f2,(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]

Cut_free proof is :
===================

f84=[f,[a]=>[b]]
f80=[th(f84,b->c),[a,b->c]=>[b]]
f81=[g,[c]=>[b]]
f77=[hia(f80,f81,b->c),[a,b->c,b->c]=>[b]]
```

```
f53=[th(f77,a),[a,b->c,a,b->c]=>[b]]
f56=[inc(f53,a,b->c),[a,a,b->c,b->c]=>[b]]
f57=[con(f56,a),[a,b->c,b->c]=>[b]]
f48=[con(f57,b->c),[a,b->c]=>[b]]
f114=[aia(f48,a&(b->c)),[a&(b->c)]=>[b]]
f110=[th(f114,b),[a&(b->c)]=>[b,b]]
f105=[id(b),[b]=>[b]]
f106=[th(f105,b),[b]=>[b,b]]
f121=[th(f106,b),[b,b]=>[b,b]]
f120=[th(f121,b),[b,b]=>[b,b,b]]
f117=[con(f120,b),[b,b]=>[b,b]]
f113=[con(f117,b),[b]=>[b,b]]
f34=[oia(f110,f113,(a&(b->c))#b),[(a&(b->c))#b]=>[b,b]]
f35=[con(f34,b),[(a&(b->c))#b]=>[b]]
f26=[th(f35,a),[a,(a&(b->c))#b]=>[b]]
f1=[aia(f26,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]


Normal proof is :
=================

f84=[f,[a]=>[b]]
f81=[g,[c]=>[b]]
f48=[hia(f84,f81,b->c),[a,b->c]=>[b]]
f34=[aia(f48,a&(b->c)),[a&(b->c)]=>[b]]
f122=[id(b),[b]=>[b]]
f35=[oia(f34,f122,(a&(b->c))#b),[(a&(b->c))#b]=>[b]]
f26=[th(f35,a),[a,(a&(b->c))#b]=>[b]]
f1=[aia(f26,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]


Insert the second arrow : comp(f,pi_left(a,(a&(b->c))#b)).


The arrow is :
==============

comp(f,pi_left(a,(a&(b->c))#b))


Proof is as follows :
=====================

f127=[id(a),[a]=>[a]]
f126=[th(f127,(a&(b->c))#b),[a,(a&(b->c))#b]=>[a]]
f125=[aia(f126,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[a]]
f124=[f,[a]=>[b]]
f123=[cut(f125,f124,a),[a&(a&(b->c))#b]=>[b]]


Cut_free proof is :
===================

f133=[f,[a]=>[b]]
f130=[th(f133,(a&(b->c))#b),[a,(a&(b->c))#b]=>[b]]
f123=[aia(f130,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]


Normal proof is :
=================

f133=[f,[a]=>[b]]
f138=[th(f133,b->c),[a,b->c]=>[b]]
f137=[aia(f138,a&(b->c)),[a&(b->c)]=>[b]]
f134=[th(f137,a),[a,a&(b->c)]=>[b]]
f136=[f,[a]=>[b]]
f135=[th(f136,b),[a,b]=>[b]]
f130=[oia(f134,f135,(a&(b->c))#b),[a,(a&(b->c))#b]=>[b]]
f123=[aia(f130,a&(a&(b->c))#b),[a&(a&(b->c))#b]=>[b]]


The answer is :
===============

comp(square(epsilon_left(f,g),id(b)),pi_right(a,(a&(b->c))#b))
                                          is not equal to
comp(f,pi_left(a,(a&(b->c))#b))


yes
| ?-
```

Example 4

For a,b in Ob(X) for some category X, the following diagram in Fbccl(X)



does not commute.

```
yes
| ?- theory(bicartcl).

yes
| ?- commutative.

Insert the first arrow : comp(pi_left_star(a,a->b),
                              pi_left(a,a->b)).


The arrow is :
==============

comp(pi_left_star(a,a->b),pi_left(a,a->b))


Proof is as follows :
=====================

f7=[id(a),[a]=>[a]]
f6=[th(f7,a->b),[a,a->b]=>[a]]
f5=[aia(f6,a&(a->b)),[a&(a->b)]=>[a]]
f4=[id(a),[a]=>[a]]
f3=[th(f4,a->b),[a]=>[a,a->b]]
f2=[ois(f3,a#(a->b)),[a]=>[a#(a->b)]]
f1=[cut(f5,f2,a),[a&(a->b)]=>[a#(a->b)]]


Cut_free proof is :
===================

f19=[id(a),[a]=>[a]]
f16=[th(f19,a->b),[a,a->b]=>[a]]
f13=[aia(f16,a&(a->b)),[a&(a->b)]=>[a]]
f10=[th(f13,a->b),[a&(a->b)]=>[a,a->b]]
f1=[ois(f10,a#(a->b)),[a&(a->b)]=>[a#(a->b)]]


Normal proof is :
=================

f19=[id(a),[a]=>[a]]
f16=[th(f19,a->b),[a]=>[a,a->b]]
f13=[ois(f16,a#(a->b)),[a]=>[a#(a->b)]]
f10=[th(f13,a->b),[a,a->b]=>[a#(a->b)]]
f1=[aia(f10,a&(a->b)),[a&(a->b)]=>[a#(a->b)]]


Insert the second arrow : comp(comp(pi_right_star(a,a->b),
|: ,                                alpha_right(pi_right(a,b))),
|:                             epsilon(a,b)).


The arrow is :
==============

comp(comp(pi_right_star(a,a->b),alpha_right(pi_right(a,b))),
     epsilon(a,b))
```

```
Proof is as follows :
=====================

f40=[id(a),[a]=>[a]]
f41=[id(b),[b]=>[b]]
f39=[hia(f40,f41,a->b),[a,a->b]=>[b]]
f38=[ala(f39,a&(a->b)),[a&(a->b)]=>[b]]
f36=[id(a),[a]=>[a]]
f34=[th(f36,b),[a,b]=>[a]]
f37=[id(b),[b]=>[b]]
f35=[th(f37,a),[a,b]=>[b]]
f33=[ais(f34,f35,a&b),[a,b]=>[a&b]]
f31=[id(b),[b]=>[b]]
f30=[th(f31,a),[a,b]=>[b]]
f29=[ala(f30,a&b),[a&b]=>[b]]
f32=[cut(f33,f29,a&b),[a,b]=>[b]]
f28=[his(f32,a->b),[b]=>[a->b]]
f26=[id(a),[a]=>[a]]
f27=[id(b),[b]=>[b]]
f25=[hia(f26,f27,a->b),[a,a->b]=>[b]]
f24=[his(f25,a->b),[a->b]=>[a->b]]
f23=[th(f24,a),[a->b]=>[a,a->b]]
f22=[ois(f23,a#(a->b)),[a->b]=>[a#(a->b)]]
f21=[cut(f28,f22,a->b),[b]=>[a#(a->b)]]
f20=[cut(f38,f21,b),[a&(a->b)]=>[a#(a->b)]]


Cut_free proof is :
===================

f168=[id(a),[a]=>[a]]
f152=[id(a),[a]=>[a]]
f151=[id(b),[b]=>[b]]
f148=[hia(f152,f151,a->b),[a,a->b]=>[b]]
f145=[ala(f148,a&(a->b)),[a&(a->b)]=>[b]]
f169=[th(f145,a),[a,a&(a->b)]=>[b]]
f167=[th(f169,b),[a,b,a&(a->b)]=>[b]]
f164=[hia(f168,f167,a->b),[a,a,a->b,a&(a->b)]=>[b]]
f161=[ala(f164,a&(a->b)),[a,a&(a->b),a&(a->b)]=>[b]]
f158=[th(f161,a),[a,a&(a->b),a,a&(a->b)]=>[b]]
f155=[inc(f158,a,a&(a->b)),[a,a,a&(a->b),a&(a->b)]=>[b]]
f128=[con(f155,a),[a,a&(a->b),a&(a->b)]=>[b]]
f123=[con(f128,a&(a->b)),[a,a&(a->b)]=>[b]]
f120=[his(f123,a->b),[a&(a->b)]=>[a->b]]
f117=[th(f120,a),[a&(a->b)]=>[a,a->b]]
f20=[ois(f117,a#(a->b)),[a&(a->b)]=>[a#(a->b)]]


Normal proof is :
=================

f152=[id(a),[a]=>[a]]
f151=[id(b),[b]=>[b]]
f148=[hia(f152,f151,a->b),[a,a->b]=>[b]]
f128=[ala(f148,a&(a->b)),[a&(a->b)]=>[b]]
f123=[th(f128,a),[a,a&(a->b)]=>[b]]
f120=[his(f123,a->b),[a&(a->b)]=>[a->b]]
f117=[th(f120,a),[a&(a->b)]=>[a,a->b]]
f20=[ois(f117,a#(a->b)),[a&(a->b)]=>[a#(a->b)]]


The answer is :
===============

comp(pi_left_star(a,a->b),pi_left(a,a->b))
                         is not equal to
comp(comp(pi_right_star(a,a->b),alpha_right(pi_right(a,b))),
     epsilon(a,b))


yes
| ?-
```

Chapter 6

Notes on the Implementation

This chapter describes how the algorithms have been implemented and how the implementation works. The implementation is coded in C-Prolog and is divided into the following main parts, corresponding to the theoretical work described in chapters 1-5 :-

(6.1) Proof Checker Procedure

A sequent calculus derivation is represented in our system by a 'step-collection' of proof-steps. The syntax of proof-steps is given in the user manual (appendix 5), where a proof-step is represented by a name followed by a list of information about the axiom or the applied rule, the antecedent and the succedent of the sequent. Because of the nature of the cut-elimination and the normalisation algorithms, this representation of the derivation is chosen in order to simplify the calling, retracting, replacing and manipulating of parts of the stored proof without any logical changes to the other parts of the proof.

The axioms and rules of the unlabelled derivation systems have been implemented in the general case as in chapter 5 with some restrictions, where appropriate, for the specific cases of the deductive systems in chapters 1-5 (e.g the 'and-introduction-succedent' rule is implemented in its general case

$$\frac{G \rightarrow FaE \qquad G \rightarrow FbE}{G \rightarrow F\ a\&b\ E}$$

where we restrict the two lists F and E to be empty in the case of the theories of [cartesian, bicartesian, cartesian closed] categories.

The predicate "rule(X)" where X is an axiom or the conclusion of a rule, is used to construct proofs in the deductive systems. (e.g. for the 'and-introduction-succedent' rule X is

```
L = [ais(M,N,A & B), G => T],
```

where

M = [_, G => T1],

N = [_, G => T2],

and

T = E(A&B)F, T1 = EAF, T2 = EBF).

The proof is checked one step at a time from bottom up using the predicate "prove(X)", where X represents a proof-step, then it will take this X and try to satisfy the subgoal "rule(X)" which by turn tries to match one of the implemented axioms and rules. If the step is correct, it will be kept in the database for further use.

The predicate "proof" is used, as another way to check the correctness and assert derivations without using the formal syntax for proof-step. In order to check a proof-tree, call the predicate "proof", the system repeatedly waits to read either the name of the rule (or axiom) or the word 'end'. If the name is of an axiom or rule, the system asks for the name to be given to the proof-step, the name (or names) of the previous step (or steps) , if any , and the antecedent and the succedent formulae; then the system uses this information to construct a proof-step. The above predicate "prove(X)", where X is the constructed proof-step, is used to check the correctness of this step. The system repeats this process, until the word 'end', and then returns control to top-level.

(6.2) Interpretation Procedure

In this section, we discuss the code of the interpretation function S which interprets sequent calculus derivations as arrows of the free [cartesian,.......] categories. All the details in chapters 1-5 concerning the interpretation of the axioms and rules have been implemented using the predicate "interpr(Z,H)", where Z is the name of the last step of a derivation, and H is then instantiated as the interpretation of this derivation.

In order to implement the interpretation function S as defined in chapters 1-5, we must implement all the special cases for each rule (e.g the thinning rule is of the form

$$\frac{GD \rightarrow F}{GaD \rightarrow F}$$

in order to implement the interpretation of this rule, we have the following three possible cases which have three different interpretations:

i) G is empty.
ii) D is empty.
iii) G and D are not empty.)

We can divide the code of the interpretation function S into two parts according to the axioms and rules. In case of axioms A1, A2 and A3, the arrow H providing the interpretation is the arrow in A1, id(tr) and id(bo) respectively. In the case of the rules, to interpret the conclusion of the rule, the system goes recursively to compute first the interpretation of the premisses.

In order to interpret a proof as an arrow, we use the predicate "interpr(Z,H)", where Z is the name of the last step in the proof. The system reads the proof-step with name Z, then matches this step with one of the implemented axioms and rules and continues recursively to reach the leaves of the proof. H is then instantiated to the arrow representing Z.

(6.3) Cut-Elimination Procedure

This section discusses the implementation of the cut elimination algorithm described in Appendix 2. The clauses therein have been implemented in the same order as in that Appendix; using the predicate "cut_elim(Z = [cut(Z1,Z2,C), G => T])", where Z1 and Z2 depend on the clause being implemented (e.g when

Z1 = [als(Z3,Z4,A & B), G1 => T1],
Z2 = [th(Z5,A & B), G2 => T2],

this is the implementation of the clause (C.3) in Appendix 2).

In order to eliminate the cuts from a proof, we can use the predicate "cut_free(Z)", where Z represents the name of the last proof-step; the system will start by satisfying the subgoal "one_step" and keep satisfying this goal using the "repeat" technique in Prolog until there are no instances of cuts. In order to satisfy the goal "one_step", the system looks for an instance of the cut rule; if there is one, it will use the pattern-matching facilities of Prolog to match this instance with one of the "cut_elim" clauses. Then, satisfying the predicates "expcon", "expinc" and "expth" which are designed to solve the problem which arises in some clauses (e.g (C.3), (C.4), (C.20.3), ...) by producing proof-steps (with lists of contraction , thinning and interchange formulae), so these proofs must be expanded in the normal way, where the active formulae must be single and not lists. After satisfying the goals of "expcon", "expinc" and "expth", the system will purge; in other words it will retract all the unwanted proof-steps and put the proof in the right order (By this is meant, ordering the proof-tree in the database using the Traversal Postorder technique defined by:

traverse left subtree in postorder
traverse right subtree in postorder
visit root

e.g. the following proof-tree

```
                G
               ---
   D            E         F
  -------------          ---
        B                 C
       ------------------------
                A
```

is ordered in the database according to the postorder technique as follows :

D G E B F C A

So the system will be able to pick up the nearest instance of the 'cut' rule to the leaves.

The system will repeat the procedure of "one_step" until there are no more instances of 'cuts' ; the system then prints the proof, using the predicate "printall(Z)", where Z is the name of the last step in the original proof.

(6.4) Normalisation Procedure

This section discusses the implementation of the normalisation algorithms in appendices 3 & 4. The normalisation procedure is divided into two procedures

i) for elimination of contractions & interchanges.

ii) for ordering the application of rules according to the priority in the definition of the normal form.

(6.4.1) Contraction and Interchange-Elimination Procedure

This subsection concerns the clauses of Appendix 3 which are used to eliminate the instances of the contraction and interchange rules in cut-free proofs, for the theories of cartesian, bicartesian and distributive bicartesian categories. As for the theories of cartesian closed and bicartesian closed, there are some clauses to remove all the instances of the contraction and interchange rules with active formulae of the form a † b and a v b. That appendix contains some conditions on the thinning formulae. All these clauses have been implemented using the predicate "norm1(A)" , where A represents the last step in each clause.

In order to produce a contraction & interchange-free proof, we use the predicate "con_inc_free(X)" , where X represents the name of the last step in a cut-free proof. The system will first satisfy the subgoal "normall" , which will be satisfied by repeatedly satisfying the goal "do1_step" until no more reductions can be carried out on the proof. The procedure "do1_step" is satisfied by satisfying a number of subgoals as follows :

1) The predicate "put_in_list(X,L1)", where X represents the name of the last proof-step, copies the proof into a list L1.

2) The predicate "assert(proved(mark))" is used to assert the clause "proved(mark)" at the end of the proof in the database.

3) The predicate "norm1(proved(Z))" is used to match one proof-step at a time using the pattern-matching facilities, with one of the implemented clauses.

4) The system will continue the above step for the whole proof, when Z in the predicate "norm1(proved(Z))" is instantiated for 'mark' , the system then copies the result proof in a list "L2" using the predicate "put_in_list", abolishes all the proof-steps from the database and reasserts "L2" again in the database in the right order.

5) Finally, the system compares the two lists L1 (the proof before the reduction processes) and L2 (the proof after the reduction processes). If they are not the same, the predicate "do1_step" is repeated many times until the two lists are the same, this means that there are no more reductions. The system then prints the proof, using the predicate "printall".

(6.4.2) Priority Procedure

In this subsection, we discuss the implementation of the clauses in appendix 4. These are used to complete the normalisation of a proof by ordering the instances of the rules applied, according to the priority in the definition of 'normal form'.

After the procedure of the predicate "normal1" in the above subsection has been carried out, the system will continue to perform the priority procedure by satisfying the predicate "normal2". The code of the procedure "normal2" is similar to the code of the procedure "normal1", by replacing the predicate "do1_step" in "normal1" by the predicate "do2_step". The two procedures "do1_step" and "do2_step" differs only in one predicate; the first has the predicate "norm1(Z)" (which implements the clauses of the Appendix 3) and the second has the predicate "norm2(Z)" (which implements the clauses in the Appendix4 with Z representing the conclusion of each clause). This procedure ends by printing out the normal form proof.

Note

We constructed one procedure "normal" as a combination of three procedures altogether: "cut_free", "normal1" and "normal2", in addition to other little procedures such as "hook_terminal" and "term_inti" to characterise the normal proofs for the initial and terminal arrows. This "normal" procedure is used to produce normal form proofs.

(6.5) Representation Procedure

This section discusses the implementation of the completeness theorems mentioned in chapters 1-5, for the representation of arrows of the free [-cartesian, ...] categories by means of sequent calculus proofs. The arrow "Y" is represented by a proof by using the procedure "rep_of(Y)" . To satisfy the goal "rep_of(Y)", the system generates an atom, using the predicate "gensym", to represent the name of the last step in the result representation. Since each of the arrows in chapters 1 - 5 is written as a term with argument inside (e.g. the arrow 'comp(f,g)'), the system appends the generated name to the arguments of the arrow (e.g. in the above arrow, the result is 'comp(f,g,n)', where n is the name generated by "gensym"). The system matches this with one of implemented arrows (e.g. we may find the representation F of pi_left(A,B) by satisfying the goal pi_left(A,B,F)).

Note

By combining the two procedures "rep_of(Y)" and "normal", we constructed a procedure "norm_rep(Y)", which produces the normal representation for an arrow 'Y'.

(6.6) Commutativity Procedure

This procedure has been constructed to help deciding the equality between two arrows of a category. In order to do this we use the predicate "commutative" : to satisfy the goal "commutative" the system satisfies the following subgoals :

1) The system waits to read the first arrow Y.

2) The predicate "norm_rep(Y)" is satisfied by producing the normal representation for the arrow Y.

3) The predicate "tree(Z,L1)" is constructed in order to save the normal representation of the first arrow as a data structure in a list L1 . This must be done, since the database needs to be cleaned from the representation of the first arrow, in order to insert the second order. As an example, the following proof :

```
l = [f, [a] => [b]],
m = [th(l,c), [a,c] => [b]],
n = [aia(m,a & c), [a & c] => [b]].
```

is saved as a data structure in a list as follows :

```
[f : [a] => [b], th(c) : [a,c] => [b], aia(a&c) : [a&c] => [b]]).
```

This data structure is ordered in the list from right to left in the same order for the proof in the database, e.g in the above example the proof-step representing the conclusion of applying thinning comes directly after the proof-step representing the premiss of the thinning rule.

4) Then the predicate "clean" cleans all trace of the first proof from the database.

5) The predicate "read" causes the system to read the second arrow.

6) The same processes, as in the steps 2 - 4 above, for the second

arrow to produce a list L2 with the data structure for the normal representation of the second arrow.

7) Finally, the system compares the two lists L1 and L2; if they are the same then the two arrows are equal, and if they are not then the two arrows are not equal.

(6.7) Other subprocedures

There are other procedures for manipulating lists such as

i) append(L1,L2,L) : list L1 prepended before list L2 is list L.

ii) islist(A) : to check if A is a list or not.

iii) gensym(f,F) : to generate a new name F using the atom f.

iv) member(H,L) : to check if H is a member of the list L.

v) sub(X,C,L,L1) : this is to substitute C for X in the list L ; the result list is L1.

vi) clean : it is used after each session of use, in order to use the system using the same theory and the same arrows, if any, of the underlying category for another session.

vii) new : Cleans up the database from all the processes of the last session and puts the system again to the top level.

Conclusion

In this thesis, we have implemented, in Prolog, the cut-elimination and the normalisation algorithms for a variant of Gentzen's sequent calculus, in order to provide a mechanised version of the decision procedure for the word problem for various kinds of category. These algorithms were outlined in SZABO, for both the cartesian and monoidal kinds of category: we have restricted our work to the cartesian kind. Other algorithms have been implemented to interpret sequent calculus derivations as arrows in the [cartesian, ...] categories, and to represent arrows of these categories by means of sequent calculus derivations. Some of the details in SZABO's algorithm have been modified or expanded to allow implementation.

The pattern-matching and debugging facilities of Prolog were found to be well-suited for this type of work. But the algorithms are both space and time expensive, and our implementation in turn takes a lot of space and time. Our representation of the sequent calculus derivations in Prolog is too complicated; this seems to be the main reason behind the inefficiency of the system.

We do not propose to extend our implementation with other transformation rules to handle predicate logic, or other categories with more complicated structure such as limits and colimits . Another representation for sequent calculus derivations should be adopted, perhaps as in the type theoretic work of the PRL team (CONSTABLE [1985]).

In conclusion, this implementation has been useful as an educational exercise in formalising a piece of logic and category theory. Proof theoretic studies of the sequent calculus are of continuing importance for theorem-proving in predicate logic and type theory ; the present approach to category theory via the sequent calculus seems, in view of the recent work of CURIEN [1985], no longer to be the best approach. Nevertheless, as noted by HUET [1985a], proof theoretic techniques may yet be useful in proving, for example, termination properties of the rewriting systems studied by CURIEN.

References

CLOCKSIN, W. F. and MELLISH, C. S.

[1981] Programming in Prolog (Springer-Verlag, Berlin).

COLMERAUER, A., KANOUI, H. and VAN CANEGHEM, M.

[1983] Reports and Documents, Prolog, Theoretical principles and Current trends, Technology and Science of Informatics, vol. 2, no. 4, 255 - 292.

CONSTABLE, R.

[1985] Implementing mathematics. The NuPRL book, draft, Cornell Univ.

CURIEN, P. -L.

[1985] Categorical combinators, sequential algorithms and functional programming; (these d'etat) (CNRS - University Paris VII, LITP, No 85-26).

EILENBERG, S., HARRISON, D. K., MAcLANE, S. and ROHRL, H.(editors)

[1966] Proceedings of the conference on categorical algebra, La Jolla 1965 (Springer, New York).

EILENBERG, S. and KELLY, G. M.

[1966] Closed Categories, in : EILENBERG et al. [1966] pp. 421 - 562.

FEFERMAN, S.

[1968] Lectures on proof theory, Lecture Notes in Mathematics, vol. 70, pp. 1 - 107.

FENSTAD, J. (editor)

[1971] Proceedings of the Second Scandinavian Logic Colloquium (North-Holland, Amsterdam).

FITTING, M. C.

[1969] Intuitionist Logic, Model Theory and Forcing (North-Holland, Amsterdam).

HERRLICH, H. and STRECKER, G. E.

[1973] Category Theory (Allyn and Bacon, Boston).

HILTON, P. (editor)

[1969a] Category Theory, Homology Theory and their Applications I, Lecture Notes in Mathematics, vol. 86 (Springer, Berlin).

[1969b] Category Theory, Homology Theory and their Applications II, Lecture Notes in Mathematics, vol. 92 (Springer, Berlin).

HUET, G.

[1985a] Equational systems for category theory and intuitionistic logic; talk at Rewriting Techniques & Applications Conference, Dijon.

[1985b] Cartesian Closed Categories and Lambda-Calculus, preprint CMU and INRIA.

HUET, G. and OPPEN, D.

[1980] Equations and rewrite rules : A survey, in "Formal Language: Perspectives and Open Problems " (R. BOOK Ed.), Academic Press, New York.

KELLY, G. M.

[1964] On MAcLANE's conditions for coherence of natural associativities, etc., Journal of Algebra 1, 397 - 402.

KELLY, G. M. and MAcLANE, S.

[1971] Coherence in closed categories, Journal of Pure and Applied Algebra 1, 97 - 140.

KREISEL, G.

[1968] A survey of proof theory I, Journal of Symbolic Logic 33, 321 - 388.

[1971] A survey of proof theory II, in : FENSTAD [1971] pp. 109-170.

LAMBEK, J.

[1968] Deductive systems and categories I, Mathematical Systems Theory 2, 287 - 318.

[1969] Deductive systems and categories II, Lecture Notes in Mathematics, volume 86, pp. 76 - 122.

[1972] Deductive systems and categories III, Lecture Notes in Mathematics, volume 274, pp. 57 - 82.

[1974] Functional completeness of cartesian categories, Annals Mathematical Logic 6, 259 - 292.

LAMBEK, J. and SCOTT, P. J.

[1984] Cartesian closed categories and lambda-calculus, provisional version of part I of a book on Higher Order Categorical Logic (to appear, Cambridge University Press).

LAPLAZA, M.

[1972] Coherence for distributivity, Lecture Notes in Mathematics, vol. 281, pp. 29 - 65.

LAWVERE, F. W.

[1969] Diagonal arguments and cartesian closed categories, Lecture Notes in Mathematics, vol. 86, pp. 134 - 145.

MAcLANE, S.

[1963] Natural associativity and commutativity, Rice University studies 49, 28 - 46.

[1971] Categories for the Working Mathematician (Springer, Berlin).

[1972] (editor) Coherence in Categories, Lecture Notes in Mathematics, vol. 281 (Springer Verlag, Berlin).

MANN, C.

[1975] The connection between equivalences of proofs and cartesian closed categories, Proceedings of the London Mathematical Society, Series 3 31, 289 - 310.

NAUR, P. et al.

[1960] Report on the algorithmic language Algol 60 (1960), comm. ACM 3:5, 299 - 314, revised in comm. ACM 6 : 1, 1 - 17.

PEREIRA, F.

[1984] C-Prolog User's Manual, version 1.5, Edinburgh Computer Aided Architectural Design.

POTTINGER, G.

[1979] The Church-Rosser Theorem for the typed lambda-calculus with Extensional Pairing, preprint, Carngie-Mellon University, Pittsburgh.

PRAWITZ, D.

[1965] Natural Deduction (Almqvist & Wiksell, Stockholm).

[1971] Ideas and results in proof theory, in : FENSTAD [1971] pp. 235 – 307.

ROSE, H. E. and SHEPHERDSON, J. C. (editors)

[1975] Logic Colloquium 73 (North-Holland, Amsterdam).

SZABO, M. E.

[1969] (editor) The collected papers of Gerhard Gentzen (North-Holland, Amsterdam).

[1971] Gentzen methods applied to category theory I : Free symmetric monoidal closed categories, Journal of Symbolic Logic 36, 705.

[1972] Gentzen methods applied to category theory II : Free cartesian closed multicategories, Journal of Symbolic Logic 37, 434.

[1973] Multinomial coefficients in closed categories, Notice of the American Mathematical Society 73T, A190.

[1974] A categorical equivalence of proofs, Noter Dame Journal of Formal Logic 15, 177 – 191.

[1975] A counter-example to coherence in cartesian closed categories, Canadian Mathematical Bulletin 18, 111 – 114.

[1976] An addendum to my paper " A categorical equivalence of proofs", Noter Dame Journal of Formal Logic 17, 78.

[1977] The logic of closed categories, Noter Dame Journal of Formal Logic 18, 441 – 457.

[1978] Algebra of Proofs (North-Holland, Amsterdam).

Appendix 1

Apparent errors in the printing of SZABO [1978]

In this appendix, we list some apparent errors in the printing of SZABO, as observed in those chapters and appendices concerning the theories of: cartesian, bicartesian, distributive bicartesian, cartesian closed, and bicartesian closed categories.

Notation :

The following notations are used in this appendix to correspond to SZABO's notations, because of the lack of convenient printer characters for Greek letters :

a	for	α	||	b	for	β
c	for	γ	||	d	for	δ
alpha_star	for	α^*	||	alpha_star_inverse	for	$(\alpha^*)^{-1}$
L	for	Λ	||	G	for	Γ
D	for	Δ	||	F	for	Φ
H	for	Θ	||	E	for	Ψ
Z	for	Ξ	||	P	for	Π
O	for	Ω				

page 59

line 3 from the bottom replace a -> b by
a -> c

page 60

line 2 from the bottom replace delta_star(A) : A -> A v A by
delta_star(A) : A v A -> A

page 61

in (6), line 4 replace alpha_star_inverse by alpha_star

page 66

line 2, replace A v T by A ↑ T

page 68

line 4 replace a v b and a ↑ b respectively by
a ↑ b and a v b respectively

page 75

line 2 replace (A v B) ↑ (A ↑ C) by
(A v B) ↑ (A v C)

page 80

In the diagram, replace occurrences of v D
by ↑ D

page 85

line 4 in lemma(6.6.4), omit ' respectively ' (cf (3.5.7)).

In lemma (6.6.5), replace ⊥ v ⊥ by T v T

page 87

Remove (D.72) from the proof of the normalization theorem (6.6.9) : its effect is trivial, (and also the deductive system dbcD(X) does not include the interchange rule).

page 137

line 1 replace (R2) and (R3)
by (R2), (R3), and (R4)

page 230

In (C.4), penultimate line, interchange L, G.

page 231

In (C.10), line 2 from the bottom replace LGa
by LDa

page 233

In (C.19.1) and (C.19.2), replace H
by FHE in lines 2,3 of the right-hand part of the clause. Similarly in line 3 of the left-hand part of the clause.

page 237

In (C.30.1), line 2 replace ZDcLDa => bP -> F
by ZDcLa => bP -> F

page 238

In (C.32.3), line 2 replace => by <=

page 240

In (C.39.1), line 3 replace DGL -> FOHbaH
by DGL -> FOEbaH

In (C.39.3), line 3 replace DGL -> FOdE
by DGL -> FHdE

page 247

In (D.11.1), line 2 replace f by g

page 250

In (D.19.1), replace the last D in all the lines of the left-hand part by L.

page 261

clause (D.49) is replaced by

$$\frac{\dfrac{GabD \xrightarrow{f} FcE}{Ga \uparrow bD \to FcE} \qquad \dfrac{GabD \xrightarrow{g} FdE}{Ga \uparrow bD \to FdE}}{Ga \uparrow bD \to Fc \uparrow dE} \quad > \quad \frac{\dfrac{GabD \xrightarrow{f} FcE \qquad GabD \xrightarrow{g} FdE}{GabD \to Fc \uparrow dE}}{Ga \uparrow bD \to Fc \uparrow dE}$$

page 266

In (D.65.2), line 2 replace DG => bLcZ -> d
by DGa => bLcZ -> d

In (D.65.3), line 1 replace big phi
by small phi

page 268

In (D.72.1), replace

$$\frac{GdDaL \xrightarrow{g} F}{GdcDaL \to F} \quad \text{by} \quad \frac{GdDbL \xrightarrow{g} F}{GdcDbL \to F}$$

page 271

In (D.76.7), in the last line replace a v b by b v a

page 272

In (D.76.9), line 2 replace GbcD -> F
by GbaD -> F

page 273

In (D.77.1), line 1 replace GcddDbL -> FbE
by GcddD -> FbE

Appendix 2

Cut Elimination Algorithm

In this appendix, we collect all the clauses used to prove the cut-elimination theorems. These are implemented in our system, in order to produce an equivalent cut-free representation for every arrow of each of the free categories defined in chapters 1 - 5.

(A1,A1) For all A,B,C in Ob(X) , all f in X(A,B) , and all g in X(B,C),

```
   f         g
A -> B    B -> C                comp(g,f)
-----------------   >    A  ----------->  C          (C.1)
     A -> C
```

(A2,R2) For all A,B in Ob(X) , and all f in X(A,B) ,

```
                    g
                GD -> F
   f           ---------                g
A -> B         GBD -> F          GD -> F
------------------------    >    --------            (C.2.1)
       GAD -> F                  GAD -> F
```

(A3,R2)

```
                    f
                GD -> F
  .            ---------
 -> T          GTD -> F    >     GD -> F             (C.2.3)
------------------------
       GD -> F
```

(R10,R2)

```
                                                                        h
                                                                    DL -> H
                                                                   ---------
                                                                   DbL -> H
                                                                  ----------
                                                            g
                                                       G -> FbE   DabL -> H
                                                       --------------------
                                            f
                                       G -> FaE         DaGL -> FHE
                                       ----------------------------
                                             DGGL -> FFHEE
                                             -------------
   f          g                  h           DGGL -> FFHE
G -> FaE   G -> FbE         DL -> H          ------------
-------------------         ---------        DGGL -> FHE
  G -> Fa ↑ bE              Da↑bL -> H       -----------
  --------------------------------     >     DGL -> FHE          (C.3)
          DGL -> FHE
```

(R13,R2)

```
                                                     g
                                                 DL -> H
                                    f           ---------         g
                                 G -> FabE      DbL -> H      DL -> H
                                 -----------------------      --------
                                      DGL -> FaHE             DaL -> H
                                      --------------------------------
                                            DDGLL -> FHHE
                                            -------------
   f                         g              DDGLL -> FHE
G -> FabE               DL -> H             ------------
----------            ---------             DDGL -> FHE
G -> Fa v bE        Da v bL -> H            -----------
----------------------------------     >    DGL -> FHE           (C.4)
          DGL -> FHE
```

(R15,R2)

```
          f                              g                  g
GaD  ->  b                         LZ  ->  H          LZ  ->  H
---------------              ---------------          ---------
GD  ->  a  =>  b         La  =>  bZ  ->  H           LGZ  ->  H
-----------------------------------------------  >  ------------
            LGDZ  ->  H                             LGDZ  ->  H
```

(C.5)

(R5,A1)

```
    f
G  ->  FE
----------
                 g                f
G  ->  FAE     A  ->  B       G  ->  FE
-------------------------  >  ----------
     G  ->  FBE               G  ->  FBE
```

(C.7.1)

(R5,A4)

```
    f
G  ->  FE
----------
                                  f
G  ->  F  E      ⊥  ->        G  ->  FE
-------------------------  >
     G  ->  FE
```

(C.7.2)

(R5,R11)

```
                                                                       f
                                                                   G  ->  FE
                                                          f        ----------                  g
                                                      G  ->  FE    G  ->  FbE       DabL  ->  H
                                                      ----------   ---------------------------
                                                      G  ->  FaE             DaGL  ->  FHE
                                                      ------------------------------------
                                                                DDGL  ->  FFHEE
    f                                    g                      ---------------
G  ->  FE                      DabL  ->  H                      DGGL  ->  FFHE
--------------            ----------------                      --------------
G  ->  Fa  ↑  bE          Da  ↑  bL  ->  H                      DGGL  ->  FHE
------------------------------------------  >                   --------------
           DGL  ->  FHE                                         DGL  ->  FHE
```

(C.8)

(R5,R12)

```
                                                                   f
                                                               G  ->  FE
                                                               ----------
                                                               G  ->  FbE
                                                               -----------            g
                                                               G  ->  FabE     DaL  ->  H
                                                               --------------------------          h
                                                                      DGL  ->  FHbE          DbL  ->  H
                                                                      --------------------------------
                                                                          DDGLL  ->  FHHE
    f                          g              h                           ---------------
G  ->  FE               DaL  ->  H     DbL  ->  H                         DDGLL  ->  FHE
--------------          ------------------------                          --------------
G  ->  Fa  v  bE            Da  v  bL  ->  H                              DDGL  ->  FHE
------------------------------------------------  >                       --------------
            DGL  ->  FHE                                                  DGL  ->  FHE
```

(C.9)

(R5,R14)

```
                                                                            f
                                                                       G  ->  FE
                                                             g         ----------
                                                          D  ->  a     aG  ->  FE
                                                          -------------------------
    f                        g             h                  DG  ->  FE
G  ->  FE                 D  ->  a     LbZ  ->  H             ----------                h
---------------           --------------------                DG  ->  FbE        LbZ  ->  H
G  ->  Fa  =>  bE          LDa  =>  bZ  ->  H                 ---------------------------
-------------------------------------------------  >                 LDGZ  ->  FHE
          LDGZ  ->  FHE
```

(C.10)

(R5,R2)

```
                                                       f
                                                 G --> FE
         f              g                        --------
   G --> FE        DL --> H                      G --> FHE
   --------        --------                      ---------
   G --> FaE       DaL --> H                     GL --> FHE
   -------------------------                 > ----------      (C.12)
         DGL --> FHE                           DGL --> FHE
```

(R10,R11)

```
                                                          g             h
                                                      G --> FbE   DabL --> H
                                              f       ------------------------
                                          G --> FaE         DaGL --> FHE
                                          ----------------------------------
     f          g             h                      DGGL --> FFHEE
 G --> FaE   G --> FbE    DabL --> H                 -------------
 ----------------------   ----------                 DGGL --> FFHE
     G --> Fa + bE        DatbL --> H                -------------
     -------------------------------                 DGGL --> FHE
                                              >      ------------       (C.14)
           DGL --> FHE                               DGL --> FHE
```

(R13,R12)

```
                                                 f              g
                                             G --> FabE   DaL --> H
                                             ----------------------      h
                                                 DGL --> FHbE       DbL --> H
                                                 ----------------------------
     f              g            h                      DDGLL --> FHHE
 G --> FabE    DaL --> H    DbL --> H                   --------------
 ----------    ------------------------                 DDGLL --> FHE
 G --> Fa v bE     Da v bL --> H                        -------------
 ---------------------------------------                DDGL --> FHE
                                              >         ------------      (C.15)
          DGL --> FHE                                   DGL --> FHE
```

(R15,R14)

```
                                                    g          f
                                                 L --> a   GaD --> b
        f               g           h            -------------------      h
 GaD --> b           L --> a    ZbS --> H            GLD --> b       ZbS --> H
 ---------           ---------------------           ------------------------
 GD --> a => b       ZLa => bS --> H                       ZGLDS --> H
 ---------------------------------------      >            -----------   (C.16)
          ZLGDS --> H                                      ZLGDS --> H
```

(-,R2)

```
                                g                 f             g
                         DcLZ --> H           G --> FcE   DcLZ --> H
        f                ----------           ----------------------
 G --> FcE              DcLaZ --> H               DGLZ --> FHE
 ----------------------------------       >       ------------             (C.18.1)
          DGLaZ --> FHE                           DGLaZ --> FHE
```

```
                                g                 f             g
                         DLcZ --> H           G --> FcE   DLcZ --> H
        f                ----------           ----------------------
 G --> FcE              DaLcZ --> H               DLGZ --> FHE
 ----------------------------------       >       ------------             (C.18.2)
          DaLGZ --> FHE                           DaLGZ --> FHE
```

(-,R3)

```
                                 g                f               g
                        DcLaaZ --> H          G --> FcE   DcLaaZ --> H
        f               ------------          ------------------------
 G --> FcE              DcLaZ --> H               DGLaaZ --> FHE
 ----------------------------------       >       --------------           (C.19.1)
          DGLaZ --> FHE                           DGLaZ --> FHE
```

```
                                 g                f               g
                        DaaLcZ --> H          G --> FcE   GaaLcZ --> H
        f               ------------          ------------------------
 G --> FcE              DaLcZ --> H               DaaLGZ --> FHE
 ----------------------------------       >       --------------           (C.19.2)
          DaLGZ --> FHE                           DaLGZ --> FHE
```

```
                                                   f                  g
                                                G -> FcE      DccL -> H
                                      f         ---------------------------
                                   G -> FcE          DcGL -> FHE
                                   ------------------------------------
                                        DGGL -> FFHEE
                       g                ---------------
                 DccL -> H              DGGL -> FFHE
        f        ---------              -------------
     G -> FcE     DcL -> H              DGGL -> FHE
     ------------------------    >      -------------           (C.19.3)
         DGL -> FHE                     DGL -> FHE
```

(-,R4)

```
                          g              f                 g
                DcLabZ -> H           G -> FcE   DcLabZ -> H
        f       -----------           ------------------------
     G -> FcE   DcLbaZ -> H                DGLabZ -> FHE
     ------------------------    >         --------------        (C.20.1)
        DGLbaZ -> FHE                      DGLbaZ -> FHE

                          g              f                 g
                DabLcZ -> H           G -> FcE   GabLcZ -> H
        f       -----------           ------------------------
     G -> FcE   DbaLcZ -> H                DabLGZ -> FHE
     ------------------------    >         --------------        (C.20.2)
        DbaLGZ -> FHE                      DbaLGZ -> FHE

                        g                f               g
                 DdcL -> H            G -> FcE    DdcL -> H
        f        ---------            ----------------------
     G -> FcE    DcdL -> H                DdGL -> FHE
     ------------------------    >        -----------            (C.20.3)
        DGdL -> FHE                       DGdL -> FHE

                        g                f               g
                 DcdL -> H            G -> FcE    DcdL -> H
        f        ---------            ----------------------
     G -> FcE    DdcL -> H                DGdL -> FHE
     ------------------------    >        -----------            (C.20.4)
        DdGL -> FHE                       DdGL -> FHE
```

(-,R5)

```
                       g                 f              g
                 DcL -> HO            G -> FcE    DcL -> HO
        f        ----------           ----------------------
     G -> FcE    DcL -> HaO                DGL -> FHOE
     ------------------------    >         -------------         (C.21)
        DGL -> FHaOE                       DGL -> FHaOE
```

(-,R6)

```
                       g                 f              g
                 DcL -> HaaO          G -> FcE    DcL -> HaaO
        f        -----------          ------------------------
     G -> FcE    DcL -> HaO                DGL -> FHaaOE
     ------------------------    >         --------------        (C.22)
        DGL -> FHaOE                       DGL -> FHaOE
```

(-,R7)

```
                       g                 f              g
                 DcL -> HabO          G -> FcE    DcL -> HabO
        f        -----------          ------------------------
     G -> FcE    DcL -> HbaO                DGL -> FHabOE
     ------------------------    >          --------------       (C.23)
        DGL -> FHbaOE                       DGL -> FHbaOE
```

(-,R10)

```
                     g              h          f            g            f             h
             DcL -> HaO     DcL -> HbO      G -> FcE   DcL -> HaO     G -> FcE   DcL -> HbO
     f       ----------------------------   ----------------------    ----------------------
  G -> FcE        DcL -> Ha ↑ bO                DGL -> FHaOE               DGL -> FHbOE
  ------------------------------------  >       ----------------------------------------
        DGL -> FHa ↑ bOE                                  DGL -> FHa ↑ bOE
                                                                                 (C.26)
```

(-,R11)

```
                          g                    f                 g
                 DcLabZ  -> H           G  -> FcE    DcLabZ  -> H
       f         ------------          -----------------------------
G  -> FcE    DcLa ↑ bZ -> H                 DGLabZ -> FHE
------------------------------   >          ----------------
     DGLa ↑ bZ -> FHE                       DGLa ↑ bZ -> FHE
                                                     (C.27.1)
```

```
                          g                    f                 g
                 DabLcZ  -> H           G  -> FcE    DabLcZ  -> H
       f         ------------          -----------------------------
G  -> FcE    Da ↑ bLcZ -> H                 DabLGZ -> FHE
------------------------------   >          ----------------
     Da ↑ bLGZ -> FHE                       Da ↑ bLGZ -> FHE
                                                     (C.27.2)
```

(-,R12)

```
                   g             h          f             g          f             h
            DcLaZ -> H    DcLbZ -> H    G -> FcE   DcLaZ -> H    G -> FcE   DcLbZ -> H
    f       ------------------------    ----------------------   ----------------------
G -> FcE        DcLa v bZ -> H              DGLaZ -> FHE              DGLbZ -> FHE
--------------------------------   >        -------------------------------------------
    DGLa v bZ -> FHE                                  DGLa v bZ -> FHE
                                                              (C.28.1)
```

```
                   g             h          f             g          f             h
            DaLcZ -> H    DbLcZ -> H    G -> FcE   DaLcZ -> H    G -> FcE   DbLcZ -> H
    f       ------------------------    ----------------------   ----------------------
G -> FcE        Da v bLcZ -> H              DaLGZ -> FHE              DbLGZ -> FHE
--------------------------------   >        -------------------------------------------
    Da v bLGZ -> FHE                                  Da v bLGZ -> FHE
                                                              (C.28.2)
```

(-,R13)

```
                      g                    f           g
                 DcL -> HabO          G -> FcE    DcL -> HabO
       f         -----------          ------------------------
G  -> FcE        DcL -> Ha v bO            DGL -> FHabOE
-------------------------------   >        ----------------
     DGL -> FHa v bOE                      DGL -> FHa v bOE (C.29)
```

(-,R14)

```
                    g          h           f         g
               DcL -> a   ZbP -> F     G -> c   DcL -> a
      f        -------------------     ----------------           h
G -> c          ZDcLa => bP -> F           DGL -> a          ZbP -> F
-------------------------------   >        ----------------------------
     ZDGLa => bP -> F                          ZDGLa => bP -> F
                                                        (C.30.1)
```

```
                  g           h                   f           h
               D -> a   LcZbP -> F             G -> c   LcZbP -> F
      f        -------------------             -------------------
G -> c          LcZDa => bP -> F          g          LGZbP -> F
-------------------------------   >    D -> a
                                       ------------------------------
     LGZDa => bP -> F                    LGZDa => bP -> F (C.30.2)
```

```
                  g           h                   f           h
               D -> a   LbZcP -> F             G -> c   LbZcP -> F
      f        -------------------             -------------------
G -> c          LDa => bZcP -> F          g          LbZGP -> F
-------------------------------   >    D -> a
                                       ------------------------------
     LDa => bZGP -> F                    LDa => bZGP -> F (C.30.3)
```

(-,R15)

```
                         g              f            g
                 DcLaZ -> b         G -> c   DcLaZ -> b
      f          ------------       ----------------------
G -> c           DcLZ -> (a => b)      DGLaZ -> b
---------------------------------  >   ----------------
     DGLZ -> (a => b)                  DGLZ -> (a => b) (C.31.1)
```

```
                    g                        f             g
           DaLcZ   -> b               G  -> c   DaLcZ    -> b
           -----------------           --------------------
      f                                   DaLGZ  -> b
G  -> c    DLcZ  -> (a => b)             ---------------
-----------------------------   >         DLGZ  -> (a => b)  (C.31.2)
    DLGZ  -> (a => b)
```

(R2,-)

```
     f                                 f              g
GD  -> FcE                       GD  -> FcE     LcZ  -> H
-----------                      ------------------------
                    g                 LGDZ  -> FHE
GaD  -> FcE    LcZ  -> H              -------------
-------------------------   >         LGaDZ  -> FHE        (C.34)
    LGaDZ  -> FHE
```

(R3,-)

```
      f                                 f              g
GaaD  -> FcE                     GaaD  -> FcE    LcZ  -> H
-------------                    -------------------------
                    g                LGaaDZ  -> FHE
GaD  -> FcE    LcZ  -> H             --------------
-------------------------   >        LGaDZ  -> FHE         (C.35)
    LGaDZ  -> FHE
```

(R4,-)

```
      f                                 f              g
GabD  -> FcE                     GabD  -> FcE    LcZ  -> H
-------------                    -------------------------
                     g               LGabDZ  -> FHE
GbaD  -> FcE    LcZ  -> H            --------------
-------------------------   >        LGbaDZ  -> FHE        (C.36)
   LGbaDZ  -> FHE
```

(R5,-)

```
     f                                f            g
G  -> FcEH                       G  -> FcEH   DcL  -> O
-----------                      ----------------------
                     g               DGL  -> FOEH
G  -> FcEaH     DcL  -> O            -------------
--------------------------   >       DGL  -> FOEaH         (C.37.1)
    DGL  -> FOEaH

     f                                f            g
G  -> FEcH                       G  -> FEcH   DcL  -> O
-----------                      ----------------------
                     g               DGL  -> FEOH
G  -> FaEcH     DcL  -> O            -------------
--------------------------   >       DGL  -> FaEOH         (C.37.2)
    DGL  -> FaEOH
```

(R6,-)

```
     f                                f              g
G  -> FcEaaH                     G  -> FcEaaH   DcL  -> O
-------------                    ------------------------
                     g               DGL  -> FOEaaH
G  -> FcEaH     DcL  -> O            --------------
--------------------------   >       DGL  -> FOEaH         (C.38.1)
    DGL  -> FOEaH

     f                                f              g
G  -> FaaEcH                     G  -> FaaEcH   DcL  -> O
-------------                    ------------------------
                     g               DGL  -> FaaEOH
G  -> FaEcH     DcL  -> O            --------------
--------------------------   >       DGL  -> FaEOH         (C.38.2)
    DGL  -> FaEOH

                                      f             g
                                 G  -> FccE    DcL  -> H
                                 -----------------------
                                                               g
                                      DGL  -> FcHE        DcL  -> H
                                      ---------------------------------
                                             DDGLL  -> FHHE
                                             --------------
                                             DDGLL  -> FHE
     f                                       -------------
G  -> FccE                                   DDGL  -> FHE
-----------                                  ------------
                     g                       DGL  -> FHE   (C.38.3)
G  -> FcE       DcL  -> H
--------------------------   >
    DGL  -> FHE
```

(R7,-)

```
   f                                    f              g
G -> FcEabH                          G -> FcEabH   DcL -> O
-----------                          ----------------------
                   g                    DGL -> FOEabH
G -> FcEbaH    DcL -> O                 -------------
-----------------------     >           DGL -> FOEbaH       (C.39.1)
    DGL -> FOEbaH
```

```
   f                                    f              g
G -> FabEcH                          G -> FabEcH   DcL -> O
-----------                          ----------------------
                   g                    DGL -> FabEOH
G -> FbaEcH    DcL -> O                 -------------
-----------------------     >           DGL -> FbaEOH       (C.39.2)
    DGL -> FbaEOH
```

```
   f                                    f           g
G -> FdcE                            G -> FdcE  DcL -> H
---------                            -------------------
                   g                     DGL -> FdHE
G -> FcdE      DcL -> H                  -----------
-----------------------     >            DGL -> FHdE        (C.39.3)
      DGL -> FHdE
```

```
   f                                    f           g
G -> FcdE                            G -> FcdE  DcL -> H
---------                            -------------------
                   g                     DGL -> FHdE
G -> FdcE      DcL -> H                  -----------
-----------------------     >            DGL -> FdHE        (C.39.4)
      DGL -> FdHE
```

(R10,-)

```
   f          g                          f           h          g            h
G -> FcEaH  G -> FcEbH                G -> FcEaH DcL -> O    G -> FcEbH  DcL -> O
----------------------                -------------------    --------------------
                       h                 DGL -> FOEaH            DGL -> FOEbH
   G -> FcEa ↑ bH  DcL -> O              --------------------------------------
   ------------------------    >              DGL -> FOEa ↑ bH
       DGL -> FOEa ↑ bH                                           (C.41.1)
```

```
   f          g                          f           h          g            h
G -> FaEcH  G -> FbEcH                G -> FaEcH DcL -> O    G -> FbEcH  DcL -> O
----------------------                -------------------    --------------------
                       h                 DGL -> FaEOH            DGL -> FbEOH
   G -> Fa ↑ bEcH  DcL -> O              --------------------------------------
   ------------------------    >              DGL -> Fa ↑ bEOH
       DGL -> Fa ↑ bEOH                                           (C.41.2)
```

(R11,-)

```
        f                                   f         g
GabD -> FcE                         GabD -> FcE   LcZ -> H
-----------                         ----------------------
                      g                 LGabDZ -> FHE
Ga ↑ bD -> FcE    LcZ -> H              -----------------
--------------------------    >         LGa ↑ bDZ -> FHE(C.42)
    LGa ↑ bDZ -> FHE
```

(R12,-)

```
     f          g                         f         h          g         h
GaD -> FcE  GbD -> FcE               GaD -> FcE LcZ -> H   GbD -> FcE LcZ -> H
----------------------               -------------------   -------------------
                      h                 LGaDZ -> FHE           LGbDZ -> FHE
   Ga v bD -> FcE  LcZ -> H             ------------------------------------
   ------------------------    >             LGa v bDZ -> FHE
     LGa v bDZ -> FHE                                          (C.43)
```

(R13,-)

```
   f                                      f              g
G -> FcEabH                            G -> FcEabH   DcL -> O
-----------                            ----------------------
                        g                 DGL -> FOEabH
G -> FcEa v bH      DcL -> O              ----------------
----------------------------    >         DGL -> FOEa v bH
     DGL -> FOEa v bH                                      (C.44.1)
```

```
     f                                       f                 g
G  ->  FabEcH                           G  ->  FabEcH   DcL  ->  O
-------------                           -------------------------
                            g
G  ->  Fa v bEcH     DcL  ->  O              DGL  ->  FabEOH
-------------------------------   >          ---------------
       DGL  ->  Fa v bEOH                    DGL  ->  Fa v bEOH
                                                          (C.44.2)
```

(R14,-)

```
     f          g                                      g             h
G  ->  a   DbL  ->  FcE                         DbL  ->  FcE   ZcP  ->  H
-------------------                                ----------------------------
                              h                  f
  DGa => bL  ->  FcE   ZcP  ->  H           G  ->  a        ZDbLP  ->  FHE
  ------------------------------   >      ------------------------------
       ZDGa => bLP  ->  FHE                  ZDGa => bLP  ->  FHE
                                                          (C.45)
```

Appendix 3

Contraction & Interchange Elimination Algorithm

In this appendix, we extend the relation > of appendix 2 to the global reducibility relation >= which is implemented in the system. The clauses of this appendix prove that for every cut-free derivation p which represents an arrow of one of the free categories in chapters 1-3, and which contains an instance of contraction or interchange, there exists a contraction-and-interchange-free q deriving the same sequent. As for chapters 4 and 5, it will remove all the instances of contraction and interchange with active formulae of the form a † b . The algorithm is constructed to prove some lemmas stated without proofs in SZABO, in order to complete the description of the normal form.

(R2,R2)

```
             f                            f
     GDL -----> F                 GDL -----> F
   --------------               --------------
    GaDL -----> F      >=        GDbL -----> F          (E.1)
   --------------               --------------
   GaDbL -----> F               GaDbL -----> F
```

(R2,R3)

```
             f
     GaD -----> F
   --------------                         f
    GaaD -----> F      >=         GaD -----> F          (E.2.1)
   --------------
     GaD -----> F
```

```
              f                          f
    GDaaL ----> F                GDaaL ----> F
   --------------               --------------
   GbDaaL ----> F      >=         GDaL ----> F          (E.2.2)
   --------------               --------------
    GbDaL ----> F                GbDaL ----> F
```

```
              f                          f
    GaaDL ----> F                GaaDL ----> F
   --------------               --------------
   GaaDbL ----> F      >=         GaDL ----> F          (E.2.3)
   --------------               --------------
    GaDbL ----> F                GaDbL ----> F
```

(R2,R4)

```
             f
     GaD -----> F                         f
   --------------                 GaD -----> F
    GabD -----> F      >=       --------------          (E.3.1)
   --------------                GbaD -----> F
    GbaD -----> F
```

```
             f
     GaD -----> F                         f
   --------------                 GaD -----> F
    GbaD -----> F      >=       --------------          (E.3.2)
   --------------                GabD -----> F
    GabD -----> F
```

```
                  f                         f
        GDabL ----> F             GDabL ----> F
      ---------------           ---------------
      GcDabL ----> F      >=      GDbaL ----> F           (E.3.3)
      ---------------           ---------------
      GcDbaL ----> F            GcDbaL ----> F

                  f                         f
        GabDL ----> F             GabDL ----> F
      ---------------           ---------------
      GabDcL ----> F      >=      GbaDL ----> F           (E.3.4)
      ---------------           ---------------
      GbaDcL ----> F            GbaDcL ----> F
```

(R2,R6)

```
             f                         f
      GD -----> FaaE            GD -----> FaaE
    ----------------          ----------------
    GbD -----> FaaE           GD ----->   FaE             (E.4)
    ----------------          ----------------
    GbD ----->   FaE          GbD -----> FaE
```

(R2,R7)

```
             f                         f
      GD -----> FabE            GD -----> FabE
    ----------------          ----------------
    GcD -----> FabE     >=    GD -----> FbaE              (E.5)
    ----------------          ----------------
    GcD -----> FbaE           GcD ----> FbaE
```

(R3,R4)

```
                   f                           f
    GaaDbcL -----> F              GaaDbcL -----> F
    -----------------             -----------------
     GaDbcL -----> F      >=      GaaDcbL -----> F        (E.6.1)
    -----------------             -----------------
     GaDcbL -----> F               GaDcbL -----> F

                   f                           f
    GbcDaaL -----> F              GbcDaaL -----> F
    -----------------             -----------------
    GbcDaL  -----> F      >=      GcbDaaL -----> F        (E.6.2)
    -----------------             -----------------
    GcbDaL  -----> F              GcbDaL  -----> F

                                               f
                   f                GaabL -----> F
        GaabL -----> F             ---------------
      ---------------               GabaL -----> F
         GabL -----> F    >=       ---------------        (E.6.3)
      ---------------               GbaaL -----> F
         GbaL -----> F             ---------------
                                     GbaL -----> F

                                               f
                   f                GbaaL -----> F
        GbaaL -----> F             ---------------
      ---------------               GabaL -----> F
         GbaL -----> F    >=       ---------------        (E.6.4)
      ---------------               GaabL -----> F
         GabL -----> F             ---------------
                                     GabL -----> F
```

(R3,R5)

```
               f                          f
     GaaD -----> FE              GaaD -----> FE
    ---------------             ---------------
     GaaD ----> FbE     >=        GaD -----> FE           (E.7)
    ---------------             ---------------
      GaD ----> FbE              GaD -----> FbE
```

(R3,R7)

```
                  f                                 f
        GccD ------> FabE                  GccD ------> FabE
        -----------------                  -----------------
         GcD ------> FabE        >=        GccD ------> FbaE        (E.8)
        -----------------                  -----------------
         GcD ------> FbaE                   GcD ------> FbaE
```

(R3,R10)

```
         f              g                   f            g
 GccD -> FaE    GccD -> FbE         GccD -> FaE    GccD -> FbE
 --------------------------         -----------    -----------
     GccD -> Fa ↑ bE          >=     GcD -> FaE     GcD -> FbE   (E.9)
 --------------------------         --------------------------
      GcD -> Fa ↑ bE                      GcD -> Fa ↑ bE
```

(R3,R11)

```
                 f                                  f
      GccDabL ------> F                  GccDabL ------> F
      -----------------                  -----------------
      GccDa ↑ bL ---> F        >=         GcDabL ------> F          (E.10.1)
      -----------------                  -----------------
       GcDa ↑ bL ---> F                  GcDa ↑ bL ---> F

                 f                                  f
      GabDccL ------> F                  GabDccL ------> F
      -----------------                  -----------------
      Ga ↑ bDccL ---> F        >=         GabDcL ------> F          (E.10.2)
      -----------------                  -----------------
       Ga ↑ bDcL ---> F                  Ga ↑ bDcL ---> F

                 f                                  f
        GaaaaD ------> F                   GaaaaD ------> F
   --------------------                    ---------------
     Ga ↑ aaaD ------> F                    GaaaD ------> F
   --------------------        >=          ---------------          (E.10.3)
   Ga ↑ aa ↑ aD ---> F                       GaaD ------> F
   --------------------                    ---------------
         Ga ↑ aD ---> F                    Ga ↑ aD ----> F

                 f                                  f
        GaaaaD ------> F                   GaaaaD ------> F
   --------------------                    ---------------
     Gaaa ↑ aD ------> F                    GaaaD ------> F
   --------------------        >=          ---------------          (E.10.4)
   Ga ↑ aa ↑ aD ---> F                       GaaD ------> F
   --------------------                    ---------------
         Ga ↑ aD ---> F                    Ga ↑ aD ----> F

                                                    f
                 f                         GababD ------> F
        GababD ------> F                   ---------------
   --------------------                    GaabbD ------> F
     Ga ↑ babD ------> F        >=         ---------------          (E.10.5)
   --------------------                     GaabD ------> F
   Ga ↑ ba ↑ bD ---> F                     ---------------
   --------------------                      GabD ------> F
         Ga ↑ bD ---> F                    ---------------
                                           Ga ↑ bD ----> F

                                                    f
                 f                         GababD ------> F
        GababD ------> F                   ---------------
   --------------------                    GaabbD ------> F
     Gaba ↑ bD ------> F        >=         ---------------          (E.10.6)
   --------------------                     GaabD ------> F
   Ga ↑ ba ↑ bD ---> F                     ---------------
   --------------------                      GabD ------> F
         Ga ↑ bD ---> F                    ---------------
                                           Ga ↑ bD ----> F
```

(R3,R12)

```
        f                 g                       f                g
GaDccL -> F    GbDccL -> F             GaDccL -> F    GbDccL -> F
--------------------------             -----------    -----------
    Ga v bDccL -> F           >=        GaDcL -> F     GbDcL -> F    (E.11.1)
--------------------------             --------------------------
     Ga v bDcL -> F                          Ga v bDcL -> F
```

```
          f                 g                          f             g
GccDaL -> F     GccDbL -> F             GccDaL -> F    GccDbL -> F
-----------------------------           -----------    -----------
     GccDa v bL -> F               >=   GcDaL -> F     GcDbL -> F      (E.11.2)
-----------------------------           --------------------------
     GcDa v bL -> F                           GcDa v bL -> F

   p          q              r          s
GaD -> F   GbD -> F       GaD -> F   GbD -> F
--------   --------       --------   --------
GaaD -> F  GbaD -> F      GabD -> F  GbbD -> F                 p         s
--------------------      --------------------            GaD -> F  GbD -> F
   Ga v baD -> F             Ga v bbD -> F           >=   ------------------
   ----------------------------------------                 Ga v bD -> F
           Ga v ba v bD -> F
           -----------------
              Ga v bD -> F                                     (E.11.3)

   p          q              r          s
GaD -> F   GbD -> F       GaD -> F   GbD -> F
--------   --------       --------   --------
GaaD -> F  GabD -> F      GbaD -> F  GbbD -> F                 p         s
--------------------      --------------------            GaD -> F  GbD -> F
   Gaa v bD -> F             Gba v bD -> F           >=   ------------------
   ----------------------------------------                 Ga v bD -> F
           Ga v ba v bD -> F
           -----------------
              Ga v bD -> F
                                                               (E.11.4)
```

(R3,R13)

```
          f                               f
   GccD ------> FabE              GccD ------> FabE
   ------------------             ------------------
   GccD ------> Fa v bE     >=     GcD ------> FabE       (E.12)
   ------------------             ------------------
    GcD ------> Fa v bE            GcD ------> Fa v bE
```

(R3,R14)

```
        f            g                 f
   GccD -> a    LbS -> e          GccD -> a
   ---------------------          ---------          g
     LGccDa => bS -> e      >=    GcD -> a      LbS -> e     (E.13.1)
     -----------------            ----------------------
     LGcDa => bS -> e               LGcDa => bS -> e

     h            k                                 k
   G -> a    DccLbS -> e                      DccLbS -> e
   ---------------------               h      -----------
     DccLGa => bS -> e      >=       G -> a   DcLbS -> e     (E.13.2)
     -----------------               --------------------
     DcLGa => bS -> e                 DcLGa => bS -> e

     h            j                                 j
   G -> a    DbLccS -> e                      DbLccS -> e
   ---------------------               h      -----------
     DGa => bLccS -> e      >=       G -> a   DbLcS -> e     (E.13.3)
     -----------------               --------------------
     DGa => bLcS -> e                 DGa => bLcS -> e
```

(R3,R15)

```
              f                          f
        GccDaL -> b                GccDaL -> b
        ------------------         -----------
        GccDL -> (a => b)    >=    GcDaL -> b              (E.14.1)
        ------------------         -----------------
        GcDL -> (a => b)           GcDL -> (a => b)

              f                          f
        GaDccL -> b                GaDccL -> b
        ------------------         -----------
        GDccL -> (a => b)    >=    GaDcL -> b              (E.14.2)
        ------------------         -----------------
        GDcL -> (a => b)           GDcL -> (a => b)
```

```
                 f                              f
          GcacD -> b                     GcacD -> b
          -------------------            -----------
          GccD -> (a => b)       >=      GccaD -> b
          -------------------            -----------            (E.14.3)
          GcD -> (a => b)                GcaD -> b
                                         ---------------
                                         GcD -> (a => b)

(R4,R5)
             f                                f
      GabD ------> FE                  GabD ------> FE
      -----------------                -----------------
      GabD ------> FcE        >=       GbaD ------> FE          (E.15)
      -----------------                -----------------
      GbaD ------> FcE                 GbaD ------> FcE

(R4,R6)
             f                                f
      GabD ------> FccE                GabD ------> FccE
      ------------------               ------------------
      GabD ------> FcE        >=       GbaD ------> FccE        (E.16)
      ------------------               ------------------
      GbaD ------> FcE                 GbaD ------> FcE

(R4,R10)
        f              g                  f              g
  GcdD -> FaE    GcdD -> FbE        GcdD -> FaE    GcdD -> FbE
  ---------------------------       -----------    -----------
      GcdD -> Fa ↑ bE         >=    GdcD -> FaE    GdcD -> FbE   (E.17)
  ---------------------------       --------------------------
      GdcD -> Fa ↑ bE                     GdcD -> Fa ↑ bE

(R4,R11)
                 f                            f
       GcdDabL -> F                   GcdDabL -> F
      -------------                  -------------
      Gc↑dDabL -> F        >=         GcdDbaL -> F              (E.18.1)
      -------------                  -------------
      Gc↑dDbaL -> F                  Gc↑dDbaL -> F

                 f                            f
       GabDcdL -> F                   GabDcdL -> F
      -------------                  -------------
      GabDc↑dL -> F        >=         GbaDcdL -> F              (E.18.2)
      -------------                  -------------
      GbaDc↑dL -> F                  GbaDc↑dL -> F

                                              f
                 f                      GabcD -> F
         GabcD -> F                  -------------
      -------------                     GacbD -> F
      Ga ↑ bcD -> F        >=        -------------              (E.18.3)
      -------------                     GcabD -> F
      Gca ↑ bD -> F                  -------------
                                     Gca ↑ bD -> F

                                              f
                 f                      GcabD -> F
         GcabD -> F                  -------------
      -------------                     GacbD -> F
      Gca ↑ bD -> F        >=        -------------              (E.18.4)
      -------------                     GabcD -> F
      Ga ↑ bcD -> F                  -------------
                                     Ga ↑ bcD -> F

(R4,R12)
          f              g                 f             g
  GcdDaL -> F    GcdDbL -> F       GcdDaL -> F    GcdDbL -> F
  ---------------------------      -----------    -----------
      GcdDa v bL -> F         >=   GdcDaL -> F    GdcDbL -> F   (E.19.1)
  ---------------------------      --------------------------
      GdcDa v bL -> F                  GdcDa v bL -> F
```

```
            f               g                    f               g
GaDcdL -> F     GbDcdL -> F         GaDcdL -> F      GbDcdL -> F
--------------------------------    -----------      -----------
     Ga v bDcdL -> F            >=  GaDdcL -> F      GbDdcL -> F    (E.19.2)
--------------------------------    ------------------------------
     Ga v bDdcL -> F                     Ga v bDdcL -> F

        f                   g              f                   g
GacD -> F           GbcD -> F       GacD -> F          GbcD -> F
--------------------------------    ---------          ---------
        Ga v bcD -> F           >=  GcaD -> F          GcbD -> F    (E.19.3)
--------------------------------    ------------------------------
        Gca v bD -> F                       Gca v bD -> F

        f                   g              f                   g
GcaD -> F           GcbD -> F       GcaD -> F          GcbD -> F
--------------------------------    ---------          ---------
        Gca v bD -> F           >=  GacD -> F·         GbcD -> F    (E.19.4)
--------------------------------    ------------------------------
        Ga v bcD -> F                       Ga v bcD -> F
```

(R4,R13)

```
          f                               f
     GcdD -> FabE                    GcdD -> FabE
     ----------------                ----------------
     GcdD -> Fa v bE         >=      GdcD -> FabE            (E.20)
     ----------------                ----------------
     GdcD -> Fa v bE                 GdcD -> Fa v bE
```

(R4,R14)

```
         f              g                  f
    GcdD -> a      LbS -> e           GcdD -> a
    ---------------------------       ---------          g
       LGcdDa => bS -> e        >=    GdcD -> a     LbS -> e    (E.21.1)
       -----------------              ----------------------
       LGdcDa => bS -> e                 LGdcDa => bS -> e

       h              k                                   k
    G -> a    DcdLbS -> e                        DcdLbS -> e
    ----------------------                    h  -----------
       DcdLGa => bS -> e        >=        G -> a  DdcLbS -> e   (E.21.2)
       -----------------              ----------------------
       DdcLGa => bS -> e                 DdcLGa => bS -> e

       h              j                                   j
    G -> a    DbLcdS -> e                        DbLcdS -> e
    ----------------------                    h  -----------
       DGa => bLcdS -> e        >=        G -> a  DbLdcS -> e   (E.21.3)
       -----------------              ----------------------
       DGa => bLdcS -> e                 DGa => bLdcS -> e
```

(R4,R15)

```
                  f                          f
           GcdDaL -> b                GcdDaL -> b
           ------------------         -----------
           GcdDL -> (a => b)   >=     GdcDaL -> b              (E.22.1)
           ------------------         ------------------
           GdcDL -> (a => b)          GdcDL -> (a => b)

                  f                          f
           GaDcdL -> b                GaDcdL -> b
           ------------------         -----------
           GDcdL -> (a => b)   >=     GaDdcL -> b              (E.22.2)
           ------------------         ------------------
           GDdcL -> (a => b)          GDdcL -> (a => b)
```

(R5,R5)

```
         f                              f
      G -> FEH                       G -> FEH
      ----------                     ----------
      G -> FaEH          >=          G -> FEbH                 (E.23)
      ----------                     ----------
      G -> FaEbH                     G -> FaEbH
```

(R5,R6)

```
      f                                    f
G -> FaE                              G -> FaE             (E.24.1)
----------
G -> FaaE          >=
----------
G -> FaE
```

```
      f                                    f
G -> FaaEH                            G -> FaaEH
------------                          -----------
G -> FaaEbH        >=                 G -> FaEH            (E.24.2)
------------                          -----------
G -> FaEbH                            G -> FaEbH
```

```
      f                                    f
G -> FEaaH                            G -> FEaaH
------------                          -----------
G -> FbEaaH        >=                 G -> FEaH            (E.24.3)
------------                          -----------
G -> FbEaH                            G -> FbEaH
```

(R5,R7)

```
      f                                    f
G -> FEabH                            G -> FEabH
------------                          ------------
G -> FcEabH        >=                 G -> FEbaH           (E.25.1)
------------                          ------------
G -> FcEbaH                           G -> FcEbaH
```

```
      f                                    f
G -> FabEH                            G -> FabEH
------------                          ------------
G -> FabEcH        >=                 G -> FbaEH           (E.25.2)
------------                          ------------
G -> FbaEcH                           G -> FbaEcH
```

```
      f
G -> FaE                                   f
----------                            G -> FaE
G -> FabE          >=                 ----------           (E.25.3)
----------                            G -> FbaE
G -> FbaE
```

```
      f
G -> Fae                                   f
----------                            G -> FaE
G -> FbaE          >=                 ----------           (E.25.4)
----------                            G -> FabE
G -> FabE
```

(R6,R7)

```
      f                                    f
G -> FccEabH                          G -> FccEabH
------------                          ------------
G -> FcEabH        >=                 G -> FccEbaH         (E.26.1)
------------                          ------------
G -> FcEbaH                           G -> FcEbaH
```

```
      f                                    f
G -> FabEccH                          G -> FabEccH
------------                          ------------
G -> FabEcH        >=                 G -> FbaEccH         (E.26.2)
------------                          ------------
G -> FbaEcH                           G -> FbaEcH
```

```
                                           f
      f                               G -> FaabE
G -> FaabE                            ----------
----------                            G -> FabaE
G -> FabE          >=                 ----------           (E.26.3)
----------                            G -> FbaaE
G -> FbaE                             ----------
                                      G -> FbaE
```

```
        f                                    f
  G -> FbaaE                            G -> FbaaE
  ----------                            ----------
  G -> FbaE              >=             G -> FabaE             (E.26.4)
  ----------                            ----------
  G -> FabE                             G -> FaabE
                                        ----------
                                        G -> FabE
```

(R6,R10)

```
   f             g                        f            g
 G -> FccEaH   G -> FccEbH              G -> FccEaH  G -> FccEbH
 -------------------------              -----------  -----------
      G -> FccEa ↑ bH          >=       G -> FcEaH   G -> FcEbH
 -------------------------              -------------------------
      G -> FcEa ↑ bH                         G -> FcEa ↑ bH
                                                                 (E.27.1)
```

```
   f             g                        f            g
 G -> FaEccH   G -> FbEccH              G -> FaEccH  G -> FbEccH
 -------------------------              -----------  -----------
      G -> Fa ↑ bEccH          >=       G -> FaEcH   G -> FbEcH
 -------------------------              -------------------------
      G -> Fa ↑ bEcH                         G -> Fa ↑ bEcH
                                                                 (E.27.2)
```

```
   f          g            h          k
 G -> FaE   G -> FbE     G -> FaE   G -> FbE
 --------   --------     --------   --------
 G -> FaaE  G -> FabE    G -> FbaE  G -> FbbE          f          k
 -------------------     -------------------         G -> FaE   G -> FbE
    G -> Faa ↑ bE           G -> Fba ↑ bE      >=    -------------------
    --------------------------------------              G -> Fa ↑ bE
            G -> Fa ↑ ba ↑ bE
            -----------------
              G -> Fa ↑ bE
                                                                 (E.27.3)
```

```
   f          g            h          k
 G -> FaE   G -> FbE     G -> FaE   G -> FbE
 --------   --------     --------   --------
 G -> FaaE  G -> FbaE    G -> FabE  G -> FbbE          f          k
 -------------------     -------------------         G -> FaE   G -> FbE
    G -> Fa ↑ baE           G -> Fa ↑ bbE      >=    -------------------
    --------------------------------------              G -> Fa ↑ bE
            G -> Fa ↑ ba ↑ bE
            -----------------
              G -> Fa ↑ bE
                                                                 (E.27.4)
```

```
   f          f            k          k
 G -> FaE   G -> FaE     G -> FbE   G -> FbE
 --------   --------     --------   --------
 G -> FaaE  G -> FabE    G -> FbaE  G -> FbbE          f          k
 -------------------     -------------------         G -> FaE   G -> FbE
    G -> Faa ↑ bE           G -> Fba ↑ bE      >=    -------------------
    --------------------------------------              G -> Fa ↑ bE
            G -> Fa ↑ ba ↑ bE
            -----------------
              G -> Fa ↑ bE
                                                                 (E.27.5)
```

```
   f          f            k          k
 G -> FaE   G -> FaE     G -> FbE   G -> FbE
 --------   --------     --------   --------
 G -> FaaE  G -> FbaE    G -> FabE  G -> FbbE          f          k
 -------------------     -------------------         G -> FaE   G -> FbE
    G -> Fa ↑ baE           G -> Fa ↑ bbE      >=    -------------------
    --------------------------------------              G -> Fa ↑ bE
            G -> Fa ↑ ba ↑ bE
            -----------------
              G -> Fa ↑ bE
                                                                 (E.27.6)
```

```
     f         g               h        k
G --> FaE  G --> FbE       G --> FaE  G --> FbE
--------------------       --------------------
   G -> Fa ↑ bE              G -> Fa ↑ bE                 f         g
   ------------              ------------            G --> FaE  G --> FbE
   G -> Faa ↑ bE             G -> Fba ↑ bE      >=   --------------------
   ---------------------------------------             G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                           (E.27.7)
```

```
     f         g               h        k
G --> FaE  G --> FbE       G --> FaE  G --> FbE
--------------------       --------------------
   G -> Fa ↑ bE              G -> Fa ↑ bE                 f         g
   ------------              ------------            G --> FaE  G --> FbE
   G -> Fa ↑ baE             G -> Fa ↑ bbE      >=   --------------------
   ---------------------------------------             G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                           (E.27.8)
```

```
     f         g               k          k
G --> FaE  G --> FbE       G --> FbE  G --> FbE
--------------------       ---------  ---------
   G -> Fa ↑ bE            G -> FbaE  G -> FbbE              f         g
   ------------            --------------------         G --> FaE  G --> FbE
   G -> Faa ↑ bE             G -> Fba ↑ bE         >=   --------------------
   ---------------------------------------                G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                              (E.27.9)
```

```
     f         g               k          k
G --> FaE  G --> FbE       G --> FbE  G --> FbE
--------------------       ---------  ---------
   G -> Fa ↑ bE            G -> FabE  G -> FbbE              f         g
   ------------            --------------------         G --> FaE  G --> FbE
   G -> Fa ↑ baE             G -> Fa ↑ bbE         >=   --------------------
   ---------------------------------------                G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                              (E.27.10)
```

```
     f          f              h        k
G --> FaE  G --> FaE       G --> FaE  G --> FbE
---------  ---------       --------------------
G -> FaaE  G -> FabE         G -> Fa ↑ bE                    h         k
--------------------         ------------               G --> FaE  G --> FbE
   G -> Faa ↑ bE             G -> Fba ↑ bE         >=   --------------------
   ---------------------------------------                G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                              (E.27.11)
```

```
     f          f              h        k
G --> FaE  G --> FaE       G --> FaE  G --> FbE
---------  ---------       --------------------
G -> FaaE  G -> FbaE         G -> Fa ↑ bE                    h         k
--------------------         ------------               G --> FaE  G --> FbE
   G -> Fa ↑ baE             G -> Fa ↑ bbE         >=   --------------------
   ---------------------------------------                G -> Fa ↑ bE
           G -> Fa ↑ ba ↑ bE
           -----------------
              G -> Fa ↑ bE
                                                              (E.27.12)
```

(R6,R11)

```
           f                                   f
     GabD --> FccE                       GabD --> FccE
    --------------                     ---------------
    Ga ↑ bD -> FccE          >=          GabD -> FcE          (E.28)
    ---------------                    ---------------
    Ga ↑ bD -> FcE                     Ga ↑ bD -> FcE
```

(R6,R12)

```
          f                  g                         f             g
GaD  ->  FccE     GbD  ->  FccE         GaD  ->  FccE     GbD  ->  FccE
-----------------------------------     --------------    --------------
       Ga  v  bD  ->  FccE          >=  GaD  ->  FcE      GbD  ->  FcE       (E.29)
-----------------------------------     --------------------------------
       Ga  v  bD  ->  FcE                       Ga  v  bD  ->  FcE
```

(R6,R13)

```
   f                                 f
G  ->  FabEccH                    G  ->  FabEccH
--------------------              --------------------
G  ->  Fa  v  bEccH        >=     G  ->  FabEcH                  (E.30.1)
--------------------              --------------------
G  ->  Fa  v  bEcH                G  ->  Fa  v  bEcH

   f                                 f
G  ->  FccEabH                    G  ->  FccEabH
--------------------              --------------------
G  ->  FccEa  v  bH        >=     G  ->  FcEabH                  (E.30.2)
--------------------              --------------------
G  ->  FcEa  v  bH                G  ->  FcEa  v  bH

                                     f
                                  G  ->  FababE
   f                              -----------------
G  ->  FababE                     G  ->  FaabbE
-----------------------           -----------------
G  ->  Fa  v  babE                G  ->  FaabE                   (E.30.3)
-----------------------    >=     -----------------
G  ->  Fa  v  ba  v  bE           G  ->  FabE
-----------------------           -----------------
G  ->  Fa  v  bE                  G  ->  Fa  v  bE

                                     f
                                  G  ->  FababE
   f                              -----------------
G  ->  FababE                     G  ->  FaabbE
-----------------------           -----------------
G  ->  Faba  v  bE         >=     G  ->  FaabE                   (E.30.4)
-----------------------           -----------------
G  ->  Fa  v  ba  v  bE           G  ->  FabE
-----------------------           -----------------
G  ->  Fa  v  bE                  G  ->  Fa  v  bE
```

(R6,R14)

```
   f             g                                          g
G  ->  a    DbL  ->  FccE                         DbL  ->  FccE
---------------------------              f        --------------
  DGa  =>  bL  ->  FccE        >=    G  ->  a     DbL  ->  FcE        (E.31)
---------------------------          ---------------------------
  DGa  =>  bL  ->  FcE                  DGa  =>  bL  ->  FcE
```

(R7,R10)

```
   f                  g                         f                g
G  ->  FaEcdH    G  ->  FbEcdH               G  ->  FaEcdH    G  ->  FbEcdH
----------------------------------           --------------   --------------
     G  ->  Fa  ↑  bEcdH               >=    G  ->  FaEdcH    G  ->  FbEdcH     (E.32.1)
----------------------------------           --------------------------------
     G  ->  Fa  ↑  bEdcH                          G  ->  Fa  ↑  bEdcH

   f                  g                         f                g
G  ->  FcdEaH    G  ->  FcdEbH               G  ->  FcdEaH    G  ->  FcdEbH
----------------------------------           --------------   --------------
     G  ->  FcdEa  ↑  bH               >=    G  ->  FdcEaH    G  ->  FdcEbH     (E.32.2)
----------------------------------           --------------------------------
     G  ->  FdcEa  ↑  bH                          G  ->  FdcEa  ↑  bH

   f                g                           f              g
G  ->  FacE     G  ->  FbcE                  G  ->  FacE    G  ->  FbcE
----------------------------                 -----------    -----------
     G  ->  Fa  ↑  bcE                 >=    G  ->  FcaE    G  ->  FcbE         (E.32.3)
----------------------------                 --------------------------
     G  ->  Fca  ↑  bE                            G  ->  Fca  ↑  bE
```

```
        f                   g                          f               g
  G  ->, FcaE         G  -> FcbE              G  -> FcaE         G  -> FcbE
  ----------------------------------          ----------         ----------
        G  -> Fca ↑ bE                >=      G  -> FacE         G  -> FbcE      (E.32.4)
  ----------------------------------          -------------------------------
        G  -> Fa ↑ bcE                                G  -> Fa ↑ bcE
```

(R7,R11)

```
              f                                      f
        GabD  -> FcdE                          GabD  -> FcdE
    ----------------------                 ----------------------
    Ga ↑ bD  -> FcdE             >=            GabD  -> FdcE              (E.33)
    ----------------------                 ----------------------
    Ga ↑ bD  -> FdcE                       Ga ↑ bD  -> FdcE
```

(R7,R12)

```
        f                g                        f                 g
  GaD  -> FcdE     GbD  -> FcdE             GaD  -> FcdE      GbD  -> FcdE
  -----------------------------             ------------      ------------
       Ga v bD  -> FcdE            >=       GaD  -> FdcE      GbD  -> FdcE     (E.34)
  -----------------------------             ------------------------------
       Ga v bD  -> FdcE                            Ga v bD  -> FdcE
```

(R7,R13)

```
       f                                    f
  G  -> FabEcdH                        G  -> FabEcdH
  ------------------                   ------------------
  G  -> Fa v bEcdH           >=        G  -> FabEdcH              (E.35.1)
  ------------------                   ------------------
  G  -> Fa v bEdcH                     G  -> Fa v bEdcH

       f                                    f
  G  -> FcdEabH                        G  -> FcdEabH
  ------------------                   ------------------
  G  -> FcdEa v bH           >=        G  -> FdcEabH              (E.35.2)
  ------------------                   ------------------
  G  -> FdcEa v bH                     G  -> FdcEa v bH

                                            f
       f                               G  -> FabcH
  G  -> FabcH                          ---------------
  ---------------                      G  -> FacbH
  G  -> Fa v bcH             >=        ---------------            (E.35.3)
  ---------------                      G  -> FcabH
  G  -> Fca v bH                       ---------------
                                       G  -> Fca v bH

                                            f
       f                               G  -> FcabH
  G  -> FcabH                          ---------------
  ---------------                      G  -> FacbH
  G  -> Fca v bH             >=        ---------------            (E.35.4)
  ---------------                      G  -> FabcH
  G  -> Fa v bcH                       ---------------
                                       G  -> Fa v bcH
```

(R7,R14)

```
      f              g                                         g
  G  -> a      DbL  -> FcdE                             DbL  -> FcdE
  -------------------------                  f          -------------
    DGa => bL  -> FcdE          >=      G  -> a         DbL  -> FdcE      (E.36)
  -------------------------             ---------------------------------
    DGa => bL  -> FdcE                     DGa => bL  -> FdcE
```

(R2,R3,R11)

```
                  f
       Ga ↑ bL  -> F
  -----------------------
      Ga ↑ bbL  -> F
  -----------------------                               f
     Ga ↑ babL  -> F             >=         Ga ↑ bL  -> F
  -----------------------
  Ga ↑ ba ↑ bL  -> F
  -----------------------                                         (E.37.1)
       Ga ↑ bL  -> F
```

```
              f
    Ga ↑ bL  -> F
-----------------
   Gba ↑ bL  -> F
-----------------                              f
  Gaba ↑ bL  -> F          >=       Ga ↑ bL -> F
-----------------
Ga ↑ ba ↑ bL -> F                                        (E.37.2)
-----------------
    Ga ↑ bL  -> F
```

```
               f
       GabbD  -> F
------------------                         f
    Ga ↑ bbD  -> F                   GabbD -> F
------------------                   -----------
   Ga ↑ babD  -> F         >=         GabD -> F      (E.37.3)
------------------                   -----------
Ga ↑ ba ↑ bD  -> F                   Ga ↑ bD -> F
------------------
     Ga ↑ bD  -> F
```

```
               f
       GbabD  -> F                         f
------------------                   GbabD -> F
    Gba ↑ bD  -> F                   -----------
------------------                   GabbD -> F
   Gaba ↑ bD  -> F         >=        -----------     (E.37.4)
------------------                    GabD -> F
Ga ↑ ba ↑ bD  -> F                   -----------
------------------                   Ga ↑ bD -> F
     Ga ↑ bD  -> F
```

```
               f
       GabaD  -> F                         f
------------------                   GabaD -> F
    Ga ↑ baD  -> F                   -----------
------------------                   GaabD -> F
   Ga ↑ babD  -> F         >=        -----------     (E.37.5)
------------------                    GabD -> F
Ga ↑ ba ↑ bD  -> F                   -----------
------------------                   Ga ↑ bD -> F
     Ga ↑ bD  -> F
```

```
               f
       GaabD  -> F
------------------                         f
    Gaa ↑ bD  -> F                   GaabD -> F
------------------                   -----------
   Gaba ↑ bD  -> F         >=         GabD -> F      (E.37.6)
------------------                   -----------
Ga ↑ ba ↑ bD  -> F                   Ga ↑ bD -> F
------------------
     Ga ↑ bD  -> F
```

(R5,R6,R13)

```
  f
G -> Fa v bE
-----------------
G -> Fa v bbE
-----------------                      f
G -> Fa v babE             >=       G -> Fa v bE
-----------------
G -> Fa v ba v bE                                    (E.38.1)
-----------------
G -> Fa v bE
```

```
  f
G -> Fa v bE
-----------------
G -> Fba v bE
-----------------                      f
G -> Faba v bE             >=       G -> Fa v bE
-----------------
G -> Fa v ba v bE                                    (E.38.2)
-----------------
G -> Fa v bE
```

```
      f
 G -> FabbE
 ---------------
 G -> Fa v bbE                        G -> FabbE
 ---------------                      ------------
 G -> Fa v babE               >=      G -> FabE        (E.38.3)
 ---------------                      ------------
 G -> Fa v ba v bE                    G -> Fa v bE
 ---------------
 G -> Fa v bE

      f
 G -> FbabE
 ---------------                      G -> FbabE
 G -> Fba v bE                        ------------
 ---------------                      G -> FabbE
 G -> Faba v bE               >=      ------------     (E.38.4)
 ---------------                      G -> FabE
 G -> Fa v ba v bE                    ------------
 ---------------                      G -> Fa v bE
 G -> Fa v bE

      f
 G -> FabaE
 ---------------                      G -> FabaE
 G -> Fa v baE                        ------------
 ---------------                      G -> FaabE
 G -> Fa v babE               >=      ------------     (E.38.5)
 ---------------                      G -> FabE
 G -> Fa v ba v bE                    ------------
 ---------------                      G -> Fa v bE
 G -> Fa v bE

      f
 G -> FaabE
 ---------------
 G -> Faa v bE                        G -> FaabE
 ---------------                      ------------
 G -> Faba v bE               >=      G -> FabE        (E.38.6)
 ---------------                      ------------
 G -> Fa v ba v bE                    G -> Fa v bE
 ---------------
 G -> Fa v bE

(R3,R12,R14)
     f           g                 h              k
 aG -> c     bG -> c         DadL -> F      DbdL -> F
 -------------------         ------------------------
   a v bG -> c                     Da v bdL -> F
   ------------------------------------------------
          Da v b a v bGc => dL -> F
          -------------------------
              Da v bGc => dL -> F

             f              h               g              k
          aG -> c     DadL -> F          bG -> c     DbdL -> F
          ---------------------          ---------------------
           DaaGc => dL -> F                DbbGc => dL -> F
           ----------------                ----------------
             DaGc => dL -> F                 DbGc => dL -> F
 >=          ---------------------------------------------      (E.39)
                          Da v bGc => dL -> F

(R4,R12,R14)
    f        g                              f          k          g          k
 aG -> c  bG -> c                        aG -> c  DedL -> F    bG -> c  DedL -> F
 ----------------               k        ------------------    ------------------
   a v bG -> c           DedL -> F       DeaGc => dL -> F      DebGc => dL -> F
   ------------------------------  >=    ----------------      ----------------
      Dea v bGc => dL -> F                DaeGc => dL -> F     DbeGc => dL -> F
      --------------------                ---------------------------------------
      Da v beGc => dL -> F                          Da v beGc => dL -> F

                                                                            (E.40.1)
```

```
                   g          h             f        g          f        h
             DadL -> F  DbdL -> F      eG -> c  DadL -> F  eG -> c  DbdL -> F
             --------------------      ------------------  ------------------
      f
 eG -> c         Da v bdL -> F         DaeGc => dL -> F    DbeGc => dL -> F
 ------------------------------  >=    ----------------    ----------------
    Da v beGc => dL -> F               DeaGc => dL -> F    DebGc => dL -> F
    --------------------               ------------------------------------
    Dea v bGc => dL -> F                      Dea v bGc => dL -> F
                                                                 (E.40.2)

                   g          h            f        g         f         h
             DdaL -> F  DdbL -> F      G -> c  DdaL -> F  G -> c  DdbL -> F
             --------------------      -----------------  -----------------
      f
 G -> c          Dda v bL -> F         DGc => daL -> F     DGc => dbL -> F
 ------------------------------  >=    ---------------     ---------------
    DGc => da v bL -> F                DGac => dL -> F     DGbc => dL -> F
    -------------------                -----------------------------------
    DGa v bc => dL -> F                       DGa v bc => dL -> F
                                                                 (E.40.3)

     f         g                        f        k         g        k
 Ga -> c   Gb -> c                  Ga -> c  DdL -> F  Gb -> c  DdL -> F
 -----------------                  -----------------  -----------------
                          k
    Ga v b -> c       DdL -> F      DGac => dL -> F    DGbc => dL -> F
    --------------------------  >=  ---------------    ---------------
       DGa v bc => dL -> F          DGc => daL -> F    DGc => dbL -> F
       -------------------          ----------------------------------
       DGc => da v bL -> F                 DGc => da v bL -> F
                                                                 (E.40.4)
```

(R2)

```
                                          f
                                      GD -> F
           f                          -------
       GD -> F                        GbD -> F
   -------------        >=            --------
   Ga ↑ bD -> F                       GabD -> F                  (E.41.1)
                                      ---------
                                      Ga ↑ bD -> F

                                  f            f
           f                  GD -> F      GD -> F
       GD -> F                -------      -------
   -------------        >=    GaD -> F     GbD -> F              (E.41.2)
   Ga v bD -> F               ---------------------
                                  Ga v bD -> F
```

(R5)

```
                                 f             f
     f                       G -> FE       G -> FE
 G -> FE                     -------       -------
 ------------           >=   G -> FaE      G -> FbE              (E.42.1)
 G -> Fa ↑ bE                ----------------------
                                  G -> Fa ↑ bE

                                 f
                             G -> FE
     f                       --------
 G -> FE                     G -> FbE
 ------------           >=   --------                            (E.42.2)
 G -> Fa v bE                G -> FabE
                             ---------
                             G -> Fa v bE
```

Appendix 4

Normalization Algorithm

In this appendix, we collect from appendix (D) in SZABO all those clauses which determine the choice and order of application of Rules (R2)-(R15) in case of the categories considered above. This completes the description of the normal form in chapters 1-5.

(R2,R2)

$$\frac{\dfrac{GDL \xrightarrow{f} F}{GaDL \to F}}{GaDbL \to F} \quad >= \quad \frac{\dfrac{GDL \xrightarrow{f} F}{GDbL \to F}}{GaDbL \to F} \qquad (D.1)$$

(R2,R3)

$$\frac{\dfrac{GaaDL \xrightarrow{f} F}{GaaDbL \to F}}{GaDbL \to F} \quad >= \quad \frac{\dfrac{GaaDL \xrightarrow{f} F}{GaDL \to F}}{GaDbL \to F} \qquad (D.2.1)$$

$$\frac{\dfrac{GDaaL \xrightarrow{f} F}{GbDaaL \to F}}{GbDaL \to F} \quad >= \quad \frac{\dfrac{GDaaL \xrightarrow{f} F}{GDaL \to F}}{GbDaL \to F} \qquad (D.2.2)$$

$$\frac{\dfrac{GaD \xrightarrow{f} F}{GabD \to F}}{GaD \to F} \quad >= \quad GaD \xrightarrow{f} F, \text{ with } a = b. \qquad (D.2.3)$$

$$\frac{\dfrac{GaD \xrightarrow{f} F}{GbaD \to F}}{GaD \to F} \quad >= \quad GaD \xrightarrow{f} F, \text{ with } a = b. \qquad (D.2.4)$$

(R2,R4)

$$\frac{\dfrac{GabDL \xrightarrow{f} F}{GbaDL \to F}}{GbaDcL \to F} \quad >= \quad \frac{\dfrac{GabDL \xrightarrow{f} F}{GabDcL \to F}}{GbaDcL \to F} \qquad (D.3.1)$$

$$\frac{\dfrac{GDabL \xrightarrow{f} F}{GDbaL \to F}}{GcDbaL \to F} \quad >= \quad \frac{\dfrac{GDabL \xrightarrow{f} F}{GcDabL \to F}}{GcDbaL \to F} \qquad (D.3.2)$$

$$\frac{\dfrac{GaD \xrightarrow{f} F}{GabD \to F}}{GbaD \to F} \quad >= \quad \frac{GaD \xrightarrow{f} F}{GbaD \to F} \qquad (D.3.3)$$

```
                        f
                   GaD -> F
                   ---------
                   GbaD -> F              f
                   ---------        GaD -> F
                   GabD -> F   >=   ---------              (D.3.4)
                                    GabD -> F

(R2,R5)
                      f                  f
                   GD -> FE           GD -> FE
                   ---------          ---------
                   GaD -> FE          GD -> FbE
                   ----------  >=     ----------           (D.4)
                   GaD -> FbE         GaD -> FbE

(R2,R10)
        f                g                  f             g
     GD -> FaE      GD -> FbE          GD -> FaE    GD -> FbE
     ---------      ---------          ------------------------
     GcD -> FaE     GcD -> FbE             GD -> Fa ↑ bE
     -------------------------   >=        -------------          (D.7)
          GcD -> Fa ↑ bE                   GcD -> Fa ↑ bE

(R2,R11)
                        f                    f
                 GDabL -> F            GDabL -> F
                 ----------            ----------
                 GcDabL -> F           GDa ↑ bL -> F
                 --------------  >=    --------------
                 GcDa ↑ bL -> F        GcDa ↑ bL -> F              (D.8.1)

                        g                    g
                 GabDL -> F            GabDL -> F
                 ----------            ----------
                 GabDcL -> F           Ga ↑ bDL -> F
                 --------------  >=    --------------              (D.8.2)
                 Ga ↑ bDcL -> F        Ga ↑ bDcL -> F
(R2,R12)
        f                 g                  f             g
     GDaL -> F       GDbL -> F          GDaL -> F    GDbL -> F
     ---------       ---------          -----------------------
     GcDaL -> F      GcDbL -> F             GDa v bL -> F
     --------------------------   >=        -------------         (D.9.1)
          GcDa v bL -> F                    GcDa v bL -> F

        f                 g                  f             g
     GaDL -> F       GbDL -> F          GaDL -> F    GbDL -> F
     ---------       ---------          -----------------------
     GaDcL -> F      GbDcL -> F             Ga v bDL -> F
     --------------------------   >=        -------------         (D.9.2)
          Ga v bDcL -> F                    Ga v bDcL -> F

(R2,R13)
                      f                    f
                 GD -> FabE            GD -> FabE
                 ----------            ----------
                 GcD -> FabE           GD -> Fa v bE
                 -------------   >=    -------------             (D.10)
                 GcD -> Fa v bE        GcD -> Fa v bE
(R2,R14)
              f                           f          g
           GD -> a                     GD -> a    LbZ -> F
           -------           g         -------------------
           GcD -> a       LbZ -> F        LGDa => bZ -> F
           ------------------------  >=   ---------------        (D.11.1)
             LGcDa => bZ -> F             LGcDa => bZ -> F

                             k            h           k
                        DLbZ -> F      G -> a    DLbZ -> F
             h          ---------      ------------------
           G -> a       DcLbZ -> F        DLGa => bZ -> F
           ------------------------  >=   ---------------        (D.11.2)
             DcLGa => bZ -> F             DcLGa => bZ -> F
```

$$\frac{G \xrightarrow{h} a \qquad \dfrac{DbLZ \xrightarrow{m} F}{DbLcZ \to F}}{DGa \Rightarrow bLcZ \to F} \;\geq\; \frac{\dfrac{G \xrightarrow{h} a \qquad DbLZ \xrightarrow{m} F}{DGa \Rightarrow bLZ \to F}}{DGa \Rightarrow bLcZ \to F} \qquad (D.11.3)$$

$$\frac{G \xrightarrow{h} a \qquad \dfrac{DL \xrightarrow{p} F}{DbL \to F}}{DGa \Rightarrow bL \to F} \;\geq\; \frac{\dfrac{DL \xrightarrow{p} F}{Da \Rightarrow bL \to F}}{DGa \Rightarrow bL \to F} \qquad (D.11.4)$$

(R2,R15)

$$\frac{\dfrac{GaDL \xrightarrow{f} b}{GaDcL \to b}}{GDcL \to (a \Rightarrow b)} \;\geq\; \frac{\dfrac{GaDL \xrightarrow{f} b}{GDL \to (a \Rightarrow b)}}{GDcL \to (a \Rightarrow b)} \qquad (D.12.1)$$

$$\frac{\dfrac{GDaL \xrightarrow{f} b}{GcDaL \to b}}{GcDL \to (a \Rightarrow b)} \;\geq\; \frac{\dfrac{GDaL \xrightarrow{f} b}{GDL \to (a \Rightarrow b)}}{GcDL \to (a \Rightarrow b)} \qquad (D.12.2)$$

(R3,R3)

$$\frac{\dfrac{GaaDbbL \xrightarrow{f} F}{GaDbbL \to F}}{GaDbL \to F} \;\geq\; \frac{\dfrac{GaaDbbL \xrightarrow{f} F}{GaaDbL \to F}}{GaDbL \to F} \qquad (D.15.1)$$

(R3,R4)

$$\frac{\dfrac{GaaDbcL \xrightarrow{f} F}{GaaDcbL \to F}}{GaDcbL \to F} \;\geq\; \frac{\dfrac{GaaDbcL \xrightarrow{f} F}{GaDbcL \to F}}{GaDcbL \to F} \qquad (D.16.1)$$

$$\frac{\dfrac{GbcDaaL \xrightarrow{g} F}{GcbDaaL \to F}}{GcbDaL \to F} \;\geq\; \frac{\dfrac{GbcDaaL \xrightarrow{g} F}{GbcDaL \to F}}{GcbDaL \to F} \qquad (D.16.2)$$

$$\frac{\dfrac{GabD \xrightarrow{h} F}{GbaD \to F}}{GbD \to F} \;\geq\; \frac{GabD \xrightarrow{h} F}{GbD \to F} \qquad (D.16.3)$$

(R3,R5)

$$\frac{\dfrac{GaaD \xrightarrow{f} FE}{GaD \to FE}}{GaD \to FbE} \;\geq\; \frac{\dfrac{GaaD \xrightarrow{f} FE}{GaaD \to FbE}}{GaD \to FbE} \qquad (D.17)$$

(R3,R10)

$$\frac{\dfrac{GccD \xrightarrow{f} FaE}{GcD \to FaE} \qquad \dfrac{GccD \xrightarrow{g} FbE}{GcD \to FbE}}{GcD \to Fa \uparrow bE} \;\geq\; \frac{\dfrac{GccD \xrightarrow{f} FaE \qquad GccD \xrightarrow{g} FbE}{GccD \to Fa \uparrow bE}}{GcD \to Fa \uparrow bE} \qquad (D.18)$$

(R3,R11)

```
                        f                         f
          GaaDbcL  -> F            GaaDbcL -> F
          ------------             ------------
          GaaDb ↑ cL -> F          GaDbcL -> F
          ---------------->=       --------------
          GaDb ↑ cL -> F           GaDb ↑ cL -> F          (D.19.1)


                        g                         g
          GbcDaaL  -> F            GbcDaaL -> F
          ------------             ------------
          Gb ↑ cDaaL -> F          GbcDaL -> F
          ---------------->=       --------------          (D.19.2)
          Gb ↑ cDaL -> F           Gb ↑ cDaL -> F

                            h                        h
                  GaaaaD   -> F    Gaa.aa.D  -> F
               --------------      -------------
          Gaa(a ↑ a)D -> F         G.aa..a.D -> F
     ----------------------        --------------
     G(a ↑ a)(a ↑ a)D -> F              GaaD -> F
     ----------------------  >=       ------------          (D.19.3)
                 Ga ↑ aD -> F         Ga ↑ aD -> F
```

(R3,R12)

```
             f               g                     f              g
     GccDaL -> F     GccDbL -> F        GccDaL -> F     GccDbL -> F
     -----------     -----------        ----------------------------
     GcDaL -> F      GcDbL -> F              GccDa v bL -> F
     ---------------------------  >=         ---------------        (D.20.1)
            GcDa v bL -> F                   GcDa v bL -> F

             f               g                     f              g
     GaDccL -> F     GbDccL -> F        GaDccL -> F     GbDccL -> F
     -----------     -----------        ----------------------------
     GaDcL -> F      GbDcL -> F              Ga v bDccL -> F
     ---------------------------  >=         ---------------        (D.20.2)
            Ga v bDcL -> F                   Ga v bDcL -> F
```

(R3,R13)

```
                  f                              f
          GccD -> FabE                 GccD -> FabE
          ------------                 ------------
          GcD -> FabE                  GccD -> Fa v bE
          -------------  >=            ---------------        (D.21)
          GcD -> Fa v bE               GcD -> Fa v bE
```

(R3,R14)

```
              f                               f            g
       GccD -> a                       GccD -> a     LbZ -> F
       ---------                       ----------------------
                         g                LGccDa => bZ -> F
       GcD -> a     LbZ -> F  >=          -----------------        (D.22.1)
       ------------------------           LGcDa => bZ -> F
          LGcDa => bZ -> F

                             k          h                 k
                  DccLbZ -> F          G -> a     DccLbZ -> F
       h          -----------          ----------------------
       G -> a     DcLbZ -> F              DccLGa => bZ -> F
       ------------------------  >=       -----------------        (D.22.2)
          DcLGa => bZ -> F                DcLGa => bZ -> F

                             m          h                m
                  DbLccZ -> F          G -> a    DbLccZ -> F
       h          -----------          ---------------------
       G -> a     DbLcZ -> F              DGa => bLccZ -> F
       ------------------------  >=       -----------------        (D.22.3)
          DGa => bLcZ -> F                DGa => bLcZ -> F
```

(R3,R15)

```
                    f                         f
          GccDaL -> b               GccDaL -> b
          -----------               ------------
          GcDaL -> b                GccDL -> (a => b)
          ------------------>=      -----------------        (D.23.1)
          GcDL -> (a => b)          GcDL -> (a => b)
```

```
                    f                    f
          GaDccL   -> b        GaDccL   -> b
          ------------         ----------------
          GaDcL -> b           GDccL -> (a => b)
          ----------------->=  ----------------          (D.23.2)
          GDcL -> (a => b)     GDcL -> (a => b)

(R4,R5)
                 f                    f
          GabD  -> FE          GabD  -> FE
          -----------          -----------
          GbaD -> FE           GabD -> FcE
          -----------    >=    -----------               (D.25)
          GbaD -> FcE          GbaD -> FcE

(R4,R10)
            f               g                  f               g
   GcdD    -> FaE    GcdD  -> FbE       GcdD  -> FaE    GcdD  -> FbE
   ----------        ----------         -----------------------------
   GdcD -> FaE       GdcD -> FbE             GcdD -> Fa ↑ bE
   ----------------------------- >=          ---------------     (D.28)
          GdcD -> Fa ↑ bE                    GdcD -> Fa ↑ bE

(R4,R11)
                   f                       f
          GabDcdL -> F             GabDcdL -> F
          ----------------         -------------
          GabDc ↑ dL -> F          GbaDcdL -> F
          ---------------->=       ---------------
          GbaDc ↑ dL -> F          GbaDc ↑ dL -> F               (D.29.1)

                   f                       f
          GabDcdL -> F             GabDcdL -> F
          ----------------         -------------
          Ga ↑ bDcdL -> F          GabDdcL -> F
          ---------------->=       ---------------               (D.29.2)
          Ga ↑ bDdcL -> F          Ga ↑ bDdcL -> F

                                          g
                                   GabcD -> F
                           g       ----------
                   GabcD  -> F     GacbD -> F
            ----------------       ----------
            G(a ↑ b)cD -> F        GcabD -> F
            ----------------  >=   ---------------               (D.29.3)
            Gc(a ↑ b)D -> F        Gc(a ↑ b)D -> F

                                          g
                                   GabcD -> F
                           g       ----------
                   GabcD  -> F     GbacD -> F
            ----------------       ----------
            Ga(b ↑ c)D -> F        GbcaD -> F
            ----------------  >=   ---------------               (D.29.4)
            G(b ↑ c)aD -> F        G(b ↑ c)aD -> F

(R4,R12)
             f               g                 f                g
   GcdDaL   -> F     GcdDbL -> F       GcdDaL -> F     GcdDbL  -> F
   ------------      ------------      ------------------------------
   GdcDaL -> F       GdcDbL -> F           GcdDa v bL -> F
   ----------------------------- >=        ---------------       (D.30.1)
          GdcDa v bL -> F                  GdcDa v bL -> F

             h               k                 h                k
   GaDcdL   -> F     GbDcdL -> F       GaDcdL -> F     GbDcdL  -> F
   ------------      ------------      ------------------------------
   GaDdcL -> F       GbDdcL -> F           Ga v bDcdL -> F
   ----------------------------- >=        ---------------       (D.30.2)
          Ga v bDdcL -> F                  Ga v bDdcL -> F
```

(R4,R13)

```
               f                            f
       GcdD  -> FabE               GcdD  -> FabE
       ----------------            ----------------
       GdcD  -> FabE               GcdD  -> Fa v bE
       ------------------- >=      ----------------
       GdcD  -> Fa v bE            GdcD  -> Fa v bE          (D.31)
```

(R4,R14)

```
          f                             f             g
   GcdD  -> a                    GcdD  -> a     LbZ  -> F
   ----------                    -------------------------
                   g
   GdcD  -> a   LbZ -> F           LGcdDa => bZ -> F
   ---------------------- >=       ------------------
     LGdcDa => bZ -> F             LGdcDa => bZ -> F         (D.32.1)

                   k              h                 k
            DcdLbZ -> F          G -> a     DcdLbZ -> F
            -----------          -----------------------
      h
   G -> a   DdcLbZ -> F            DcdLGa => bZ -> F
   ---------------------- >=       ------------------
     DdcLGa => bZ -> F             DdcLGa => bZ -> F         (D.32.2)

                   m              h                m
            DbLcdZ -> F          G -> a    DbLcdZ -> F
            -----------          ----------------------
      h
   G -> a   DbLdcZ -> F            DGa => bLcdZ -> F
   ---------------------- >=       ------------------
     DGa => bLdcZ -> F             DGa => bLdcZ -> F         (D.32.3)

                                  h              p
                                 G -> a    DcbL -> F
                                 -------------------
                 p
            DcbL -> F              DcGa => bL -> F
            ---------              ----------------
      h
   G -> a   DbcL -> F            DGc(a => b)L -> F
   -------------------- >=       -----------------
    DG(a => b)cL -> F            DG(a => b)cL -> F           (D.32.4)

                                  h              q
                                 G -> a    DbcL -> F
                                 -------------------
                 q
            DbcL -> F            DG(a => b)cL -> F
            ---------            -----------------
      h
   G -> a   DcbL -> F            DGc(a => b)L -> F
   -------------------- >=       -----------------
    DcG(a => b)L -> F            DcG(a => b)L -> F           (D.32.5)
```

(R4,R15)

```
              f                         f
     GaDcdL -> b               GaDcdL -> b
     -----------               --------------------
     GaDdcL -> b               GDcdL -> (a => b)
     -------------------->=    --------------------
     GDdcL -> (a => b)         GDdcL -> (a => b)             (D.33.1)

              g                         g
     GcdDaL -> b               GcdDaL -> b
     -----------               --------------------
     GdcDaL -> b               GcdDL -> (a => b)
     -------------------->=    --------------------
     GdcDL -> (a => b)         GdcDL -> (a => b)             (D.33.2)

              h
         GcaD -> b
         ---------                      h
         GacD -> b               GcaD -> b
         --------------->=       ------------
         GcD -> a => b           GcD -> a => b               (D.33.3)

              k
         GacD -> b
         ---------                      k
         GcaD -> b               GacD -> b
         --------------->=       ------------
         GcD -> a => b           GcD -> a => b               (D.33.4)
```

(R5,R5)

$$\cfrac{\cfrac{G \xrightarrow{f} FEH}{G \to FaEH}}{G \to FaEbH} \;\geq\; \cfrac{\cfrac{G \xrightarrow{f} FEH}{G \to FEbH}}{G \to FaEbH} \qquad (D.34)$$

(R5,R10)

$$\cfrac{\cfrac{G \xrightarrow{f} FEaH \qquad G \xrightarrow{g} FEbH}{G \to FEa \uparrow bH}}{G \to FcEa \uparrow bH} \;\geq\; \cfrac{\cfrac{G \xrightarrow{f} FEaH}{G \to FcEaH} \qquad \cfrac{G \xrightarrow{g} FEbH}{G \to FcEbH}}{G \to FcEa \uparrow bH} \qquad (D.35.1)$$

$$\cfrac{\cfrac{G \xrightarrow{f} FaEH \qquad G \xrightarrow{g} FbEH}{G \to Fa \uparrow bEH}}{G \to Fa \uparrow bEcH} \;\geq\; \cfrac{\cfrac{G \xrightarrow{f} FaEH}{G \to FaEcH} \qquad \cfrac{G \xrightarrow{g} FbEH}{G \to FbEcH}}{G \to Fa \uparrow bEcH} \qquad (D.35.2)$$

$$\cfrac{\cfrac{G \xrightarrow{f} Fa}{G \to Fab} \qquad \cfrac{G \xrightarrow{g} Fc}{G \to Fac}}{G \to Fa(b \uparrow c)} \;\geq\; \cfrac{\cfrac{G \xrightarrow{f} Fa}{G \to Fab} \qquad \cfrac{G \xrightarrow{f} Fa}{G \to Fac}}{G \to Fa(b \uparrow c)} \qquad (D.35.3)$$

$$\cfrac{\cfrac{G \xrightarrow{f} dE}{G \to bdE} \qquad \cfrac{G \xrightarrow{g} cE}{G \to cdE}}{G \to (b \uparrow c)dE} \;\geq\; \cfrac{\cfrac{G \xrightarrow{f} dE}{G \to bdE} \qquad \cfrac{G \xrightarrow{f} dE}{G \to cdE}}{G \to (b \uparrow c)dE} \qquad (D.35.3)$$

$$\cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{h} FdE}{G \to FadE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \;\geq\; \cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \qquad (D.35.5)$$

$$\cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{h} FcE}{G \to FcdE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \;\geq\; \cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \qquad (D.35.6)$$

$$\cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{k} FdE}{G \to FcdE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \;\geq\; \cfrac{\cfrac{\cfrac{G \xrightarrow{k} FdE}{G \to FbdE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{k} FdE}{G \to FcdE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \qquad (D.35.7)$$

$$\cfrac{\cfrac{\cfrac{G \xrightarrow{m} FbE}{G \to FbdE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \;\geq\; \cfrac{\cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FabdE} \qquad \cfrac{\cfrac{G \xrightarrow{f} FaE}{G \to FadE}}{G \to FacdE}}{G \to Fa(b \uparrow c)dE} \qquad (D.35.8)$$

```
   m               k                      k            k
G -> FbE        G -> FdE              G -> FdE     G -> FdE
---------       ---------             ---------    ---------
G -> FbdE       G -> FcdE             G -> FbdE    G -> FcdE
----------      ----------            ----------   ----------
G -> FabdE      G -> FacdE            G -> FabdE   G -> FacdE
--------------------------    >=      -------------------------
   G -> Fa(b ↑ c)dE                     G -> Fa(b ↑ c)dE          (D.35.9)

   k               f                      k            k
G -> FdE        G -> FaE              G -> FdE     G -> FdE
---------       ---------             ---------    ---------
G -> FbdE       G -> FadE             G -> FbdE    G -> FcdE
----------      ----------            ----------   ----------
G -> FabdE      G -> FacdE            G -> FabdE   G -> FacdE
--------------------------    >=      -------------------------
   G -> Fa(b ↑ c)dE                     G -> Fa(b ↑ c)dE          (D.35.10)

   k               h                      k            k
G -> FdE        G -> FcE              G -> FdE     G -> FdE
---------       ---------             ---------    ---------
G -> FbdE       G -> FcdE             G -> FbdE    G -> FcdE
----------      ----------            ----------   ----------
G -> FabdE      G -> FacdE            G -> FabdE   G -> FacdE
--------------------------    >=      -------------------------
   G -> Fa(b ↑ c)dE                     G -> Fa(b ↑ c)dE          (D.35.11)

   k               n                      k            k
G -> FdE        G -> FdE              G -> FdE     G -> FdE
---------       ---------             ---------    ---------
G -> FbdE       G -> FcdE             G -> FbdE    G -> FcdE
----------      ----------            ----------   ----------
G -> FabdE      G -> FacdE            G -> FabdE   G -> FacdE
--------------------------    >=      -------------------------
   G -> Fa(b ↑ c)dE                     G -> Fa(b ↑ c)dE          (D.35.12)
```

(R5,R11)

```
           f                    f
      GabD -> FE           GabD -> FE
      -----------          -----------
      Ga ↑ bD -> FE        GabD -> FcE
      -------------   >=   -------------
      Ga ↑ bD -> FcE       Ga ↑ bD -> FcE                 (D.36)
```

(R5,R12)

```
      f          g                 f            g
  GaD -> FE   GbD -> FE         GaD -> FE    GbD -> FE
  ---------------------         ---------    ---------
      Ga v bD -> FE             GaD -> FcE   GbD -> FcE
      --------------     >=     -----------------------
      Ga v bD -> FcE                Ga v bD -> FcE            (D.37)
```

(R5,R13)

```
         f                        f
      G -> FEabH               G -> FEabH
      ------------             ------------
      G -> FEa v bH            G -> FcEabH
      --------------    >=     --------------
      G -> FcEa v bH           G -> FcEa v bH               (D.38.1)

         g                        g
      G -> FabEH               G -> FabEH
      ------------             ------------
      G -> Fa v bEH            G -> FabEcH
      --------------    >=     --------------
      G -> Fa v bEcH           G -> Fa v bEcH               (D.38.2)
```

(R5,R14)

```
     f           g                              f
  G -> a     DbL -> FE                      DbL -> FE
  --------------------                      ----------
   DGa => bL -> FE                  f
   ----------------      >=      G -> a     DbL -> FcE
   DGa => bL -> FcE              ---------------------
                                   DGa => bL -> FcE         (D.39)
```

```
(R10,R10)
      f              g                   h              k
  G -> FaEaH     G -> FaEaH         G -> FaEaH     G -> FaEaH
  -------------------------         -------------------------
       G -> Fa ↑ aEaH                    G -> Fa ↑ aEaH
       ----------------------------------------------------
                     G -> Fa ↑ aEa ↑ aH

            f              g              h              k
       G -> FaEaH     G -> FaEaH     G -> FaEaH     G -> FaEaH
       -------------------------     -------------------------
            G -> FaEa ↑ aH                G -> FaEa ↑ aH
   >=       ----------------------------------------------
                     G -> Fa ↑ aEa ↑ aH                          (D.48)

(R10,R11)
       f                   g                        f                g
 GabD -> FcE        GabD -> FdE             GabD -> FcE      GabD -> FdE
 ------------       ------------            -----------------------------
 Ga ↑ bD -> FcE     Ga ↑ bD -> FdE                GabD -> Fc ↑ dE
 ----------------------------------  >=           ------------------
       Ga ↑ bD -> Fc ↑ dE                         Ga ↑ bD -> Fc ↑ dE   (D.49)

(R10,R12)
      f             g                    h             k
  GaD -> FcE    GbD -> FcE          GaD -> FdE    GbD -> FdE
  ------------------------          ------------------------
       Ga v bD -> FcE                    Ga v bD -> FdE
       ----------------------------------------------------
                     Ga v bD -> Fc ↑ dE

            f             h              g             k
       GaD -> FcE    GaD -> FdE     GbD -> FcE    GbD -> FdE
       ------------------------     ------------------------
            GaD -> Fc ↑ dE               GbD -> Fc ↑ dE
   >=       ----------------------------------------------
                     Ga v bD -> Fc ↑ dE                          (D.50)

(R10,R13)
      f                 g                      f                  g
  G -> FcdEaH      G -> FcdEbH            G -> FcdEaH        G -> FcdEbH
  ----------------------------            -----------        -----------
       G -> FcdEa ↑ bH                    G -> Fc v dEaH     G -> Fc v dEbH
       ---------------           >=       ---------------------------------
       G -> Fc v dEa ↑ bH                          G -> Fc v dEa ↑ bH
                                                                   (D.51.1)

      h                 k                      h                  k
  G -> FaEcdH      G -> FbEcdH            G -> FaEcdH        G -> FbEcdH
  ----------------------------            -----------        -----------
       G -> Fa ↑ bEcdH                    G -> FaEc v dH     G -> FbEc v dH
       ---------------           >=       ---------------------------------
       G -> Fa ↑ bEc v dH                          G -> Fa ↑ bEc v dH
                                                                   (D.51.2)

(R10,R14)
    f           g                f           h                          g               h
 G -> a    DbL -> FcE        G -> a    DbL -> FdE                 DbL -> FcE    DbL -> FdE
 ---------------------       ---------------------         f      -------------------------
  DGa => bL -> FcE            DGa => bL -> FdE           G -> a       DbL -> Fc ↑ dE
  ---------------------------------------------- >=      --------------------------------
            DGa => bL -> Fc ↑ dE                             DGa => bL -> Fc ↑ dE
                                                                         (D.52)

(R11,R11)
                          f                         f
                GabDcdL -> F              GabDcdL -> F
                ------------              ------------
                Ga ↑ bDcdL -> F           GabDc ↑ dL -> F
                ------------------   >=   ------------------    (D.53)
                Ga ↑ bDc ↑ dL -> F        Ga ↑ bDc ↑ dL -> F
```

(R11,R12)

```
              f                  g                          f               g
GcdDaL -> F        GcdDbL -> F        GcdDaL -> F    GcdDbL -> F
-------------      -------------      --------------------------
Gc ↑ dDaL -> F     Gc ↑ dDbL -> F         GcdDa v bL -> F
--------------------------------->=       ------------------
     Gc ↑ dDa v bL -> F                   Gc ↑ dDa v bL -> F
                                                          (D.54.1)

        h                   k                       h               k
GaDcdL -> F        GbDcdL -> F        GaDcdL -> F    GbDcdL -> F
-------------      -------------      --------------------------
GaDc ↑ dL -> F     GbDc ↑ dL -> F         Ga v bDcdL -> F
--------------------------------->=       ------------------
     Ga v bDc ↑ dL -> F                   Ga v bDc ↑ dL -> F
                                                          (D.54.2)
```

(R11,R13)

```
        f                          f
   GabD -> FcdE               GabD -> FcdE
   -------------              ----------------
   Ga ↑ bD -> FcdE            GabD -> Fc v dE
   ---------------   >=       ----------------
   Ga ↑ bD -> Fc v dE         Ga ↑ bD -> Fc v dE          (D.55)
```

(R11,R14)

```
      f                            f            g
 GcdD -> a                   GcdD -> a     LbZ -> F
 ----------                  ---------------------
                 g
 Gc ↑ dD -> a   LbZ -> F      LGcdDa => bZ -> F
 -----------------------  >=  -------------------
   LGc ↑ dDa => bZ -> F       LGc ↑ dDa => bZ -> F         (D.56.1)

                  k                h               k
          DcdLbZ -> F          G -> a      DcdLbZ -> F
   h      -----------          ----------------------
 G -> a   Dc ↑ dLbZ -> F         DcdLGa => bZ -> F
 -----------------------  >=     -------------------
   Dc ↑ dLGa => bZ -> F          Dc ↑ dLGa => bZ -> F      (D.56.2)

                  m                h             m
          DbLcdZ -> F          G -> a     DbLcdZ -> F
   h      -----------          ---------------------
 G -> a   DbLc ↑ dZ -> F        DGa => bLcdZ -> F
 -----------------------  >=    -------------------
   DGa => bLc ↑ dZ -> F         DGa => bLc ↑ dZ -> F       (D.56.3)
```

(R11,R15)

```
          f                          f
   GaDcdL -> b                GaDcdL -> b
   ------------               ------------
   GaDc ↑ dL -> b             GDcdL -> (a => b)
   ------------------->=      -----------------
   GDc ↑ dL -> (a => b)       GDc ↑ dL -> (a => b)         (D.57.1)

         g                          g
   GcdDaL -> b                GcdDaL -> b
   ------------               ------------
   Gc ↑ dDaL -> b             GcdDL -> (a => b)
   ------------------->=      -----------------
   Gc ↑ dDL -> (a => b)       Gc ↑ dDL -> (a => b)         (D.57.2)

                   h                               h
             GabDcdL -> e                   GabDcdL -> e
           --------------                  --------------
           GabDc ↑ dL -> e                 Ga ↑ bDcdL -> e
         -----------------            ----------------------
         Ga ↑ bDc ↑ dL -> e           GDcdL -> (a ↑ b) => e
 ----------------------------- >=     -------------------------
 GDc ↑ dL -> (a ↑ b) => e             GDc ↑ dL -> (a ↑ b) => e
                                                           (D.57.3)
```

(R12,R12)

```
               f              g                    h              k
     GaDaL -> F     GaDaL -> F          GaDaL -> F     GaDaL -> F
     --------------------------          --------------------------
           Ga v aDaL -> F                     Ga v aDaL -> F
           --------------------------------------------------
                          Ga v aDa v aL -> F

                 f              h                 g              k
           GaDaL -> F     GaDaL -> F        GaDaL -> F     GaDaL -> F
           -------------------------        -------------------------
                 GaDa v aL -> F                   GaDa v aL -> F
 >=              ------------------------------------------------
                              Ga v aDa v aL -> F                  (D.58)
```

(R12,R13)

```
        f             g                      f                  g
   GaD -> FcdE   GbD -> FcdE           GaD -> FcdE         GbD -> FcdE
   -------------------------           -----------         -----------
       Ga v bD -> FcdE                 GaD -> Fc v dE      GbD -> Fc v dE
       -------------------      >=     ----------------------------------
       Ga v bD -> Fc v dE                     Ga v bD -> Fc v dE
                                                                  (D.59)
```

(R12,R14)

```
  f           g            f           h                              g          h
G -> a  DbLcZ -> F     G -> a  DbLdZ -> F                    DbLcZ -> F  DbLdZ -> F
------------------     ------------------                f   ----------------------
 DGa => bLcZ -> F       DGa => bLdZ -> F             G -> a     DbLc v dZ -> F
 ---------------------------------------   >=        ------------------------
      DGa => bLc v dZ -> F                            DGa => bLc v dZ -> F
                                                                    (D.60.1)

  f           g            f           h                              g          h
G -> a  DcLbZ -> F     G -> a  DdLbL -> F                    DcLbL -> F  DdLbL -> F
------------------     ------------------                f   ----------------------
 DcLGa => bZ -> F       DdLGa => bZ -> F             G -> a     Dc v dLbZ -> F
 ---------------------------------------   >=        ------------------------
      Dc v dLGa => bZ -> F                            Dc v dLGa => bZ -> F
                                                                    (D.60.2)
```

(R12,R15)

```
                 f              g                  f                  g
           GcDaL -> d    GcDbL -> d          GcDaL -> d         GcDbL -> d
           ------------------------          -----------        -----------
             GcDa v bL -> d                  GDaL -> c => d     GDbL -> c => d
             --------------                  ---------------------------------
             GDa v bL -> c => d                     GDa v bL -> c => d
                                                                    (D.61.1)

                 h              k                  h                  k
           GaDcL -> d    GbDcL -> d          GaDcL -> d         GbDcL -> d
           ------------------------          -----------        -----------
             Ga v bDcL -> d                  GaDL -> c => d     GbDL -> c => d
             --------------                  ---------------------------------
             Ga v bDL -> c => d                     Ga v bDL -> c => d
                                                                    (D.61.2)
```

(R13,R13)

```
              f                               f
           G -> FabEcdH                    G -> FabEcdH
           ------------                    ------------
           G -> Fa v bEcdH                 G -> FabEc v dH
           ------------------     >=       ------------------
           G -> Fa v bEc v dH              G -> Fa v bEc v dH        (D.62)
```

(R13,R14)

```
       f            g                               g
     G -> a   DbL -> FcdE                      DbL -> FcdE
     --------------------                  f   -----------
     DGa => bL -> FcdE                   G -> a  DbL -> Fc v dE
     ------------------------    >=      -----------------------
     DGa => bL -> Fc v dE                  DGa => bL -> Fc v dE      (D.63)
```

(R14,R14)

```
                  g             h                        f             h
              D -> a   LbZdP -> F               G -> c   LbZdP -> F
              --------------------          ---------------------------
       f                                     g
   G -> c   LDa => bZdP -> F             D -> a    LbZGc => dP -> F
   --------------------------   >=      ---------------------------
    LDa => bZGc => dP -> F               LDa => bZGc => dP -> F      (D.64)
```

(R14,R15)

```
         f               g                                 g
       G -> a    DcLbZ -> d                       DcLbZ -> d
       ------------------------               f   -------------
       DcLGa => bZ -> d                   G -> a  DLbZ -> c => d
       -------------------------- >=     --------------------------
       DLGa => bZ -> c => d                DLGa => bZ -> c => d
                                                                     (D.65.1)

         f               g                                 g
       G -> a    DbLcZ -> d                       DbLcZ -> d
       ------------------------               f   -------------
       DGa => bLcZ -> d                   G -> a  DbLZ -> c => d
       -------------------------- >=     --------------------------
       DGa => bLZ -> c => d                DGa => bLZ -> c => d
                                                                     (D.65.2)

                  f            h                         g            h
              G -> c   LbZdP -> e                    D -> a   LbZdP -> e
              --------------------                   --------------------
       g                                              LDa => bZdP -> e
   D -> a    LbZGc => dP -> e                      f  ---------------------------
   ---------------------------                 G -> c  LDZdP -> (a => b) => e
     LDa => bZGc => dP -> e                    -----------------------------------
   ------------------------------------- >=
   LDZGc => dP -> (a => b) => e                  LDZGc => dP -> (a => b) => e
                                                                     (D.65.3)

         m                 n                                    n
       G -> a    DcdLbZ -> e                          DcdLbZ -> e
       ------------------------                     ---------------
        DcdLGa => bZ -> e                            Dc ↑ dLbZ -> e
        -------------------------               m   -----------------------
        Dc ↑ dLGa => bZ -> e                G -> a  DLbZ -> (c ↑ d) => e
        ------------------------------ >=  ------------------------------
        DLGa => bZ -> (c ↑ d) => e           DLGa => bZ -> (c ↑ d) => e
                                                                     (D.65.4)

         m                 r                                    r
       G -> a    DbLcdZ -> e                          DbLcdZ -> e
       ------------------------                     ---------------
        DGa => bLcdZ -> e                            DbLc ↑ dZ -> e
        -------------------------               m   -----------------------
        DGa => bLc ↑ dZ -> e                G -> a  DbLZ -> (c ↑ d) => e
        ------------------------------ >=  ------------------------------
        DGa => bLZ -> (c ↑ d) => e           DGa => bLZ -> (c ↑ d) => e
                                                                     (D.65.5)

             m               s                                  s
           G -> a    DbLZ -> e                          DbLZ -> e
           ------------------------                     -------------
            DGa => bLZ -> e                             DbLcZ -> e
            -----------------------                 m   -------------------
            DGa => bLcZ -> e                     G -> a DbLZ -> c => e
            ----------------------- >=          ---------------------------
            DGa => bLZ -> c => e                  DGa => bLZ -> c => e
                                                                     (D.65.6)

             m               t                                  t
           G -> a    DLbZ -> e                          DLbZ -> e
           ------------------------                     -------------
            DLGa => bZ -> e                             DcLbZ -> e
            -----------------------                 m   -------------------
            DcLGa => bZ -> e                     G -> a DLbZ -> c => e
            ----------------------- >=          ---------------------------
            DLGa => bZ -> c => e                  DLGa => bZ -> c => e
                                                                     (D.65.7)
```

```
(R2,R2,R10)
                                                    f              g
                                              GcD -> FaE     GdD -> FbE
       f               g                      -----------    -----------
 GcD -> FaE      GdD -> FbE                   GdcD -> FaE    GdcD -> FbE
 -----------     -----------                  --------------------------
 GcdD -> FaE     GcdD -> FbE                      GdcD -> Fa ↑ bE
 ---------------------------      >=              ---------------
       GcdD -> Fa ↑ bE                            GcdD -> Fa ↑ bE        (D.71)

(R2,R2,R12)
                                                      f              g
                                              GcDaL -> F     GdDbL -> F
         f               g                    -----------    -----------
 GcDaL -> F      GdDbL -> F                   GdcDaL -> F    GdcDbL -> F
 -----------     -----------                  --------------------------
 GcdDaL -> F     GcdDbL -> F                      GdcDa v bL -> F
 ---------------------------      >=              ---------------
       GcdDa v bL -> F                            GcdDa v bL -> F        (D.72.1)

                                                      h              k
                                              GaDcL -> F     GbDdL -> F
         h               k                    -----------    -----------
 GaDcL -> F      GbDdL -> F                   GaDdcL -> F    GbDdcL -> F
 -----------     -----------                  --------------------------
 GaDcdL -> F     GbDcdL -> F                      Ga v bDdcL -> F
 ---------------------------      >=              ---------------
       Ga v bDcdL -> F                            Ga v bDcdL -> F        (D.72.2)

(R2,R3,R10)
                                                    f
                                              GcD -> FaE
                        g                     -----------          g
                 GccD -> FbE                  GccD -> FaE    GccD -> FbE
      f          -----------                  --------------------------
 GcD -> FaE      GcD -> FbE                       GccD -> Fa ↑ bE
 ---------------------------      >=              ---------------
       GcD -> Fa ↑ bE                             GcD -> Fa ↑ bE         (D.73.1)

                                                                   k
                                                              GcD -> FbE
       h                                            h         -----------
 GccD -> FaE                                  GccD -> FaE    GccD -> FbE
 -----------           k                      --------------------------
 GcD -> FaE      GcD -> FbE                       GccD -> Fa ↑ bE
 ---------------------------      >=              ---------------
       GcD -> Fa ↑ bE                             GcD -> Fa ↑ bE         (D.73.2)

(R2,R3,R12)
                                                      f
                                              GcDaL -> F
                          g                   -----------            g
                 GccDbL -> F                  GccDaL -> F    GccDbL -> F
         f       -----------                  --------------------------
 GcDaL -> F      GcDbL -> F                       GccDa v bL -> F
 ---------------------------      >=              ---------------
       GcDa v bL -> F                             GcDa v bL -> F         (D.74.1)

                                                                     k
                                                              GcDbL -> F
          h                                            h      -----------
 GccDaL -> F                                  GccDaL -> F    GccDbL -> F
 -----------             k                    --------------------------
 GcDaL -> F      GcDbL -> F                       GccDa v bL -> F
 ---------------------------      >=              ---------------
       GcDa v bL -> F                             GcDa v bL -> F         (D.74.2)

                                                      m
                                              GaDcL -> F
                          n                   -----------            n
                 GbDccL -> F                  GaDccL -> F    GbDccL -> F
         m       -----------                  --------------------------
 GaDcL -> F      GbDcL -> F                       Ga v bDccL -> F
 ---------------------------      >=              ---------------
       Ga v bDcL -> F                             Ga v bDcL -> F         (D.74.3)
```

```
                                                                    q
                                                             GbDcL -> F
            p                                   p            ----------
      GaDccL -> F                         GaDccL -> F     GbDccL -> F
      -----------                         -------------------------
                          q
       GaDcL -> F    GbDcL -> F               Ga v bDccL -> F
       ------------------------- >=           ---------------
            Ga v bDcL -> F                     Ga v bDcL -> F        (D.74.4)

(R2,R4,R10)
                                                  f
                                             GcD -> FaE
          f              g                  -----------              g
       GcD -> FaE   GcdD -> FbE             GcdD -> FaE        GcdD -> FbE
      -----------   -----------             ------------------------------
      GdcD -> FaE   GdcD -> FbE                   GcdD -> Fa ↑ bE
      ------------------------- >=                ---------------
         GdcD -> Fa ↑ bE                          GdcD -> Fa ↑ bE    (D.75.1)

                                                  h
                                             GdD -> FaE
          h              g                  -----------              g
       GdD -> FaE   GcdD -> FbE             GcdD -> FaE        GcdD -> FbE
      -----------   -----------             ------------------------------
      GdcD -> FaE   GdcD -> FbE                   GcdD -> Fa ↑ bE
      ------------------------- >=                ---------------
         GdcD -> Fa ↑ bE                          GdcD -> Fa ↑ bE    (D.75.2)

                                                                     f
                                                              GcD -> FaE
          g                f                   g             -----------
      GcdD -> FbE     GcD -> FaE          GcdD -> FbE        GcdD -> FaE
      -----------   -----------           --------------------------------
      GdcD -> FbE   GdcD -> FaE                   GcdD -> Fb ↑ aE
      ------------------------- >=                ---------------
         GdcD -> Fb ↑ aE                          GdcD -> Fb ↑ aE    (D.75.3)

                                                                     h
                                                              GdD -> FaE
          g                h                   g             -----------
      GcdD -> FbE     GdD -> FaE          GcdD -> FbE        GcdD -> FaE
      -----------   -----------           --------------------------------
      GdcD -> FbE   GdcD -> FaE                   GcdD -> Fb ↑ aE
      ------------------------- >=                ---------------
         GdcD -> Fb ↑ aE                          GdcD -> Fb ↑ aE    (D.75.4)

(R2,R4,R12)
                                                   f
                                            GcDaL -> F
             f               g              ----------             g
       GcDaL -> F   GcdDbL -> F             GcdDaL -> F     GcdDbL -> F
      -----------   -----------             ------------------------------
      GdcDaL -> F   GdcDbL -> F                   GcdDa v bL -> F
      ------------------------- >=                ---------------
         GdcDa v bL -> F                          GdcDa v bL -> F    (D.76.1)

                                                   h
                                            GdDaL -> F
             h               g              ----------             g
       GdDaL -> F   GcdDbL -> F             GcdDaL -> F     GcdDbL -> F
      -----------   -----------             ------------------------------
      GdcDaL -> F   GdcDbL -> F                   GcdDa v bL -> F
      ------------------------- >=                ---------------
         GdcDa v bL -> F                          GdcDa v bL -> F    (D.76.2)

                                                                     f
                                                             GcDaL -> F
              g              f                     g        -----------
      GcdDbL -> F     GcDaL -> F           GcdDbL -> F      GcdDaL -> F
      -----------   -----------            -------------------------------
      GdcDbL -> F   GdcDaL -> F                   GcdDb v aL -> F
      ------------------------- >=                ---------------
         GdcDb v aL -> F                          GdcDb v aL -> F    (D.76.3)
```

```
                                                              h
                                                         GdDaL -> F
          g             h                      g        ------------
GcdDbL -> F    GdDaL -> F          GcdDbL -> F     GcdDaL -> F
-----------    -----------         ---------------------------
GdcDbL -> F   GdcDaL -> F               GcdDb v aL -> F
------------------------- >=            ---------------
      GdcDb v aL -> F                   GdcDb v aL -> F       (D.76.4)


                                           k
                                    GaDcL -> F
        k                m          -----------            m
 GaDcL -> F    GbDcdL -> F         GaDcdL -> F    GbDcdL -> F
 -----------   -----------         ----------------------------
GaDdcL -> F   GbDdcL -> F              Ga v bDcdL -> F
------------------------- >=           ---------------
     Ga v bDdcL -> F                   Ga v bDdcL -> F        (D.76.5)


                                           n
                                    GaDdL -> F
        n                m          -----------            m
 GaDdL -> F    GbDcdL -> F         GaDcdL -> F    GbDcdL -> F
 -----------   -----------         ----------------------------
GaDdcL -> F   GbDdcL -> F              Ga v bDcdL -> F
------------------------- >=           ---------------
     Ga v bDdcL -> F                   Ga v bDdcL -> F        (D.76.6)


                                                          k
                                                    GaDcL -> F
         m               k                 m        -----------
GbDcdi -> F     GaDcL -> F        GbDcdL -> F     GaDcdL -> F
-----------     -----------       ----------------------------
GbDdcL -> F   GaDdcL -> F              Gb v aDcdL -> F
------------------------- >=           ---------------
     Gb v aDdcL -> F                   Gb v aDdcL -> F        (D.76.7)


                                                          n
                                                    GaDdL -> F
         m               n                 m        -----------
GbDcdL -> F     GaDdL -> F        GbDcdL -> F     GaDcdL -> F
-----------     -----------       ----------------------------
GbDdcL -> F   GaDdcL -> F              Gb v aDcdL -> F
------------------------- >=           ---------------
     Gb v aDdcL -> F                   Gb v aDdcL -> F        (D.76.8)


          f             g
     GaD -> F     GaD -> F
     --------     --------                 f              g
    GbaD -> F    GcaD -> F            GaD -> F       GaD -> F
    ----------------------            ---------      ---------
       G(b v c)aD -> F               GabD -> F      GacD -> F
       ---------------        >=     ------------------------
       Ga(b v c)D -> F                  Ga(b v c)D -> F        (D.76.9)


          h             g
     GbD -> F     GaD -> F
     --------     --------                 h              g
    GbaD -> F    GcaD -> F            GbD -> F       GaD -> F
    ----------------------            ---------      ---------
       G(b v c)aD -> F               GabD -> F      GacD -> F
       ---------------        >=     ------------------------
       Ga(b v c)D -> F                  Ga(b v c)D -> F        (D.76.10)


          f             k
     GaD -> F     GcD -> F
     --------     --------                 f              k
    GbaD -> F    GcaD -> F            GaD -> F       GcD -> F
    ----------------------            ---------      ---------
       G(b v c)aD -> F               GabD -> F      GacD -> F
       ---------------        >=     ------------------------
       Ga(b v c)D -> F                  Ga(b v c)D -> F        (D.76.11)
```

```
       h            k
  GbD -> F     GcD -> F
  --------     --------                   h            k
 GbaD -> F    GcaD -> F              GbD -> F     GcD -> F
 ----------------------              --------     --------
    G(b v c)aD -> F                 GabD -> F    GacD -> F
    ---------------         >=      ----------------------
    Ga(b v c)D -> F                    Ga(b v c)D -> F        (D.76.12)

       f            g
  GaD -> F     GaD -> F
  --------     --------                   f            g
 GabD -> F    GacD -> F              GaD -> F     GaD -> F
 ----------------------              --------     --------
    Ga(b v c)D -> F                 GbaD -> F    GcaD -> F
    ---------------         >=      ----------------------
    G(b v c)aD -> F                    G(b v c)aD -> F        (D.76.13)

       h            g
  GbD -> F     GaD -> F
  --------     --------                   h            g
 GabD -> F    GacD -> F              GbD -> F     GaD -> F
 ----------------------              --------     --------
    Ga(b v c)D -> F                 GbaD -> F    GcaD -> F
    ---------------         >=      ----------------------
    G(b v c)aD -> F                    G(b v c)aD -> F        (D.76.14)

       f            k
  GaD -> F     GcD -> F
  --------     --------                   f            k
 GabD -> F    GacD -> F              GaD -> F     GcD -> F
 ----------------------              --------     --------
    Ga(b v c)D -> F                 GbaD -> F    GcaD -> F
    ---------------         >=      ----------------------
    G(b v c)aD -> F                    G(b v c)aD -> F        (D.76.15)

       h            k
  GbD -> F     GcD -> F
  --------     --------                   h            k
 GabD -> F    GacD -> F              GbD -> F     GcD -> F
 ----------------------              --------     --------
    Ga(b v c)D -> F                 GbaD -> F    GcaD -> F
    ---------------         >=      ----------------------
    G(b v c)aD -> F                    G(b v c)aD -> F        (D.76.16)
```

(R3,R3,R10)

```
                                             f                     g
                                      GccD -> FaE         GcdD -> FbE
                                      -----------         -----------
                                     GccdD -> FaE        GccdD -> FbE
         f               g           --------------------------------
 GccD -> FaE     GcdD -> FbE                GccdD -> Fa ↑ bE
 -----------     -----------                ----------------
  GcdD -> FaE     GcdD -> FbE                GccD -> Fa ↑ bE
  ---------------------------  >=            ---------------
       GcdD -> Fa ↑ bE                        GcdD -> Fa ↑ bE  (D.77.1)

                                             h                    k
                                      GcdD -> FaE         GccD -> FbE
                                      -----------         -----------
                                     GccdD -> FaE        GccdD -> FbE
         h               k           --------------------------------
 GcdD -> FaE     GccD -> FbE                GccdD -> Fa ↑ bE
 -----------     -----------                ----------------
  GcdD -> FaE     GcdD -> FbE                GccD -> Fa ↑ bE
  ---------------------------  >=            ---------------
       GcdD -> Fa ↑ bE                        GcdD -> Fa ↑ bE  (D.77.2)
```

(R3,R3,R12)

```
                                                 f                      g
                                      GccDaL -> F          GcdDbL -> F
                                      -----------          -----------
                                     GccdDaL -> F         GccdDbL -> F
           f                 g       ---------------------------------
 GccDaL -> F      GcdDbL -> F               GccdDav bL -> F
 -----------      -----------               ---------------
  GcdDaL -> F      GcdDbL -> F               GccDa v bL -> F
  ---------------------------  >=            ---------------
       GcdDa v bL -> F                        GcdDa v bL -> F  (D.78.1)
```

```
                                                                 h                    k
                                                       GcddDaL  -> F       GccdDbL  -> F
                                                      --------------      --------------
                 h                   k                GccddDaL  -> F     GccddDbL  -> F
   GcddDaL  -> F       GccdDbL  -> F                  ----------------------------------
   --------------      --------------                         GccddDa v bL  -> F
     GcdDaL  -> F        GcdDbL  -> F                         ------------------
     --------------------------------- >=                       GccdDa v bL  -> F
              GcdDa v bL  -> F                                  ------------------
                                                                  GcdDa v bL  -> F (D.78.2)

                                                                 f                    g
                                                       GaDccdL  -> F       GbDcddL  -> F
                                                      --------------      --------------
                 f                   g                GaDccddL  -> F     GbDccddL  -> F
   GaDccdL  -> F       GbDcddL  -> F                  ----------------------------------
   --------------      --------------                         Ga v bDccddL  -> F
     GaDcdL  -> F        GbDcdL  -> F                         ------------------
     --------------------------------- >=                       Ga v bDccdL  -> F
              Ga v bDcdL  -> F                                  ------------------
                                                                  Ga v bDcdL  -> F (D.78.3)

                                                                 h                    k
                                                       GaDcddL  -> F       GbDccdL  -> F
                                                      --------------      --------------
                 h                   k                GaDccddL  -> F     GbDccddL  -> F
   GaDcddL  -> F       GbDccdL  -> F                  ----------------------------------
   --------------      --------------                         Ga v bDccddL  -> F
     GaDcdL  -> F        GbDcdL  -> F                         ------------------
     --------------------------------- >=                       Ga v bDccdL  -> F
              Ga v bDcdL  -> F                                  ------------------
                                                                  Ga v bDcdL  -> F (D.78.4)
```

(R4,R4,R10)

```
                                                                      g
                                                              GdcD  -> FbE
           f                                           f      ------------
   GcdD  -> FaE                                 GcdD  -> FaE  GcdD  -> FbE
   ------------              g                  --------------------------
   GdcD  -> FaE      GdcD  -> FbE                    GcdD  -> Fa ↑ bE
   ------------------------------ >=                 ----------------
        GdcD  -> Fa ↑ bE                             GdcD  -> Fa ↑ bE      (D.79)
```

(R4,R4,R12)

```
                                                                          g
                                                              GdcDbL  -> F
           f                                          f       ------------
   GcdDaL  -> F                                 GcdDaL  -> F  GcdDbL  -> F
   ------------                g                --------------------------
   GdcDaL  -> F      GdcDbL  -> F                    GcdDa v bL  -> F
   ------------------------------ >=                 ----------------
        GdcDa v bL  -> F                             GdcDa v bL  -> F      (D.80.1)

                                                                          g
                                                              GbDdcL  -> F
           f                                          f       ------------
   GaDcdL  -> F                                 GaDcdL  -> F  GbDcdL  -> F
   ------------                g                --------------------------
   GaDdcL  -> F      GbDdcL  -> F                    Ga v bDcdL  -> F
   ------------------------------ >=                 ----------------
        Ga v bDdcL  -> F                             Ga v bDdcL  -> F      (D.80.2)
```

Appendix (5)

User Manual

$1. Introduction

szabo is an implementation, running under prolog, to automate some reasoning about cartesian, bicartesian, distributive bicartesian, cartesian closed and bicartesian closed categories.

The implementation is based on the algorithms outlined in SZABO [1978], to interpret derivations from the sequent propositional calculus as arrows of the corresponding free category and to represent the arrows of this free category by means of derivations with cuts. Also it includes procedures to eliminate the cut from the derivations and to produce the normal form, according to the algorithms of cut-elimination and normal ization due to SZABO . All the work behind this implementation is discussed in detail in chapters 1-5.

$2. ' szabo ' Program Syntax

We define here the syntax, using BNF (NAUR [1960]), of the formal language processed by our implementation. Note the use of lower case letters in the object language, since prolog restricts upper case letters to be names of meta-variables.

Each proof as input to the program is a list of named steps, such that each step is either a representation of an axiom or derived from another (or others) by using a rule of inference. The conclusion of the original proof will be represented by the last step in the input proof. The syntax of each step will be described in BNF as follows :-

```
PROOF_STEP          ::=  STEP_NAME " = " STEP
STEP_NAME           ::=  LOWER_CASE_WORD | NUMBERED_LOWER_CASE_WORD
STEP                ::=  "[" JUSTIFICATION "," ANTECEDENT "=>" <SUCCEDENT "]"
ANTECEDENT          ::=  "[" FORMULA_SEQUENCE "]" | "[" "]"
SUCCEDENT           ::=  "[" FORMULA_SEQUENCE "]" | "[" "]"
FORMULA_SEQUENCE    ::=  FORMULA | FORMULA_SEQUENCE "," FORMULA
FORMULA             ::=  ATOMIC_FORMULA | FORMULA  CONNECTIVE  ATOMIC_FORMULA
ATOMIC_FORMULA      ::=  PROPOSITIONAL_SYMBOL | bo | tr | id
PROPOSITIONAL_SYMBOL ::= LOWER_CASE_WORD
                          [ other than one of {bo, tr, CURRENT_NUMBERED_f}]
NUMBERED_LOWER_CASE_WORD ::= LOWER_CASE_WORD  INTEGER
```

```
INTEGER              ::=  DIGIT | INTEGER  DIGIT
DIGIT                ::=  0| 1| 2| 3| 4| 5| 6| 7| 8| 9
LOWER_CASE_WORD      ::=  LOWER_CASE_LETTER| LOWER_CASE_WORD  LOWER_CASE_LETTER
LOWER_CASE_LETTER::=  a| b| c| d| e| f| g| h| i| j| k| l| m|
                          n| o| p| q| r| s| t| u| v| w| x| y| z.
CURRENT_NUMBERED_f     ::=  f  INTEGER
CONNECTIVE             ::=  &| #| ->
JUSTIFICATION          ::=  AXIOM_NAME | RULE
AXIOM_NAME             ::=  ARROW | true | bottom
ARROW                  ::=  LOWER_CASE_WORD| NUMBERED_LOWER_CASE_WORD
                                 s.t.not one of ( true, bottom )
RULE                   ::=  ( th|con|aia|ois|his) "(" ARGS1 ")" |
                            inc "(" ARGS2 ")"| (cut|ais|oia|hia) "(" ARGS3 ")"
ARGS1                    ::=  STEP_NAME "," FORMULA
ARGS2                    ::=  STEP_NAME "," FORMULA "," FORMULA
ARGS3                    ::=  STEP_NAME "," STEP_NAME "," FORMULA
```

$3. Abbreviations & Syntax of Connectives

We show here how to translate between our syntax and that of the sequent calculus as follows :-

1. Names of the axioms

program syntax	Sequent Calculus syntax
f	A --f--> B
true	----------> T
bottom	⊥ ---------->

2. Names of the rules of inference

a) Structural rules :-

program syntax	Sequent Calculus syntax
th	Thinning --> or --> Thinning (Thinning in the antecedent and in the succedent)
con	Contraction --> or --> Contraction (Contraction in the antecedent and in the succedent)
inc	Interchange --> or --> Interchange (Interchange in the antecedent and in the succedent)
cut	cut (Cut rule)

b) Operational rules

program syntax	Sequent Calculus syntax
aia	↑ I --> (and introduction antecedent)
ais	--> ↑ I (and introduction succedent)
oia	v I --> (or introduction antecedent)
ois	--> v I (or introduction succedent)
hia	==>I --> (hook introduction antecedent)
his	--> ==>I (hook introduction succedent)

We summarize the syntax to be used in the implementation corresponding to the traditional propositional calculus syntax for the connectives as follows :-

Connective	Prop.Calculus syntax	Program syntax
conjunction	a ↑ b	a & b
disjunction	a v b	a # b
implication	a => b	a -> b
true	T	tr
bottom	⊥	bo

NB .

Here we remark on the syntax for the rules of inference defined in $2. Any rule of inference name must be followed by a number of arguments, and these arguments must be in the order as shown in $2. We explain this order for some of them which are easily confused :

1) inc(l,a,b)

This refers to the instance of the interchange rule which, operating on a step 'l' having adjacent formulae 'b,a' (in that order) in antecedent or succedent, produces a new step with these two formulae in order ' a,b '.

2) cut(l,m,c)

This means that the cut rule is applied to the steps which have names 'l','m', with the cut formula c; 'l' is the name of the step which has 'c' in its succedent, and 'm' is the name of the step which has 'c' in its antecedent.

3) ais(l,m,a & b),

'l' is the name of a step with 'a' in its succedent; 'm' has 'b' in its succedent : this construct performs 'and-introduction' on the succedents.

4) oia(l,m,a # b)

'l' is the name of a step with 'a' in its antecedent; 'm' has 'b' in its antecedent : this construct performs 'or-introduction' on the antecedents.

5) hia(l,m,a -> b)

'l' is the name of a step with 'a' in its succedent; 'm' has 'b' in its antecedent : this construct performs 'hook-introduction' on the antecedents.

$4. Canonical arrows in category theory

In this section we give some canonical arrows from different free categories (These particular arrows, and their representations by means of derivations of the sequent propositional calculus, are built into the system for the user's convenience) .

(4.1) Canonical arrows in Fc(X) (The free cartesian category)

Let X be a category and Fc(X) is the free cartesian category generated by X and a terminal object T and connective ↑ ; the following arrows are in ArFc(X) :

i) pi_left(a,b) : a ↑ b -> a , for a,b in ObFc(X) ;

ii) pi_right(a,b) : a ↑ b -> b , for a,b in ObFc(X) ;

iii) delta(a) : a -> a ↑ a , for a in ObFc(X) ;

iv) alpha(a,b,c) : a ↑ (b ↑ c) -> (a ↑ b) ↑ c, for a,b,c in ObFc(X);

v) alpha_inverse(a,b,c) : (a ↑ b) ↑ c -> a ↑ (b ↑ c), for a,b,c in ObFc(X);

vi) for f : a -> b and g : c -> d in ArFc(X),
product(f,g) : a ↑ c -> b ↑ d ;

vii) sigma(a,b) : a ↑ b -> b ↑ a , for a,b in ObFc(X) ;

viii) sigma_inverse(a,b) : b ↑ a -> a ↑ b , for a,b in ObFc(X) ;

ix) for f : a -> b and g : b -> c in ArFc(X),
comp(g,f) : a -> c ;

x) for f : a -> b and g : a -> c in ArFc(X),
angle(f,g) : a -> b ↑ c .

(4.2) <u>Canonical arrows in Fbc(X)</u> (The free bicartesian category)

We extend the canonical arrows in Fc(X) by the following :

xi) pi_left_star(a,b) : a -> a v b , for a,b in ObFbc(X) ;

xii) pi_right_star(a,b) : b -> a v b , for a,b in ObFbc(X) ;

xiii) delta_star(a) : a v a -> a , for a in ObFbc(X) ;

xiv) alpha_star(a,b,c) : a v (b v c) -> (a v b) v c, for a,b,c in ObFbc(X) ;

xv) alpha_star_inverse(a,b,c) : (a v b) v c -> a v (b v c) , for a,b,c in ObFbc(X) ;

xvi) sigma_star(a,b) : a v b -> b v a , for a,b in ObFbc(X) ;

xvii) sigma_star_inverse(a,b) : b v a -> a v b, for a,b in ObFbc(X);

xviii) for f : a -> b and g : c -> d in ArFbc(X),
union(f,g) : a v c -> b v d ;

xix) for f : a -> c and g : b -> c in ArFbc(X),
square(f,g) : a v b -> c .

(4.3) <u>Canonical arrows in Fdbc(X)</u>(free distributive bicartesian category)

We extend the canonical arrows in Fbc(X) by the following :

xx) delta_left(a,b,c) : a ↑ (b v c) -> (a ↑ b) v (a ↑ c) ;

xxi) delta_right(a,b,c) : (a v b) ↑ c -> (a ↑ c) v (b ↑ c) ;

xxii) delta_left_star(a,b,c) : (a v b) ↑ (a v c) -> a v (b ↑ c) ;

xxiii) delta_right_star(a,b,c) : (a v c) ↑ (b v c) -> (a ↑ b) v c ;

xxiv) kapa(a,b,c,d,e) : (a↑((b v c) v d))↑e -> (b v ((a↑c)↑e)) v d;

(4.4) Canonical arrows in Fccl(X) (The free cartesian closed category)

We extend the canonical arrows in Fc(X) by the following :

xi) for f : a -> b and g : c -> d in ArFccl(X),
epsilon_left(f,g) : a ↑ (b => c) -> d ;

xii) for f : a ↑ b -> c in ArFccl(X),
alpha_right(f) : b -> (a => c) ;

xiii) epsilon(a,b) : a ↑ (a => b) -> b , for a,b in ObFccl(X);

xiv) ita(a,b) : b -> a => (a ↑ b), for a,b in ObFccl(X);

xv) for f : a -> b and g : c -> d in ArFccl(X),
hook(f,g) : (b => c) -> (a => d) .

(4.5) Canonical arrows in Fbccl(X)(The free bicartesian closed category)

The canonical arrows in Fbccl(X) are combination of all canonical arrows in Fc(X), Fbc(X), Fdbc(X) and Fccl(X).

$5. Predicates (Commands)

In order to use this system there are two types of command (we call them predicates, because of prolog convention) :

I) Proof theory predicates.

II) Category theory predicates.

I) Proof theory predicates (commands)

The following predicates are used to manipulate derivations from the classes of sequent propositional calculus discussed in chapters 1-5; the derivations must be asserted to the system according to the syntax described in $2.

1- theory(X) : Tells the system that the theory we intend to work with is X, where X represents one of the following abbreviations cart (cartesian), bicart (bicartesian), dbicart (distributive bicartesian), cartcl (cartesian closed) or bicartcl (bicartesian closed).

2- composite(F,G,H) : This predicate is used, when arrows F,G of the underlying category X with composition H are used in the construction of the derivation in operation.

3- prove(X) : Checks the correctness of the current instance of a proof-step X; if it is correct, it will be added to the database for further use. We keep using this command until the proof is complete.

4- proof : This command is used, as another way to check the correctness and assert derivations without using the formal syntax for proof-step in $2 and the predicate ' prove(X) ' above. It will switch on the system to communicate with the user to assert the derivation in informal way.

5- interpr(X,K) : After asserting a derivation, this command is used to compute the interpretation of this derivation as an arrow in the free category. X represents the name of the last proof-step and H will be instantiated as the arrow computed to interpret the derivation.

6- cut_free(X) : Tells the system to start the processes of transforming the derivation in question to its equivalent cut-free one. X represents the name of the last proof-step.

7- con_inc_free(X) : where X represents the name of the last step of a cut-free proof. This predicate cause the system to start the processes of transforming the cut-free proof into its equivalent contraction & interchange-free one. This is used only if we dealing with the theories of <u>cart</u>, <u>bicart</u> or <u>dbicart</u>.

8- normal(X) : This predicate is a combination of three processes : cut-elimination, contraction & interchange-elimination and normalization; so that this general command is used in place of cut_free and con_inc_free to produce the equivalent normal form for a derivation. X represents the name of the last step of the derivation.

II) <u>Category theory predicates</u> (Commands)

This subsection describes predicates for manipulating arrows of the free cartesian, bicartesian, distributive bicartesian, cartesian closed and bicartesian closed categories. These arrows will be represented by means of derivations from the sequent propositional calculus. By the processes of cut-elimination and normalization, these representations will be transformed to their equivalent normal form. The system also is used to test whether diagrams, from the theories mentioned above, dose commute or not. The following are the predicates of the system to operate the above methods.

1- arrow(A,F,B) : Insert an arrow , of the underlying category X, being used in the construction of the arrow of the free category in question. F represents an arrow from A to B (F : A -> B).

2- rep_of(X) : Will produce a representation of an arrow X as a derivation.

3- norm_rep(X) : If X represents an arrow of some free category, this predicate causes the system to produce a representation of X by means of a derivation, then calling the processes of cut-elimination and normalization to produce the corresponding normal representation.

4- commutative : Tests whether two arrows are equal or not, in order to test the commutativity of the diagrams in category theory. After reading this command, the system waits to read the LHS arrow of the equality in question and responds by computing its normal representation. The system waits again to read the RHS arrow of the equality and responds by computing its normal representation. Finally it compares these two normal representations and responds with the conclusion.

5- inter_norm_rep(X): This is a combination of two processes; one for normal representation of the arrow X, the other is to interpret this normal representation to an arrow of the same theory of the arrow X.

The following are general commands to be used :

1- help : This displays the commands available for the system.

2- clean : This command is to be used after each session of use, in order to use the system using the same theory and the same arrows, if any, of the underling category for another session.

3- new : Cleans up the database from all the processes of the last session and puts the system again to the top level.

$6. How To Use The System ?

To use this system for both proof theory and category theory there are two ways :

1) Direct interaction with the system.
2) Using a text editor to create predicates in a file, then telling the system to be directed by the file.

1) Direct interaction with the system :-

Type the UNIX command ' szabo ' at your terminal ; this will prompt by ' : ?- ' , and will then wait for you to start your session, using the predicates described in $5 above.

2) Indirectly from a file.

In this way there two options :

a) Printing the results on the screen

In this option create a session of input in a file using a text editor. After you have the session of input to the system in a file (which has a name 'file_name') you can type the command " szabo file_name " at your terminal, it will respond by printing out the answer you are looking for.

b) Collecting the results in a new file :-

Using the redirection facility of UNIX, the output sent to the screen as in (a) above can be sent to a file .
The command " szabo file_name > newfile_name " will do the process.

Example :-

Suppose the following proof has been created in a file ' ex ' as follows :

```
theory(cartcl).

prove(l = [id(a),[a]=>[a]]).
prove(m = [th(l,b),[a,b]=>[a]]).
prove(n = [id(b),[b]=>[b]]).
prove(o = [th(n,a),[a,b]=>[b]]).
prove(p = [ais(m,o,a&b),[a,b]=>[a&b]]).
prove(t = [h,[a]=>[c]]).
prove(u = [th(t,b),[a,b]=>[c]]).
prove(q = [ala(u,a&b),[a&b]=>[c]]).
prove(r = [cut(p,q,a&b),[a,b]=>[c]]).
prove(s = [his(r,a->c),[b]=>[a->c]]).

normal(s).
```

we then use the command;

$* szabo ex > ex.lis

will read the predicates from ' ex ' and output the proof and its equivalent normal form to file ' ex.lis '.

$7. Input & Output Sessions with the System

In this section, we give some examples to show the user how to use the system for both proof theory and category theory.

(7.1) Proof theory Examples

In the following example we use the predicate ' proof ' to check the correctness and assert a derivation to the system in an informal way without using the syntax of $2 above.

Example 1

For f in X(a,b), g in X(a,c) and h in X(e,d), for some category X, the following derivation in the class of derivations of the theory of cartesian categories :

```
   f                     g                             h
a ---> b              a ---> c                      e ---> d
------------------------------                 ---------------
          a ---> b ↑ c                         b ↑ c,e ---> d
          ----------------------------------------------------
                               a,e ---> d
```

can be checked and asserted in the system as follows :

```
yes
| ?- theory(cart).
yes
| ?- proof.
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : axiom.

Enter name of proof step : l.

Enter name of the arrow  : f.

Enter the antecedent included in a list : [a].

Enter the succedent included in a list  : [b].


next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : axiom.

Enter name of proof step : m.

Enter name of the arrow  : g.

Enter the antecedent included in a list : [a].

Enter the succedent included in a list : [c].
```

```
next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : ais.

Enter name of proof step : n.

Enter name of LHS of previous steps : l.

Enter name of RHS of previous steps : m.

Enter name of ais formula : b & c.

Enter the antecedent included in a list : [a].

Enter the succedent included in a list  : [b & c].


next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : axiom.

Enter name of proof step : o.

Enter name of the arrow  : h.

Enter the antecedent included in a list : [e].

Enter the succedent included in a list  : [d].


next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : th.

Enter name of proof step : p.

Enter name of previous step : o.

Enter name of th formula : b & c.

Enter the antecedent included in a list : [b & c,e].

Enter the succedent included in a list  : [d].


next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : cut.

Enter name of proof step : q.

Enter name of LHS of previous steps : n.

Enter name of RHS of previous steps : p.

Enter name of cut formula : b & c.

Enter the antecedent included in a list : [a,e].

Enter the succedent included in a list  : [d].


next step
Enter name of the applied rule,axiom,id(a) or write end if
the proof has been finished : end.

the inserted proof has been checked and is correct .

yes
| ?-
```

In the following two examples, we use the syntax of the proof-step in $2 and the predicate ' prove(X) ' to check and assert derivations for further use :

Example 2

Let h be an arrow in X(a,c) and a,b,c be objects of a category X, the following derivation of the class of derivations in the free cartesian closed category over X :

```
       id(a)                    id(b)                        h
    A ------> A             B ------> B                A ------> C
 ---------------         ----------------           ---------------
 A,B ------> A           A,B ------> B              A,B ------> C
 ------------------------------------------        ------------------
        A,B ------> A ↑ B                          A ↑ B ------> C
        ----------------------------------------------------------
                           A,B ------> C
                       ------------------------
                       B ------> (A => C)
```

is interpreted by means of an arrow of the cartesian closed category as follows :

```
yes
| ?- theory(cartcl).

yes
| ?- prove(s1 = [id(a), [a] => [a]]).

yes
| ?- prove(s2 = [th(s1,b), [a,b] => [a]]).

yes
| ?- prove(s3 = [id(b), [b] => [b]]).

yes
| ?- prove(s4 = [th(s3,a), [a,b] => [b]]).

yes
| ?- prove(s5 = [ais(s2,s4,a&b), [a,b] => [a & b]]).

yes
| ?- prove(s6 = [h, [a] => [c]]).

yes
| ?- prove(s7 = [th(s6,b), [a,b] => [c]]).

yes
| ?- prove(s8 = [ala(s7,a & b), [a & b] => [c]]).

yes
| ?- prove(s9 = [cut(s5,s8,a & b), [a,b] => [c]]).

yes
| ?- prove(s = [his(s9,a -> c), [b] => [a -> c]]).

yes
| ?- interpr(s,K).

K = comp(hook(id(a),comp(comp(h,pi_left(a,b)),
                         comp(product(pi_left(a,b),
                                      pi_right(a,b)),
                              delta(a#b)))),
         ita(a,b))

yes
| ?-
```

Example 3

For arrows f in X(x,a), g in X(a,c), h in X(a,d), i in X(b,c) and j in X(b,d) and the composition of g,f is k1 and the composition of h,f is k2 in a category X, the following derivation from the theory of bicartesian categories :

```
      f
   x ---> a
----------------
x,y ---> a
------------------       g              h                    i             j
x,y ---> a,b          a ---> c      a ---> d            b ---> c      b ---> d
---------------------  -------------------------------  --------------------------------
x,y ---> a,b,v              a ---> c ↑ d                       b ---> c ↑ d
-----------------------     ---------------------------------------------------------------
x,y ---> a v b,v                              a v b ---> c ↑ d
-------------------------------------------------------------------------
                      x,y ---> c ↑ d,v
```

is transformed to its equivalent normal form as follows :

```
yes
| ?- theory(bicart).
yes
| ?- composite(g,f,k1).
yes
| ?- composite(h,f,k2).
yes
| ?- prove(l1 = [f, [x] => [a]]).
yes
| ?- prove(l2 = [th(l1,y), [x,y] => [a]]).
yes
| ?- prove(l3 = [th(l2,b), [x,y] => [a,b]]).
yes
| ?- prove(l4 = [th(l3,v), [x,y] => [a,b,v]]).
yes
| ?- prove(l5 = [ois(l4,a#b), [x,y] => [a#b,v]]).
yes
| ?- prove(l6 = [g, [a] => [c]]).
yes
| ?- prove(l7 = [h, [a] => [d]]).
yes
| ?- prove(l8 = [ais(l6,l7,c & d), [a] => [c & d]]).
yes
| ?- prove(l9 = [i, [b] => [c]]).
yes
| ?- prove(l10 = [j, [b] => [d]]).
yes
| ?- prove(l11 = [ais(l9,l10,c & d), [b] => [c & d]]).
yes
| ?- prove(l12 = [oia(l8,l11,a#b), [a#b] => [c&d]]).
yes
| ?- prove(l13 = [cut(l5,l12,a#b), [x,y] => [c&d,v]]).
yes
| ?- normal(l13).

Proof is as follows :
=====================

l1=[f,[x]=>[a]]
l2=[th(l1,y),[x,y]=>[a]]
l3=[th(l2,b),[x,y]=>[a,b]]
l4=[th(l3,v),[x,y]=>[a,b,v]]
l5=[ois(l4,a#b),[x,y]=>[a#b,v]]
l6=[g,[a]=>[c]]
l7=[h,[a]=>[d]]
l8=[ais(l6,l7,c&d),[a]=>[c&d]]
l9=[i,[b]=>[c]]
```

```
l10=[],[b]=>[d]]
l11=[als(l9,l10,c&d),[b]=>[c&d]]
l12=[ola(l8,l11,a#b),[a#b]=>[c&d]]
l13=[cut(l5,l12,a#b),[x,y]=>[c&d,v]]

Cut_free proof is :
===================

f22=[k1,[x]=>[c]]
f47=[th(f22,y),[x,y]=>[c]]
f46=[th(f47,c),[x,y]=>[c,c]]
f40=[th(f46,v),[x,y]=>[c,c,v]]
f31=[k2,[x]=>[d]]
f51=[th(f31,y),[x,y]=>[d]]
f50=[th(f51,c),[x,y]=>[d,c]]
f43=[th(f50,v),[x,y]=>[d,c,v]]
f34=[als(f40,f43,c&d),[x,y]=>[c&d,c,v]]
f22=[k1,[x]=>[c]]
f61=[th(f22,y),[x,y]=>[c]]
f60=[th(f61,d),[x,y]=>[c,d]]
f54=[th(f60,v),[x,y]=>[c,d,v]]
f31=[k2,[x]=>[d]]
f65=[th(f31,y),[x,y]=>[d]]
f64=[th(f65,d),[x,y]=>[d,d]]
f57=[th(f64,v),[x,y]=>[d,d,v]]
f37=[als(f54,f57,c&d),[x,y]=>[c&d,d,v]]
f5=[als(f34,f37,c&d),[x,y]=>[c&d,c&d,v]]
l13=[con(f5,c&d),[x,y]=>[c&d,v]]

Contraction and interchange free proof is :
===========================================

f22=[k1,[x]=>[c]]
f47=[th(f22,y),[x,y]=>[c]]
f46=[th(f47,v),[x,y]=>[c,v]]
f31=[k2,[x]=>[d]]
f65=[th(f31,y),[x,y]=>[d]]
f64=[th(f65,v),[x,y]=>[d,v]]
l13=[als(f46,f64,c&d),[x,y]=>[c&d,v]]

Normal proof is :
=================

f22=[k1,[x]=>[c]]
f47=[th(f22,v),[x]=>[c,v]]
f31=[k2,[x]=>[d]]
f65=[th(f31,v),[x]=>[d,v]]
f46=[als(f47,f65,c&d),[x]=>[c&d,v]]
l13=[th(f46,y),[x,y]=>[c&d,v]]


yes
| ?-
```

which in tree form is :

```
      k1                  k2
   x ---> c            x ---> d
   -----------         -----------
   x ---> c,v          x ---> d,v
   ------------------------------
          x ---> c ↑ d,v
        --------------------
        x,y ---> c ↑ d,v
```

(7.2) Category theory examples

In this subsection, we give some examples; in order to use the system to automate reasoning about category theory.

Example 1

For f in X(a,b), for some category X, the normal representation of the arrow :

h = comp(comp(pi_right_star(a,b),f),pi_left(a,b))

in the following diagram :

$$a \uparrow b \xrightarrow{\text{pi\_left(a,b)}} a \xrightarrow{f} b \xrightarrow{\text{pi\_right\_star(a,b)}} a \vee b$$

can be obtained as follows :

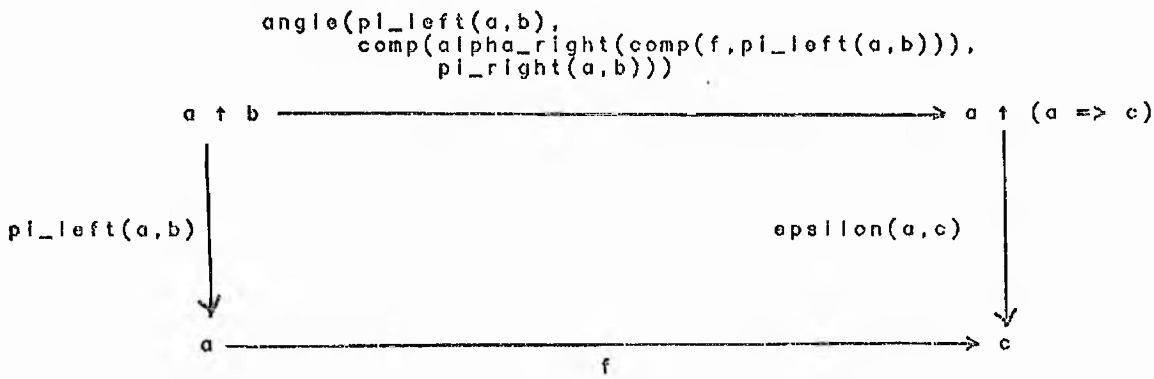
```
yes
| ?- theory(bicart).

yes
| ?- arrow(a,f,b).

yes
| ?- norm_rep(comp(comp(pi_right_star(a,b),f),
                    pi_left(a,b))).

The arrow is :
==============

comp(comp(pi_right_star(a,b),f),pi_left(a,b))

Proof is as follows :
=====================

f9=[id(a),[a]=>[a]]
f8=[th(f9,b),[a,b]=>[a]]
f7=[ala(f8,a&b),[a&b]=>[a]]
f6=[f,[a]=>[b]]
f5=[id(b),[b]=>[b]]
f4=[th(f5,a),[b]=>[a,b]]
f3=[ols(f4,a#b),[b]=>[a#b]]
f2=[cut(f6,f3,b),[a]=>[a#b]]
f1=[cut(f7,f2,a),[a&b]=>[a#b]]

Cut_free proof is :
===================

f27=[f,[a]=>[b]]
f24=[th(f27,b),[a,b]=>[b]]
f21=[ala(f24,a&b),[a&b]=>[b]]
f18=[th(f21,a),[a&b]=>[a,b]]
f1=[ols(f18,a#b),[a&b]=>[a#b]]

Contraction and interchange free proof is :
===========================================

f27=[f,[a]=>[b]]
f24=[th(f27,b),[a,b]=>[b]]
f21=[ala(f24,a&b),[a&b]=>[b]]
f18=[th(f21,a),[a&b]=>[a,b]]
f1=[ols(f18,a#b),[a&b]=>[a#b]]

Normal proof is :
=================

f27=[f,[a]=>[b]]
f24=[th(f27,a),[a]=>[a,b]]
f21=[ols(f24,a#b),[a]=>[a#b]]
f18=[th(f21,b),[a,b]=>[a#b]]
f1=[ala(f18,a&b),[a&b]=>[a#b]]

yes
| ?-
```

which in tree form is :

```
             f
        a ---> b
        ----------
        a ---> a,b
        ------------
        a ---> a v b
     ---------------
     a,b ---> a v b
  -------------------
  a † b ---> a v b
```

Example 2

Let f be an arrow in X(a,c), for some category X; the following diagram of the theory of cartesian closed categories :



commutes.

```
yes
| ?-
yes
| ?- theory(cartcl).

yes
| ?- arrow(a,f,c).

yes
| ?- commutative.

Insert the first arrow :
|: comp(epsilon(a,c),
|:       angle(pi_left(a,b),
|:             comp(alpha_right(comp(f,pi_left(a,b))),
|:                  pi_right(a,b)))).


The arrow is :
==============

comp(epsilon(a,c),
     angle(pi_left(a,b),
           comp(alpha_right(comp(f,pi_left(a,b))),
                pi_right(a,b))))
```

Proof is as follows :

```
f379=[id(a),[a]=>[a]]
f378=[th(f379,b),[a,b]=>[a]]
f377=[aia(f378,a&b),[a&b]=>[a]]
f395=[id(b),[b]=>[b]]
f394=[th(f395,a),[a,b]=>[b]]
f393=[aia(f394,a&b),[a&b]=>[b]]
f391=[id(a),[a]=>[a]]
f389=[th(f391,b),[a,b]=>[a]]
f392=[id(b),[b]=>[b]]
f390=[th(f392,a),[a,b]=>[b]]
f388=[ais(f389,f390,a&b),[a,b]=>[a&b]]
f386=[id(a),[a]=>[a]]
f385=[th(f386,b),[a,b]=>[a]]
f384=[aia(f385,a&b),[a&b]=>[a]]
f383=[f,[a]=>[c]]
f382=[cut(f384,f383,a),[a&b]=>[c]]
f387=[cut(f388,f382,a&b),[a,b]=>[c]]
f381=[his(f387,a->c),[b]=>[a->c]]
f380=[cut(f393,f381,b),[a&b]=>[a->c]]
f376=[ais(f377,f380,a&(a->c)),[a&b]=>[a&(a->c)]]
f374=[id(a),[a]=>[a]]
f375=[id(c),[c]=>[c]]
f373=[hia(f374,f375,a->c),[a,a->c]=>[c]]
f372=[aia(f373,a&(a->c)),[a&(a->c)]=>[c]]
f371=[cut(f376,f372,a&(a->c)),[a&b]=>[c]]
```

Cut_free proof is :

```
f582=[f,[a]=>[c]]
f579=[th(f582,b),[a,b]=>[c]]
f576=[aia(f579,a&b),[a&b]=>[c]]
f573=[th(f576,b),[a&b,b]=>[c]]
f570=[th(f573,a),[a&b,a,b]=>[c]]
f567=[aia(f570,a&b),[a&b,a&b]=>[c]]
f564=[th(f567,b),[a&b,b,a&b]=>[c]]
f561=[th(f564,a),[a&b,a,b,a&b]=>[c]]
f558=[aia(f561,a&b),[a&b,a&b,a&b]=>[c]]
f555=[th(f558,a),[a&b,a&b,a,a&b]=>[c]]
f552=[th(f555,b),[a&b,a&b,a,b,a&b]=>[c]]
f537=[aia(f552,a&b),[a&b,a&b,a&b,a&b]=>[c]]
f532=[con(f537,a&b),[a&b,a&b,a&b]=>[c]]
f474=[con(f532,a&b),[a&b,a&b]=>[c]]
f371=[con(f474,a&b),[a&b]=>[c]]
```

Normal proof is :

```
f582=[f,[a]=>[c]]
f579=[th(f582,b),[a,b]=>[c]]
f371=[aia(f579,a&b),[a&b]=>[c]]
```

Insert the second arrow : comp(f,pi_left(a,b)).

The arrow is :

comp(f,pi_left(a,b))

Proof is as follows :

```
f587=[id(a),[a]=>[a]]
f586=[th(f587,b),[a,b]=>[a]]
f585=[aia(f586,a&b),[a&b]=>[a]]
f584=[f,[a]=>[c]]
f583=[cut(f585,f584,a),[a&b]=>[c]]
```

Cut_free proof is :

```
f593=[f,[a]=>[c]]
f590=[th(f593,b),[a,b]=>[c]]
f583=[aia(f590,a&b),[a&b]=>[c]]
```

```
Normal proof is :
================

f593=[f,[a]=>[c]]
f590=[th(f593,b),[a,b]=>[c]]
f583=[ala(f590,a&b),[a&b]=>[c]]


The answer is :
=============

comp(epsilon(a,c),
     angle(pi_left(a,b),
            comp(alpha_right(comp(f,pi_left(a,b))),
                 pi_right(a,b))))
= comp(f,pi_left(a,b))

because they have the same normal representation .


yes
| ?-
```

Appendix 6

Predicate Index

Appendix 7

Maintenance Information

In this appendix, we give some information to help in maintaining the system. The system can be run by typing the command " szabo ", where szabo is a executable consisting of the line Prolog szabo.sav (together with instruction setting the heap size etc) . In order to added or change some rules in the system, edit the text " szabo.pro " using . Then invoke Prolog and compile the program using the directive:

```
|?- ['szabo.pro'].
```

which will instruct the Prolog-interpreter to consult the program. Then save the compiled form of the program into the file " szabo.sav ", by performing the command :

```
|?- save('szabo.sav').
```

Then exit from Prolog, the " szabo " command can now be used as before.

Appendix 8

Program Listing

```
:- op(30, fx, '\').
:- op(100, xfy, '#').
:- op(100, xfy, '&').
:- op(150, xfy, '->').
:- op(200, xfy, '=>').
:- op(150, xfy, '<->').
:- op(250, xfy, ':').

                           /* doc */

help :-
    write('Commands'),nl,
    write('========'),nl,nl,
    write('1-  theory          2-  composite 3-  prove'),nl,
    write('4-  proof           5-  interpr   6-  cut_free'),nl,
    write('7-  con_inc_free    8-  normal    9-  arrow '),nl,
    write('10- rep_of          11- norm_rep  12- commutative'),nl,
    write('13- inter_norm_rep  14- clean     15- new '), nl,nl,

    write('Help on any of these commands may be had by typing,'),
    nl,write('help_* where * is one of the above commands.'),nl.

help_theory :-
  write('theory(X) : Tells the system that the theory we intend'),
  nl, write(' to work with is X, where X represents one of the '),
  nl, write(' following abbreviations cart (cartesian), bicart
                                       ----               ------ '),
  nl, write(' (bicartesian), dbicart (distributive bicartesian),
                             -------  '),nl,
  write(' cartcl (cartesian closed) or bicartcl (bicartesian closed).
          ------                       --------  '),nl.

help_composite :-
  write('composite(F,G,H) : This predicate is used, when arrows'),
  nl, write(' F,G of the underlying category X with composition'),
  nl, write(' H are used in the construction of the derivation '),
  nl, write(' in operation. '),nl.

help_prove :-
  write('prove(X) : Checks the correctness of the current instance'),
  nl, write('of a proof-step X; if it is correct, it will be added'),
  nl, write(' to the database for further use. We keep using this'),
  nl, write('      command until the proof is complete.'),nl.

help_proof :-
  write('proof : This command is used, as another way to check the'),
  nl, write('correctness and assert derivations without using the'),
  nl, write('formal syntax for proof-step in $2 and the predicate'),
  nl, write('" prove(X) " above. It will switch on the system to'),
  nl, write('communicate with the user to assert the derivation'),
  nl, write('in informal way.'),nl.

help_interpr :-
  write('interpr(X,K) : After asserting a derivation, this command'),
  nl,write('is used to compute the interpretation of this derivation'),
  nl,write('as an arrow in the free category. X represents the name'),
  nl,write('of the last proof-step and  H  will be instantiated as'),
  nl,write('the arrow computed to interpret the derivation.'),nl.

help_cut_free :-
  write('cut_free(X) : Tells the system to start the processes'),
  nl, write('of transforming the derivation in question to its'),
  nl, write('equivalent cut-free one. X represents the name'),nl,
  write('     of the last proof-step.'),nl.
```

```
help_con_inc_free :-
  write('con_inc_free(X) : where X represents the name of the'),
  nl, write('last step of a cut-free proof. This predicate cause'),
  nl, write('the system to start the processes of transforming '),
  nl, write('the cut-free proof into its equivalent contraction &'),
  nl, write('interchange-free one. This is used only if we dealing'),
  nl, write('with the theories of cart, bicart or dbicart.
                                                         '),nl.

help_normal :-
  write('normal(X) :This predicate is a combination of three'),nl,
  write('processes: cut-elimination, contraction & interchange-'),nl,
  write('elimination and normalization; so that this general'),nl,
  write('command is used in place or cut_free and con_inc_free'),nl,
  write('to produce the equivalent normal form for a derivation.'),nl,
  write('X represents the name of the last step of the derivation.'),nl.

help_arrow :-
  write('arrow(A,F,B) : Insert all arrows, of the underling category'),
  nl, write('X, being used in the construction of the arrow of '),
  nl, write('the free category in question. F  represents an '),
  nl, write('arrow from A to B ( F : A -> B ).'),nl.

help_rep_of :-
  write('rep_of(X) : Will produce a representation of an arrow'),
  nl, write('X as a derivation.'),nl.

help_norm_rep :-
  write('norm_rep(X) : If X represents an arrow of some free'),nl,
  write('category, this predicate causes the system to produce '),
  nl, write('a representation of X by means of a derivation,'),nl,
  write('  then calling the processes of cut-elimination and'),nl,
  write('  normalization to produce the corresponding normal'),nl,
  write('  representation.'),nl.

help_commutative :-
  write('commutative : Tests whether two arrows are equal or not,'),
  nl, write('in order to test the commutativity of the diagrams '),
  nl, write('in category theory. After reading this command, the'),
  nl, write('system waits to read the  LHS  arrow of the equality'),
  nl, write('in question and responds by computing its normal'),nl,
  write('representation. The system waits again to read the'),nl,
  write('RHS arrow of the equality and responds by computing'),nl,
  write('its normal representation. Finally it compares these'),nl,
  write('two normal representations and responds with the'),nl,
  write('conclusion.'),nl.

help_inter_norm_rep :-
  write('inter_norm_rep(X): This is a combination of two processes;'),
  nl, write('one for normal representation of the arrow X, the other'),
  nl, write('is to interpret this normal representation to an arrow'),
  nl, write('of the same theory of the arrow X.'),nl.

help_clean :-
  write('clean : This command is to be used after each session of'),
  nl,write('use,in order to use the system using the same theory'),
  nl,write('and the same arrows,if any, of the underling category'),
  nl,write('for another session.'),nl.

help_new :-
  write('new : Cleans up the database from all the processes of'),
  nl, write('the last session and puts the system again to the top
           level.'),nl.

                   /* Proof Checker Program */

/* This program is to help the user to insert the proof in more
   detail rather than in the list form */

proof :-
     write('Enter name of the applied rule,axiom,id(a) or write'),
     nl, write('end if the proof has been finished : '),
     read(X),
    ((X = end,nl,
```

```
        write('the inserted proof has been checked and is correct.'),nl)
      ;(write('Enter name of proof step : '),
        read(Y),
        ((X =.. [id,A],
          prove(Y = [id(A), [A] => [A]]) )
        ;((nul(X,X1)
          ;unar(X,X1)
          ;inc(X,X1)
          ;binar(X,X1) ),
          write('Enter the antecedent included in a list : '),
          read(A),
          write('Enter the succedent included in a list  : '),
          read(S),
          prove(Y = [X1, A => S]) ) ),
        nl,nl, write('next step'),nl,
        proof) ).

nul(X,X1) :-
          X = axiom,
          write('Enter name of the arrow : '),
          read(Z),
          X1 = Z.

inc(X,X1) :-
      X = inc,
      write('Enter name of previous step :'),
      read(Y),
      write('Enter name of the first formula of the two
             interchangable formulae as appear in this step :'),
      read(L),
      write('Enter name of the second formula of the two
             interchangable formulae as appear in this step :'),
      read(M),
      X1 = inc(Y,L,M).

unar(X,X1) :-
        (X = th
        ;X = con
        ;X = aia
        ;X = ois
        ;X = his),
        write('Enter name of previous step :'),
        read(Y),
        write('Enter name of '), write(X), write(' formula : '),
        read(C),
        X1 =.. [X,Y,C].

binar(X,X1) :-
        (X = cut
        ;X = ais
        ;X = oia
        ;X = hia),
        write('Enter name of LHS of previous steps :'),
        read(Z1),
        write('Enter name of RHS of previous steps :'),
        read(Z2),
        write('Enter name of '), write(X), write(' formula : '),
        read(C),
        X1 =.. [X,Z1,Z2,C].

/* This is a proof checker program using the syntax of the
   proof-step */
theory(X) :-
             assert(thry(X)).

prove(X) :- X = ( Y = Z),
             rule(X),
             assert(proved(X)).

rule(X) :- X = ( L = [Y,[A] => [B]]),
           (Y =.. [_] ; Y =.. [id,_]),
            !.

rule(X) :- X = ( L = [true, [] => [tr]]),
            !.
```

```
rule(X) :- X = ( L = [bottom, [bo] => []]),
           !.

rule(X) :- X = ( L = [ th(M,A), G => T]),
           !,
           ((proved(M = [_, G1 => T]),
             append(Gamma, [A|Delta], G),
             append(Gamma, Delta, G1))
           ;(proved(M = [_, G => T1]),
             append(Phi, [A|Psi], T),
             append(Phi, Psi, T1),
             (thry(bicart); thry(dbicart); thry(bicartcl)))).

rule(X) :- X = ( L = [con(M,A), G => T]),
           !,
           ((proved(M = [_, G1 => T]),
             append(Gamma, [A|[A|Delta]], G1),
             append(Gamma, [A|Delta], G))
           ;(proved(M = [_, G => T1]),
             append(Phi, [A|[A|Psi]], T1),
             append(Phi, [A|Psi], T),
             (thry(bicart); thry(dbicart); thry(bicartcl)))).

rule(X) :- X = ( L = [inc(M,B,A), G => T]),
           !,
           ((proved(M = [_, G1 => T]),
             append(Gamma, [B|[A|Delta]], G),
             append(Gamma, [A|[B|Delta]], G1))
           ;(proved(M = [_, G => T1]),
             append(Phi, [B|[A|Psi]], T),
             append(Phi, [A|[B|Psi]], T1),
             (thry(bicart); thry(dbicart); thry(bicartcl)))).

rule(X) :- X = ( L = [cut(M,N,A), E => F]),
           !,
           proved(M = [_, Gamma => T1]),
           proved(N = [_, G1 => Theta]),
           append(Delta, [A|Lambda], G1),
           append3(Delta, Gamma, Lambda, G),
           append(Phi, [A|Psi], T1),
           append3(Phi, Theta, Psi, T),
        (((thry(cart) ; thry(cartcl) ; thry(bicart)),
          Phi = [], Psi = [])
        ;(thry(bicart), Delta = [], Lambda = [])
        ;thry(dbicart)
        ;thry(bicartcl)).

rule(X) :- X = ( L = [ais(M,N,A & B),G => T]),
           !,
           proved( M = [_,G => T1]),
           proved( N = [_, G => T2]),
           append(Phi, [A|Psi], T1),
           append(Phi, [B|Psi], T2),
           append(Phi, [A & B|Psi], T),
        (((thry(cart); thry(bicart); thry(cartcl)),
          Phi = [], Psi = [])
        ;thry(dbicart)
        ;thry(bicartcl)).

rule(X) :- X = ( L = [aia(M,A & B),G => T]),
           !,
           proved(M = [_,G1 => T]),
           append(Gamma, [A & B|Delta], G),
           append(Gamma, [A|[B|Delta]], G1).

rule(X) :- X = ( L = [oia(M,N,A # B), G => T]),
           !,
           proved( M = [_,G1 => T]),
           proved( N = [_,G2 => T]),
           append(Gamma, [A # B|Delta], G),
           append(Gamma, [A|Delta], G1),
           append(Gamma, [B|Delta], G2),
         ((thry(bicart), Gamma = [], Delta = [])
         ;thry(dbicart)
         ;thry(bicartcl)).
```

```
rule(X) :- X = ( L = [ois(M,A # B), G => T]),
            !,
            proved(M = [_, G => T1]),
            append(Phi, [A # B|Psi], T),
            append(Phi, [A|[B|Psi]], T1),
           (thry(bicart) ; thry(dbicart) ; thry(bicartcl)).

rule(X) :- X = ( L = [his(M,A -> B), G => [A -> B]]),
            !,
            proved( M = [_,G1 => [B]]),
            append(Gamma, [A|Delta], G1),
            append(Gamma, Delta, G),
           (thry(cartcl) ; thry(bicartcl)).

rule(X) :- X = ( L = [hia(M,N,A -> B), G => T]),
            !,
            proved( M = [_,Gamma => [A]]),
            proved( N = [_,G1 => T]),
            append(Gamma,[A -> B],G2),
            append(Delta, [B|Lambda], G1),
            append3(Delta, G2, Lambda, G),
          ((thry(cartcl), T = [_])
          ;thry(bicartcl)).

                                /* interpr */

/* this part is to interpret sequent calculus proofs as
   arrows of category theory */

inter_norm_rep(X) :-
          norm_rep(X),
          main(Y),
          interpr(Y,H),
          write('The irreducible arrow is :'), nl,
          write('=========================  '), nl, nl,
          write(H).

interpr(Z,H) :-
                 proved(Z = [F, [A] => [B]]),
                (F =.. [_] ; F =.. [id,A]),
               !,
                H = F.

interpr(Z,H) :-
                 proved(Z = [true, [] => [tr]]),
               !,
                H = id(tr).

interpr(Z,H) :-
       proved(Z = [cut(Z1,Z2,C), G => T]),
       proved(Z1 = [_, Gamma => T1]),
       proved(Z2 = [_, G1 => Theta]),
       append(Delta, [C|Lambda], G1),
       append3(Delta,Gamma,Lambda,G),
       append(Phi, [C|Psi], T1),
       append3(Phi, Theta, Psi, T),
     !,
       interpr(Z1,H1),
       interpr(Z2,H2),
       pi(Gamma, Gamma1),
       pi(Delta, Delta1),
       pi(Lambda, Lambda1),
       un(Phi, Phi1),
       un(Psi, Psi1),
       un(Theta, Theta1),
     ((Phi = [], Psi = [],
       (Delta = [], Lambda = [],
        H3 = comp(H2,H1) )
      ;(Delta = [], Lambda \== [],
        H3 = comp(H2, product(H1, id(Lambda1))) )
      ;(Lambda = [], Delta \== [],
        H3 = comp(H2, product(id(Delta1), H1)) )
      ;(Delta \== [], Lambda \== [],
        H3 = comp(H2,product(product(id(Delta1),H1),id(Lambda1)))))
     ;(Delta = [], Lambda = [],
```

```
        (Phi = [], Psi \== [],
         H3 = comp(union(H2, id(Psi1)), H1) )
       ;(Psi = [], Phi \== [],
         H3 = comp(union(id(Phi1), H2), H1) )
       ;(Phi \== [], Psi \== [],
         H3 = comp(union(union(id(Phi1), H2), id(Psi1)), H1)) )
     ;(Phi = [], Delta = [], Psi \== [], Lambda \== [],
       H3 = comp(comp(union(H2, pi_left(Psi1,Lambda1)),
                         delta_right(C, Psi1, Lambda1)),
                    product(H1, id(Lambda1))) )
     ;(Phi = [], Lambda = [], Psi \== [], Delta \== [],
       H3 = comp(comp(union(H2, pi_right(Delta1,Psi1)),
                         delta_left(Delta1, C, Psi1)),
                  product(id(Delta1), H1)) )
     ;(Psi = [], Delta = [], Phi \== [], Lambda \== [],
       H3 = comp(comp(union(pi_left(Phi1, Lambda1), H2),
                         delta_right(Phi1, C, Lambda1)),
                  product(H1, id(Lambda1))) )
     ;(Psi = [], Lambda = [], Phi \== [], Delta \== [],
       H3 = comp(comp(union(pi_right(Delta1,Phi1), H2),
                         delta_left(Delta1, Phi1, C)),
                  product(id(Delta1), H1)) )
     ;(Phi = [], Psi \== [], Delta \== [], Lambda \== [],
       H3 =comp(comp(comp(comp(union(id(Theta1),pi_right(Delta1,Psi1)),
                                union(H2,pi_left(Delta1&Psi1,Lambda1))),
                            delta_right(Delta1&C, Delta1&Psi1, Lambda1)),
                      product(delta_left(Delta1,C,Psi1),id(Lambda1))),
                 product(product(id(Delta1), H1), id(Lambda1))))
     ;(Psi = [], Phi \== [], Delta \== [], Lambda \== [],
       H3 =comp(comp(comp(comp(union(pi_right(Delta1,Phi1),id(Theta1)),
                                union(pi_left(Delta1&Phi1,Lambda1),H2)),
                            delta_right(Delta1 & Phi1,Delta1&C,Lambda1)),
                      product(delta_left(Delta1,Phi1,C), id(Lambda1))),
                 product(product(id(Delta1), H1), id(Lambda1))) )
     ;(Delta = [], Phi \== [], Psi \== [], Lambda \== [],
       H3=comp(comp(comp(union(union(pi_left(Phi1,Lambda1),H2),id(Psi1)),
                         union(delta_right(Phi1, C, Lambda1),
                               pi_left(Psi1, Lambda1))),
                     delta_right(Phi1 # C, Psi1, Lambda1)),
                product(H1, id(Lambda1))))
      ;(Lambda = [], Phi \== [], Psi \== [], Delta \== [],
       H3=comp(comp(comp(union(union(pi_right(Delta1,Phi1),H2),id(Psi1)),
                         union(delta_left(Delta1,Phi1,C),
                               pi_right(Delta1,Psi1))),
                     delta_left(Delta1, Phi1 # C, Psi1)),
                product(id(Delta1), H1)))
      ;(Phi \== [], Psi \== [], Delta \== [], Lambda \== [],
        H3 = comp(comp(union(union(id(Phi1), H2), id(Psi1)),
                       kappa(Delta1, Phi1, C, Psi1, Lambda1)),
                  product(product(id(Delta1), H1), id(Lambda1))))),
        rmcomp(H3,H).


interpr(Z,H) :-
         proved(Z = [th(Z1,A), G => T]),
         proved(Z1 = [_, G1 => T]),
         append(Gamma, [A|Delta], G),
         append(Gamma, Delta, G1),
       !,
         interpr(Z1,H1),
         pi(Gamma, Gamma1),
         pi(Delta, Delta1),
       ((Gamma = [],
         H2 = comp(H1, pi_right(A,Delta1)))
       ;(Delta = [],
         H2 = comp(H1, pi_left(Gamma1,A)))
       ; H2 = comp(H1, product(pi_left(Gamma1,A),id(Delta1)))),
         rmcomp(H2,H).


interpr(Z,H) :-
        proved(Z = [als(Z1,Z2,A & B), G => T]),
        proved(Z1 = [_, G => T1]),
        proved(Z2 = [_, G => T2]),
        append(Phi, [A & B|Psi], T),
        append(Phi, [A|Psi], T1),
        append(Phi, [B|Psi], T2),
      !,
        interpr(Z1,H1),
        interpr(Z2,H2),
        un(Phi, Phi1),
        un(Psi, Psi1),
        un(G, G1),
```

```
      ((Phi = [],
        Psi = [],
        H3 = comp(product(H1,H2),delta(G1)))
      ;(Phi = [],
        H3 = comp(comp(delta_right_star(A,B,Psi1),product(H1,H2)),
                   delta(G1)))
      ;(Psi = [],
        H3 = comp(comp(delta_left_star(Phi1,A,B),product(H1,H2)),
                   delta(G1)))
      ; H3 = comp(comp(comp(product(delta_left_star(Phi1,A,B),id(Psi1)),
                        delta_right_star(Phi1 # A, Phi1 # B, Psi1)),
                   product(H1,H2)),
              delta(G1))),
        rmcomp(H3,H).

interpr(Z,H) :-
          proved(Z = [con(Z1,A), G => T]),
          proved(Z1 = [_, G1 => T]),
          append(Gamma, [A|[A|Delta]], G1),
          append(Gamma, [A|Delta], G),
        !,
          interpr(Z1,H1),
          pi(Gamma, Gamma1),
          pi(Delta, Delta1),
        ((Gamma = [], Delta = [],
          H2 = comp(H1, delta(A)) )
        ;(Gamma = [],
          H2 = comp(H1, product(delta(A), id(Delta1))) )
        ;(Delta = [],
          H2 = comp(comp(H1, alpha(Gamma1, A, A)),
                    product(id(Gamma1), delta(A))) )
        ; H2=comp(comp(H1, product(alpha(Gamma1,A,A),id(Delta1))),
                 product(product(id(Gamma1),delta(A)),id(Delta1)))),
          rmcomp(H2,H).

interpr(Z,H) :-
          proved(Z = [aia(Z1,A & B), G => T]),
          proved(Z1 = [_, G1 => T]),
          append(Gamma, [A & B|Delta], G),
          append(Gamma, [A|[B|Delta]], G1),
        !,
          interpr(Z1,H1),
          pi(Gamma, Gamma1),
          pi(Delta, Delta1),
        ((Gamma = [],
          Delta = [],
          H = H1)
        ;(Gamma = [],
          H2 = comp(H, alpha_inverse(A,B,Delta1)))
        ;(Delta = [],
          H2 = comp(H1,alpha(Gamma1,A,B)))
        ; H2 = comp(H1,product(alpha(Gamma1,A,B),id(Delta1)))),
          rmcomp(H2,H).

interpr(Z,H) :-
                proved(Z = [bottom, [bo] => []]),
              !,
                H = id(bo).

interpr(Z,H) :-
          proved(Z = [th(Z1,A), G => T]),
          proved(Z1 = [_, G => T1]),
          append(Phi, [A|Psi], T),
          append(Phi, Psi, T1),
        !,
          interpr(Z1,H1),
          un(Phi, Phi1),
          un(Psi, Psi1),
        ((Phi = [],
          H2 = comp(pi_right_star(A,Psi1), H1))
        ;(Psi = [],
          H2 = comp(pi_left_star(Phi1,A), H1))
        ; H2 = comp(union(pi_left_star(Phi1,A), id(Psi1)), H1)),
          rmcomp(H2,H).

interpr(Z,H) :-
      proved(Z = [con(Z1,A), G => T]),
      proved(Z1 = [_, G => T1]),
```

```
        append(Phi, [A|[A|Psi]], T1),
        append(Phi, [A|Psi], T),
      !,
        interpr(Z1,H1),
        un(Phi, Phi1),
        un(Psi, Psi1),
      ((Phi = [],
        Psi = [],
        H2 = comp(delta_star(A), H1))
      ;(Phi = [],
        H2 = comp(union(delta_star(A), id(Psi1)), H1))
      ;(Psi = [],
        H2 = comp(comp(union(id(Phi1), delta_star(A)),
                      alpha_star_inverse(Phi1, A, A)),
                 H1))
      ; H2 = comp(comp(union(union(id(Phi1),delta_star(A)),
                                 id(Psi1)),
                       union(alpha_star_inverse(Phi1,A,A),id(Psi1))),
                 H1)),
        rmcomp(H2,H).

interpr(Z,H) :-
        proved(Z = [ola(Z1,Z2,A # B), G => T]),
        proved(Z1 = [_, G1 => T]),
        proved(Z2 = [_, G2 => T]),
        append(Gamma, [A # B|Delta], G),
        append(Gamma, [A|Delta], G1),
        append(Gamma, [B|Delta], G2),
      !,
        interpr(Z1,H1),
        interpr(Z2,H2),
        pi(Gamma,Gamma1),
        pi(Delta,Delta1),
        un(T,T1),
      ((Gamma = [],
        Delta = [],
        H3 = comp(delta_star(T1),union(H1,H2)))
      ;(Gamma = [],
        H3 = comp(comp(delta_star(T1),union(H1,H2)),
                  delta_right(A,B,Delta1)))
      ;(Delta = [],
        H3 = comp(comp(delta_star(T1),union(H1,H2)),
                  delta_left(Gamma1,A,B)))
      ; H3 = comp(comp(comp(delta_star(T1),union(H1,H2)),
                       delta_right(Gamma1 & A,Gamma1 & B,Delta1)),
                  product(delta_left(Gamma1,A,B),id(Delta1)))),
        rmcomp(H3,H).

interpr(Z,H) :-
        proved(Z = [ois(Z1,A # B), G => T]),
        proved(Z1 = [_, G => T1]),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A|[B|Psi]], T1),
      !,
        interpr(Z1,H1),
        un(Phi,Phi1),
        un(Psi,Psi1),
      ((Phi = [],
        Psi = [],
        H2 = H1)
      ;(Phi = [],
        H2 = H1)
      ;(Psi = [],
        H2 = comp(alpha_star_inverse(Phi1,A,B),H1))
      ; H2 = comp(union(alpha_star_inverse(Phi1,A,B),id(Psi1)),H1)),
        rmcomp(H2,H).

interpr(Z,H) :-
        proved(Z = [inc(Z1,B,A), G => T]),
        proved(Z1 = [_, G1 => T]),
        append(Gamma, [B|[A|Delta]], G),
        append(Gamma, [A|[B|Delta]], G1),
      !,
        interpr(Z1,H1),
        pi(Gamma, Gamma1),
        pi(Delta, Delta1),
      ((Gamma = [],
        Delta = [],
        H2 = comp(H1,sigma_inverse(A,B)))
      ;(Gamma = [],
```

```
        H2 = comp(H1,product(sigma_inverse(A,B),id(Delta1))))
      ;(Delta = [],
        H2 = comp(comp(comp(H1,alpha(Gamma1,A,B)),
                    product(id(Gamma1),sigma_inverse(A,B))),
                  alpha_inverse(Gamma1,B,A)))
      ; H2 = comp(comp(comp(H1,product(alpha(Gamma1,A,B),id(Delta1))),
                    product(product(id(Gamma1),sigma_inverse(A,B)),
                            id(Delta1))),
                  product(alpha_inverse(Gamma1,B,A),id(Delta1)))),
        rmcomp(H2,H).


interpr(Z,H) :-
        proved(Z = [inc(Z1,B,A), G => T]),
        proved(Z1 = [_, G => T1]),
        append(Phi, [B|[A|Psi]], T),
        append(Phi, [A|[B|Psi]], T1),
      !,
        interpr(Z1,H1),
        un(Phi, Phi1),
        un(Psi, Psi1),
      ((Phi = [], Psi = [],
        H2 = comp(sigma_star(A,B), H1) )
      ;(Phi = [],
        H2 = comp(union(sigma_star(A,B), id(Psi1)), H1) )
      ;(Psi = [],
        H2 = comp(comp(comp(alpha_star(Phi1,B,A),
                        union(id(Phi1), sigma_star(A,B))),
                    alpha_star_inverse(Phi1,A,B)),
                 H1) )
      ; H2 = comp(comp(comp(union(alpha_star(Phi1,B,A), id(Psi1)),
                        union(union(id(Phi1), sigma_star(A,B)),
                              id(Psi1))),
                    union(alpha_star_inverse(Phi1,A,B), id(Psi1))),
                 H1) ),
        rmcomp(H2,H).


interpr(Z,H) :-
       proved(Z = [hia(Z1,Z2,A -> B), G => T]),
       proved(Z1 = [_, Gamma => [A]]),
       proved(Z2 = [_, G1 => T]),
       append(Gamma, [A -> B], M),
       append(Delta, [B|Lambda], G1),
       append3(Delta, M, Lambda, G),
     !,
       interpr(Z1,H1),
       interpr(Z2,H2),
       pi(Gamma,Gamma1),
       pi(Delta,Delta1),
       pi(Lambda,Lambda1),
     ((Delta = [],
       Lambda = [],
       H3 = comp(comp(H2,epsilon(A,B)),
                 product(H1,id(A -> B))))
     ;(Delta = [],
       H3 = comp(comp(H2,product(epsilon(A,B),id(Lambda1))),
                 product(product(H1,id(A -> B)),id(Lambda1))))
     ;(Lambda = [],
       H3 = comp(comp(comp(H2,product(id(Delta1),epsilon(A,B))),
                      product(id(Delta1),product(H1,id(A -> B)))),
                 alpha_inverse(Delta1,Gamma1,A -> B)))
     ;H3=comp(comp(comp(H2,product(product(id(Delta1),epsilon(A,B)),
                                   id(Lambda1))),
                 product(product(product(id(Delta1),H1),id(A -> B)),
                         id(Lambda1))),
            product(alpha_inverse(Delta1,Gamma1,A->B),id(Lambda1)))),
       rmcomp(H3,H).


interpr(Z,H) :-
        proved(Z = [his(Z1,A -> B), G => [A -> B]]),
        proved(Z1 = [_, G1 => [B]]),
        append(Gamma, [A|Delta], G1),
        append(Gamma, Delta, G),
      !,
        interpr(Z1,H1),
        pi(Gamma,Gamma1),
        pi(Delta,Delta1),
      ((Gamma = [],
        H2 = comp(hook(id(A),H1),ita(A,Delta1)))
      ;(Delta = [],
        H2 = comp(comp(hook(id(A),H1),hook(id(A),sigma(A,Gamma1))),
```

```
                    lta(A,Gamma1)))
     ; H2 = comp(comp(comp(hook(id(A),H1),
                              hook(id(A),product(sigmao(A,Gamma1),
                                                  id(Delta1)))),
                         hook(id(A),alpha(A,Gamma1,Delta1))),
                    lta(A,Gamma1 & Delta1))),
     rmcomp(H2,H).


/* pi(Z,H) means transiting the list of elements Z into just
   one element using &, for example [a,b,c,d,e]  translates
   to a&(b&(c&(d&e))) */

pi([A],A).

pi(Z,L) :-
          append(Z1,[A],Z),
          Z1 \== [],
          pi(Z1,L1),
        ((L1 =..[_],
          L = L1 & A)
        ;(L1 =.. [_|_],
          L = (L1) & A )).

pi([],[]).


/* un the same like pi but using # instead of & */

un([A],A).

un(Z,L) :-
          append(Z1,[A],Z),
          Z1 \== [],
          un(Z1,L1),
        ((L1 =..[_],
          L = L1 # A)
        ;(L1 =.. [_|_],
          L = (L1) # A )).

un([],[]).


/* rmcomp(H1,H) means that if H1 = comp(id(a),f),we put H = f,
   or, if H1 = comp(g,id(a)), we put H = g,otherwise H = H1 */

rmcomp(H1,H) :-
               ((H1 = comp(id(A),F), H = F)
               ;(H1 = comp(G,id(B)), H = G)
               ;H = H1).


/* This command to test the equality of arrows */

commutative :-
         write('Insert the first arrow : '),
         read(X),nl,nl,nl,
         norm_rep(X),
         main(Z1),
         tree(Z1,L1),
         clean,
         write('Insert the second arrow : '),
         read(Y),nl,nl,nl,
         norm_rep(Y),
         main(Z2),
         tree(Z2,L2),
         write('The answer is :'),nl,
         write('=============  '),nl,nl,
       ((L1 == L2,
         write(X), write(' = '), nl,
         write(Y), nl, nl,
         write('because they have the same normal representation.'),
         nl,nl)
       ;(write(X) , nl,
         write('                         is not equal to'),nl,
         write(Y),nl,nl)).


tree(X,L) :-
             tree(X, [], L).
```

```
tree(X, Sofar, K) :-
                  proved(X = [Y,Z]),
               (((Y =.. [_] ; Y =.. [id,_]),
                 append(Sofar, [Y:Z], K))
               ;(Y =.. [th,L1,A],
                 tree(L1, Sofar, K1),
                 append(K1, [th(A) : Z], K))
                ;(Y =.. [con,L1,A],
                  tree(L1, Sofar, K1),
                  append(K1, [con(A) : Z], K))
               ;(Y =.. [aia,L1,A],
                 tree(L1, Sofar, K1),
                 append(K1, [aia(A) : Z], K))
               ;(Y =.. [ois,L1,A],
                 tree(L1, Sofar, K1),
                 append(K1, [ois(A) : Z], K))
               ;(Y =.. [his,L1,A],
                 tree(L1, Sofar, K1),
                 append(K1, [his(A) : Z], K))
               ;(Y =.. [inc,L1,B,A],
                 tree(L1, Sofar, K1),
                 append(K1, [inc(B,A) : Z], K))
               ;(Y =.. [ais,L1,L2,A],
                 tree(L1, Sofar, K1),
                 tree(L2, K1, K2),
                 append(K2, [ais(A) : Z], K))
               ;(Y =.. [oia,L1,L2,A],
                 tree(L1, Sofar, K1),
                 tree(L2, K1, K2),
                 append(K2, [oia(A) : Z], K))
               ;(Y =.. [hia,L1,L2,A],
                 tree(L1, Sofar, K1),
                 tree(L2, K1, K2),
                 append(K2, [hia(A) : Z], K))).

/* This is a program to produce representations of arrows of the
   free (cartesian) categories by means of Gentezen derivations */

norm_rep(Y) :-
       write('The arrow is :'), nl,
       write('============  '), nl, nl,
       assert(inserted(Y)),   /* this needed only to tell the
                                 system when the processes of the
                                 normal start that we are working
                                 on category theory and don't satisfy
                                 'cop', since there is no need */
       write(Y), nl,nl,
       gensym(f,Z),
     ((Y =.. [X], Y1 = [ar,X])
     ; Y =.. Y1),
       append(Y1,[Z],L1),
       L =.. L1,
       L,
       normal(Z).


new :- clean,
        abolish(arr,3),
        abolish(composition,3),
        abolish(thry,1).


clean :- abolish(proved,1),
          abolish(main,1),
          abolish(inserted,1).


rep_of(Y) :-
     write('The arrow is :'), nl,
     write('============  '), nl, nl,
     write(Y), nl,nl,
     gensym(f,Z),
     Y =.. Y1,
     append(Y1,[Z],L1),
     L =.. L1,
     L,
     write('Representation :-'),nl,
     write('================   '),nl,nl,
     printall(Z), nl.
```

```
/* arrow(A,F,S) means  F : A -------> S */
arrow(A,F,B) :-
                assert(arr(A,F,B)).


composite(G,F,H) :-
                assert(composition(G,F,H)).


ar(F,Z) :- arr(A,F,B),
           assert(proved(Z = [F, [A] => [B]])).


/* id(A)  : A    -------------> A  */
id(K,X) :-
     ((K =.. [true], assert(proved(X = [true, [] => [tr]])))
     ;(K =.. [bottom],assert(proved(X = [bottom,[bo] => []])))
     ;(K =.. [A],
       assert(proved(X = [id(A), [A] =>`[A]])))
     ;(K =.. [&,A,B],
       gensym(f,Z1),
       assert(proved(X = [aia(Z1,A & B),[A & B] => [A & B]])),
       gensym(f,Z2),
       gensym(f,Z3),
       assert(proved(Z1 = [ais(Z2,Z3,A & B),[A,B] => [A & B]])),
       gensym(f,Z4),
       assert(proved(Z2 = [th(Z4,B), [A,B] => [A]])),
       gensym(f,Z5),
       assert(proved(Z3 = [th(Z5,A), [A,B] => [B]])),
       id(A,Z4),
       id(B,Z5))
     ;(K =.. [#,A,B],
       gensym(f,Z1),
       assert(proved(X = [ois(Z1,A # B),[A # B] => [A # B]])),
       gensym(f,Z2),
       gensym(f,Z3),
       assert(proved(Z1 = [ola(Z2,Z3,A # B),[A # B] => [A,B]])),
       gensym(f,Z4),
       assert(proved(Z2 = [th(Z4,B), [A] => [A,B]])),
       gensym(f,Z5),
       assert(proved(Z3 = [th(Z5,A), [B] => [A,B]])),
       id(A,Z4),
       id(B,Z5))
     ;(K =.. [->,A,B],
       gensym(f,Z1),
       assert(proved(X = [his(Z1,A -> B),[A -> B] => [A -> B]])),
       gensym(f,Z2),
       gensym(f,Z3),
       assert(proved(Z1 = [hia(Z2,Z3,A -> B),[A,A -> B] => [B]])),
       id(A,Z2),
       id(B,Z3))).


/* Canonical Arrows for Fc(X) */
/* ============================ */

/* tau(A)  :  A  ---------------> T    where T is true  */
tau(A,X) :-
          gensym(f,F1),
          assert(proved(F1 = [true, [] => [tr]])),
          assert(proved(X = [th(F1,A), [A] => [tr]])).


/*  pi_left(A,B)  :  A & B  ----------> A         */
pi_left(A,B,X) :-
        gensym(f,F1),
        assert(proved(X = [aia(F1,A & B), [A & B] => [A]])),
        gensym(f,F2),
        assert(proved(F1 = [th(F2,B), [A,B] => [A]])),
        id(A,F2).


/* pi_right(A,B)  :  A & B  ------------->  B       */
pi_right(A,B,X) :-
        gensym(f,F1),
        assert(proved(X = [aia(F1,A & B), [A & B] => [B]])),
        gensym(f,F2),
        assert(proved(F1 = [th(F2,A), [A,B] => [B]])),
        id(B,F2).
```

```
/*  comp(V,U)  :  A  ---------------->  D   where
     U  :   A  ---------> B  ,  V :  B  -------->  D    */

comp(V,U,Z) :-
     part(V,Z2),
     part(U,Z1),
     proved(Z1 = [_, A => [C]]),
     proved(Z2 = [_, [C] => S]),
     assert(proved(Z = [cut(Z1,Z2,C), A => S])).


/* angle(U,V) = <U,V>  : A  ----------> B & C     where
   U : A  -------------> B  ,   V  : A  ---------------> C     */

angle(U,V,Z) :-
     part(U,Z1),
     part(V,Z2),
     proved(Z1 = [_, A => [B]]),
     proved(Z2 = [_, A => [C]]),
     assert(proved(Z = [ais(Z1,Z2,B & C), A => [B & C]])).


/* delta(A) : A ------------> A & A  */

delta(A,Z) :-
     gensym(f,F1),
     gensym(f,F2),
     assert(proved(Z = [ais(F1,F2,A & A), [A] => [A & A]])),
     id(A,F1),
     id(A,F2).


/* alpha(A,B,C) :  A & (B & C)  -------> (A & B) & C   */
alpha(A,B,C,Z) :-
     angle(angle(pi_left(A,B&C),comp(pi_left(B,C),pi_right(A,B&C))),
            comp(pi_right(B,C),pi_right(A,B&C)),Z).


/* alpha_inverse(A,B,C) : (A & B) & C  ---> A & (B & C)  */
alpha_inverse(A,B,C,Z) :-
     angle(comp(pi_left(A,B),pi_left(A&B,C)),
            angle(comp(pi_right(A,B),pi_left(A&B,C)),pi_right(A&B,C)),
            Z).


/* product(U,V) = U & V  : A & B  ---------> C & D   where
   U : A  -----------> C   ,   V : B  ------> D          */

product(U,V,Z) :-
     part(U,Z1),
     part(V,Z2),
     proved(Z1 = [_, [A] => [C]]),
     proved(Z2 = [_, [B] => [D]]),
     part(pi_left(A,B),Z3),
     gensym(f,Z4),
     assert(proved(Z4 = [cut(Z3,Z1,A), [A & B] => [C]])),
     part(pi_right(A,B),Z5),
     gensym(f,Z6),
     assert(proved(Z6 = [cut(Z5,Z2,B), [A & B] => [D]])),
     assert(proved(Z = [ais(Z4,Z6,C & D),[A & B] => [C & D]])).


/* sigma(A,B) : A & B  ------------> B & A     */
sigma(A,B,Z) :-
               angle(pi_right(A,B),pi_left(A,B), Z).


/* sigma_inverse(A,B)  :  B & A  ---------------> A & B   */
sigma_inverse(A,B,Z) :-
                angle(pi_right(B,A),pi_left(B,A), Z).


/* Canonical Arrows for Fbc(X) */
/* ============================ */

/* tau_star(A)  :  bottom  -------------> A         */
tau_star(A,X) :-
     gensym(f,F1),
     assert(proved(F1 = [bottom, [bo] => []])),
     assert(proved(X = [th(F1,A), [bo] => [A]])).
```

```
/* pi_left_star(A,B)   : A   ---------------->   A # B        */
pi_left_star(A,B,X) :-
      gensym(f,F1),
      assert(proved(X = [ois(F1,A # B), [A] => [A # B]])),
      gensym(f,F2),
      assert(proved(F1 = [th(F2,B), [A] => [A,B]])),
      id(A,F2).

/* pi_right_star(A,B)   : B   ---------------->  A # B      */
pi_right_star(A,B,X) :-
      gensym(f,F1),
      assert(proved(X = [ois(F1,A # B), [B] => [A # B]])),
      gensym(f,F2),
      assert(proved(F1 = [th(F2,A), [B] => [A,B]])),
      id(B,F2).


/* delta_star(A) : A # A ------------> A   */
delta_star(A,Z) :-
      gensym(f,F1),
      gensym(f,F2),
      assert(proved(Z = [oia(F1,F2,A # A), [A # A] => [A]])),
      id(A,F1),
      id(A,F2).


/* square(U,V) = [U,V] :   A # B   ------------> C     where
     U  : A   ----------> C      ,    V : B -------> C         */
square(U,V,Z) :-
      part(U,Z1),
      part(V,Z2),
      proved(Z1 = [_, [A] => S]),
      proved(Z2 = [_, [B] => S]),
      assert(proved(Z = [oia(Z1,Z2,A # B), [A # B] => S])).


/* union(U,V)   = U # V   : A # C   -----------> B # D .   where
     U  : A   -----------> B      ,    V : C   ----------> D        */
union(U,V,Z) :-
       part(U,Z1),
       part(V,Z2),
       proved(Z1 = [_, [A] => [B]]),
       proved(Z2 = [_, [C] => [D]]),
       part(pi_left_star(B,D),Z3),
       gensym(f,Z4),
       assert(proved(Z4 = [cut(Z1,Z3,B), [A] => [B # D]])),
       part(pi_right_star(B,D),Z5),
       gensym(f,Z6),
       assert(proved(Z6 = [cut(Z2,Z5,D), [C] => [B # D]])),
       assert(proved(Z = [oia(Z4,Z6,A # C),[A # C] => [B # D]])).


/* alpha_star(A,B,C) : A # (B # C)  ----->  (A # B) # C    */
alpha_star(A,B,C,Z) :-
      square(comp(pi_left_star(A#B,C),pi_left_star(A,B)),
      square(comp(pi_left_star(A#B,C),pi_right_star(A,B)),
              pi_right_star(A#B,C)), Z).


/* alpha_star_inverse(A,B,C) : (A # B) # C  -> A # (B # C) */
alpha_star_inverse(A,B,C,Z) :-
      square(square(pi_left_star(A,B#C),comp(pi_right_star(A,B#C),
                                        pi_left_star(B,C))),
       comp(pi_right_star(A,B#C),pi_right_star(B,C)), Z).


/* sigma_star(A,B)   : A # B   ---------------> B # A      */
sigma_star(A,B,Z) :-
      square(pi_right_star(B,A),pi_left_star(B,A), Z).


/* sigma_star_inverse(A,B)   : B # A   ----------> A # B   */
sigma_star_inverse(A,B,Z) :-
      square(pi_right_star(A,B),pi_left_star(A,B), Z).
```

```
/* Canonical Arrows for Fdbc(X) */
/* ============================== */

/* round(U,V) = (U,V) : A & (B # C)  -------> D     where
   U  : A & B  -------> D   ,  V  : A & C  -------> D    */
round(U,V,Z) :-
     part(U,Z1),
     part(V,Z2),
     proved(Z1 = [_, [A & B] => S]),
     proved(Z2 = [_, [A & C] => S]),
     gensym(f,F1),
     assert(proved(Z = [ala(F1,A & (B # C)),[A & (B # C)] => S])),
     gensym(f,F2),
     gensym(f,F3),
     assert(proved(F1 = [ola(F2,F3,B # C), [A,B # C] => S])),
     gensym(f,F4),
     assert(proved(F2 = [cut(F4,Z1,A & B), [A,B] => S])),
     gensym(f,F5),
     assert(proved(F3 = [cut(F5,Z2,A & C), [A,C] => S])),
     gensym(f,F6),
     gensym(f,F7),
     assert(proved(F4 = [ais(F6,F7,A & B), [A,B] => [A & B]])),
     gensym(f,F8),
     gensym(f,F9),
     assert(proved(F5 = [ais(F8,F9,A & C), [A,C] => [A & C]])),
     gensym(f,F10),
     assert(proved(F6 = [th(F10,B), [A,B] => [A]])),
     gensym(f,F11),
     assert(proved(F7 = [th(F11,A), [A,B] => [B]])),
     gensym(f,F12),
     assert(proved(F8 = [th(F12,C), [A,C] => [A]])),
     gensym(f,F13),
     assert(proved(F9 = [th(F13,A), [A,C] => [C]])),
     id(A,F10),
     id(B,F11),
     id(A,F12),
     id(C,F13).


/* delta_left(A,B,C) : A & (B # C) -> (A & B) # (A & C) */
delta_left(A,B,C,Z) :-
     round(pi_left_star(A&B,A&C),pi_right_star(A&B,A&C),Z).


/* delta_right(A,B,C) : (A # B) & C -> (A & C) # (B & C) */
delta_right(A,B,C,Z) :-
     comp(union(sigma(C,A),sigma(C,B)),comp(delta_left(C,A,B),
                                      sigma(A#B)), Z).


/* delta_left_star(A,B,C) : (A # B) & (A # C) -> A # (B & C) */
delta_left_star(A,B,C,Z) :-
     comp(comp(comp(comp(comp(delta_star(A#(B&C)),
                          union(id(A#(B&C)),union(pi_right(B,A),
                                                  id(B&C)))),
                   union(id(A#(B&C)),delta_left(B,A,C))),
            union(pi_left_star(A,B&C),id(B&(A#C)))),
        union(pi_left(A,A#C),id(B&(A#C)))),
    delta_right(A,B,A#C),
    Z).


/* delta_right_star(A,B,C) : (A # C) & (B # C) -> (A & B) # C */
delta_right_star(A,B,C,Z) :-
     comp(comp(sigma_star(C,A & B), delta_left_star(C,A,B)),
    product(sigma_star(A,C),sigma_star(B,C)),
    Z).


/* kappa(A,B,C,D,E) :(A&((B # C) # D))&E -> (B#((A&C)&E))#D */
kappa(A,B,C,D,E,Z) :-
   comp(comp(comp(comp(comp(union(union(pi_right(A,B),id((A&C)&E)),
                               pi_right(A,D)),
                        union(union(pi_left(A&B,E),id((A&C)&E)),
                              pi_left(A&D,E))),
                 union(delta_right(A&B,A&C,E),id((A&D)&E))),
          delta_right((A&B) # (A&C),A&D,E)),
      product(union(delta_left(A,B,C),id(A & D)),id(E))),
  product(delta_left(A,B # C,D),id(E)),
  Z).
```

```
/* Canonical arrows in Fccl(X) :- */
/* ================================== */

/* epsilon_left(U,V) : A & (B -> C)  -------> D    where
     U  : A   -------> B        ,     V : C   -------> D       */
epsilon_left(U,V,Z) :-
        part(U,Z1),
        part(V,Z2),
        proved(Z1 = [_, [A] => [B]]),
        proved(Z2 = [_, [C] => S]),
        gensym(f,F1),
        assert(proved(Z = [ala(F1,A&(B -> C)),[A&(B -> C)] => S])),
        assert(proved(F1 = [hia(Z1,Z2,B -> C),[A,B -> C] => S])).


/* alpha_right(U) :  B  ---------------> (A -> C)   where
     U  : A & B  ------------> C     */
alpha_right(U,Z) :-
        part(U,Z1),
        proved(Z1 = [_, [A & B] => [C]]),
        gensym(f,F1),
        assert(proved(Z = [his(F1,A -> C), [B] => [A -> C]])),
        gensym(f,F2),
        assert(proved(F1 = [cut(F2,Z1,A & B), [A,B] => [C]])),
        gensym(f,F3),
        gensym(f,F4),
        assert(proved(F2 = [als(F3,F4,A & B), [A,B] => [A & B]])),
        gensym(f,F5),
        assert(proved(F3 = [th(F5,B), [A,B] => [A]])),
        gensym(f,F6),
        assert(proved(F4 = [th(F6,A), [A,B] => [B]])),
        id(A,F5),
        id(B,F6).


/* epsilon(A,B) : A & (A => B) ---------------> B */
epsilon(A,B,Z) :-
      gensym(f,F1),
      assert(proved(Z = [ala(F1,A&(A -> B)),[A&(A -> B)] => [B]])),
      gensym(f,F2),
      gensym(f,F3),
      assert(proved(F1 = [hia(F2,F3,A -> B), [A,A -> B] => [B]])),
      id(A,F2),
      id(B,F3).


/* ita(A,B) : B -------------> A => (A & B)       */
ita(A,B,Z) :-
      gensym(f,F1),
      assert(proved(Z = [his(F1,A -> (A&B)),[B] => [A -> (A&B)]])),
      gensym(f,F2),
      gensym(f,F3),
      assert(proved(F1 = [ais(F2,F3,A & B), [A,B] => [A & B]])),
      gensym(f,F4),
      assert(proved(F2 = [th(F4,B), [A,B] => [A]])),
      gensym(f,F5),
      assert(proved(F3 = [th(F5,A), [A,B] => [B]])),
      id(A,F4),
      id(B,F5).


/* hook(U,V) = (U => V) : (B => C) -----> (A => D)  where
   U : A --------> B   ,   V : C --------> D        */
hook(U,V,Z) :-
       alpha_right(epsilon_left(U,V), Z).


part(X,Z) :-
        gensym(f,Z),
      ((X =.. [X1], ar(X,Z))
      ;(X =.. X1,
        append(X1,[Z],L1),
        L =.. L1,
        L)).
```

```
                /* Cut-Elimination Program */

normal(Z) :-
          cut_free(Z),
        (((thry(cartcl) ; thry(bicartcl)),
          hook_terminal)
        ; true),
          term_intl,
          expth,
          put_in_list(Z,L),
          abolish(proved,1),
          ass_list(L),
          normal1,
          normal2,
       (((thry(cartcl) ; thry(bicartcl)),
          assert(second),
          normal1,
          normal2,
          retract(second) )
        ; true ).

cut_free(Z) :-
        ( hook_cut
        ;((inserted(X) ; cop),
          write('Proof is as follows :'), nl,
          write('===================='), nl, printall(Z),
          nl, nl, nl, assert(main(Z)))),
          repeat,
              one_step,
              not(proved(_ = [cut(_,_,_), _ => _])),
        !,
        ( hook_cut
        ;(write('Cut_free proof is :'), nl,
          write('================='), nl, nl,
          printall(Z), nl, nl)).

one_step :-
        proved(Z = [cut(Z1,Z2,C), G => T]),
        proved(Z1 = [K1,_]), K1 =.. [H1|_], H1 \== cut,
        proved(Z2 = [K2,_]), K2 =.. [H2|_], H2 \== cut,
        cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])),
        expcon,
        expinc,
        expth,
        purge,
 !.
one_step.    /* must succeed if no cuts in proof */

purge :-   /* purges unwanted proof steps */
        main(Y),
        assertz(proved(mark)),
        reassert_all(Y),
        repeat,
            retract(proved(Z)),
            Z = mark,    /* retracts all upto mark */
      !.

reassert_all(X) :-
        proved(X = Y),
        ( nullary(X)
        ;(unary(X,L), reassert_all(L))
        ;(binary(X,L,M), reassert_all(L), reassert_all(M))),
        retract(proved(X = Y)),
        assertz(proved(X = Y)).

/* (A1,A1) - (C.1) */
cut_elm(proved(Z = [cut(Z1,Z2,B), G => T])) :-
         proved(Z1 = [L, G => [B]]),
         proved(Z2 = [M, [B] => T]),
         (L =.. [X] ; L =.. [id,B]),
         (M =.. [Y] ; M =.. [id,B]),
      !,
         ((L =.. [id,B], N = M)
         ;(M =.. [id,B], N = L)
         ;composition(M,L,N)),
         assert(proved(Z = [N, G => T])),
         retract(proved(Z = [cut(Z1,Z2,B), G => T])).
```

```
/* (A1,R2) - (C.2.1) */
cut_elm(proved(Z = [cut(Z1,Z2,B), G => T])) :-
              proved(Z1 = [K1,[A]=>[B]]),
              (K1 =.. [X] ; K1 =.. [id,A]),
              proved(Z2 = [th(Z3,B),G1=>T]),
              proved(Z3 = [K,G3=>T]),
              append(Gamma, [A|Delta], G),
              append(Gamma, [B|Delta], G1),
              append(Gamma, Delta, G3),
           !,
              gensym(f,F1),
              assert(proved(F1 = [K,G3 => T])),
              assert(proved(Z = [th(F1,A),G => T])),
              retract(proved(Z = [cut(Z1,Z2,B), G => T])).


/* (A3,R2) - (C.2.3) */
cut_elm(proved(Z = [cut(Z1,Z2,tr), G => T])) :-
              proved(Z1 = [true,[]=>[tr]]),
              proved(Z2 = [th(Z3,tr),G1=>T]),
              proved(Z3 = [K,G=>T]),
           !,
              assert(proved(Z = [K,G=>T])),
              retract(proved(Z = [cut(Z1,Z2,tr),G=>T])).

/* (R10,R2) - (C.3) */
cut_elm(proved(Z = [cut(Z1,Z2,A & B), G => T])) :-
              proved(Z1 = [ais(Z3,Z4,A & B), G1 => T1]),
              proved(Z2 = [th(Z5,A & B), G2 => T2]),
              proved(Z3 = [K1,G1=>T3]),
              proved(Z4 = [K2,G1=>T4]),
              proved(Z5 = [K3,G3=>T2]),
              append(Phi, [A & B|Psi], T1),
              append(Phi, [A|Psi], T3),
              append(Phi, [B|Psi], T4),
              append3(Phi, T2, Psi, T),
              append(Delta, [A & B|Lambda], G2),
              append(Delta, Lambda, G3),
              append3(Delta, G1, Lambda, G),
           !,
              gensym(f,F1),
              assert(proved(F1 = [K3,G3=>T2])),
              append(Delta, [B|Lambda], G4),
              gensym(f,F2),
              assert(proved( F2 = [th(F1,B),G4 => T2])),
              append(Delta, [A|[B|Lambda]], G5),
              gensym(f,F3),
              assert(proved( F3 = [th(F2,A),G5 => T2])),
              gensym(f,F4),
              assert(proved(F4 = [K2,G1=>T4])),
              append3(Delta, [A|G1], Lambda, G6),
              gensym(f,F5),
              assert(proved( F5 = [cut(F4,F3,B),G6 => T])),
              gensym(f,F6),
              assert(proved(F6 = [K1,G1=>T3])),
              append4(Delta, G1, G1, Lambda, G7),
              append(Phi, T, T6),
              append(T6, Psi, T5),
              gensym(f,F7),
              assert(proved( F7 = [cut(F6,F5,A),G7 => T5])),
              gensym(f,F8),
              assert(proved( F8 = [con(F7,Psi),G7 => T6])),
              gensym(f,F9),
              assert(proved( F9 = [con(F8,Phi),G7 => T])),
              assert(proved( Z = [con(F9,G1),G => T])),
              retract(proved(Z = [cut(Z1,Z2,A & B),G => T])).


/* (R13,R2) - (C.4) */
cut_elm(proved(Z = [cut(Z1,Z2,A # B), G => T])) :-
              proved(Z1 = [ois(Z3,A # B),G1=>T1]),
              proved(Z2 = [th(Z4,A # B),G2=>T2]),
              proved(Z3 = [K1,G1=>T3]),
              proved(Z4 = [K2,G3=>T2]),
              append(Phi, [A # B|Psi], T1),
              append(Phi, [A|[B|Psi]], T3),
              append3(Phi, T2, Psi, T),
              append(Delta, [A # B|Lambda], G2),
              append(Delta, Lambda, G3),
              append3(Delta, G1, Lambda, G),
           !,
```

```
                gensym(f,F1),
                assert(proved(F1 = [K2,G3=>T2])),
                append(Delta, [B|Lambda], G4),
                gensym(f,F2),
                assert(proved( F2 = [th(F1,B),G4 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [K1,G1=>T3])),
                append3(Phi, [A|T2], Psi, T4),
                gensym(f,F4),
                assert(proved( F4 = [cut(F3,F2,B),G => T4])),
                gensym(f,F5),
                assert(proved(F5 = [K2, G3 => T2])),
                append(Delta, [A|Lambda], G5),
                gensym(f,F6),
                assert(proved( F6 = [th(F5,A),G5 => T2])),
                append(Delta, G, G7),
                append(G7, Lambda, G6),
                append4(Phi, T2, T2, Psi, T5),
                gensym(f,F7),
                assert(proved( F7 = [cut(F4,F6,A),G6 => T5])),
                gensym(f,F8),
                assert(proved( F8 = [con(F7,T2),G6 => T])),
                gensym(f,F9),
                assert(proved( F9 = [con(F8,Lambda),G7 => T])),
                assert(proved( Z = [con(F9,Delta),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A # B),G => T])).

/* (R15,R2) - (C.5) */
cut_elm(proved(Z = [cut(Z1,Z2,A -> B), G => T])) :-
                proved(Z1 = [his(Z3,A -> B),G1=>[A -> B]]),
                proved(Z2 = [th(Z4,A -> B),G2=>T]),
                proved(Z3 = [K1,G3=>[B]]),
                proved(Z4 = [K2,G4=>T]),
                append(Gamma, [A|Delta], G3),
                append(Gamma, Delta, G1),
                append(Lambda, [A -> B|Zeta], G2),
                append(Lambda, Zeta, G4),
                append3(Lambda, G1, Zeta, G),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K2,G4 => T])),
                append3(Lambda, Gamma, Zeta, G5),
                gensym(f,F2),
                assert(proved( F2 = [th(F1,Gamma),G5 => T])),
                assert(proved( Z = [th(F2,Delta),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A -> B),G => T])).

/* (R5,A1) - (C.7.1) */
cut_elm(proved(Z = [cut(Z1,Z2,A),G => T])) :-
                proved(Z1 = [th(Z3,A),G=>T1]),
                proved(Z2 = [K,[A]=>[B]]),
                (K =.. [X] ; K =.. [id,A]),
                proved(Z3 = [K1,G=>T2]),
                append(Phi, [B|Psi], T),
                append(Phi, [A|Psi], T1),
                append(Phi, Psi, T2),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K1,G => T2])),
                assert(proved(Z=[th(F1,B),G=>T])),
                retract(proved(Z=[cut(Z1,Z2,A),G=>T])).

/* (R5,A4) - (C.7.2) */
cut_elm(proved(Z = [cut(Z1,Z2,bo), G => T])) :-
                proved(Z1 = [th(Z3,bo),G=>T1]),
                proved(Z2 = [bottom,[bo]=>[]]),
                proved(Z3 = [K1,G=>T]),
            !,
                assert(proved(Z = [K1, G => T])),
                retract(proved(Z = [cut(Z1,Z2,bo),G=>T])).

/* (R5,R11) - (C.8) */
cut_elm(proved(Z = [cut(Z1,Z2,A & B), G => T])) :-
                proved(Z1 = [th(Z3,A & B),G1 => T1]),
                proved(Z2 = [ala(Z4,A & B),G2 => T2]),
                proved(Z3 = [K1,G1=>T3]),
                proved(Z4 = [K2,G3=>T2]),
                append(Phi, [A & B|Psi], T1),
                append(Phi, Psi, T3),
```

```
                append3(Phi, T2, Psi, T),
                append(Delta, [A & B|Lambda], G2),
                append(Delta, [A|[B|Lambda]], G3),
                append3(Delta, G1, Lambda, G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T3])),
                append(Phi, [A|Psi], T4),
                gensym(f,F2),
                assert(proved( F2 = [th(F1,A),G1 => T4])),
                gensym(f,F3),
                assert(proved(F3 = [K1, G1 => T3])),
                append(Phi, [B|Psi], T5),
                gensym(f,F4),
                assert(proved( F4 = [th(F3,B),G1 => T5])),
                gensym(f,F5),
                assert(proved(F5 = [K2, G3 => T2])),
                append3(Delta, [A|G1], Lambda, G4),
                gensym(f,F6),
                assert(proved( F6 = [cut(F4,F5,B),G4 => T])),
                append4(Delta, G1, G1, Lambda, G5),
                append(Phi, T, T7),
                append(T7, Psi, T6),
                gensym(f,F7),
                assert(proved( F7 = [cut(F2,F6,A),G5 => T6])),
                gensym(f,F8),
                assert(proved( F8 = [con(F7,Psi),G5 => T7])),
                gensym(f,F9),
                assert(proved( F9 = [con(F8,Phi),G5 => T])),
                assert(proved( Z = [con(F9,G1),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A & B),G => T])).

/* (R5,R12) - (C.9) */
cut_elm(proved(Z = [cut(Z1,Z2,A # B), G => T])) :-
                proved(Z1 = [th(Z3,A # B),G1=>T1]),
                proved(Z2 = [ola(Z4,Z5,A # B),G2=>T2]),
                proved(Z3 = [K1,G1=>T3]),
                proved(Z4 = [K2,G3=>T2]),
                proved(Z5 = [K3,G4=>T2]),
                append(Phi, [A # B|Psi], T1),
                append(Phi, Psi, T3),
                append3(Phi, T2, Psi, T),
                append(Delta, [A # B|Lambda], G2),
                append(Delta, [A|Lambda], G3),
                append(Delta, [B|Lambda], G4),
                append3(Delta, G1, Lambda, G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1,G1 => T3])),
                append(Phi, [B|Psi], T4),
                gensym(f,F2),
                assert(proved( F2 = [th(F1,B),G1 => T4])),
                append(Phi, [A|[B|Psi]], T5),
                gensym(f,F3),
                assert(proved( F3 = [th(F2,A),G1 => T5])),
                gensym(f,F4),
                assert(proved(F4 = [K2, G3 => T2])),
                append3(Phi, T2, [B|Psi], T6),
                gensym(f,F5),
                assert(proved( F5 = [cut(F3,F4,A),G => T6])),
                gensym(f,F6),
                assert(proved(F6 = [K3, G4 => T2])),
                append(Delta, G, G6),
                append(G6, Lambda, G5),
                append4(Phi, T2, T2, Psi, T7),
                gensym(f,F7),
                assert(proved( F7 = [cut(F5,F6,B),G5 => T7])),
                gensym(f,F8),
                assert(proved( F8 = [con(F7,T2),G5 => T])),
                gensym(f,F9),
                assert(proved( F9 = [con(F8,Lambda),G6 => T])),
                assert(proved( Z = [con(F9,Delta),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A # B),G => T])).

/* (R5,R14) - (C.10) */
cut_elm(proved(Z = [cut(Z1,Z2,A -> B), G => T])) :-
                proved(Z1 = [th(Z3,A -> B),G1=>T1]),
                proved(Z2 = [hia(Z4,Z5,A -> B),G2=>T2]),
                proved(Z3 = [K1,G1=>T3]),
                proved(Z4 = [K2,G3=>[A]]),
                proved(Z5 = [K3,G4=>T2]),
                append(Phi, [A -> B|Psi], T1),
```

```
                append(Phi, Psi, T3),
                append3(Phi, T2, Psi, T),
                append(S2, [A -> B|Zeta], G2),
                append(Lambda, [B|Zeta], G4),
                append(Lambda, G3, S2),
                append3(S2, G1, Zeta, G),
            !,
                append([A],G1,G5),
                gensym(f,F1),
                assert(proved(F1 = [K1,G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [th(F1,A), G5 => T3])),
                append(G3,G1,G6),
                gensym(f,F3),
                assert(proved(F3 = [K2, G3 => [A]])),
                gensym(f,F4),
                assert(proved(F4 = [cut(F3,F2,A), G6 => T3])),
                append(Phi, [B|Psi], T4),
                gensym(f,F5),
                assert(proved(F5 = [th(F4,B), G6 => T4])),
                gensym(f,F6),
                assert(proved(F6 = [K3,G4 => T2])),
                assert(proved(Z = [cut(F5,F6,B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,A -> B),G => T])).

/* (R5,R2) - (C.12) */
cut_elm(proved(Z = [cut(Z1,Z2,A), G => T])) :-
                proved(Z1 = [th(Z3,A),G1=>T1]),
                proved(Z2 = [th(Z4,A),G2=>T2]),
                proved(Z3 = [K1,G1=>T3]),
                proved(Z4 = [_,G3=>T2]),
                append(Phi, [A|Psi], T1),
                append(Phi, Psi, T3),
                append3(Phi, T2, Psi, T),
                append(Delta, [A|Lambda], G2),
                append(Delta, Lambda, G3),
                append(G1, Lambda, G4),
                append(Delta, G4, G),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [th(F1,T2),G1 => T])),
                gensym(f,F3),
                assert(proved(F3 = [th(F2,Lambda),G4 => T])),
                assert(proved(Z = [th(F3,Delta),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A),G => T])).

/* (R10,R11) - (C.14) */
cut_elm(proved(Z = [cut(Z1,Z2,A & B), G => T])) :-
                proved(Z1 = [ais(Z3,Z4,A & B),G1 => T1]),
                proved(Z2 = [aia(Z5,A & B),G2 => T2]),
                proved(Z3 = [K1,G1=>T3]),
                proved(Z4 = [K2,G1=>T4]),
                proved(Z5 = [K3,G3=>T2]),
                append(Phi, [A & B|Psi], T1),
                append(Phi, [A|Psi], T3),
                append(Phi, [B|Psi], T4),
                append3(Phi, T2, Psi, T),
                append(Delta, [A & B|Lambda], G2),
                append(Delta, [A|[B|Lambda]], G3),
                append3(Delta, G1, Lambda, G),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G1 => T4])),
                gensym(f,F2),
                assert(proved(F2 = [K3, G3 => T2])),
                append3(Delta, [A|G1], Lambda, G4),
                gensym(f,F3),
                assert(proved( F3 = [cut(F1,F2,B),G4 => T])),
                gensym(f,F4),
                assert(proved(F4 = [K1, G1 => T3])),
                append4(Delta, G1, G1, Lambda, G5),
                append(Phi, T, T6),
                append(T6, Psi, T5),
                gensym(f,F5),
                assert(proved( F5 = [cut(F4,F3,A),G5 => T5])),
                gensym(f,F6),
                assert(proved( F6 = [con(F5,Psi),G5 => T6])),
                gensym(f,F7),
                assert(proved( F7 = [con(F6,Phi),G5 => T])),
                assert(proved( Z = [con(F7,G1),G => T])),
                retract(proved(Z = [cut(Z1,Z2,A & B),G => T])).
```

```
/* (R13,R12) - (C.15) */
cut_elm(proved(Z = [cut(Z1,Z2,A # B), G => T])) :-
            proved(Z1 = [ois(Z3,A # B),G1=>T1]),
            proved(Z2 = [oia(Z4,Z5,A # B),G2=>T2]),
            proved(Z3 = [K1,G1=>T3]),
            proved(Z4 = [K2,G3=>T2]),
            proved(Z5 = [K3,G4=>T2]),
            append(Phi, [A # B|Psi], T1),
            append(Phi, [A|[B|Psi]], T3),
            append3(Phi, T2, Psi, T),
            append(Delta, [A # B|Lambdc], G2),
            append(Delta, [A|Lambda], G3),
            append(Delta, [B|Lambda], G4),
            append3(Delta, G1, Lambda, G),
          !,
            gensym(f,F1),
            assert(proved(F1 = [K1, G1 => T3])),
            gensym(f,F2),
            assert(proved(F2 = [K2, G3 => T2])),
            append3(Phi, T2, [B|Psi], T4),
            gensym(f,F3),
            assert(proved( F3 = [cut(F1,F2,A),G => T4])),
            gensym(f,F4),
            assert(proved(F4 = [K3, G4 => T2])),
            append(Delta, G, G6),
            append(G6, Lambda, G5),
            append4(Phi, T2, T2, Psi, T5),
            gensym(f,F5),
            assert(proved( F5 = [cut(F3,F4,B),G5 => T5])),
            gensym(f,F6),
            assert(proved( F6 = [con(F5,T2),G5 => T])),
            gensym(f,F7),
            assert(proved( F7 = [con(F6,Lambda),G6 => T])),
            assert(proved( Z = [con(F7,Delta),G => T])),
            retract(proved(Z = [cut(Z1,Z2,A # B),G => T])).

/* (R15,R14) - (C.16) */
cut_elm(proved(Z = [cut(Z1,Z2,A -> B), G => T])) :-
            proved(Z1 = [his(Z3,A -> B),G1=>[A->B]]),
            proved(Z2 = [hia(Z4,Z5,A -> B),G2=>T]),
            proved(Z3 = [K1,G3=>[B]]),
            proved(Z4 = [K2,G4=>[A]]),
            proved(Z5 = [K3,G5=>T]),
            append(Gamma, [A|Delta], G3),
            append(Gamma, Delta, G1),
            append(S1, [A -> B|Sigma], G2),
            append(Zeta, [B|Sigma], G5),
            append(Zeta, G4, S1),
            append3(S1, G1, Sigma, G),
          !,
            gensym(f,F1),
            assert(proved(F1 = [K2,G4 => [A]])),
            gensym(f,F2),
            assert(proved(F2 = [K1,G3 => [B]])),
            append3(Gamma, G4, Delta, G6),
            gensym(f,F3),
            assert(proved( F3 = [cut(F1,F2,A),G6 => [B]])),
            gensym(f,F4),
            assert(proved(F4 = [K3, G5 => T])),
            append3(Zeta, G6, Sigma, G7),
            gensym(f,F5),
            assert(proved( F5 = [cut(F3,F4,B),G7 => T])),
            assert(proved( Z = [inc(F5,G4,Gamma),G => T])),
            retract(proved(Z = [cut(Z1,Z2,A -> B),G => T])).

/* (_,R2) - (C.18.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
            proved(Z1 = [K1,G1=>T1]),
            proved(Z2 = [th(Z3,A),G2=>T2]),
            proved(Z3 = [K2,G3=>T2]),
            append(Phi, [C|Psi], T1),
            append3(Phi, T2, Psi, T),
          ((append(S1, [A|Zeta], G2),
            append(S1, Zeta, G3),
            append(Delta, [C|Lambda], S1),
            append4(Delta, G1, Lambda, [A|Zeta], G),
            append4(Delta, G1, Lambda, Zeta, G4))
          ;(append(Delta, [A|E1], G2),
            append(Delta, E1, G3),
```

```
                append(Lambda, [C|Zeta], E1),
                append4(Delta, [A|Lambda], G1, Zeta, G),
                append4(Delta, Lambda, G1, Zeta, G4))),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved(Z = [th(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (_,R3) - (C.19.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [con(Z3,A),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
             ((append(S1, [A|[A|Zeta]], G3),
                append(S1, [A|Zeta], G2),
                append(Delta, [C|Lambda], S1),
                append4(Delta, G1, Lambda, [A|Zeta], G),
                append4(Delta, G1, Lambda, [A|[A|Zeta]], G4))
             ;(append(Delta, [A|[A|E1]], G3),
                append(Delta, [A|E1], G2),
                append(Lambda, [C|Zeta], E1),
                append4(Delta, [A|Lambda], G1, Zeta, G),
                append4(Delta, [A|[A|Lambda]], G1, Zeta, G4))),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved(Z = [con(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (_,R3) - (C.19.3) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [con(Z3,C),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
                append(Delta, [C|[C|Lambda]], G3),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                append3(Delta, [C|G1], Lambda, G4),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                gensym(f,F4),
                assert(proved(F4 = [K1, G1 => T1])),
                append4(Delta, G1, G1, Lambda, G5),
                append(Phi, T, T4),
                append(T4, Psi, T3),
                gensym(f,F5),
                assert(proved( F5 = [cut(F4,F3,C),G5 => T3])),
                gensym(f,F6),
                assert(proved( F6 = [con(F5,Psi),G5 => T4])),
                gensym(f,F7),
                assert(proved( F7 = [con(F6,Phi),G5 => T])),
                assert(proved( Z = [con(F7,G1),G => T])),
                retract(proved(Z = [cut(Z1,Z2,C),G => T])).

/* (_,R4) - (C.20.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [inc(Z3,B,A),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
             ((append(S1, [B|[A|Zeta]], G2),
```

```
                append(S1, [A|[B|Zeta]], G3),
                append(Delta, [C|Lambda], S1),
                append4(Delta, G1, Lambda, [B|[A|Zeta]], G),
                append4(Delta, G1, Lambda, [A|[B|Zeta]], G4))
              ;(append(Delta, [B|[A|E1]], G2),
                append(Delta, [A|[B|E1]], G3),
                append(Lambda, [C|Zeta], E1),
                append4(Delta, [B|[A|Lambda]], G1, Zeta, G),
                append4(Delta, [A|[B|Lambda]], G1, Zeta, G4))),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved(Z = [inc(F3,B,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (_,R4) - (C.20.3) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [inc(Z3,C,B),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
                append(Delta, [C|[B|Lambda]], G2),
                append(Delta, [B|[C|Lambda]], G3),
                append3(Delta, G1, [B|Lambda], G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                append3(Delta, [B|G1], Lambda, G4),
                gensym(f,F3),
                assert(proved( F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved( Z = [inc(F3,G1,B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (_,R4) - (C.20.4) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [inc(Z3,B,C),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
                append(Delta, [B|[C|Lambda]], G2),
                append(Delta, [C|[B|Lambda]], G3),
                append3(Delta, [B|G1], Lambda, G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                append3(Delta, G1, [B|Lambda], G4),
                gensym(f,F3),
                assert(proved( F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved( Z = [inc(F3,B,G1), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (_,R5) - (C.21) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [th(Z3,A),G2=>T2]),
                proved(Z3 = [K2,G2=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Theta, [A|Omiga], T2),
                append(Theta, Omiga, T3),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G2 => T3])),
                append3(Phi, T3, Psi, T4),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [th(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).
```

```
/* (_,R6) - (C.22) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [con(Z3,A),G2=>T2]),
                proved(Z3 = [K2,G2=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Theta, [A|[A|Omiga]], T3),
                append(Theta, [A|Omiga], T2),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G2 => T3])),
                append3(Phi, T3, Psi, T4),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [con(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R7) - (C.23) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [inc(Z3,B,A),G2=>T2]),
                proved(Z3 = [K2,G2=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Theta, [B|[A|Omiga]], T2),
                append(Theta, [A|[B|Omiga]], T3),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G2 => T3])),
                append3(Phi, T3, Psi, T4),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [inc(F3,B,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R10) - (C.26) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [ais(Z3,Z4,A & B),G2 => T2]),
                proved(Z3 = [K2,G2=>T3]),
                proved(Z4 = [K3,G2=>T4]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Theta, [A & B|Omiga], T2),
                append(Theta, [A|Omiga], T3),
                append(Theta, [B|Omiga], T4),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K1,G1=>T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G2 => T3])),
                append3(Phi, T3, Psi, T5),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T5])),
                gensym(f,F4),
                assert(proved(F4 = [K1,G1=>T1])),
                gensym(f,F5),
                assert(proved(F5 = [K3, G2 => T4])),
                append3(Phi, T4, Psi, T6),
                gensym(f,F6),
                assert(proved(F6 = [cut(F4,F5,C), G => T6])),
                assert(proved(Z = [ais(F3,F6,A & B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).
```

```
/* (_,R11) - (C.27.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [ola(Z3,A & B),G2 => T2]),
                proved(Z3 = [K2,G3=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
            ((append(S1, [A & B|Zeta], G2),
                append(S1, [A|[B|Zeta]], G3),
                append(Delta, [C|Lambda], S1),
                append4(Delta, G1, Lambda, [A & B|Zeta], G),
                append4(Delta, G1, Lambda, [A|[B|Zeta]], G4))
            ;(append(Delta, [A & B|E1], G2),
                append(Delta, [A|[B|E1]], G3),
                append(Lambda, [C|Zeta], E1),
                append4(Delta, [A & B|Lambda], G1, Zeta, G),
                append4(Delta, [A|[B|Lambda]], G1, Zeta, G4))),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                assert(proved(Z = [ola(F3,A & B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R12) - (C.28.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [ola(Z3,Z4,A # B),G2=>T2]),
                proved(Z3 = [K2,G3=>T2]),
                proved(Z4 = [K3,G4=>T2]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
            ((append(S1, [A # B|Zeta], G2),
                append(S1, [A|Zeta], G3),
                append(S1, [B|Zeta], G4),
                append(Delta, [C|Lambda], S1),
                append4(Delta, G1, Lambda, [A # B|Zeta], G),
                append4(Delta, G1, Lambda, [A|Zeta], G5),
                append4(Delta, G1, Lambda, [B|Zeta], G6))
            ;(append(Delta, [A # B|E1], G2),
                append(Delta, [A|E1], G3),
                append(Delta, [B|E1], G4),
                append(Lambda, [C|Zeta], E1),
                append4(Delta, [A # B|Lambda], G1, Zeta, G),
                append4(Delta, [A|Lambda], G1, Zeta, G5),
                append4(Delta, [B|Lambda], G1, Zeta, G6))),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G5 => T])),
                gensym(f,F4),
                assert(proved(F4 = [K1, G1 => T1])),
                gensym(f,F5),
                assert(proved(F5 = [K3, G4 => T2])),
                gensym(f,F6),
                assert(proved(F6 = [cut(F4,F5,C), G6 => T])),
                assert(proved(Z = [ola(F3,F6,A # B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R13) - (C.29) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>T1]),
                proved(Z2 = [ols(Z3,A # B),G2=>T2]),
                proved(Z3 = [K2,G2=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Theta, [A # B|Omiga], T2),
                append(Theta, [A|[B|Omiga]], T3),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
            !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => T1])),
```

```
                gensym(f,F2),
                assert(proved(F2 = [K2, G2 => T3])),
                append3(Phi, T3, Psi, T4),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [cis(F3,A # B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R14) - (C.30.1) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>[C]]),
                proved(Z2 = [hia(Z3,Z4,A -> B),G2=>T]),
                proved(Z3 = [K2,G3=>[A]]),
                proved(Z4 = [K3,G4=>T]),
                append(Zeta, [B|Pi], G4),
                append3(Zeta, G3, [A -> B|Pi], G2),
                append(Delta, [C|Lambda], G3),
                append5(Zeta, Delta, G1, Lambda, [A -> B|Pi], G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => [C]])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => [A]])),
                append3(Delta, G1, Lambda, G5),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G5 => [A]])),
                gensym(f,F4),
                assert(proved(F4 = [K3, G4 => T])),
                assert(proved(Z = [hia(F3,F4,A -> B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R14) - (C.30.2,3) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [K1,G1=>[C]]),
                proved(Z2 = [hia(Z3,Z4,A -> B),G2=>T]),
                proved(Z3 = [K2,G3=>[A]]),
                proved(Z4 = [K3,G4=>T]),
              ((append(S1, [A -> B|Pi], G2),
                append(S2, [B|Pi], G4),
                append(S2, G3, S1),
                append(Lambda, [C|Zeta], S2),
                append5(Lambda, G1, Zeta, G3, [A -> B|Pi], G),
                append4(Lambda, G1, Zeta, [B|Pi], G5))
              ;(append(S3, [A -> B|E1], G2),
                append(Lambda, [B|E1], G4),
                append(Lambda, G3, S3),
                append(Zeta, [C|Pi], E1),
                append5(Lambda, G3, [A -> B|Zeta], G1, Pi, G),
                append4(Lambda, [B|Zeta], G1, Pi, G5))),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => [C]])),
                gensym(f,F2),
                assert(proved(F2 = [K3, G4 => T])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G5 => T])),
                gensym(f,F4),
                assert(proved(F4 = [K2, G3 => [A]])),
                assert(proved(Z = [hia(F4,F3,A -> B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).


/* (_,R15) - (C.31.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => [A -> B]])) :-
              proved(Z1 = [K1,G1=>[C]]),
              proved(Z2 = [his(Z3,A -> B),G2=>[A->B]]),
              proved(Z3 = [K2,G3=>[B]]),
            ((append(S1, [A|Zeta], G3),
              append(S1, Zeta, G2),
              append(Delta, [C|Lambda], S1),
              append4(Delta, G1, Lambda, Zeta, G),
              append4(Delta, G1, Lambda, [A|Zeta], G4))
            ;(append(Delta, [A|E1], G3),
              append(Delta, E1, G2),
              append(Lambda, [C|Zeta], E1),
              append4(Delta, Lambda, G1, Zeta, G),
              append4(Delta, [A|Lambda], G1, Zeta, G4))),
            !,
```

```
                gensym(f,F1),
                assert(proved(F1 = [K1, G1 => [C]])),
                gensym(f,F2),
                assert(proved(F2 = [K2, G3 => [B]])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G4 => [B]])),
                assert(proved(Z = [his(F3,A -> B), G => [A -> B]])),
                retract(proved(Z = [cut(Z1,Z2,C), G => [A -> B]])).

/* (R2,_) - (C.34) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [th(Z3,A),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
                proved(Z3 = [K2,G3=>T1]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
                append(Gamma, [A|Delta], G1),
                append(Gamma, Delta, G3),
                append(Lambda, [C|Zeta], G2),
                append3(Lambda, G1, Zeta, G),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G3 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                append3(Lambda, G3, Zeta, G4),
                gensym(f,F3),
                assert(proved(F3 =[cut(F1,F2,C), G4 => T])),
                assert(proved(Z = [th(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R3,_) - (C.35) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                 proved(Z1 = [con(Z3,A),G1=>T1]),
                 proved(Z2 = [K1,G2=>T2]),
                 proved(Z3 = [K2,G3=>T1]),
                 append(Phi, [C|Psi], T1),
                 append3(Phi, T2, Psi, T),
                 append(Gamma, [A|[A|Delta]], G3),
                 append(Gamma, [A|Delta], G1),
                 append(Lambda, [C|Zeta], G2),
                 append3(Lambda, G1, Zeta, G),
               !,
                 gensym(f,F1),
                 assert(proved(F1 = [K2, G3 => T1])),
                 gensym(f,F2),
                 assert(proved(F2 = [K1, G2 => T2])),
                 append3(Lambda, G3, Zeta, G4),
                 gensym(f,F3),
                 assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                 assert(proved(Z = [con(F3,A), G => T])),
                 retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R4,_) - (C.36) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                 proved(Z1 = [inc(Z3,B,A),G1=>T1]),
                 proved(Z2 = [K1,G2=>T2]),
                 proved(Z3 = [K2,G3=>T1]),
                 append(Phi, [C|Psi], T1),
                 append3(Phi, T2, Psi, T),
                 append(Gamma, [B|[A|Delta]], G1),
                 append(Gamma, [A|[B|Delta]], G3),
                 append(Lambda, [C|Zeta], G2),
                 append3(Lambda, G1, Zeta, G),
               !,
                 gensym(f,F1),
                 assert(proved(F1 = [K2, G3 => T1])),
                 gensym(f,F2),
                 assert(proved(F2 = [K1, G2 => T2])),
                 append3(Lambda, G3, Zeta, G4),
                 gensym(f,F3),
                 assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                 assert(proved(Z = [inc(F3,B,A), G => T])),
                 retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R5,_) - (C.37.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [th(Z3,A),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
```

```
                proved(Z3 = [K2,G1=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
              ((append(S1, [A|Theta], T1),
                append(S1, Theta, T3),
                append(Phi, [C|Psi], S1),
                append4(Phi, T2, Psi, [A|Theta], T),
                append4(Phi, T2, Psi, Theta, T4))
              ;(append(Phi, [A|E1], T1),
                append(Phi, E1, T3),
                append(Psi, [C|Theta], E1),
                append4(Phi, [A|Psi], T2, Theta, T),
                append4(Phi, Psi, T2, Theta, T4))),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [th(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R6,_) - (C.38.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [con(Z3,A),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
                proved(Z3 = [K2,G1=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
              ((append(S1, [A|[A|Theta]], T3),
                append(S1, [A|Theta], T1),
                append(Phi, [C|Psi], S1),
                append4(Phi, T2, Psi, [A|Theta], T),
                append4(Phi, T2, Psi, [A|[A|Theta]], T4))
              ;(append(Phi, [A|[A|E1]], T3),
                append(Phi, [A|E1], T1),
                append(Psi, [C|Theta], E1),
                append4(Phi, [A|Psi], T2, Theta, T),
                append4(Phi, [A|[A|Psi]], T2, Theta, T4))),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [con(F3,A), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R6,_) - (C.38.3) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [con(Z3,C),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
                proved(Z3 = [K2,G1=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
                append(Phi, [C|[C|Psi]], T3),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
              !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                append3(Phi, [C|T2], Psi, T4),
                gensym(f,F3),
                assert(proved( F3 = [cut(F1,F2,C), G => T4])),
                gensym(f,F4),
                assert(proved(F4 = [K1, G2 => T2])),
                append3(Delta, G, Lambda, G3),
                append4(Phi, T2, T2, Psi, T5),
                gensym(f,F5),
                assert(proved( F5 = [cut(F3,F4,C),G3 => T5])),
                gensym(f,F6),
                assert(proved( F6 = [con(F5,T2),G3 => T])),
                append(Delta, G, G4),
                gensym(f,F7),
                assert(proved( F7 = [con(F6,Lambda),G4 => T])),
                assert(proved( Z = [con(F7,Delta),G => T])),
                retract(proved(Z = [cut(Z1,Z2,C),G => T])).
```

```
/* (R7,_) - (C.39.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
           proved(Z1 = [inc(Z3,B,A),G1=>T1]),
           proved(Z2 = [K1,G2=>T2]),
           proved(Z3 = [K2,G1=>T3]),
           append(Delta, [C|Lambda], G2),
           append3(Delta, G1, Lambda, G),
         ((append(S1, [B|[A|Theta]], T1),
           append(S1, [A|[B|Theta]], T3),
           append(Phi, [C|Psi], S1),
           append4(Phi, T2, Psi, [B|[A|Theta]], T),
           append4(Phi, T2, Psi, [A|[B|Theta]], T4))
         ;(append(Phi, [B|[A|E1]], T1),
           append(Phi, [A|[B|E1]], T3),
           append(Psi, [C|Theta], E1),
           append4(Phi, [B|[A|Psi]], T2, Theta, T),
           append4(Phi, [A|[B|Psi]], T2, Theta, T4))),
         !,
           gensym(f,F1),
           assert(proved(F1 = [K2, G1 => T3])),
           gensym(f,F2),
           assert(proved(F2 = [K1, G2 => T2])),
           gensym(f,F3),
           assert(proved(F3 = [cut(F1,F2,C), G => T4])),
           assert(proved(Z = [inc(F3,B,A), G => T])),
           retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R7,_) - (C.39.3) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
            proved(Z1 = [inc(Z3,C,D),G1=>T1]),
            proved(Z2 = [K1,G2=>T2]),
            proved(Z3 = [K2,G1=>T3]),
            append(Delta, [C|Lambda], G2),
            append3(Delta, G1, Lambda, G),
            append(Phi, [C|[D|Psi]], T1),
            append(Phi, [D|[C|Psi]], T3),
            append3(Phi, T2, [D|Psi], T),
          !,
            gensym(f,F1),
            assert(proved(F1 = [K2, G1 => T3])),
            gensym(f,F2),
            assert(proved(F2 = [K1, G2 => T2])),
            append3(Phi, [D|T2], Psi, T4),
            gensym(f,F3),
            assert(proved(F3 = [cut(F1,F2,C), G => T4])),
            assert(proved(Z = [inc(F3,T2,D), G =>T])),
            retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R7,_) - (C.39.4) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
            proved(Z1 = [inc(Z3,D,C),G1=>T1]),
            proved(Z2 = [K1,G2=>T2]),
            proved(Z3 = [K2,G1=>T3]),
            append(Delta, [C|Lambda], G2),
            append3(Delta, G1, Lambda, G),
            append(Phi, [D|[C|Psi]], T1),
            append(Phi, [C|[D|Psi]], T3),
            append3(Phi, [D|T2], Psi, T),
          !,
            gensym(f,F1),
            assert(proved(F1 = [K2, G1 => T3])),
            gensym(f,F2),
            assert(proved(F2 = [K1, G2 => T2])),
            append3(Phi, T2, [D|Psi], T4),
            gensym(f,F3),
            assert(proved(F3 = [cut(F1,F2,C), G => T4])),
            assert(proved(Z = [inc(F3,D,T2), G =>T])),
            retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R10,_) - (C.41.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
            proved(Z1 = [ais(Z3,Z4,A & B),G1 => T1]),
            proved(Z2 = [K1,G2=>T2]),
            proved(Z3 = [K2,G1=>T3]),
            proved(Z4 = [K3,G1=>T4]),
            append(Delta, [C|Lambda], G2),
            append3(Delta, G1, Lambda, G),
          ((append(S1, [A & B|Theta], T1),
```

```
                    append(S1, [A|Theta], T3),
                    append(S1, [B|Theta], T4),
                    append(Phi, [C|Psi], S1),
                    append4(Phi, T2, Psi, [A & B|Theta], T),
                    append4(Phi, T2, Psi, [A|Theta], T5),
                    append4(Phi, T2, Psi, [B|Theta], T6))
                 ;(append(Phi, [A & B|E1], T1),
                    append(Phi, [A|E1], T3),
                    append(Phi, [B|E1], T4),
                    append(Psi, [C|Theta], E1),
                    append4(Phi, [A & B|Psi], T2, Theta, T),
                    append4(Phi, [A|Psi], T2, Theta, T5),
                    append4(Phi, [B|Psi], T2, Theta, T6))),
                 !,
                    gensym(f,F1),
                    assert(proved(F1 = [K2, G1 => T3])),
                    gensym(f,F2),
                    assert(proved(F2 = [K1, G2 => T2])),
                    gensym(f,F3),
                    assert(proved(F3 = [cut(F1,F2,C), G => T5])),
                    gensym(f,F4),
                    assert(proved(F4 = [K3, G1 => T4])),
                    gensym(f,F5),
                    assert(proved(F5 = [K1, G2 => T2])),
                    gensym(f,F6),
                    assert(proved(F6 = [cut(F4,F5,C), G => T6])),
                    assert(proved(Z = [ais(F3,F6,A & B), G => T])),
                    retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R11,_) - (C.42) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                    proved(Z1 = [aia(Z3,A & B),G1=>T1]),
                    proved(Z2 = [K1,G2=>T2]),
                    proved(Z3 = [K2,G3=>T1]),
                    append(Phi, [C|Psi], T1),
                    append3(Phi, T2, Psi, T),
                    append(Gamma, [A & B|Delta], G1),
                    append(Gamma, [A|[B|Delta]], G3),
                    append(Lambda, [C|Zeta], G2),
                    append3(Lambda, G1, Zeta, G),
                 !,
                    gensym(f,F1),
                    assert(proved(F1 = [K2, G3 => T1])),
                    gensym(f,F2),
                    assert(proved(F2 = [K1, G2 => T2])),
                    append3(Lambda, G3, Zeta, G4),
                    gensym(f,F3),
                    assert(proved(F3 = [cut(F1,F2,C), G4 => T])),
                    assert(proved(Z = [aia(F3,A & B), G => T])),
                    retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R12,_) - (C.43) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                    proved(Z1 = [oia(Z3,Z4,A # B),G1=>T1]),
                    proved(Z2 = [K1,G2=>T2]),
                    proved(Z3 = [K2,G3=>T1]),
                    proved(Z4 = [K3,G4=>T1]),
                    append(Phi, [C|Psi], T1),
                    append3(Phi, T2, Psi, T),
                    append(Gamma, [A # B|Delta], G1),
                    append(Gamma, [A|Delta], G3),
                    append(Gamma, [B|Delta], G4),
                    append(Lambda, [C|Zeta], G2),
                    append3(Lambda, G1, Zeta, G),
                 !,
                    gensym(f,F1),
                    assert(proved(F1 = [K2, G3 => T1])),
                    gensym(f,F2),
                    assert(proved(F2 = [K1, G2 => T2])),
                    append3(Lambda, G3, Zeta, G5),
                    gensym(f,F3),
                    assert(proved(F3 = [cut(F1,F2,C), G5 => T])),
                    gensym(f,F4),
                    assert(proved(F4 = [K3, G4 => T1])),
                    gensym(f,F5),
                    assert(proved(F5 = [K1, G2 => T2])),
                    append3(Lambda, G4, Zeta, G6),
                    gensym(f,F6),
                    assert(proved(F6 = [cut(F4,F5,C), G6 => T])),
                    assert(proved(Z = [oia(F3,F6,A # B), G => T])),
                    retract(proved(Z = [cut(Z1,Z2,C), G => T])).
```

```
/* (R13,_) - (C.44.1,2) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [ois(Z3,A # B),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
                proved(Z3 = [K2,G1=>T3]),
                append(Delta, [C|Lambda], G2),
                append3(Delta, G1, Lambda, G),
             ((append(S1, [A # B|Theta], T1),
                append(S1, [A|[B|Theta]], T3),
                append(Phi, [C|Psi], S1),
                append4(Phi, T2, Psi, [A # B|Theta], T),
                append4(Phi, T2, Psi, [A|[B|Theta]], T4))
             ;(append(Phi, [A # B|E1], T1),
                append(Phi, [A|[B|E1]], T3),
                append(Psi, [C|Theta], E1),
                append4(Phi, [A # B|Psi], T2, Theta, T),
                append4(Phi, [A|[B|Psi]], T2, Theta, T4))),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K2, G1 => T3])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G => T4])),
                assert(proved(Z = [ois(F3,A # B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).

/* (R14,_) - (C.45) */
cut_elm(proved(Z = [cut(Z1,Z2,C), G => T])) :-
                proved(Z1 = [hia(Z3,Z4,A -> B),G1=>T1]),
                proved(Z2 = [K1,G2=>T2]),
                proved(Z3 = [K2,G3=>[A]]),
                proved(Z4 = [K3,G4=>T1]),
                append(Phi, [C|Psi], T1),
                append3(Phi, T2, Psi, T),
                append(S1, [A -> B|Lambda], G1),
                append(Delta, [B|Lambda], G4),
                append(Delta, G3, S1),
                append(Zeta, [C|Pi], G2),
                append3(Zeta, G1, Pi, G),
             !,
                gensym(f,F1),
                assert(proved(F1 = [K3, G4 => T1])),
                gensym(f,F2),
                assert(proved(F2 = [K1, G2 => T2])),
                append3(Zeta, G4, Pi, G5),
                gensym(f,F3),
                assert(proved(F3 = [cut(F1,F2,C), G5 => T])),
                gensym(f,F4),
                assert(proved(F4 = [K2, G3 => [A]])),
                assert(proved(Z = [hia(F4,F3,A -> B), G => T])),
                retract(proved(Z = [cut(Z1,Z2,C), G => T])).
```

```
/* this part is to remove all steps of (R14)(hook_introduction
   in the antecedent with the property that the active formulas
   in their right premisses are isomorphic to T (the terminal
   object) */

hook_terminal :-
                (thry(cartcl) ; thry(bicartcl)),
                proved(Z = [hia(Z1,Z2,A -> B), G => T]),
                isom_tr(B),
                hook_termin(Z = [hia(Z1,Z2,A -> B), G => T]),
                fail.

hook_terminal.


hook_termin(Z = [hia(Z1,Z2,A -> B), G => T]) :-
            proved(Z1 = [K1, G1 => [A]]),
            proved(Z2 = [K, G2 => T]),
            append(Delta, [B|Lambda], G2),
            append3(Delta, G1, [A -> B|Lambda], G),
            not(thinning(Z2,B,Delta,Lambda)),
            gensym(f,Z3),
            expterm(Z3, B),
            gensym(f,Z4),
            append(Delta, Lambda, G3),
            assert(proved(Z4 = [cut(Z3,Z2,B), G3 => T])),
            append(Delta, [A -> B|Lambda], G4),
            gensym(f,Z5),
            assert(proved(Z5 = [th(Z4,A -> B), G4 => T])),
            assert(proved(Z = [th(Z5,G1), G => T])),
            retract(proved(Z1 = [K1, G1 => [A]])),
            retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
            main(X), assert(hook_cut),
            cut_free(X),
            retract(hook_cut),
          !.

hook_termin(Z).


thinning(Z2,B,Delta,Lambda) :-
                proved(Z2 = [th(Z3,B), G2 => T]),
                proved(Z3 = [_, G3 => T]),
                append(Delta, Lambda, G3).


/* this part is to deal with the terminal and initial arrows */
term_inti :-
            proved(X),
            terminal(X),
            initial(X),
            fail.

term_inti.


/* deal with terminal arrows */
terminal(X = [L, G => T]) :-
            G \== [],
          (((thry(cart) ; thry(cartcl)),
            proved(X = [L, G => [Y]]),
            isom_tr(Y),
            gensym(f,X1),
            expterm(X1, Y),
            asserta(proved(X = [th(X1,G), G => T])))
          ;((thry(bicart) ; thry(dbicart) ; thry(bicartcl)),
            append(Delta, [Y|Phi], T),
            isom_tr(Y),
            gensym(f,X1),
            expterm(X1, Y),
            gensym(f,X2),
            append([Y], Phi, T1),
            asserta(proved(X2 = [th(X1,Phi), [] => T1])),
            gensym(f,X3),
            asserta(proved(X3 = [th(X2,Delta), [] => T])),
            asserta(proved(X = [th(X3,G), G => T])))),
            retract(proved(X = [L, G => T])),
          !,
```

```
terminal(X).


/* to check if an object X is isomorphic to a termial object */
isom_tr(X) :-
            X = tr,
            !.


isom_tr(X) :-
            X =.. [&,A,B],
            !,
            isom_tr(A), isom_tr(B).


isom_tr(X) :-
            X =.. [->,B,A],
            !,
            (((thry(cartcl) ; thry(bicartcl)),
              isom_tr(A), assert(term_hook(A)))
            ;(thry(bicartcl),
              isom_bo(B), assert(bo_hook(B)))).


isom_tr(X) :-
            X =.. [#,A,B],
            !,
            ((isom_tr(A), isom_bo(B), assert(term_or(A)))
            ;(isom_tr(B), isom_bo(A), assert(term_or(B)))).


/* to produce a unique derivation of --> Y, where Y is
   isomorphic to a terminal object */
expterm(X1, Y) :-
     ((Y = tr,
       asserta(proved(X1 = [true, [] => [tr]])))
     ;(Y =.. [&,A,B],
       gensym(f,X2),
       gensym(f,X3),
       expterm(X2, A),
       expterm(X3, B),
       asserta(proved(X1 = [ais(X2,X3,A&B), [] => [A & B]])))
     ;(Y =.. [->,B,A],
       term_hook(A), retract(term_hook(A)),
       gensym(f,X2),
       expterm(X2, A),
       gensym(f, X3),
       asserta(proved(X3 = [th(X2,B), [B] => [A]])),
       asserta(proved(X1 = [his(X3,B -> A),[] => [B -> A]])))
     ;(Y =.. [->,B,A],
       bo_hook(B), retract(bo_hook(B)),
       gensym(f,X2),
       expinti(X2, B),
       gensym(f, X3),
       asserta(proved(X3 = [th(X2,A), [B] => [A]])),
       asserta(proved(X1 = [his(X3,B -> A),[] => [B -> A]])))
     ;(Y =.. [#,A,B],
       gensym(f,X2),
       gensym(f,X3),
       ((term_or(A), retract(term_or(A)),
         expterm(X2, A),
         asserta(proved(X3 = [th(X2,B), [] => [A,B]])))
       ;(term_or(B), retract(term_or(B)),
         expterm(X2, B),
         asserta(proved(X3 = [th(X2,A), [] => [A,B]])))),
       asserta(proved(X1 = [ois(X3,A # B),[] => [A # B]])))).


/* deal with initial arrows */
initial(X = [L, G => T]) :-
            (thry(bicart) ; thry(dbicart) ; thry(bicartcl)),
             T \== [],
             append(Phi, [E|Psi], T),
             not(isom_tr(E)),
             append(Delta, [Y|Lambda], G),
             isom_bo(Y),
             gensym(f,X1),
             expinti(X1, Y),
             gensym(f,X2),
```

```
               append([Y], Lambda, G1),
               asserta(proved(X2 = [th(X1,Lambda), G1 => []])),
               gensym(f,X3),
               asserta(proved(X3 = [th(X2,Delta), G => []])),
               asserta(proved(X = [th(X3,T), G => T])),
               retract(proved(X = [L, G => T])),
             !.

initial(X).


/* to check if an object X is isomorphic to an initial object */
isom_bo(X) :-
               X = bo,
             !.


isom_bo(X) :-
               X =.. [#,A,B],
             !,
               isom_bo(A), isom_bo(B).


isom_bo(X) :-
             X =.. [&,A,B],
           !,
           ((thry(bicart),
             ((isom_tr(A), isom_bo(B), assert(inti_and(B)))
             ;(isom_tr(B), isom_bo(A), assert(inti_and(A)))))
           ;((thry(dbicart) ; thry(bicartcl)),
             ((isom_bo(A), assert(inti_and(A)))
             ;(isom_bo(B), assert(inti_and(B)))))).


/* to produce a unique derivation of Y ---> , where Y is
   isomorphic to an initial object */

expinti(X1,Y) :-
     ((Y = bo,
        asserta(proved(X1 = [bottom, [bo] => []])))
     ;(Y =.. [#,A,B],
        gensym(f,X2),
        gensym(f,X3),
        expinti(X2, A),
        expinti(X3, B),
        asserta(proved(X1 = [ola(X2,X3,A#B), [A#B] => []])))
     ;(Y =.. [&,A,B],
        gensym(f,X2),
        gensym(f,X3),
        ((inti_and(A), retract(inti_and(A)),
          expinti(X2, A),
          asserta(proved(X3 = [th(X2,B), [A,B] => []])))
        ;(inti_and(B), retract(inti_and(B)),
          expinti(X2, B),
          asserta(proved(X3 = [th(X2,A), [A,B] => []])))),
        asserta(proved(X1 = [ala(X3,A&B), [A&B] => []])))).
```

```
                    /* Contraction & Interchange Elimination */

/* This is a program to eliminate contraction and interchange
   in the case of Fc(X), Fbc(X) and Fdbc(X) */

con_inc_free(X) :-
        cop,
        write('Proof is as follows :'),nl,
        write('=====================   '),nl,nl,
        printall(X),nl,nl,
        assert(main(X)),
        normal1.


normal1 :-
      main(Z),
      repeat,
           do1_step,
    (thry(cartcl)
    ; thry(bicartcl)
    ;(write('Contraction and interchange free proof is :'),nl,
      write('============================================   '),nl,nl,
      printall(Z), nl, nl)).


do1_step :-
         main(X),
         put_in_list(X,L),
         assertz(proved(mark)),
         proved(Z),
         norm1(proved(Z)),
         Z = mark,
         put_in_list(X,L1),
         abolish(proved,1),
         ass_list(L1), !,
         L == L1.


/* (R2,R2) - (E.1) */
norm1(proved(Z = [th(Z1,B), G => T])) :-
         proved(Z1 = [th(Z2,A), G1 => T]),
         proved(Z2 = [_, G2 => T]),
         append(Gamma, [A|E1], G1),
         append(Gamma, E1, G2),
         append(Gamma, [A|E], G),
         append(Delta, [B|Lambda], E),
         append(Delta, Lambda, E1),
       !,
         assertz(proved(Z = [th(Z1,A), G => T])),
         append(Gamma, E, G3),
         assertz(proved(Z1 = [th(Z2,B), G3 => T])),
         retract(proved(Z = [th(Z1,B), G => T])),
         retract(proved(Z1 = [th(Z2,A), G1 => T])),
       !.


/* (R2,R3) - (E.2.1) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
         proved(Z1 = [th(Z2,A), G1 => T]),
         proved(Z2 = [F, G => T]),
         append(Gamma, [A|[A|Delta]], G1),
         append(Gamma, [A|Delta], G),
       !,
         assertz(proved(Z = [F, G => T])),
         retract(proved(Z = [con(Z1,A), G => T])),
         retract(proved(Z1 = [th(Z2,A), G1 => T])),
         retract(proved(Z2 = [F, G => T])),
       !.


/* (R2,R3) - (E.2.2,3) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
         proved(Z1 = [th(Z2,B), G1 => T]),
         proved(Z2 = [_, G2 => T]),
       ((append(Gamma, [A|[A|E]], G1),
         append(Gamma, [A|E], G),
         append(Gamma, [A|[A|E1]], G2),
         append(Delta, [B|Lambda], E),
         append(Delta, Lambda, E1),
```

```
         append(Gamma, [A|E1], G3))
       ;(append(S, [A|[A|Lambda]], G1),
         append(S, [A|Lambda], G),
         append(S1, [A|[A|Lambda]], G2),
         append(Gamma, [B|Delta], S),
         append(Gamma, Delta, S1),
         append(S1, [A|Lambda], G3))),
       !,
         assertz(proved(Z = [th(Z1,B), G => T])),
         assertz(proved(Z1 = [con(Z2,A), G3 => T])),
         retract(proved(Z = [con(Z1,A), G => T])),
         retract(proved(Z1 = [th(Z2,B), G1 => T])),
       !.

/* (R2,R4) - (E.3.1) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
         proved(Z1 = [th(Z2,B), G1 => T]),
         proved(Z2 = [_, G2 => T]),
         append(Gamma,[A|[B|Delta]],G1),
         append(Gamma,[B|[A|Delta]],G),
         append(Gamma,[A|Delta],G2),
       !,
         assertz(proved(Z = [th(Z2,B), G => T])),
         retract(proved(Z1 = [th(Z2,B), G1 => T])),
         retract(proved(Z = [inc(Z1,B,A), G => T])),
       !.

/* (R2,R4) - (E.3.2) */
norm1(proved(Z = [inc(Z1,A,B), G => T])) :-
         proved(Z1 = [th(Z2,B), G1 => T]),
         proved(Z2 = [_, G2 => T]),
         append(Gamma,[A|[B|Delta]],G),
         append(Gamma,[B|[A|Delta]],G1),
         append(Gamma,[A|Delta],G2),
       !,
         assertz(proved(Z = [th(Z2,B), G => T])),
         retract(proved(Z1 = [th(Z2,B), G1 => T])),
         retract(proved(Z = [inc(Z1,A,B), G => T])),
       !.

/* (R2,R4) - (E.3.3,4) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
         proved(Z1 = [th(Z2,C), G1 => T]),
         proved(Z2 = [_, G2 => T]),
       ((append(Gamma, [B|[A|E]], G),
         append(Gamma, [A|[B|E]], G1),
         append(Gamma, [A|[B|E1]], G2),
         append(Delta, [C|Lambda], E),
         append(Delta, Lambda, E1),
         append(Gamma, [B|[A|E1]], G3))
       ;(append(S, [B|[A|Lambda]], G),
         append(S, [A|[B|Lambda]], G1),
         append(S1, [A|[B|Lambda]], G2),
         append(Gamma, [C|Delta], S),
         append(Gamma, Delta, S1),
         append(S1, [B|[A|Lambda]], G3))),
       !,
         assertz(proved(Z = [th(Z1,C), G => T])),
         assertz(proved(Z1 = [inc(Z2,B,A), G3 => T])),
         retract(proved(Z1 = [th(Z2,C), G1 => T])),
         retract(proved(Z = [inc(Z1,B,A), G => T])),
       !.

/* (R2,R6) - (E.4) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
         proved(Z1 = [th(Z2,B), G => T1]),
         proved(Z2 = [_, G1 => T1]),
         append(Gamma, [B|Delta], G),
         append(Gamma, Delta, G1),
         append(Phi, [A|[A|Psi]], T1),
         append(Phi, [A|Psi], T),
       !,
         assertz(proved(Z = [th(Z1,B), G => T])),
         assertz(proved(Z1 = [con(Z2,A), G1 => T])),
         retract(proved(Z = [con(Z1,A), G => T])),
         retract(proved(Z1 = [th(Z2,B), G => T1])),
       !.
```

```
/* (R2,R7) - (E.5) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [th(Z2,C), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [C|Delta], G),
        append(Gamma, Delta, G1),
        append(Phi, [B|[A|Psi]], T),
        append(Phi, [A|[B|Psi]], T1),
      !,
        assertz(proved(Z = [th(Z1,C), G => T])),
        assertz(proved(Z1 = [inc(Z2,B,A), G1 => T])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [th(Z2,C), G => T1])),
      !.


/* (R3,R4) - (E.6.1,2) */
norm1(proved(Z = [inc(Z1,C,B), G => T])) :-
        proved(Z1 = [con(Z2,A), G1 => T]),
        proved(Z2 = [_, G2 => T]),
      ((append(S1, [C|[B|Lambda]], G),
        append(S1, [B|[C|Lambda]], G1),
        append(S,  [B|[C|Lambda]], G2),
        append(Gamma, [A|[A|Delta]], S),
        append(Gamma, [A|Delta], S1),
        append(S, [C|[B|Lambda]], G3))
      ;(append(Gamma, [C|[B|E1]], G),
        append(Gamma, [B|[C|E1]], G1),
        append(Gamma, [B|[C|E]],  G2),
        append(Delta, [A|[A|Lambda]], E),
        append(Delta, [A|Lambda], E1),
        append(Gamma, [C|[B|E]], G3))),
      !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        assertz(proved(Z1 = [inc(Z2,C,B), G3 => T])),
        retract(proved(Z = [inc(Z1,C,B), G => T])),
        retract(proved(Z1 = [con(Z2,A), G1 => T])),
      !.


/* (R3,R4) - (E.6.3) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [con(Z2,A), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        append(Gamma, [B|[A|Lambda]], G),
        append(Gamma, [A|[B|Lambda]], G1),
        append(Gamma, [A|[A|[B|Lambda]]], G2),
      !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        append(Gamma, [B|[A|[A|Lambda]]], G3),
        gensym(f,F),
        assertz(proved(Z1 = [inc(F,B,A), G3 => T])),
        append(Gamma, [A|[B|[A|Lambda]]], G4),
        assertz(proved(F = [inc(Z2,B,A), G4 => T])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [con(Z2,A), G1 => T])),
      !.


/* (R3,R4) - (E.6.4) */
norm1(proved(Z = [inc(Z1,A,B), G => T])) :-
        proved(Z1 = [con(Z2,A), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        append(Gamma, [A|[B|Lambda]], G),
        append(Gamma, [B|[A|Lambda]], G1),
        append(Gamma, [B|[A|[A|Lambda]]], G2),
      !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        append(Gamma, [A|[A|[B|Lambda]]], G3),
        gensym(f,F),
        assertz(proved(Z1 = [inc(F,A,B), G3 => T])),
        append(Gamma, [A|[B|[A|Lambda]]], G4),
        assertz(proved(F = [inc(Z2,A,B), G4 => T])),
        retract(proved(Z = [inc(Z1,A,B), G => T])),
        retract(proved(Z1 = [con(Z2,A), G1 => T])),
      !.


/* (R3,R5) - (E.7) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
        proved(Z1 = [th(Z2,B), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [A|[A|Delta]], G1),
```

```
        append(Gamma, [A|Delta], G),
        append(Phi, [B|Psi], T),
        append(Phi, Psi, T1),
    !,
        assertz(proved(Z = [th(Z1,B), G => T])),
        assertz(proved(Z1 = [con(Z2,A), G => T1])),
        retract(proved(Z = [con(Z1,A), G => T])),
        retract(proved(Z1 = [th(Z2,B), G1 => T])),
    !.


/* (R3,R7) - (E.8) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [con(Z2,C), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [C|Delta], G),
        append(Gamma, [C|[C|Delta]], G1),
        append(Phi, [B|[A|Psi]], T),
        append(Phi, [A|[B|Psi]], T1),
    !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        assertz(proved(Z1 = [inc(Z2,B,A), G1 => T])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [con(Z2,C), G => T1])),
    !.


/* (R3,R10) - (E.9) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        proved(Z3 = [_, G1 => T2]),
    !,
        gensym(f,F1),
        assertz(proved(Z = [ais(Z1,F1,A & B), G => T])),
        assertz(proved(Z1 = [con(Z2,C), G => T1])),
        assertz(proved(F1 = [con(Z3,C), G => T2])),
        retract(proved(Z = [con(Z1,C), G => T])),
        retract(proved(Z1 = [ais(Z2,Z3,A & B), G1 => T])),
    !.

/* (R3,R11) - (E.10.1,2) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [aia(Z2,A & B), G1 => T]),
        proved(Z2 = [_, G2 => T]),
     ((append(S1, [A & B|Lambda], G1),
        append(S1, [A|[B|Lambda]], G2),
        append(Gamma, [C|[C|Delta]], S1),
        append3(Gamma, [C|Delta], [A & B|Lambda], G),
        append3(Gamma, [C|Delta], [A|[B|Lambda]], G3))
     ;(append(Gamma, [A & B|E1], G1),
        append(Gamma, [A|[B|E1]], G2),
        append(Delta, [C|[C|Lambda]], E1),
        append3(Gamma, [A & B|Delta], [C|Lambda], G),
        append3(Gamma, [A|[B|Delta]], [C|Lambda], G3))),
    !,
        assertz(proved(Z = [aia(Z1,A & B), G => T])),
        assertz(proved(Z1 = [con(Z2,C), G3 => T])),
        retract(proved(Z = [con(Z1,C), G => T])),
        retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
    !.


/* (R3,R11) - (E.10.3,4) */
norm1(proved(Z = [con(Z1,A & A), G => T])) :-
        proved(Z1 = [aia(Z2,A & A), G1 => T]),
        proved(Z2 = [aia(Z3,A & A), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Gamma, [A & A|Delta], G),
        append(Gamma, [A & A|[A & A|Delta]], G1),
       (append(Gamma, [A|[A|[A & A|Delta]]], G2)
       ;append(Gamma, [A & A|[A|[A|Delta]]], G2)),
        append(Gamma, [A|[A|[A|[A|Delta]]]], G3),
    !,
        assertz(proved(Z = [aia(Z1,A & A), G => T])),
        append(Gamma, [A|[A|Delta]], G4),
        assertz(proved(Z1 = [con(Z2,A), G4 => T])),
        append(Gamma, [A|[A|[A|Delta]]], G5),
        assertz(proved(Z2 = [con(Z3,A), G5 => T])),
        retract(proved(Z = [con(Z1,A & A), G => T])),
        retract(proved(Z1 = [aia(Z2,A & A), G1 => T])),
        retract(proved(Z2 = [aia(Z3,A & A), G2 => T])),
    !.
```

```
/* (R3,R11) - (E.10.5,6) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
        proved(Z1 = [ala(Z2,A & B), G1 => T]),
        proved(Z2 = [ala(Z3,A & B), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A & B|[A & B|Delta]], G1),
       (append(Gamma, [A|[B|[A & B|Delta]]], G2)
       ;append(Gamma, [A & B|[A|[B|Delta]]], G2)),
        append(Gamma, [A|[B|[A|[B|Delta]]]], G3),
      !,
        assertz(proved(Z = [ala(Z1,A & B), G => T])),
        append(Gamma, [A|[B|Delta]], G4),
        assertz(proved(Z1 = [con(Z2,A), G4 => T])),
        append(Gamma, [A|[A|[B|Delta]]], G5),
        gensym(f,F),
        assertz(proved(Z2 = [con(F,B), G5 => T])),
        append(Gamma, [A|[A|[B|[B|Delta]]]], G6),
        assertz(proved(F = [inc(Z3,A,B), G6 => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
        retract(proved(Z1 = [ala(Z2,A & B), G1 => T])),
        retract(proved(Z2 = [ala(Z3,A & B), G2 => T])),
      !.


/* (R3,R12) - (E.11.1,2) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [ola(Z2,Z3,A # B), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => T]),
     ((append(S, [C|[C|Lambda]], G1),
        append(S, [C|Lambda], G),
        append(S1, [C|[C|Lambda]], G2),
        append(S2, [C|[C|Lambda]], G3),
        append(Gamma, [A # B|Delta], S),
        append(Gamma, [A|Delta], S1),
        append(Gamma, [B|Delta], S2),
        append(S1, [C|Lambda], G4),
        append(S2, [C|Lambda], G5))
     ;(append(Gamma, [C|[C|E]], G1),
        append(Gamma, [C|E], G),
        append(Gamma, [C|[C|E1]], G2),
        append(Gamma, [C|[C|E2]], G3),
        append(Delta, [A # B|Lambda], E),
        append(Delta, [A|Lambda], E1),
        append(Delta, [B|Lambda], E2),
        append(Gamma, [C|E1], G4),
        append(Gamma, [C|E2], G5))),
      !,
        gensym(f,F),
        assertz(proved(Z = [ola(Z1,F,A # B), G => T])),
        assertz(proved(Z1 = [con(Z2,C), G4 => T])),
        assertz(proved(F = [con(Z3,C), G5 => T])),
        retract(proved(Z = [con(Z1,C), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,A # B), G1 => T])),
      !.


/* (R3,R12) - (E.11.3,4) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [ola(Z2,Z3,A # B), G1 => T]),
        proved(Z2 = [ola(Z4,Z5,A # B), G2 => T]),
        proved(Z4 = [th(Z6,A), G3 => T]),
        proved(Z6 = [_, G4 => T]),
        proved(Z5 = [th(Z7,A), G5 => T]),
        proved(Z7 = [_, G6 => T]),
        proved(Z3 = [ola(Z8,Z9,A # B), G7 => T]),
        proved(Z8 = [th(Z10,B), G8 => T]),
        proved(Z10 = [_, G4 => T]),
        proved(Z9 = [th(Z11,B), G9 => T]),
        proved(Z11 = [_, G6 => T]),
        append(Gamma, [A#B|[A#B|Delta]], G1),
        append(Gamma, [A#B|Delta], G),
        append(Gamma, [A|[A|Delta]], G3),
        append(Gamma, [B|[B|Delta]], G9),
        append(Gamma, [A|Delta], G4),
        append(Gamma, [B|Delta], G6),
     ((append(Gamma, [A#B|[A|Delta]], G2),
        append(Gamma, [A#B|[B|Delta]], G7),
        append(Gamma, [B|[A|Delta]], G5),
        append(Gamma, [A|[B|Delta]], G8))
```

```
        ;(append(Gamma, [A|[A#B|Delta]], G2),
          append(Gamma, [B|[A#B|Delta]], G7),
          append(Gamma, [A|[B|Delta]], G5),
          append(Gamma, [B|[A|Delta]], G8))),
        !,
          assertz(proved(Z = [ola(Z6,Z11,A # B), G => T])),
          retract(proved(Z = [con(Z1,A # B), G => T])),
        !.


/* (R3,R13) - (E.12) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
          proved(Z1 = [ols(Z2,A # B), G1 => T]),
          proved(Z2 = [_, G1 => T1]),
        !,
          assertz(proved(Z = [ols(Z1,A # B), G => T])),
          assertz(proved(Z1 = [con(Z2,C), G => T1])),
          retract(proved(Z = [con(Z1,C), G => T])),
          retract(proved(Z1 = [ols(Z2,A # B), G1 => T])),
        !.


/* (R3,R14) - (E.13.1) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
          proved(Z1 = [hia(Z2,Z3,A -> B), G1 => T]),
          proved(Z2 = [_, G2 => [A]]),
          proved(Z3 = [_, G3 => T]),
          append(Lambda,[B|Sigma],G3),
          append3(Lambda,G2,[A->B|Sigma],G1),
          append(Gamma,[C|[C|Delta]],G2),
          append4(Lambda,Gamma,[C|Delta],[A->B|Sigma],G),
        !,
          assertz(proved(Z = [hia(Z1,Z3,A -> B), G => T])),
          append(Gamma,[C|Delta],G4),
          assertz(proved(Z1 = [con(Z2,C), G4 => [A]])),
          retract(proved(Z1 = [hia(Z2,Z3,A -> B),G1 => T])),
          retract(proved(Z = [con(Z1,C), G => T])),
        !.


/* (R3,R14) - (E.13.2,3) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
          proved(Z1 = [hia(Z2,Z3,A -> B), G1 => T]),
          proved(Z2 = [_, Gamma => [A]]),
          proved(Z3 = [_, G2 => T]),
        ((append(S,[B|Sigma],G2),
          append3(S,Gamma,[A->B|Sigma],G1),
          append(Delta,[C|[C|Lambda]],S),
          append3(S1,Gamma,[A -> B|Sigma],G),
          append(Delta,[C|Lambda],S1),
          append(S1,[B|Sigma],G3))
        ;(append(Delta,[B|E],G2),
          append3(Delta,Gamma,[A->B|E],G1),
          append(Lambda,[C|[C|Sigma]],E),
          append3(Delta,Gamma,[A->B|E1],G),
          append(Lambda,[C|Sigma],E1),
          append(Delta,[B|E1],G3))),
        !,
          assertz(proved(Z = [hia(Z2,Z1,A->B), G => T])),
          assertz(proved(Z1 = [con(Z3,C), G3 => T])),
          retract(proved(Z1 = [hia(Z2,Z3,A->B), G1 => T])),
          retract(proved(Z = [con(Z1,C), G => T])),
        !.



/* (R3,R15) - (E.14.1,2) */
norm1(proved(Z = [con(Z1,C), G => [A -> B]])) :-
          proved(Z1 = [his(Z2,A -> B), G1 => [A -> B]]),
          proved(Z2 = [_, G2 => [B]]),
        ((append(Gamma,[C|[C|E1]],G1),
          append(Gamma,[C|E1],G),
          append(Gamma,[C|[C|E]],G2),
          append(Delta,[A|Lambda],E),
          append(Delta,Lambda,E1),
          append(Gamma,[C|E],G3))
        ;(append(S1,[C|[C|Lambda]],G1),
          append(S1,[C|Lambda],G),
          append(S,[C|[C|Lambda]],G2),
          append(Gamma,[A|Delta],S),
          append(Gamma,Delta,S1),
          append(S,[C|Lambda],G3))),
        !,
```

```
        assertz(proved(Z = [his(Z1,A->B),G => [A->B]])),
        assertz(proved(Z1 = [con(Z2,C), G3 => [B]])),
        retract(proved(Z1 = [his(Z2,A->B), G1 => [A->B]])),
        retract(proved(Z = [con(Z1,C), G => [A -> B]])),
      !.


/* (R3,R15) - (E.14.3) */
norm1(proved(Z = [con(Z1,C), G => [A -> B]])) :-
        proved(Z1 = [his(Z2,A -> B), G1 => [A -> B]]),
        proved(Z2 = [_, G2 => [B]]),
        append(Gamma,[C|[A|[C|Delta]]],G2),
        append(Gamma,[C|[C|Delta]],G1),
        append(Gamma,[C|Delta],G),
      !,
        assertz(proved(Z = [his(Z1,A->B), G => [A->B]])),
        append(Gamma,[C|[A|Delta]],G3),
        assertz(proved(Z1 = [con(Z3,C), G3 => [B]])),
        append(Gamma,[C|[C|[A|Delta]]],G4),
        assertz(proved(Z3 = [inc(Z2,C,A), G4 => [B]])),
        retract(proved(Z1 = [his(Z2,A->B),G1 => [A->B]])),
        retract(proved(Z = [con(Z1,C), G => [A -> B]])),
      !.



/* (R4,R5) - (E.15) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [th(Z2,C), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [B|[A|Delta]], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [C|Psi], T),
        append(Phi, Psi, T1),
      !,
        assertz(proved(Z = [th(Z1,C), G => T])),
        assertz(proved(Z1 = [inc(Z2,B,A), G => T1])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [th(Z2,C), G1 => T])),
      !.



/* (R4,R6) - (E.16) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [con(Z2,C), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [B|[A|Delta]], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [C|[C|Psi]], T1),
        append(Phi, [C|Psi], T),
      !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        assertz(proved(Z1 = [inc(Z2,B,A), G => T1])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [con(Z2,C), G1 => T])),
      !.


/* (R4,R10) - (E.17) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        proved(Z3 = [_, G1 => T2]),
      !,
        gensym(f,F1),
        assertz(proved(Z = [ais(Z1,F1,A & B), G => T])),
        assertz(proved(Z1 = [inc(Z2,D,C), G => T1])),
        assertz(proved(F1 = [inc(Z3,D,C), G => T2])),
        retract(proved(Z = [inc(Z1,D,C), G => T])),
        retract(proved(Z1 = [ais(Z2,Z3,A & B), G1 => T])),
      !.


/* (R4,R11) - (E.18.1,2) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [aia(Z2,C & D), G1 => T]),
        proved(Z2 = [_, G2 => T]),
     ((append(S, [C & D|Lambda], G1),
        append(S, [C|[D|Lambda]], G2),
        append(Gamma, [A|[B|Delta]], S),
        append3(Gamma, [B|[A|Delta]], [C & D|Lambda], G),
        append3(Gamma,[B|[A|Delta]],[C|[D|Lambda]],G3))
```

```
        ;(append(Gamma, [C & D|E], G1),
          append(Gamma, [C|[D|E]], G2),
          append(Delta, [A|[B|Lambda]], E),
          append3(Gamma, [C & D|Delta], [B|[A|Lambda]], G),
          append3(Gamma,[C|[D|Delta]],[B|[A|Lambda]],G3))),
        !,
          assertz(proved(Z = [oia(Z1,C & D), G => T])),
          assertz(proved(Z1 = [inc(Z2,B,A), G3 => T])),
          retract(proved(Z = [inc(Z1,B,A), G => T])),
          retract(proved(Z1 = [oia(Z2,C & D), G1 => T])),
        !.


/* (R4,R11) - (E.18.3) */
norm1(proved(Z = [inc(Z1,C,A & B), G => T])) :-
          proved(Z1 = [oia(Z2,A & B), G1 => T]),
          proved(Z2 = [_, G2 => T]),
          append(Gamma, [C|[A & B|Delta]], G),
          append(Gamma, [A & B|[C|Delta]], G1),
          append(Gamma, [A|[B|[C|Delta]]], G2),
        !,
          assertz(proved(Z = [oia(Z1,A & B), G => T])),
          gensym(f,Z3),
          append(Gamma, [C|[A|[B|Delta]]], G4),
          assertz(proved(Z1 = [inc(Z3,C,A), G4 => T])),
          append(Gamma, [A|[C|[B|Delta]]], G3),
          assertz(proved(Z3 = [inc(Z2,C,B), G3 => T])),
          retract(proved(Z = [inc(Z1,C,A & B), G => T])),
          retract(proved(Z1 = [oia(Z2,A & B), G1 => T])),
        !.


/* (R4,R11) - (E.18.4) */
norm1(proved(Z = [inc(Z1,B & C,A), G => T])) :-
          proved(Z1 = [oia(Z2,B & C), G1 => T]),
          proved(Z2 = [_, G2 => T]),
          append(Gamma, [B & C|[A|Delta]], G),
          append(Gamma, [A|[B & C|Delta]], G1),
          append(Gamma, [A|[B|[C|Delta]]], G2),
        !,
          assertz(proved(Z = [oia(Z1,B & C), G => T])),
          gensym(f1,Z3),
          append(Gamma, [B|[C|[A|Delta]]], G3),
          assertz(proved(Z1 = [inc(Z3,C,A), G3 => T])),
          append(Gamma, [B|[A|[C|Delta]]], G4),
          assertz(proved(Z3 = [inc(Z2,B,A), G4 => T])),
          retract(proved(Z = [inc(Z1,B & C,A), G => T])),
          retract(proved(Z1 = [oia(Z2,B & C), G1 => T])),
        !.


/* (R4,R12) - (E.19.1,2) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
          proved(Z1 = [oia(Z2,Z3,A # B), G1 => T]),
          proved(Z2 = [_, G2 => T]),
          proved(Z3 = [_, G3 => T]),
        ((append(Gamma, [D|[C|E]], G),
          append(Gamma, [C|[D|E]], G1),
          append(Gamma, [C|[D|E1]], G2),
          append(Gamma, [C|[D|E2]], G3),
          append(Delta, [A # B|Lambda], E),
          append(Delta, [A|Lambda], E1),
          append(Delta, [B|Lambda], E2),
          append(Gamma, [D|[C|E1]], G4),
          append(Gamma, [D|[C|E2]], G5))
        ;(append(S, [D|[C|Lambda]], G),
          append(S, [C|[D|Lambda]], G1),
          append(S1, [C|[D|Lambda]], G2),
          append(S2, [C|[D|Lambda]], G3),
          append(Gamma, [A # B|Delta], S),
          append(Gamma, [A|Delta], S1),
          append(Gamma, [B|Delta], S2),
          append(S1, [D|[C|Lambda]], G4),
          append(S2, [D|[C|Lambda]], G5))),
        !,
          gensym(f,F),
          assertz(proved(Z = [oia(Z1,F,A # B), G => T])),
          assertz(proved(Z1 = [inc(Z2,D,C), G4 => T])),
          assertz(proved(F = [inc(Z3,D,C), G5 => T])),
          retract(proved(Z = [inc(Z1,D,C), G => T])),
          retract(proved(Z1 = [oia(Z2,Z3,A # B), G1 => T])),
        !.
```

```
/* (R4,R12) - (E.19.3) */
norm1(proved(Z = [inc(Z1,C,A # B), G => T])) :-
        proved(Z1 = [oia(Z2,Z3,A # B), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Gamma,[C|[A # B|Delta]], G),
        append(Gamma, [A # B|[C|Delta]], G1),
        append(Gamma, [A|[C|Delta]], G2),
        append(Gamma, [B|[C|Delta]], G3),
      !,
        gensym(f,F),
        assertz(proved(Z = [oia(Z1,F,A # B), G => T])),
        append(Gamma, [C|[A|Delta]], G4),
        assertz(proved(Z1 = [inc(Z2,C,A), G4 => T])),
        append(Gamma, [C|[B|Delta]], G5),
        assertz(proved(F = [inc(Z3,C,B), G5 => T])),
        retract(proved(Z = [inc(Z1,C,A # B), G => T])),
        retract(proved(Z1 = [oia(Z2,Z3,A # B), G1 => T])),
      !.


/* (R4,R12) - (E.19.4) */
norm1(proved(Z = [inc(Z1,A # B,C), G => T])) :-
        proved(Z1 = [oia(Z2,Z3,A # B), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Gamma,[ A # B|[C|Delta]], G),
        append(Gamma, [C|[A # B|Delta]], G1),
        append(Gamma, [C|[A|Delta]], G2),
        append(Gamma, [C|[B|Delta]], G3),
      !,
        gensym(f,F),
        assertz(proved(Z = [oia(Z1,F,A # B), G => T])),
        append(Gamma, [A|[C|Delta]], G4),
        assertz(proved(Z1 = [inc(Z2,A,C), G4 => T])),
        append(Gamma, [B|[C|Delta]], G5),
        assertz(proved(F = [inc(Z3,B,C), G5 => T])),
        retract(proved(Z = [inc(Z1,A # B,C), G => T])),
        retract(proved(Z1 = [oia(Z2,Z3,A # B), G1 => T])),
      !.


/* (R4,R13) - (E.20) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G1 => T]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [C|[D|Delta]], G1),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A|[B|Psi]], T1),
      !,
        assertz(proved(Z = [ois(Z1,A#B), G => T])),
        assertz(proved(Z1 = [inc(Z2,D,C), G => T1])),
        retract(proved(Z = [inc(Z1,D,C), G => T])),
        retract(proved(Z1 = [ois(Z2,A#B), G1 => T])),
      !.


/* (R4,R14) - (E.21.1) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,A -> B), G1 => T]),
        proved(Z2 = [_, G2 => [A]]),
        proved(Z3 = [_, G3 => T]),
        append(Lambda,[B|Sigma],G3),
        append3(Lambda,G2,[A->B|Sigma],G1),
        append(Gamma,[C|[D|Delta]],G2),
        append4(Lambda,Gamma,[D|[C|Delta]],[A->B|Sigma],G),
      !,
        assertz(proved(Z = [hia(Z1,Z3,A -> B), G => T])),
        append(Gamma,[D|[C|Delta]],G4),
        assertz(proved(Z1 = [inc(Z2,D,C), G4 => [A]])),
        retract(proved(Z1 = [hia(Z2,Z3,A -> B), G1 => T])),
        retract(proved(Z = [inc(Z1,D,C), G => T])),
      !.


/* (R4,R14) - (E.21.2,3) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,A -> B), G1 => T]),
        proved(Z2 = [_, Gamma => [A]]),
        proved(Z3 = [_, G2 => T]),
      ((append(S,[B|Sigma],G2),
```

```
            append3(S,Gamma,[A->B|Sigma],G1),
            append(Delta,[C|[D|Lambda]],S),
            append3(S1,Gamma,[A -> B|Sigma],G),
            append(Delta,[D|[C|Lambda]],S1),
            append(S1,[B|Sigma],G3))
        ;(append(Delta,[B|E],G2),
            append3(Delta,Gamma,[A->B|E],G1),
            append(Lambda,[C|[D|Sigma]],E),
            append3(Delta,Gamma,[A->B|E1],G),
            append(Lambda,[D|[C|Sigma]],E1),
            append(Delta,[B|E1],G3))),
        !,
            assertz(proved(Z = [hia(Z2,Z1,A->B), G => T])),
            assertz(proved(Z1 = [inc(Z3,D,C), G3 => T])),
            retract(proved(Z1 = [hia(Z2,Z3,A->B), G1 => T])),
            retract(proved(Z = [inc(Z1,D,C), G => T])),
        !.


/* (R4,R15) - (E.22.1,2) */
norm1(proved(Z = [inc(Z1,D,C), G => [A -> B]])) :-
            proved(Z1 = [his(Z2,A -> B), G1 => [A -> B]]),
            proved(Z2 = [_, G2 => [B]]),
        ((append(Gamma,[C|[D|E1]],G1),
            append(Gamma,[D|[C|E1]],G),
            append(Gamma,[C|[D|E]],G2),
            append(Delta,[A|Lambda],E),
            append(Delta,Lambda,E1),
            append(Gamma,[D|[C|E]],G3))
        ;(append(S1,[C|[D|Lambda]],G1),
            append(S1,[D|[C|Lambda]],G),
            append(S,[C|[D|Lambda]],G2),
            append(Gamma,[A|Delta],S),
            append(Gamma,Delta,S1),
            append(S,[D|[C|Lambda]],G3))),
        !,
            assertz(proved(Z = [his(Z1,A->B), G => [A->B]])),
            assertz(proved(Z1 = [inc(Z2,D,C), G3 => [B]])),
            retract(proved(Z1 = [his(Z2,A->B),G1 => [A->B]])),
            retract(proved(Z = [inc(Z1,D,C), G => [A -> B]])),
        !.


/* (R5,R5) - (E.23) */
norm1(proved(Z = [th(Z1,B), G => T])) :-
            proved(Z1 = [th(Z2,A), G => T1]),
            proved(Z2 = [_, G => T2]),
            append(S, [B|Theta], T),
            append(S, Theta, T1),
            append(S1, Theta, T2),
            append(Phi, [A|Psi], S),
            append(Phi, Psi, S1),
        !,
            assertz(proved(Z = [th(Z1,A), G => T])),
            append3(Phi, Psi, [B|Theta], T3),
            assertz(proved(Z1 = [th(Z2,B), G => T3])),
            retract(proved(Z = [th(Z1,B), G => T])),
            retract(proved(Z1 = [th(Z2,A), G => T1])),
        !.


/* (R5,R6) - (E.24.1) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
            proved(Z1 = [th(Z2,A), G => T1]),
            proved(Z2 = [K, G => T]),
            append(Phi, [A|[A|Psi]], T1),
            append(Phi, [A|Psi], T),
        !,
            assertz(proved(Z = [K, G => T])),
            retract(proved(Z = [con(Z1,A), G => T])),
            retract(proved(Z1 = [th(Z2,A), G => T1])),
        !.


/* (R5,R6) - (E.24.2,3) */
norm1(proved(Z = [con(Z1,A), G => T])) :-
            proved(Z1 = [th(Z2,B), G => T1]),
            proved(Z2 = [_, G => T2]),
        ((append(Phi, [A|E], T),
            append(Phi, [A|[A|E]], T1),
            append(Phi, [A|[A|E1]], T2),
            append(Psi, [B|Theta], E),
            append(Psi, Theta, E1),
```

```
          append(Phi, [A|E1], T3))
        ;(append(S, [A|Theta], T),
          append(S, [A|[A|Theta]], T1),
          append(S1, [A|[A|Theta]], T2),
          append(Phi, [B|Psi], S),
          append(Phi, Psi, S1),
          append(S1, [A|Theta], T3))),
        !,
          assertz(proved(Z = [th(Z1,B), G => T])),
          assertz(proved(Z1 = [con(Z2,A), G => T3])),
          retract(proved(Z = [con(Z1,A), G => T])),
          retract(proved(Z1 = [th(Z2,B), G => T1])),
        !.


/* (R5,R7) - (E.25.1,2) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
          proved(Z1 = [th(Z2,C), G => T1]),
          proved(Z2 = [_, G => T2]),
        ((append(S, [B|[A|Theta]], T),
          append(S, [A|[B|Theta]], T1),
          append(S1, [A|[B|Theta]], T2),
          append(Phi, [C|Psi], S),
          append(Phi, Psi, S1),
          append(S1, [B|[A|Theta]], T3))
        ;(append(Phi, [B|[A|E]], T),
          append(Phi, [A|[B|E]], T1),
          append(Phi, [A|[B|E1]], T2),
          append(Psi, [C|Theta], E),
          append(Psi, Theta, E1),
          append(Phi, [B|[A|E1]], T3))),
        !,
          assertz(proved(Z = [th(Z1,C), G => T])),
          assertz(proved(Z1 = [inc(Z2,B,A), G => T3])),
          retract(proved(Z = [inc(Z1,B,A), G => T])),
          retract(proved(Z1 = [th(Z2,C), G => T1])),
        !.


/* (R5,R7) - (E.25.3) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
          proved(Z1 = [th(Z2,B), G => T1]),
          proved(Z2 = [_, G => T2]),
          append(Phi, [B|[A|Psi]], T),
          append(Phi, [A|[B|Psi]], T1),
          append(Phi, [A|Psi], T2),
        !,
          assertz(proved(Z = [th(Z2,B), G => T])),
          retract(proved(Z = [inc(Z1,B,A), G => T])),
          retract(proved(Z1 = [th(Z2,B), G => T1])),
        !.


/* (R5,R7) - (E.25.4) */
norm1(proved(Z = [inc(Z1,A,B), G => T])) :-
          proved(Z1 = [th(Z2,B), G => T1]),
          proved(Z2 = [_, G => T2]),
          append(Phi, [A|[B|Psi]], T),
          append(Phi, [B|[A|Psi]], T1),
          append(Phi, [A|Psi], T2),
        !,
          assertz(proved(Z = [th(Z2,B), G => T])),
          retract(proved(Z = [inc(Z1,A,B), G => T])),
          retract(proved(Z1 = [th(Z2,B), G => T1])),
        !.

/* (R6,R7) - (E.26.1,2) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
          proved(Z1 = [con(Z2,C), G => T1]),
          proved(Z2 = [_, G => T2]),
        ((append(S, [B|[A|Theta]], T),
          append(S, [A|[B|Theta]], T1),
          append(S1, [A|[B|Theta]], T2),
          append(Phi, [C|[C|Psi]], S1),
          append(Phi, [C|Psi], S),
          append(S1, [B|[A|Theta]], T3))
        ;(append(Phi, [B|[A|E]], T),
          append(Phi, [A|[B|E]], T1),
          append(Phi, [A|[B|E1]], T2),
          append(Psi, [C|[C|Theta]], E1),
          append(Psi, [C|Theta], E),
          append(Phi, [B|[A|E1]], T3))),
        !,
```

```
        assertz(proved(Z = [con(Z1,C), G => T])),
        assertz(proved(Z1 = [inc(Z2,B,A), G => T3])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [con(Z2,C), G => T1])),
    !.


/* (R6,R7) - (E.26.3) */
norm1(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [con(Z2,A), G => T1]),
        proved(Z2 = [_, G => T2]),
        append(Phi, [B|[A|Psi]], T),
        append(Phi, [A|[B|Psi]], T1),
        append(Phi, [A|[A|[B|Psi]]], T2),
    !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        append(Phi, [B|[A|[A|Psi]]], T3),
        gensym(f,F),
        assertz(proved(Z1 = [inc(F,B,A), G => T3])),
        append(Phi, [A|[B|[A|Psi]]], T4),
        assertz(proved(F = [inc(Z2,B,A), G => T4])),
        retract(proved(Z = [inc(Z1,B,A), G => T])),
        retract(proved(Z1 = [con(Z2,A), G => T1])),
    !.

/* (R6,R7) - (E.26.4) */
norm1(proved(Z = [inc(Z1,A,B), G => T])) :-
        proved(Z1 = [con(Z2,A), G => T1]),
        proved(Z2 = [_, G => T2]),
        append(Phi, [A|[B|Psi]], T),
        append(Phi, [B|[A|Psi]], T1),
        append(Phi, [B|[A|[A|Psi]]], T2),
    !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        append(Phi, [A|[A|[B|Psi]]], T3),
        gensym(f,F),
        assertz(proved(Z1 = [inc(F,A,B), G => T3])),
        append(Phi, [A|[B|[A|Psi]]], T4),
        assertz(proved(F = [inc(Z2,A,B), G => T4])),
        retract(proved(Z = [inc(Z1,A,B), G => T])),
        retract(proved(Z1 = [con(Z2,A), G => T1])),
    !.


/* (R6,R10) - (E.27.1,2) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [_, G => T2]),
        proved(Z3 = [_, G => T3]),
    ((append(S, [C|[C|Theta]], T1),
        append(S, [C|Theta], T),
        append(S1, [C|[C|Theta]], T2),
        append(S2, [C|[C|Theta]], T3),
        append(Phi, [A & B|Psi], S),
        append(Phi, [A|Psi], S1),
        append(Phi, [B|Psi], S2),
        append(S1, [C|Theta], T4),
        append(S2, [C|Theta], T5))
    ;(append(Phi, [C|[C|E]], T1),
        append(Phi, [C|E], T),
        append(Phi, [C|[C|E1]], T2),
        append(Phi, [C|[C|E2]], T3),
        append(Psi, [A & B|Theta], E),
        append(Psi, [A|Theta], E1),
        append(Psi, [B|Theta], E2),
        append(Phi, [C|E1], T4),
        append(Phi, [C|E2], T5))),
    !,
        gensym(f,F),
        assertz(proved(Z = [ais(Z1,F,A & B), G => T])),
        assertz(proved(Z1 = [con(Z2,C), G => T4])),
        assertz(proved(F = [con(Z3,C), G => T5])),
        retract(proved(Z = [con(Z1,C), G => T])),
        retract(proved(Z1 = [ais(Z2,Z3,A & B), G => T1])),
    !.


/* (R6,R10) - (E.27.3,4) */
norm1(proved(Z = [con(Z1,A&B), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [ais(Z4,Z5,A & B), G => T2]),
        proved(Z4 = [th(Z6,A), G => T3]),
        proved(Z6 = [_, G => T4]),
```

```
                proved(Z5 = [th(Z7,A), G => T5]),
                proved(Z7 = [K1, G => T6]),
                proved(Z3 = [ais(Z8,Z9,A & B), G => T7]),
                proved(Z8 = [th(Z10,B), G => T8]),
                proved(Z10 = [K2, G => T4]),
                proved(Z9 = [th(Z11,B), G => T9]),
                proved(Z11 = [_, G => T6]),
                append(Phi, [A&B|[A&B|Psi]], T1),
                append(Phi, [A&B|Psi], T),
                append(Phi, [A|Psi], T4),
                append(Phi, [B|Psi], T6),
                append(Phi,[A|[A|Psi]],T3),
                append(Phi,[B|[B|Psi]],T9),
             ((append(Phi,[A|[A&B|Psi]],T2),
                append(Phi,[B|[A&B|Psi]],T7),
                append(Phi,[A|[B|Psi]],T5),
                append(Phi,[B|[A|Psi]],T8))
             ;(append(Phi,[A&B|[A|Psi]],T2),
                append(Phi,[A&B|[B|Psi]],T7),
                append(Phi,[B|[A|Psi]],T5),
                append(Phi,[A|[B|Psi]],T8))),
             !,
                assertz(proved(Z = [ais(Z6,Z11,A & B), G => T])),
                retract(proved(Z = [con(Z1,A & B), G => T])),
             !.


/* (R6,R10) - (E.27.5,6) */
norm1(proved(Z = [con(Z1,A&B), G => T])) :-
                proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
                proved(Z2 = [ais(Z4,Z5,A & B), G => T2]),
                proved(Z4 = [th(Z6,A), G => T3]),
                proved(Z6 = [K1, G => T4]),
                proved(Z5 = [th(Z7,B), G => T5]),
                proved(Z7 = [K1, G => T4]),
                proved(Z3 = [ais(Z8,Z9,A & B), G => T6]),
                proved(Z8 = [th(Z10,A), G => T7]),
                proved(Z10 = [K2, G => T8]),
                proved(Z9 = [th(Z11,B), G => T9]),
                proved(Z11 = [K2, G => T8]),
                append(Phi, [A&B|[A&B|Psi]], T1),
                append(Phi, [A&B|Psi], T),
                append(Phi, [A|Psi], T4),
                append(Phi, [B|Psi], T8),
                append(Phi,[A|[A|Psi]],T3),
                append(Phi,[B|[B|Psi]],T9),
             ((append(Phi,[A|[A&B|Psi]],T2),
                append(Phi,[B|[A&B|Psi]],T6),
                append(Phi,[A|[B|Psi]],T5),
                append(Phi,[B|[A|Psi]],T7))
             ;(append(Phi,[A&B|[A|Psi]],T2),
                append(Phi,[A&B|[B|Psi]],T6),
                append(Phi,[B|[A|Psi]],T5),
                append(Phi,[A|[B|Psi]],T7))),
             !,
                assertz(proved(Z = [ais(Z6,Z11,A & B), G => T])),
                retract(proved(Z = [con(Z1,A & B), G => T])),
             !.


/* (R6,R10) - (E.27.7,8) */
norm1(proved(Z = [con(Z1,A&B), G => T])) :-
                proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
                proved(Z2 = [th(Z4,A), G => T2]),
                proved(Z4 = [ais(Z5,Z6,A & B), G => T]),
                proved(Z5 = [_, G => T3]),
                proved(Z6 = [_, G => T4]),
                proved(Z3 = [th(Z7,B), G => T5]),
                proved(Z7 = [ais(Z8,Z9,A & B), G => T]),
                proved(Z8 = [_, G => T3]),
                proved(Z9 = [_, G => T4]),
                append(Phi, [A&B|[A&B|Psi]], T1),
                append(Phi, [A&B|Psi], T),
                append(Phi, [A|Psi], T3),
                append(Phi, [B|Psi], T4),
             ((append(Phi,[A|[A&B|Psi]],T2),
                append(Phi,[B|[A&B|Psi]],T5))
             ;(append(Phi,[A&B|[A|Psi]],T2),
                append(Phi,[A&B|[B|Psi]],T5))),
             !,
                assertz(proved(Z = [ais(Z5,Z6,A & B), G => T])),
                retract(proved(Z = [con(Z1,A & B), G => T])),
             !.
```

```
/* (R6,R10) - (E.27.9,10) */
norm1(proved(Z = [con(Z1,A&B), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [th(Z4,A), G => T2]),
        proved(Z4 = [ais(Z5,Z6,A & B), G => T]),
        proved(Z5 = [_, G => T3]),
        proved(Z6 = [_, G => T4]),
        proved(Z3 = [ais(Z7,Z8,A & B), G => T5]),
        proved(Z7 = [th(Z9,A), G => T6]),
        proved(Z9 = [_, G => T4]),
        proved(Z8 = [th(Z10,B), G => T7]),
        proved(Z10 = [_, G => T4]),
        append(Phi, [A&B|[A&B|Psi]], T1),
        append(Phi, [A&B|Psi], T),
        append(Phi, [A|Psi], T3),
        append(Phi, [B|Psi], T4),
        append(Phi, [B|[B|Psi]], T7),
      ((append(Phi,[A|[A&B|Psi]],T2),
        append(Phi,[B|[A&B|Psi]],T5),
        append(Phi,[B|[A|Psi]],T6))
      ;(append(Phi,[A&B|[A|Psi]],T2),
        append(Phi,[A&B|[B|Psi]],T5),
        append(Phi,[A|[B|Psi]],T6))),
      !,
        assertz(proved(Z = [ais(Z5,Z6,A & B), G => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
      !.

/* (R6,R10) - (E.27.11,12) */
norm1(proved(Z = [con(Z1,A&B), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [ais(Z4,Z5,A & B), G => T2]),
        proved(Z4 = [th(Z6,A), G => T3]),
        proved(Z6 = [_, G => T4]),
        proved(Z5 = [th(Z7,B), G => T5]),
        proved(Z7 = [_, G => T4]),
        proved(Z3 = [th(Z8,B), G => T6]),
        proved(Z8 = [ais(Z9,Z10,A & B), G => T]),
        proved(Z9 = [_, G => T4]),
        proved(Z10 = [_, G => T7]),
        append(Phi, [A&B|[A&B|Psi]], T1),
        append(Phi, [A&B|Psi], T),
        append(Phi, [A|Psi], T4),
        append(Phi, [B|Psi], T7),
        append(Phi, [A|[A|Psi]], T3),
      ((append(Phi,[A|[A&B|Psi]],T2),
        append(Phi,[B|[A&B|Psi]],T6),
        append(Phi,[A|[B|Psi]],T5))
      ;(append(Phi,[A&B|[A|Psi]],T2),
        append(Phi,[A&B|[B|Psi]],T6),
        append(Phi,[B|[A|Psi]],T5))),
      !,
        assertz(proved(Z = [ais(Z9,Z10,A & B), G => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
      !.

/* (R6,R11) - (E.28) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [aia(Z2,A & B), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [C|[C|Psi]], T1),
        append(Phi, [C|Psi], T),
      !,
        assertz(proved(Z = [aia(Z1,A & B), G => T])),
        assertz(proved(Z1 = [con(Z2,C), G1 => T])),
        retract(proved(Z = [con(Z1,C), G => T])),
        retract(proved(Z1 = [aia(Z2,A & B), G => T1])),
      !.

/* (R6,R12) - (E.29) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
        proved(Z1 = [aia(Z2,Z3,A # B), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        proved(Z3 = [_, G2 => T1]),
        append(Gamma, [A # B|Delta], G),
        append(Gamma, [A|Delta], G1),
```

```
         append(Gamma, [B|Delta], G2),
         append(Phi, [C|[C|Psi]], T1),
         append(Phi, [C|Psi], T),
       !,
         gensym(f,F),
         assertz(proved(Z = [ola(Z1,F,A # B), G => T])),
         assertz(proved(Z1 = [con(Z2,C), G1 => T])),
         assertz(proved(F = [con(Z3,C), G2 => T])),
         retract(proved(Z = [con(Z1,C), G => T])),
         retract(proved(Z1 = [ola(Z2,Z3,A # B), G => T1])),
       !.


/* (R6,R13) - (E.30.1,2) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
         proved(Z1 = [ois(Z2,A # B), G => T1]),
         proved(Z2 = [_, G => T2]),
       ((append(S, [C|[C|Theta]], T1),
         append(S, [C|Theta], T),
         append(S1, [C|[C|Theta]], T2),
         append(Phi, [A # B|Psi], S),
         append(Phi, [A|[B|Psi]], S1),
         append(S1, [C|Theta], T3))
       ;(append(Phi, [C|[C|E]], T1),
         append(Phi, [C|E], T),
         append(Phi, [C|[C|E1]], T2),
         append(Psi, [A # B|Theta], E),
         append(Psi, [A|[B|Theta]], E1),
         append(Phi, [C|E1], T3))),
       !,
         assertz(proved(Z = [ois(Z1,A # B), G => T])),
         assertz(proved(Z1 = [con(Z2,C), G => T3])),
         retract(proved(Z = [con(Z1,C), G => T])),
         retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
       !.


/* (R6,R13) - (E.30.3,4) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
         proved(Z1 = [ois(Z2,A # B), G => T1]),
         proved(Z2 = [ois(Z3,A # B), G => T2]),
         proved(Z3 = [_, G => T3]),
         append(Phi, [A # B|Psi], T),
         append(Phi, [A # B|[A # B|Psi]], T1),
        (append(Phi, [A|[B|[A # B|Psi]]], T2)
        ;append(Phi, [A # B|[A|[B|Psi]]], T2)),
         append(Phi, [A|[B|[A|[B|Psi]]]], T3),
       !,
         assertz(proved(Z = [ois(Z1,A # B), G => T])),
         append(Phi, [A|[B|Psi]], T4),
         assertz(proved(Z1 = [con(Z2,A), G => T4])),
         append(Phi, [A|[A|[B|Psi]]], T5),
         gensym(f,F),
         assertz(proved(Z2 = [con(F,B), G => T5])),
         append(Phi, [A|[A|[B|[B|Psi]]]], T6),
         assertz(proved(F = [inc(Z3,A,B), G => T6])),
         retract(proved(Z = [con(Z1,A # B), G => T])),
         retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
         retract(proved(Z2 = [ois(Z3,A # B), G => T2])),
       !.


/* (R6,R14) - (E.31) */
norm1(proved(Z = [con(Z1,C), G => T])) :-
         proved(Z1 = [hia(Z2,Z3,A -> B), G => T1]),
         proved(Z2 = [_, G1 => [A]]),
         proved(Z3 = [_, G2 => T1]),
         append(G1, [A => B], M),
         append(Delta, [B|Lambda], G2),
         append3(Delta, M, Lambda, G),
         append(Phi, [C|[C|Psi]], T1),
         append(Phi, [C|Psi], T),
       !,
         assertz(proved(Z = [hia(Z2,Z1,A -> B), G => T])),
         assertz(proved(Z1 = [con(Z3,C), G2 => T])),
         retract(proved(Z = [con(Z1,C), G => T])),
         retract(proved(Z1 = [hia(Z2,Z3,A -> B), G => T1])),
       !.
```

```
/* (R7,R10) - (E.32.1,2) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [als(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [_, G => T2]),
        proved(Z3 = [_, G => T3]),
      ((append(Phi, [D|[C|E]], T),
        append(Phi, [C|[D|E]], T1),
        append(Phi, [C|[D|E1]], T2),
        append(Phi, [C|[D|E2]], T3),
        append(Psi, [A & B|Theta], E),
        append(Psi, [A|Theta], E1),
        append(Psi, [B|Theta], E2),
        append(Phi, [D|[C|E1]], T4),
        append(Phi, [D|[C|E2]], T5))
      ;(append(S, [D|[C|Theta]], T),
        append(S, [C|[D|Theta]], T1),
        append(S1, [C|[D|Theta]], T2),
        append(S2, [C|[D|Theta]], T3),
        append(Phi, [A & B|Psi], S),
        append(Phi, [A|Psi], S1),
        append(Phi, [B|Psi], S2),
        append(S1, [D|[C|Theta]], T4),
        append(S2, [D|[C|Theta]], T5))),
      !,
        gensym(f,F),
        assertz(proved(Z = [als(Z1,F,A & B), G => T])),
        assertz(proved(Z1 = [inc(Z2,D,C), G => T4])),
        assertz(proved(F = [inc(Z3,D,C), G => T5])),
        retract(proved(Z = [inc(Z1,D,C), G => T])),
        retract(proved(Z1 = [als(Z2,Z3,A & B), G => T1])),
      !.

/* (R4,R12) - (E.32.3) */
norm1(proved(Z = [inc(Z1,C,A & B), G => T])) :-
        proved(Z1 = [als(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [_, G => T2]),
        proved(Z3 = [_, G => T3]),
        append(Phi,[C|[A & B|Psi]], T),
        append(Phi, [A & B|[C|Psi]], T1),
        append(Phi, [A|[C|Psi]], T2),
        append(Phi, [B|[C|Psi]], T3),
      !,
        gensym(f,F),
        assertz(proved(Z = [als(Z1,F,A & B), G => T])),
        append(Phi, [C|[A|Psi]], T4),
        assertz(proved(Z1 = [inc(Z2,C,A), G => T4])),
        append(Phi, [C|[B|Psi]], T5),
        assertz(proved(F = [inc(Z3,C,B), G => T5])),
        retract(proved(Z = [inc(Z1,C,A & B), G => T])),
        retract(proved(Z1 = [als(Z2,Z3,A & B), G => T1])),
      !.

/* (R4,R12) - (E.32.4) */
norm1(proved(Z = [inc(Z1,A & B,C), G => T])) :-
        proved(Z1 = [als(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [_, G => T2]),
        proved(Z3 = [_, G => T3]),
        append(Phi,[ A & B|[C|Psi]], T),
        append(Phi, [C|[A & B|Psi]], T1),
        append(Phi, [C|[A|Psi]], T2),
        append(Phi, [C|[B|Psi]], T3),
      !,
        gensym(f,F),
        assertz(proved(Z = [als(Z1,F,A & B), G => T])),
        append(Phi, [A|[C|Psi]], T4),
        assertz(proved(Z1 = [inc(Z2,A,C), G => T4])),
        append(Phi, [B|[C|Psi]], T5),
        assertz(proved(F = [inc(Z3,B,C), G => T5])),
        retract(proved(Z = [inc(Z1,A & B,C), G => T])),
        retract(proved(Z1 = [als(Z2,Z3,A & B), G => T1])),
      !.

/* (R7,R11) - (E.33) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
        proved(Z1 = [ala(Z2,A & B), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [D|[C|Psi]], T),
        append(Phi, [C|[D|Psi]], T1),
      !,
```

```
              assertz(proved(Z = [ola(Z1,A & B), G => T])),
              assertz(proved(Z1 = [inc(Z2,D,C), G1 => T])),
              retract(proved(Z = [inc(Z1,D,C), G => T])),
              retract(proved(Z1 = [ola(Z2,A & B), G => T1])),
          !.


/* (R7,R12) - (E.34) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
              proved(Z1 = [ola(Z2,Z3,A#B), G => T1]),
              proved(Z2 = [_, G1 => T1]),
              proved(Z3 = [_, G2 => T1]),
              append(Gamma, [A#B|Delta], G),
              append(Gamma, [A|Delta], G1),
              append(Gamma, [B|Delta], G2),
              append(Phi, [D|[C|Psi]], T),
              append(Phi, [C|[D|Psi]], T1),
          !,
              gensym(f,F),
              assertz(proved(Z = [ola(Z1,F,A#B), G => T])),
              assertz(proved(Z1 = [inc(Z2,D,C), G1 => T])),
              assertz(proved(F = [inc(Z3,D,C), G2 => T])),
              retract(proved(Z = [inc(Z1,D,C), G => T])),
              retract(proved(Z1 = [ola(Z2,Z3,A#B), G => T1])),
          !.


/* (R7,R13) - (E.35.1,2) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
              proved(Z1 = [ois(Z2,A#B), G => T1]),
              proved(Z2 = [_, G => T2]),
          ((append(S, [D|[C|Theta]], T),
              append(S, [C|[D|Theta]], T1),
              append(S1, [C|[D|Theta]], T2),
              append(Phi, [A # B|Psi], S),
              append(Phi, [A|[B|Psi]], S1),
              append(S1, [D|[C|Theta]], T3))
          ;(append(Phi, [D|[C|E]], T),
              append(Phi, [C|[D|E]], T1),
              append(Phi, [C|[D|E1]], T2),
              append(Psi, [A # B|Theta], E),
              append(Psi, [A|[B|Theta]], E1),
              append(Phi, [D|[C|E1]], T3))),
          !,
              assertz(proved(Z = [ois(Z1,A # B), G => T])),
              assertz(proved(Z1 = [inc(Z2,D,C), G => T3])),
              retract(proved(Z = [inc(Z1,D,C), G => T])),
              retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
          !.


/* (R7,R13) - (E.35.3) */
norm1(proved(Z = [inc(Z1,C,A # B), G => T])) :-
              proved(Z1 = [ois(Z2,A # B), G => T1]),
              proved(Z2 = [_, G => T2]),
              append(Phi, [C|[A # B|Theta]], T),
              append(Phi, [A # B|[C|Theta]], T1),
              append(Phi, [A|[B|[C|Theta]]], T2),
          !,
              assertz(proved(Z = [ois(Z1,A # B), G => T])),
              append(Phi, [C|[A|[B|Theta]]], T3),
              gensym(f,F),
              assertz(proved(Z1 = [inc(F,C,A), G => T3])),
              append(Phi, [A|[C|[B|Theta]]], T4),
              assertz(proved(F = [inc(Z2,C,B), G => T4])),
              retract(proved(Z = [inc(Z1,C,A # B), G => T])),
              retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
          !.


/* (R7,R13) - (E.35.4) */
norm1(proved(Z = [inc(Z1,A # B,C), G => T])) :-
              proved(Z1 = [ois(Z2,A # B), G => T1]),
              proved(Z2 = [_, G => T2]),
              append(Phi, [A # B|[C|Theta]], T),
              append(Phi, [C|[A # B|Theta]], T1),
              append(Phi, [C|[A|[B|Theta]]], T2),
          !,
              assertz(proved(Z = [ois(Z1,A # B), G => T])),
              append(Phi, [A|[B|[C|Theta]]], T3),
              gensym(f,F),
              assertz(proved(Z1 = [inc(F,B,C), G => T3])),
              append(Phi, [A|[C|[B|Theta]]], T4),
```

```
            assertz(proved(F = [inc(Z2,A,C), G => T4])),
            retract(proved(Z = [inc(Z1,A # B,C), G => T])),
            retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        !.


/* (R7,R14) - (E.36) */
norm1(proved(Z = [inc(Z1,D,C), G => T])) :-
            proved(Z1 = [hia(Z2,Z3,A -> B), G => T1]),
            proved(Z2 = [_, G1 => [A]]),
            proved(Z3 = [_, G2 => T1]),
            append(G1, [A -> B], M),
            append(Delta, [B|Lambda], G2),
            append3(Delta, M, Lambda, G),
            append(Phi, [D|[C|Psi]], T),
            append(Phi, [C|[D|Psi]], T1),
        !,
            assertz(proved(Z = [hia(Z2,Z1,A -> B), G => T])),
            assertz(proved(Z1 = [inc(Z3,D,C), G2 => T])),
            retract(proved(Z = [inc(Z1,D,C), G => T])),
            retract(proved(Z1 = [hia(Z2,Z3,A -> B), G => T1])),
        !.


/* (R2,R3,R11) - (E.37.1,2) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
            proved(Z1 = [aia(Z2,A & B), G1 => T]),
            proved(Z2 = [th(Z3,A), G2 => T]),
            proved(Z3 = [th(Z4,B), G3 => T]),
            proved(Z4 = [K, G => T]),
            append(Gamma, [A & B|[A & B|Delta]], G1),
        ((append(Gamma, [A & B|[A|[B|Delta]]], G2),
            append(Gamma, [A & B|[B|Delta]], G3))
        ;(append(Gamma, [A|[B|[A & B|Delta]]], G2),
            append(Gamma, [B|[A & B|Delta]], G3))),
            append(Gamma, [A & B|Delta], G),
        !,
            assertz(proved(Z = [K, G => T])),
            retract(proved(Z4 = [K, G => T])),
            retract(proved(Z3 = [th(Z4,B), G3 => T])),
            retract(proved(Z2 = [th(Z3,A), G2 => T])),
            retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
            retract(proved(Z = [con(Z1,A & B), G => T])),
        !.


/* (R2,R3,R11) - (E.37.3) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
          proved(Z1 = [aia(Z2,A & B), G1 => T]),
          proved(Z2 = [th(Z3,A), G2 => T]),
          proved(Z3 = [aia(Z4,A & B), G3 => T]),
          proved(Z4 = [_, G4 => T]),
          append(Gamma, [A & B|Delta], G),
          append(Gamma, [A & B|[A & B|Delta]], G1),
          append(Gamma, [A & B|[A|[B|Delta]]], G2),
          append(Gamma, [A & B|[B|Delta]], G3),
          append(Gamma, [A|[B|[B|Delta]]], G4),
      !,
          assertz(proved(Z = [aia(Z1,A & B), G => T])),
          append(Gamma, [A|[B|Delta]], G5),
          assertz(proved(Z1 = [con(Z4,B), G5 => T])),
          retract(proved(Z = [con(Z1,A & B), G => T])),
          retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
          retract(proved(Z2 = [th(Z3,A), G2 => T])),
          retract(proved(Z3 = [aia(Z4,A & B), G3 => T])),
      !.


/* (R2,R3,R11) - (E.37.4) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
          proved(Z1 = [aia(Z2,A & B), G1 => T]),
          proved(Z2 = [th(Z3,A), G2 => T]),
          proved(Z3 = [aia(Z4,A & B), G3 => T]),
          proved(Z4 = [_, G4 => T]),
          append(Gamma, [A & B|Delta], G),
          append(Gamma, [A & B|[A & B|Delta]], G1),
          append(Gamma, [A|[B|[A & B|Delta]]], G2),
          append(Gamma, [B|[A & B|Delta]], G3),
          append(Gamma, [B|[A|[B|Delta]]], G4),
      !,
          assertz(proved(Z = [aia(Z1,A & B), G => T])),
          append(Gamma, [A|[B|Delta]], G5),
          assertz(proved(Z1 = [con(Z2,B), G5 => T])),
```

```
        append(Gamma, [A|[B|[B|Delta]]], G6),
        assertz(proved(Z2 = [inc(Z4,A,B), G6 => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
        retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
        retract(proved(Z2 = [th(Z3,A), G2 => T])),
        retract(proved(Z3 = [aia(Z4,A & B), G3 => T])),
      !.

/* (R2,R3,R11) - (E.37.5) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
        proved(Z1 = [aia(Z2,A & B), G1 => T]),
        proved(Z2 = [th(Z3,B), G2 => T]),
        proved(Z3 = [aia(Z4,A & B), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A & B|[A & B|Delta]], G1),
        append(Gamma, [A & B|[A|[B|Delta]]], G2),
        append(Gamma, [A & B|[A|Delta]], G3),
        append(Gamma, [A|[B|[A|Delta]]], G4),
      !,
        assertz(proved(Z = [aia(Z1,A & B), G => T])),
        append(Gamma, [A|[B|Delta]], G5),
        assertz(proved(Z1 = [con(Z2,B), G5 => T])),
        append(Gamma, [A|[A|[B|Delta]]], G6),
        assertz(proved(Z2 = [inc(Z4,A,B), G6 => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
        retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
        retract(proved(Z2 = [th(Z3,B), G2 => T])),
        retract(proved(Z3 = [aia(Z4,A & B), G3 => T])),
      !.


/* (R2,R3,R11) - (E.37.6) */
norm1(proved(Z = [con(Z1,A & B), G => T])) :-
        proved(Z1 = [aia(Z2,A & B), G1 => T]),
        proved(Z2 = [th(Z3,B), G2 => T]),
        proved(Z3 = [aia(Z4,A & B), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A & B|[A & B|Delta]], G1),
        append(Gamma, [A|[B|[A & B|Delta]]], G2),
        append(Gamma, [A|[A & B|Delta]], G3),
        append(Gamma, [A|[A|[B|Delta]]], G4),
      !,
        assertz(proved(Z = [aia(Z1,A & B), G => T])),
        append(Gamma, [A|[B|Delta]], G5),
        assertz(proved(Z1 = [con(Z4,A), G5 => T])),
        retract(proved(Z = [con(Z1,A & B), G => T])),
        retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
        retract(proved(Z2 = [th(Z3,B), G2 => T])),
        retract(proved(Z3 = [aia(Z4,A & B), G3 => T])),
      !.


/* (R5,R6,R13) - (E.38.1,2) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
         proved(Z1 = [ois(Z2,A # B), G => T1]),
         proved(Z2 = [th(Z3,A), G => T2]),
         proved(Z3 = [th(Z4,B), G => T3]),
         proved(Z4 = [K, G => T]),
         append(Phi, [A # B|[A # B|Psi]], T1),
       ((append(Phi, [A # B|[A|[B|Psi]]], T2),
         append(Phi, [A # B|[B|Psi]], T3))
       ;(append(Phi, [A|[B|[A # B|Psi]]], T2),
         append(Phi, [B|[A # B|Psi]], T3))),
         append(Phi, [A # B|Psi], T),
       !,
         assertz(proved(Z = [K, G => T])),
         retract(proved(Z4 = [K, G => T])),
         retract(proved(Z3 = [th(Z4,B), G => T3])),
         retract(proved(Z2 = [th(Z3,A), G => T2])),
         retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
         retract(proved(Z = [con(Z1,A # B), G => T])),
       !.


/* (R2,R3,R11) - (E.38.3) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G => T1]),
        proved(Z2 = [th(Z3,A), G => T2]),
        proved(Z3 = [ois(Z4,A # B), G => T3]),
        proved(Z4 = [_, G => T4]),
        append(Phi, [A # B|Psi], T),
```

```
        append(Phi, [A # B|[A # B|Psi]], T1),
        append(Phi, [A # B|[A|[B|Psi]]], T2),
        append(Phi, [A # B|[B|Psi]], T3),
        append(Phi, [A|[B|[B|Psi]]], T4),
    !,
        assertz(proved(Z = [ois(Z1,A # B), G => T])),
        append(Phi, [A|[B|Psi]], T5),
        assertz(proved(Z1 = [con(Z4,B), G => T5])),
        retract(proved(Z = [con(Z1,A # B), G => T])),
        retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        retract(proved(Z2 = [th(Z3,A), G => T2])),
        retract(proved(Z3 = [ois(Z4,A # B), G => T3])),
    !.


/* (R2,R3,R11) - (E.38.4) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G => T1]),
        proved(Z2 = [th(Z3,A), G => T2]),
        proved(Z3 = [ois(Z4,A # B), G => T3]),
        proved(Z4 = [_, G => T4]),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A # B|[A # B|Psi]], T1),
        append(Phi, [A|[B|[A # B|Psi]]], T2),
        append(Phi, [B|[A # B|Psi]], T3),
        append(Phi, [B|[A|[B|Psi]]], T4),
    !,
        assertz(proved(Z = [ois(Z1,A # B), G => T])),
        append(Phi, [A|[B|Psi]], T5),
        assertz(proved(Z1 = [con(Z2,B), G => T5])),
        append(Phi, [A|[B|[B|Psi]]], T6),
        assertz(proved(Z2 = [inc(Z4,A,B), G => T6])),
        retract(proved(Z = [con(Z1,A # B), G => T])),
        retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        retract(proved(Z2 = [th(Z3,A), G => T2])),
        retract(proved(Z3 = [ois(Z4,A # B), G => T3])),
    !.


/* (R2,R3,R11) - (E.38.5) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G => T1]),
        proved(Z2 = [th(Z3,B), G => T2]),
        proved(Z3 = [ois(Z4,A # B), G => T3]),
        proved(Z4 = [_, G => T4]),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A # B|[A # B|Psi]], T1),
        append(Phi, [A # B|[A|[B|Psi]]], T2),
        append(Phi, [A # B|[A|Psi]], T3),
        append(Phi, [A|[B|[A|Psi]]], T4),
    !,
        assertz(proved(Z = [ois(Z1,A # B), G => T])),
        append(Phi, [A|[B|Psi]], T5),
        assertz(proved(Z1 = [con(Z2,B), G => T5])),
        append(Phi, [A|[A|[B|Psi]]], T6),
        assertz(proved(Z2 = [inc(Z4,A,B), G => T6])),
        retract(proved(Z = [con(Z1,A # B), G => T])),
        retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        retract(proved(Z2 = [th(Z3,B), G => T2])),
        retract(proved(Z3 = [ois(Z4,A # B), G => T3])),
    !.


/* (R2,R3,R11) - (E.38.6) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G => T1]),
        proved(Z2 = [th(Z3,B), G => T2]),
        proved(Z3 = [ois(Z4,A # B), G => T3]),
        proved(Z4 = [_, G => T4]),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A # B|[A # B|Psi]], T1),
        append(Phi, [A|[B|[A # B|Psi]]], T2),
        append(Phi, [A|[A # B|Psi]], T3),
        append(Phi, [A|[A|[B|Psi]]], T4),
    !,
        assertz(proved(Z = [ois(Z1,A # B), G => T])),
        append(Phi, [A|[B|Psi]], T5),
        assertz(proved(Z1 = [con(Z4,A), G => T5])),
        retract(proved(Z = [con(Z1,A # B), G => T])),
        retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        retract(proved(Z2 = [th(Z3,B), G => T2])),
        retract(proved(Z3 = [ois(Z4,A # B), G => T3])),
    !.
```

```
/* (R3,R12,R14) - (E.39) */
norm1(proved(Z = [con(Z1,A # B), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T]),
        proved(Z2 = [oia(Z4,Z5,A#B), [A#B|Gamma] => [C]]),
        proved(Z4 = [_, [A|Gamma] => [C]]),
        proved(Z5 = [_, [B|Gamma] => [C]]),
        proved(Z3 = [oia(Z6,Z7,A # B), G2 => T]),
        proved(Z6 = [_, G3 => T]),
        proved(Z7 = [_, G4 => T]),
        append(Delta,[A#B|[A#B|E]],G1),
        append(Delta,[A#B|E],G),
        append(Gamma,[C -> D|Lambda],E),
        append(Delta,[A # B|[D|Lambda]],G2),
        append(Delta,[A|[D|Lambda]],G3),
        append(Delta,[B|[D|Lambda]],G4),
      !,
        assertz(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        append(Delta,[A|E],G5),
        assertz(proved(Z1 = [con(Z3,A), G5 => T])),
        append(Delta,[A|[A|E]],G6),
        assertz(proved(Z3 = [hia(Z4,Z6,C->D), G6 => T])),
        append(Delta,[B|E],G7),
        assertz(proved(Z2 = [con(Z8,B), G7 => T])),
        append(Delta,[B|[B|E]],G8),
        assertz(proved(Z8 = [hia(Z5,Z7,C->D), G8 => T])),
        retract(proved(Z1 = [hia(Z2,Z3,C->D), G1 => T])),
        retract(proved(Z2=[oia(Z4,Z5,A#B),[A#B|Gamma]=>[C]])),
        retract(proved(Z3 = [oia(Z6,Z7,A # B), G2 => T])),
        retract(proved(Z = [con(Z1,A # B), G => T])),
      !.

/* (R4,R12,R14) - (E.40.1) */
norm1(proved(Z = [inc(Z1,A # B,E), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T]),
        proved(Z2 = [oia(Z4,Z5,A#B),[A#B|Gamma] => [C]]),
        proved(Z4 = [_, [A|Gamma] => [C]]),
        proved(Z5 = [_, [B|Gamma] => [C]]),
        proved(Z3 = [K, G2 => T]),
        append(Delta,[E|[A#B|R]],G1),
        append(Delta,[A#B|[E|R]],G),
        append(Gamma,[C -> D|Lambda],R),
        append(Delta,[E|[D|Lambda]],G2),
      !,
        assertz(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        append(Delta,[A|[E|R]],G3),
        assertz(proved(Z1 = [inc(Z6,A,E), G3 => T])),
        append(Delta,[E|[A|R]],G4),
        assertz(proved(Z6 = [hia(Z4,Z3,C->D), G4 => T])),
        append(Delta,[B|[E|R]],G5),
        assertz(proved(Z2 = [inc(Z7,B,E), G5 => T])),
        append(Delta,[E|[B|R]],G6),
        assertz(proved(Z7 = [hia(Z5,Z8,C->D), G6 => T])),
        assertz(proved(Z8 = [K, G2 => T])),
        retract(proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T])),
        retract(proved(Z2=[oia(Z4,Z5,A#B),[A#B|Gamma]=>[C]])),
        retract(proved(Z = [inc(Z1,A # B,E), G => T])),
      !.

/* (R4,R12,R14) - (E.40.2) */
norm1(proved(Z = [inc(Z1,E,A # B), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T]),
        proved(Z2 = [K, [E|Gamma] => [C]]),
        proved(Z3 = [oia(Z4,Z5,A # B), G2 => T]),
        proved(Z4 = [_, G3 => T]),
        proved(Z5 = [_, G4 => T]),
        append(Delta,[E|[A#B|R]],G),
        append(Delta,[A#B|[E|R]],G1),
        append(Gamma,[C -> D|Lambda],R),
        append(Delta,[A # B|[D|Lambda]],G2),
        append(Delta,[A|[D|Lambda]],G3),
        append(Delta,[B|[D|Lambda]],G4),
      !,
        assertz(proved(Z = [oia(Z1,Z3,A # B), G => T])),
        append(Delta,[E|[A|R]],G5),
        assertz(proved(Z1 = [inc(Z6,E,A), G5 => T])),
        append(Delta,[A|[E|R]],G6),
        assertz(proved(Z6 = [hia(Z2,Z4,C -> D), G6 => T])),
        append(Delta,[E|[B|R]],G7),
        assertz(proved(Z3 = [inc(Z7,E,B), G7 => T])),
```

```
            append(Delta,[B|[E|R]],G8),
            assertz(proved(Z7 = [hia(Z8,Z5,C -> D), G8 => T])),
            assertz(proved(Z8 = [K, [E|Gamma] => [C]])),
            retract(proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T])),
            retract(proved(Z3 = [oia(Z4,Z5,A # B), G2 => T])),
            retract(proved(Z = [inc(Z1,E,A # B), G => T])),
        !.

/* (R4,R12,R14) - (E.40.3) */
norm1(proved(Z = [inc(Z1,A # B,C -> D), G => T])) :-
            proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T]),
            proved(Z2 = [K, Gamma => [C]]),
            proved(Z3 = [oia(Z4,Z5,A # B), G2 => T]),
            proved(Z4 = [_, G3 => T]),
            proved(Z5 = [_, G4 => T]),
            append(S,[C->D|[A#B|Lambda]],G1),
            append(S,[A#B|[C->D|Lambda]],G),
            append(Delta,Gamma,S),
            append(Delta,[D|[A # B|Lambda]],G2),
            append(Delta,[D|[A|Lambda]],G3),
            append(Delta,[D|[B|Lambda]],G4),
        !,
            assertz(proved(Z = [oia(Z1,Z3,A # B), G => T])),
            append(S,[A|[C->D|Lambda]],G5),
            assertz(proved(Z1 = [inc(Z6,A,C->D), G5 => T])),
            append(S,[C->D|[A|Lambda]],G6),
            assertz(proved(Z6 = [hia(Z2,Z4,C -> D), G6 => T])),
            append(S,[B|[C->D|Lambda]],G7),
            assertz(proved(Z3 = [inc(Z7,B,C->D), G7 => T])),
            append(S,[C->D|[B|Lambda]],G8),
            assertz(proved(Z7 = [hia(Z8,Z5,C -> D), G8 => T])),
            assertz(proved(Z8 = [K, Gamma => [C]])),
            retract(proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T])),
            retract(proved(Z3 = [oia(Z4,Z5,A # B), G2 => T])),
            retract(proved(Z = [inc(Z1,A # B,C -> D), G => T])),
        !.

/* (R4,R12,R14) - (E.40.4) */
norm1(proved(Z = [inc(Z1,C -> D,A # B), G => T])) :-
            proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T]),
            proved(Z2 = [oia(Z4,Z5,A#B),[Gamma|A#B] => [C]]),
            proved(Z4 = [_, [Gamma|A] => [C]]),
            proved(Z5 = [_, [Gamma|B] => [C]]),
            proved(Z3 = [K, G2 => T]),
            append(S,[A#B|[C->D|Lambda]],G1),
            append(S,[C->D|[A#B|Lambda]],G),
            append(Delta,Gamma,S),
            append(Delta,[D|Lambda],G2),
        !,
            assertz(proved(Z = [oia(Z1,Z2,A # B), G => T])),
            append(S,[C->D|[A|Lambda]],G3),
            assertz(proved(Z1 = [inc(Z6,C -> D,A), G3 => T])),
            append(S,[A|[C->D|Lambda]],G4),
            assertz(proved(Z6 = [hia(Z4,Z3,C -> D), G4 => T])),
            append(S,[C->D|[B|Lambda]],G5),
            assertz(proved(Z2 = [inc(Z7,C -> D,B), G5 => T])),
            append(S,[B|[C->D|Lambda]],G6),
            assertz(proved(Z7 = [hia(Z5,Z8,C -> D), G6 => T])),
            assertz(proved(Z8 = [K, G2 => T])),
            retract(proved(Z1 = [hia(Z2,Z3,C -> D), G1 => T])),
            retract(proved(Z2=[oia(Z4,Z5,A#B),[Gamma|A#B]=>[C]])),
            retract(proved(Z = [inc(Z1,C->D,A#B), G => T])),
        !.

/* (R2) - (E.41.1) */
norm1(proved(Z = [th(Z1,A & B), G => T])) :-
            proved(Z1 = [_, G1 => T]),
            append(Gamma, [A & B|Delta], G),
            append(Gamma, Delta, G1),
        !,
            gensym(f,F1),
            assertz(proved(Z = [aia(F1,A & B), G => T])),
            append(Gamma, [A|[B|Delta]], G2),
            gensym(f,F2),
            assertz(proved(F1 = [th(F2,A), G2 => T])),
            append(Gamma, [B|Delta], G3),
            assertz(proved(F2 = [th(Z1,B), G3 => T])),
            retract(proved(Z = [th(Z1,A & B), G => T])),
        !.
```

```
/* (R2) - (E.41.2) */
norm1(proved(Z = [th(Z1,A # B), G => T])) :-
        proved(Z1 = [K, G1 => T]),
        append(Gamma, [A # B|Delta], G),
        append(Gamma, Delta, G1),
      ((thry(bicart), Gamma = [], Delta = [])
      ; thry(dbicart)
      ; thry(bicartcl)),
      !,
        gensym(f,F1),
        gensym(f,F2),
        assertz(proved(Z = [ola(F1,F2,A # B), G => T])),
        append(Gamma, [A|Delta], G2),
        append(Gamma, [B|Delta], G3),
        assertz(proved(F1 = [th(Z1,A), G2 => T])),
        assertz(proved(F2 = [th(Z1,B), G3 => T])),
        branch(F2),
        retract(proved(Z = [th(Z1,A # B), G => T])),
      !.


/* (R5) - (E.42.1) */
norm1(proved(Z = [th(Z1,A & B), G => T])) :-
        proved(Z1 = [K, G => T1]),
        append(Phi, [A & B|Psi], T),
        append(Phi, Psi, T1),
      ((thry(bicart), Phi = [], Psi = [])
      ; thry(dbicart)
      ; thry(bicartcl)),
      !,
        gensym(f,F1),
        gensym(f,F2),
        assertz(proved(Z = [als(F1,F2,A & B), G => T])),
        append(Phi, [A|Psi], T2),
        assertz(proved(F1 = [th(Z1,A), G => T2])),
        append(Phi, [B|Psi], T3),
        assertz(proved(F2 = [th(Z1,B), G => T3])),
        branch(F2),
        retract(proved(Z = [th(Z1,A & B), G => T])),
      !.


/* (R5) - (E.42.2) */
norm1(proved(Z = [th(Z1,A # B), G => T])) :-
        proved(Z1 = [_, G => T1]),
        append(Phi, [A # B|Psi], T),
        append(Phi, Psi, T1),
      !,
        gensym(f,F1),
        assertz(proved(Z = [ols(F1,A # B), G => T])),
        gensym(f,F2),
        append(Phi, [A|[B|Psi]], T2),
        assertz(proved(F1 = [th(F2,A), G => T2])),
        append(Phi, [B|Psi], T3),
        assertz(proved(F2 = [th(Z1,B), G => T3])),
        retract(proved(Z = [th(Z1,A # B), G => T])),
      !.



norm1(proved(Z)) :-
                    retract(proved(Z)),
                    assertz(proved(Z)), !.
```

```
                                 /* Normalization Program   */

normal2 :-
          main(Z),
          repeat,
          do2_step,
        (((thry(cartcl) ; thry(bicartcl)), not(second) )
        ;(write('Normal proof is :'), nl,
          write('================  '), nl, nl,
          printall(Z), nl, nl, nl) ).


do2_step          :-
                  main(X),
                  put_in_list(X,L),
                  assertz(proved(mark)),
                  proved(Z),
                  norm2(proved(Z)),
                  Z = mark,
                  expinc,
                  expcon,
                  expth,
                  put_in_list(X,L1),
                  abolish(proved,1),
                  ass_list(L1), !,
                  L == L1.


ass_list([]).
ass_list([X|Y]) :-
                   assert(proved(X)),
                   ass_list(Y).


/* (R2,R3) - (D.2.1,2) */
norm2(proved(Z = [con(Z1,A), G => T])) :-
          proved(Z1 = [th(Z2,B), G1 => T]),
          proved(Z2 = [_, G2 => T]),
        ((append(Gamma, [A|[A|E]], G1),
          append(Gamma, [A|E], G),
          append(Gamma, [A|[A|E1]], G2),
          append(Delta, [B|Lambda], E),
          append(Delta, Lambda, E1),
          append(Gamma, [A|E1], G3))
        ;(append(S, [A|[A|Lambda]], G1),
          append(S, [A|Lambda], G),
          append(S1, [A|[A|Lambda]], G2),
          append(Gamma, [B|Delta], S),
          append(Gamma, Delta, S1),
          append(S1, [A|Lambda], G3))),
        !,
          assertz(proved(Z = [th(Z1,B), G => T])),
          assertz(proved(Z1 = [con(Z2,A), G3 => T])),
          retract(proved(Z = [con(Z1,A), G => T])),
          retract(proved(Z1 = [th(Z2,B), G1 => T])),
        !.


/* (R2,R3) - (D.2.3,4) */
norm2(proved(Z = [con(Z1,A), G => T])) :-
          proved(Z1 = [th(Z2,A), G1 => T]),
          proved(Z2 = [F, G => T]),
          append(Gamma, [A|[A|Delta]], G1),
          append(Gamma, [A|Delta], G),
        !,
          assertz(proved(Z = [F, G => T])),
          retract(proved(Z = [con(Z1,A), G => T])),
          retract(proved(Z1 = [th(Z2,A), G1 => T])),
          retract(proved(Z2 = [F, G => T])),
        !.


/* (R2 , R4) - (D.3.1,2) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
          proved(Z1 = [inc(Z2,B,A), G1 => T]),
          proved(Z2 = [_, G2 => T]),
        ((append(Gamma, [B|[A|E]], G),
          append(Gamma, [B|[A|E1]], G1),
          append(Gamma, [A|[B|E1]], G2),
          append(Delta, [C|Lambda], E),
          append(Delta, Lambda, E1),
          append(Gamma, [A|[B|E]], G3))
        ;(append(S, [B|[A|Lambda]], G),
```

```
            append(S1, [B|[A|Lambda]], G1),
            append(S1, [A|[B|Lambda]], G2),
            append(Gamma, [C|Delta], S),
            append(Gamma, Delta, S1),
            append(S, [A|[B|Lambda]], G3))),
        !,
            assertz(proved(Z = [inc(Z1,B,A), G => T])),
            assertz(proved(Z1 = [th(Z2,C), G3 => T])),
            retract(proved(Z1 = [inc(Z2,B,A), G1 => T])),
            retract(proved(Z = [th(Z1,C), G => T])),
        !.


/* (R2,R4) - (D.3.3) */
norm2(proved(Z = [inc(Z1,B,A), G => T])) :-
            proved(Z1 = [th(Z2,B), G1 => T]),
            proved(Z2 = [F, G2 => T]),
            append(Gamma,[A|[B|Delta]],G1),
            append(Gamma,[B|[A|Delta]],G),
            append(Gamma,[A|Delta],G2),
        !,
            assertz(proved(Z = [th(Z2,B), G => T])),
            retract(proved(Z1 = [th(Z2,B), G1 => T])),
            retract(proved(Z = [inc(Z1,B,A), G => T])),
        !.


/* (R2,R4) - (D.3.4) */
norm2(proved(Z = [inc(Z1,A,B), G => T])) :-
            proved(Z1 = [th(Z2,B), G1 => T]),
            proved(Z2 = [F, G2 => T]),
            append(Gamma,[A|[B|Delta]],G),
            append(Gamma,[B|[A|Delta]],G1),
            append(Gamma,[A|Delta],G2),
        !,
            assertz(proved(Z = [th(Z2,B), G => T])),
            retract(proved(Z1 = [th(Z2,B), G1 => T])),
            retract(proved(Z = [inc(Z1,A,B), G => T])),
        !.


/* (R2,R5) - (D.4) */
norm2(proved(Z = [th(Z1,B), G => T])) :-
            proved(Z1 = [th(Z2,A), G => T1]),
            proved(Z2 = [_, G1 => T1]),
            append(Gamma, [A|Delta], G),
            append(Gamma, Delta, G1),
            append(Phi, [B|Psi], T),
            append(Phi, Psi, T1),
        !,
            assertz(proved(Z = [th(Z1,A), G => T])),
            assertz(proved(Z1 = [th(Z2,B), G1 => T])),
            retract(proved(Z1 = [th(Z2,A), G => T1])),
            retract(proved(Z = [th(Z1,B), G => T])),
        !.


/* (R2,R10) - (D.7) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
            proved(Z1 = [th(Z3,C), G => T1]),
            proved(Z2 = [th(Z4,C), G => T2]),
            proved(Z3 = [_, G1 => T1]),
            proved(Z4 = [_, G1 => T2]),
            append(Gamma, [C|Delta], G),
            append(Gamma, Delta, G1),
            append(Phi, [A & B|Psi], T),
            append(Phi, [A|Psi], T1),
            append(Phi, [B|Psi], T2),
        !,
            assertz(proved(Z = [th(Z1,C), G => T])),
            assertz(proved(Z1 = [ais(Z3,Z4,A & B), G1 => T])),
            retract(proved(Z1 = [th(Z3,C), G => T1])),
            retract(proved(Z2 = [th(Z4,C), G => T2])),
            retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        !.

/* (R2,R11) - (D.8.1,2) */
norm2(proved(Z = [aia(Z1,A & B), G => T])) :-
            proved(Z1 = [th(Z2,C), G1 => T]),
            proved(Z2 = [_, G2 => T]),
        ((append(S, [A & B|Lambda], G),
            append(S, [A|[B|Lambda]], G1),
```

```
            append(S1, [A|[B|Lambda]], G2),
            append(Gamma, [C|Delta], S),
            append(Gamma, Delta, S1),
            append(S1, [A & B|Lambda], G3))
          ;(append(Gamma, [A & B|E], G),
            append(Gamma, [A|[B|E]], G1),
            append(Gamma, [A|[B|E1]], G2),
            append(Delta, [C|Lambda], E),
            append(Delta, Lambda, E1),
            append(Gamma, [A & B|E1], G3))),
          !,
            assertz(proved(Z = [th(Z1,C), G => T])),
            assertz(proved(Z1 = [aia(Z2,A & B), G3 => T])),
            retract(proved(Z1 = [th(Z2,C), G1 => T])),
            retract(proved(Z = [aia(Z1,A & B), G => T])),
          !.


/* (R2,R12) - (D.9.1,2) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
            proved(Z1 = [th(Z3,C), G1 => T]),
            proved(Z2 = [th(Z4,C), G2 => T]),
            proved(Z3 = [_, G3 => T]),
            proved(Z4 = [_, G4 => T]),
          ((append(S, [A # B|Lambda], G),
            append(S, [A|Lambda], G1),
            append(S, [B|Lambda], G2),
            append(S1, [A|Lambda], G3),
            append(S1, [B|Lambda], G4),
            append(Gamma, [C|Delta], S),
            append(Gamma, Delta, S1),
            append(S1, [A # B|Lambda], G5))
          ;(append(Gamma, [A # B|E], G),
            append(Gamma, [A|E], G1),
            append(Gamma, [B|E], G2),
            append(Gamma, [A|E1], G3),
            append(Gamma, [B|E1], G4),
            append(Delta, [C|Lambda], E),
            append(Delta, Lambda, E1),
            append(Gamma, [A # B|E1], G5))),
          !,
            assertz(proved(Z = [th(Z1,C), G => T])),
            assertz(proved(Z1 = [oia(Z3,Z4,A # B), G5 => T])),
            retract(proved(Z1 = [th(Z3,C), G1 => T])),
            retract(proved(Z2 = [th(Z4,C), G2 => T])),
            retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
          !.


/* (R2,R13) - (D.10) */
norm2(proved(Z = [ois(Z1,A # B), G => T])) :-
            proved(Z1 = [th(Z2,C), G => T1]),
            proved(Z2 = [_, G1 => T1]),
          !,
            assertz(proved(Z1 = [ois(Z2,A # B), G1 => T])),
            assertz(proved(Z = [th(Z1,C), G => T])),
            retract(proved(Z1 = [th(Z2,C), G => T1])),
            retract(proved(Z = [ois(Z1,A # B), G => T])),
          !.


/* (R2,R14) - (D.11.1) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
            proved(Z1 = [th(Z3,C), G1 => [A]]),
            proved(Z2 = [_, G2 => T]),
            proved(Z3 = [_, G3 => [A]]),
            append(Gamma, [C|Delta], G1),
            append(Gamma, Delta, G3),
            append(Lambda, [B|Zeta], G2),
            append3(Lambda, G1, [A -> B|Zeta], G),
          !,
            assertz(proved(Z = [th(Z1,C), G => T])),
            append3(Lambda, G3, [A -> B|Zeta], G4),
            assertz(proved(Z1 = [hia(Z3,Z2,A -> B), G4 => T])),
            retract(proved(Z1 = [th(Z3,C), G1 => [A]])),
            retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
          !.


/* (R2,R14) - (D.11.2,3) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
            proved(Z1 = [_, G1 => [A]]),
            proved(Z2 = [th(Z3,C), G2 => T]),
```

```
        proved(Z3 = [_, G3 => T]),
    ((append(Delta, [C|E], G2),
        append(Delta, E, G3),
        append(Lambda, [B|Zeta], E),
        append4(Delta, [C|Lambda], G1, [A -> B|Zeta], G),
        append4(Delta, Lambda, G1, [A -> B|Zeta], G4))
    ;(append(S, [C|Zeta], G2),
        append(S, Zeta, G3),
        append(Delta, [B|Lambda], S),
        append4(Delta, G1, [A -> B|Lambda], [C|Zeta], G),
        append4(Delta, G1, [A -> B|Lambda], Zeta, G4))),
    !,
        assertz(proved(Z = [th(Z2,C), G => T])),
        assertz(proved(Z2 = [hia(Z1,Z3,A -> B),G4 => T])),
        retract(proved(Z2 = [th(Z3,C), G2 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
    !.


/* (R2,R14) - (D.11.4) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [K, Gamma => [C]]),
        proved(Z2 = [th(Z3,B), G1 => T]),
        proved(Z3 = [_, G2 => T]),
        append(Delta, [B|Lambda], G1),
        append(Delta, Lambda, G2),
        append3(Delta, Gamma, [A -> B|Lambda], G),
    !,
        assertz(proved(Z = [th(Z1,Gamma), G => T])),
        append(Delta, [A -> B|Lambda], G3),
        assertz(proved(Z1 = [th(Z3,A -> B), G3 => T])),
        retract(proved(Z1 = [K, Gamma => [C]])),
        retract(proved(Z2 = [th(Z3,B), G1 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
    !.


/* (R2,R15) - (D.12.1,2) */
norm2(proved(Z = [his(Z1, A -> B), G => [A -> B]])) :-
        proved(Z1 = [th(Z2,C), G1 => [B]]),
        proved(Z2 = [_, G2 => [B]]),
    ((append(S, [C|Lambda], G),
        append(S1, [C|Lambda], G1),
        append(S1, Lambda, G2),
        append(Gamma, [A|Delta], S1),
        append(Gamma, Delta, S),
        append(S, Lambda, G3))
    ;(append(Gamma, [C|E], G),
        append(Gamma, [C|E1], G1),
        append(Gamma, E1, G2),
        append(Delta, [A|Lambda], E1),
        append(Delta, Lambda, E),
        append(Gamma, E, G3))),
    !,
        assertz(proved(Z = [th(Z1,C), G => [A -> B]])),
        assertz(proved(Z1 = [his(Z2,A->B),G3 => [A->B]])),
        retract(proved(Z1 = [th(Z2,C), G1 => [B]])),
        retract(proved(Z = [his(Z1,A->B),G => [A->B]])),
    !.


/* (R3,R3) - (D.15.1) */
norm2(proved(Z = [con(Z1,B), G => T])) :-
        proved(Z1 = [con(Z2,A), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        append(E, [B|Lambda], G),
        append(E, [B|[B|Lambda]], G1),
        append(E1, [B|[B|Lambda]], G2),
        append(Gamma, [A|Delta], E),
        append(Gamma, [A|[A|Delta]], E1),
    !,
        assertz(proved(Z = [con(Z1,A), G => T])),
        append(E1, [B|Lambda], G3),
        assertz(proved(Z1 = [con(Z2,B), G3 => T])),
        retract(proved(Z1 = [con(Z2,A), G1 => T])),
        retract(proved(Z = [con(Z1,B), G => T])),
    !.


/* (R3,R4) - (D.16.1,2) */
norm2(proved(Z = [con(Z1,A), G => T])) :-
        proved(Z1 = [inc(Z2,C,B), G1 => T]),
        proved(Z2 = [_, G2 => T]),
```

```
        ((append(S, [C|[B|Lambda]], G1),
          append(S, [B|[C|Lambda]], G2),
          append(Gamma, [A|[A|Delta]], S),
          append3(Gamma, [A|Delta], [C|[B|Lambda]], G),
          append3(Gamma, [A|Delta], [B|[C|Lambda]], G3))
        ;(append(Gamma, [C|[B|E]], G1),
          append(Gamma, [B|[C|E]], G2),
          append(Delta, [A|[A|Lambda]], E),
          append3(Gamma, [C|[B|Delta]], [A|Lambda], G),
          append3(Gamma, [B|[C|Delta]], [A|Lambda], G3))),
        !,
          assertz(proved(Z = [inc(Z1,C,B), G => T])),
          assertz(proved(Z1 = [con(Z2,A), G3 => T])),
          retract(proved(Z1 = [inc(Z2,C,B), G1 => T])),
          retract(proved(Z = [con(Z1,A), G => T])),
        !.


/* (R3,R4) - (D.16.3) */
norm2(proved(Z = [con(Z1,A), G => T])) :-
          proved(Z1 = [inc(Z2,A,A), G1 => T]),
          proved(Z2 = [F, G1 => T]),
          append(Gamma, [A|Delta], G),
          append(Gamma, [A|[A|Delta]], G1),
        !,
          assertz(proved(Z = [con(Z2,A), G => T])),
          retract(proved(Z1 = [inc(Z2,A,A), G1 => T])),
          retract(proved(Z = [con(Z1,A), G => T])),
        !.


/* (R3,R5) - (D.17) */
norm2(proved(Z = [th(Z1,B), G => T])) :-
          proved(Z1 = [con(Z2,A), G => T1]),
          proved(Z2 = [_, G1 => T1]),
          append(Gamma, [A|[A|Delta]], G1),
          append(Gamma, [A|Delta], G),
          append(Phi, [B|Psi], T),
          append(Phi, Psi, T1),
        !,
          assertz(proved(Z = [con(Z1,A), G => T])),
          assertz(proved(Z1 = [th(Z2,B), G1 => T])),
          retract(proved(Z1 = [con(Z2,A), G => T1])),
          retract(proved(Z = [th(Z1,B), G => T])),
        !.


/* (R3,R10) - (D.18) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
          proved(Z1 = [con(Z3,C), G => T1]),
          proved(Z2 = [con(Z4,C), G => T2]),
          proved(Z3 = [_, G1 => T1]),
          proved(Z4 = [_, G1 => T2]),
        !,
          assertz(proved(Z1 = [ais(Z3,Z4,A & B),G1 => T])),
          assertz(proved(Z = [con(Z1,C), G => T])),
          retract(proved(Z1 = [con(Z3,C), G => T1])),
          retract(proved(Z2 = [con(Z4,C), G => T2])),
          retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        !.


/* (R3,R11) - (D.19.3) */
norm2(proved(Z = [con(Z1,A & A), G => T])) :-
          proved(Z1 = [aia(Z2,A & A), G1 => T]),
          proved(Z2 = [aia(Z3,A & A), G2 => T]),
          proved(Z3 = [_, G3 => T]),
          append(Gamma, [A & A|Delta], G),
          append(Gamma, [A & A|[A & A|Delta]], G1),
          append(Gamma, [A|[A|[A & A|Delta]]], G2),
          append(Gamma, [A|[A|[A|[A|Delta]]]], G3),
        !,
          assertz(proved(Z = [aia(Z1,A & A), G => T])),
          append(Gamma, [A|[A|Delta]], G4),
          assertz(proved(Z1 = [con(Z2,A), G4 => T])),
          append(Gamma, [A|[A|[A|Delta]]], G5),
          assertz(proved(Z2 = [con(Z3,A), G5 => T])),
          retract(proved(Z = [con(Z1,A & A), G => T])),
          retract(proved(Z1 = [aia(Z2,A & A), G1 => T])),
          retract(proved(Z2 = [aia(Z3,A & A), G2 => T])),
        !.
```

```
/* (R3,R11) - (D.19.1,2) */
norm2(proved(Z = [con(Z1,A), G => T])) :-
        proved(Z1 = [aia(Z2,B & C), G1 => T]),
        proved(Z2 = [_, G2 => T]),
      ((append(S1, [B & C|Lambda], G1),
        append(S1, [B|[C|Lambda]], G2),
        append(Gamma, [A|[A|Delta]], S1),
        append3(Gamma, [A|Delta], [B & C|Lambda], G),
        append3(Gamma, [A|Delta], [A|[B|Lambda]], G3))
      ;(append(Gamma, [B & C|E1], G1),
        append(Gamma, [B|[C|E1]], G2),
        append(Delta, [A|[A|Lambda]], E1),
        append3(Gamma, [B & C|Delta], [A|Lambda], G),
        append3(Gamma, [B|[C|Delta]], [A|Lambda], G3))),
      !,
        assertz(proved(Z = [aia(Z1,B & C), G => T])),
        assertz(proved(Z1 = [con(Z2,A), G3 => T])),
        retract(proved(Z = [con(Z1,A), G => T])),
        retract(proved(Z1 = [aia(Z2,B & C), G1 => T])),
      !.


/* (R3,R12) - (D.20.1,2) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
        proved(Z1 = [con(Z3,C), G1 => T]),
        proved(Z2 = [con(Z4,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        proved(Z4 = [_, G4 => T]),
      ((append(S, [A # B|Lambda], G),
        append(S, [A|Lambda], G1),
        append(S, [B|Lambda], G2),
        append(S1, [A|Lambda], G3),
        append(S1, [B|Lambda], G4),
        append(Gamma, [A|Delta], S),
        append(Gamma, [A|[A|Delta]], S1),
        append(S1, [A # B|Lambda], G5))
      ;(append(Gamma, [A # B|E], G),
        append(Gamma, [A|E], G1),
        append(Gamma, [B|E], G2),
        append(Gamma, [A|E1], G3),
        append(Gamma, [B|E1], G4),
        append(Delta, [C|Lambda], E),
        append(Delta, [C|[C|Lambda]], E1),
        append(Gamma, [A # B|E1], G5))),
      !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        assertz(proved(Z1 = [oia(Z3,Z4,A # B), G5 => T])),
        retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        retract(proved(Z1 = [con(Z3,C), G1 => T])),
        retract(proved(Z2 = [con(Z4,C), G2 => T])),
      !.


/* (R3,R13) - (D.21) */
norm2(proved(Z = [ois(Z1,A # B), G => T])) :-
        proved(Z1 = [con(Z2,C), G => T1]),
        proved(Z2 = [_, G1 => T1]),
      !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        assertz(proved(Z1 = [ois(Z2,A # B), G1 => T])),
        retract(proved(Z = [ois(Z1,A # B), G => T])),
        retract(proved(Z1 = [con(Z2,C), G => T1])),
      !.


/* (R3,R4) - (D.22.1) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [con(Z3,C), G1 => [A]]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => [A]]),
        append(Gamma, [C|[C|Delta]], G3),
        append(Gamma, [C|Delta], G1),
        append(Lambda, [B|Zeta], G2),
        append3(Lambda, G1, [A -> B|Zeta], G),
      !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        append3(Lambda, G3, [A -> B|Zeta], G4),
        assertz(proved(Z1 = [hia(Z3,Z2,A -> B), G4 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z1 = [con(Z3,C), G1 => [A]])),
      !.
```

```
/* (R3,R14) - (D.22.2,3) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [_, G1 => [A]]),
        proved(Z2 = [con(Z3,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
      ((append(Delta, [C|[C|E]], G3),
        append(Delta, [C|E], G2),
        append(Lambda, [B|Zeta], E),
        append4(Delta, [C|Lambda], G1, [A -> B|Zeta], G),
        append4(Delta,[C|[C|Lambda]],G1,[A -> B|Zeta],G4))
      ;(append(S, [C|[C|Zeta]], G3),
        append(S, [C|Zeta], G2),
        append(Delta, [B|Lambda], S),
        append4(Delta, G1, [A -> B|Lambda], [C|Zeta], G),
        append4(Delta,G1,[A->B|Lambda],[C|[C|Zeta]],G4))),
      !,
        assertz(proved(Z = [con(Z2,C), G => T])),
        assertz(proved(Z2 = [hia(Z1,Z3,A -> B),G4 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z2 = [con(Z3,C), G2 => T])),
      !.

/* (R3,R15) - (D.23.1,2) */
norm2(proved(Z = [his(Z1,A -> B), G => [A -> B]])) :-
        proved(Z1 = [con(Z2,C), G1 => [B]]),
        proved(Z2 = [_, G2 => [B]]),
      ((append(Gamma, [C|[C|E1]], G2),
        append(Gamma, [C|E1], G1),
        append(Delta, [A|Lambda], E1),
        append3(Gamma, [C|Delta], Lambda, G),
        append3(Gamma, [C|[C|Delta]], Lambda, G3))
      ;(append(S1, [C|[C|Lambda]], G2),
        append(S1, [C|Lambda], G1),
        append(Gamma, [A|Delta], S1),
        append3(Gamma, Delta, [C|Lambda], G),
        append3(Gamma, Delta, [C|[C|Lambda]], G3))),
      !,
        assertz(proved(Z = [con(Z1,C), G => [A -> B]])),
        assertz(proved(Z1 = [his(Z2,A->B),G3 => [A->B]])),
        retract(proved(Z = [his(Z1,A->B),G => [A->B]])),
        retract(proved(Z1 = [con(Z3,C), G1 => [B]])),
      !.

/* (R4,R5) - (D.25) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
        proved(Z1 = [inc(Z2,B,A), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Phi, [C|Psi], T),
        append(Phi, Psi, T1),
        append(Gamma, [B|[A|Delta]], G),
        append(Gamma, [A|[B|Delta]], G1),
      !,
        assertz(proved(Z = [inc(Z1,B,A), G => T])),
        assertz(proved(Z1 = [th(Z2,C), G1 => T])),
        retract(proved(Z = [th(Z1,C), G => T])),
        retract(proved(Z1 = [inc(Z2,B,A), G => T1])),
      !.

/* (R4,R10) - (D.28) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
        proved(Z1 = [inc(Z3,D,C), G => T1]),
        proved(Z2 = [inc(Z4,D,C), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G1 => T2]),
      !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        assertz(proved(Z1 = [ais(Z3,Z4,A & B), G1 => T])),
        retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        retract(proved(Z1 = [inc(Z3,D,C), G => T1])),
        retract(proved(Z2 = [inc(Z4,D,C), G => T2])),
      !.

/* (R4,R11) - (D.29.1,2) */
norm2(proved(Z = [inc(Z1,B,A), G => T])) :-
        proved(Z1 = [ala(Z2,C & D), G1 => T]),
        proved(Z2 = [_, G2 => T]),
      ((append(S, [C & D|Lambda], G1),
        append(S, [C|[D|Lambda]], G2),
        append(Gamma, [A|[B|Delta]], S),
```

```
            append3(Gamma, [B|[A|Delta]], [C & D|Lambda], G),
            append3(Gamma, [B|[A|Delta]], [C|[D|Lambda]], G3))
         ;(append(Gamma, [C & D|E], G1),
            append(Gamma, [C|[D|E]], G2),
            append(Delta, [A|[B|Lambda]], E),
            append3(Gamma, [C & D|Delta], [B|[A|Lambda]], G),
            append3(Gamma,[C|[D|Delta]],[B|[A|Lambda]],G3))),
         !,
            assertz(proved(Z = [aia(Z1,C & D), G => T])),
            assertz(proved(Z1 = [inc(Z2,B,A), G3 => T])),
            retract(proved(Z = [inc(Z1,B,A), G => T])),
            retract(proved(Z1 = [aia(Z2,C & D), G1 => T])),
         !.


/* (R4,R11) - (D.29.3) */
norm2(proved(Z = [inc(Z1,C,A & B), G => T])) :-
            proved(Z1 = [aia(Z2,A & B), G1 => T]),
            proved(Z2 = [_, G2 => T]),
            append(Gamma, [C|[A & B|Delta]], G),
            append(Gamma, [A & B|[C|Delta]], G1),
            append(Gamma, [A|[B|[C|Delta]]], G2),
         !,
            assertz(proved(Z = [aia(Z1,A & B), G => T])),
            gensym(f,Z3),
            append(Gamma, [C|[A|[B|Delta]]], G4),
            assertz(proved(Z1 = [inc(Z3,C,A), G4 => T])),
            append(Gamma, [A|[C|[B|Delta]]], G3),
            assertz(proved(Z3 = [inc(Z2,C,B), G3 => T])),
            retract(proved(Z = [inc(Z1,C,A & B), G => T])),
            retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
         !.


/* (R4,R11) - (D.29.4) */
norm2(proved(Z = [inc(Z1,B & C,A), G => T])) :-
            proved(Z1 = [aia(Z2,B & C), G1 => T]),
            proved(Z2 = [_, G2 => T]),
            append(Gamma, [B & C|[A|Delta]], G),
            append(Gamma, [A|[B & C|Delta]], G1),
            append(Gamma, [A|[B|[C|Delta]]], G2),
         !,
            assertz(proved(Z = [aia(Z1,B & C), G => T])),
            gensym(f1,Z3),
            append(Gamma, [B|[C|[A|Delta]]], G3),
            assertz(proved(Z1 = [inc(Z3,C,A), G3 => T])),
            append(Gamma, [B|[A|[C|Delta]]], G4),
            assertz(proved(Z3 = [inc(Z2,B,A), G4 => T])),
            retract(proved(Z = [inc(Z1,B & C,A), G => T])),
            retract(proved(Z1 = [aia(Z2,B & C), G1 => T])),
         !.


/* (R4,R12) - (D.30.1,2) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
            proved(Z1 = [inc(Z3,D,C), G1 => T]),
            proved(Z2 = [inc(Z4,D,C), G2 => T]),
            proved(Z3 = [_, G3 => T]),
            proved(Z4 = [_, G4 => T]),
         ((append(S, [A # B|Lambda], G),
            append(S, [A|Lambda], G1),
            append(S, [B|Lambda], G2),
            append(S1, [A|Lambda], G3),
            append(S1, [B|Lambda], G4),
            append(Gamma, [D|[C|Delta]], S),
            append(Gamma, [C|[D|Delta]], S1),
            append(S1, [A # B|Lambda], G5))
         ;(append(Gamma, [A # B|E], G),
            append(Gamma, [A|E], G1),
            append(Gamma, [B|E], G2),
            append(Gamma, [A|E1], G3),
            append(Gamma, [B|E1], G4),
            append(Delta, [D|[C|Lambda]], E),
            append(Delta, [C|[D|Lambda]], E1),
            append(Gamma, [A # B|E1], G5))),
         !,
            assertz(proved(Z = [inc(Z1,D,C), G => T])),
            assertz(proved(Z1 = [oia(Z3,Z4,A # B), G5 => T])),
            retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
            retract(proved(Z1 = [inc(Z3,D,C), G1 => T])),
            retract(proved(Z2 = [inc(Z4,D,C), G2 => T])),
         !.
```

```
/* (R4,R13) - (D.31) */
norm2(proved(Z = [ois(Z1,A # B), G => T])) :-
        proved(Z1 = [inc(Z2,D,C), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [C|[D|Delta]], G1),
        append(Phi, [A # B|Psi], T),
        append(Phi, [A|[B|Psi]], T1),
     !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        assertz(proved(Z1 = [ois(Z2,A # B), G1 => T])),
        retract(proved(Z = [ois(Z1,A # B), G => T])),
        retract(proved(Z1 = [inc(Z2,D,C), G => T1])),
     !.

/* (R4,R14) - (D.32.1) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [inc(Z3,D,C), G1 => [A]]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => [A]]),
        append(S, [A -> B|Zeta], G),
        append(Lambda, [B|Zeta], G2),
        append(Lambda, G1, S),
        append(Gamma, [D|[C|Delta]], G1),
        append(Gamma, [C|[D|Delta]], G3),
     !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        append(Lambda, G3, S1),
        append(S1, [A -> B|Zeta], G4),
        assertz(proved(Z1 = [hia(Z3,Z2,A -> B), G4 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z1 = [inc(Z3,D,C), G1 => [A]])),
     !.

/* (R4,R14) - (D.32.2,3) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [_, G1 => [A]]),
        proved(Z2 = [inc(Z3,D,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
     ((append(S, [A -> B|Zeta], G),
        append(S1, [B|Zeta], G2),
        append(S2, [B|Zeta], G3),
        append(S1, G1, S),
        append(Delta, [D|[C|Lambda]], S1),
        append(Delta, [C|[D|Lambda]], S2),
        append(S2, G1, S3),
        append(S3, [A -> B|Zeta], G4))
     ;(append(S, [A -> B|E], G),
        append(Delta, [B|E], G2),
        append(Delta, [B|E1], G3),
        append(Delta, G1, S),
        append(Lambda, [D|[C|Zeta]], E),
        append(Lambda, [C|[D|Zeta]], E1),
        append(S, [A -> B|E1], G4))),
     !,
        assertz(proved(Z = [inc(Z2,D,C), G => T])),
        assertz(proved(Z2 = [hia(Z1,Z3,A -> B),G4 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z2 = [inc(Z3,D,C), G2 => T])),
     !.


/* (R4,R14) - (D.32.4) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [_, G1 => [A]]),
        proved(Z2 = [inc(Z3,B,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Delta, [C|[B|Lambda]], G3),
        append(Delta, [B|[C|Lambda]], G2),
        append(Delta, G1, S),
        append(S, [A -> B|[C|Lambda]], G),
     !,
        assertz(proved(Z = [inc(Z2,A -> B,C), G => T])),
        append(S, [C|[A -> B|Lambda]], G4),
        gensym(f,F1),
        assertz(proved(Z2 = [inc(F1,G1,C), G4 => T])),
        append(Delta, [C|G1], S1),
        append(S1, [A -> B|Lambda], G5),
        assertz(proved(F1 = [hia(Z1,Z3,A -> B),G5 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z2 = [inc(Z3,B,C), G2 => T])),
     !.
```

```
/* (R4,R14) - (D.32.5) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
        proved(Z1 = [_, G1 => [A]]),
        proved(Z2 = [inc(Z3,C,B), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        append(Delta, [B|[C|Lambda]], G3),
        append(Delta, [C|[B|Lambda]], G2),
        append(Delta, [C|G1], S),
        append(S, [A -> B|Lambda], G),
      !,
        assertz(proved(Z = [inc(Z2,C,G1), G => T])),
        append(Delta, G1, S1),
        append(S1, [C|[A -> B|Lambda]], G4),
        gensym(f,F1),
        assertz(proved(Z2 = [inc(F1,C,A -> B), G4 => T])),
        append(S1, [A -> B|[C|Lambda]], G5),
        assertz(proved(F1 = [hia(Z1,Z3,A -> B),G5 => T])),
        retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
        retract(proved(Z2 = [inc(Z3,C,B), G2 => T])),
      !.


/* (R4,R15) - (D.33.1,2) */
norm2(proved(Z = [his(Z1,A -> B), G => [A -> B]])) :-
        proved(Z1 = [inc(Z2,D,C), G1 => [B]]),
        proved(Z2 = [_, G2 => [B]]),
      ((append(S, [D|[C|Lambda]], G),
        append(S1, [D|[C|Lambda]], G1),
        append(S1, [C|[D|Lambda]], G2),
        append(Gamma, [A|Delta], S1),
        append(Gamma, Delta, S),
        append(S, [C|[D|Lambda]], G3))
      ;(append(Gamma, [D|[C|E]], G),
        append(Gamma, [D|[C|E1]], G1),
        append(Gamma, [C|[D|E1]], G2),
        append(Delta, [A|Lambda], E1),
        append(Delta, Lambda, E),
        append(Gamma, [C|[D|E]], G3))),
      !,
        assertz(proved(Z = [inc(Z1,D,C), G => [A -> B]])),
        assertz(proved(Z1 = [his(Z2,A->B),G3 => [A->B]])),
        retract(proved(Z = [his(Z1,A->B),G => [A->B]])),
        retract(proved(Z1 = [inc(Z2,D,C), G1 => [B]])),
      !.


/* (R4,R15) - (D.33.3,4) */
norm2(proved(Z = [his(Z1,A -> B), G => [A -> B]])) :-
      ((proved(Z1 = [inc(Z2,A,C), G1 => [B]]),
        proved(Z2 = [_, G2 => [B]]),
        append(Gamma, [C|[A|Delta]], G2),
        append(Gamma, [A|[C|Delta]], G1))
      ;(proved(Z1 = [inc(Z2,C,A), G1 => [B]]),
        proved(Z2 = [_, G2 => [B]]),
        append(Gamma, [A|[C|Delta]], G2),
        append(Gamma, [C|[A|Delta]], G1))),
        append(Gamma, [C|Delta], G),
      !,
        assertz(proved(Z = [his(Z2,A -> B),G => [A->B]])),
        retract(proved(Z = [his(Z1,A -> B),G => [A->B]])),
        retract(proved(Z1 = [inc(Z2,A,C), G1 => [B]])),
      !.


/* (R5,R5) - (D.34) */
norm2(proved(Z = [th(Z1,B), G => T])) :-
        proved(Z1 = [th(Z2,A), G => T1]),
        proved(Z2 = [_, G => T2]),
        append(Phi, [A|E], T1),
        append(Phi, E, T2),
        append(Psi, Theta, E),
        append3(Phi, [A|Psi], [B|Theta], T),
      !,
        assertz(proved(Z = [th(Z1,A), G => T])),
        append3(Phi, Psi, [B|Theta], T3),
        assertz(proved(Z1 = [th(Z2,B), G => T3])),
        retract(proved(Z = [th(Z1,B), G => T])),
        retract(proved(Z1 = [th(Z2,A), G => T1])),
      !.
```

```
/* (R5,R10) - (D.35.1,2) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
            proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
            proved(Z2 = [_, G => T2]),
            proved(Z3 = [_, G => T3]),
         ((append(S1, [A & B|Theta], T1),
            append(S1, [A|Theta], T2),
            append(S1, [B|Theta], T3),
            append(S, [A & B|Theta], T),
            append(Phi, [C|Psi], S),
            append(Phi, Psi, S1),
            append(S, [A|Theta], T4),
            append(S, [B|Theta], T5))
         ;(append(Phi, [A & B|E1], T1),
            append(Phi, [A|E1], T2),
            append(Phi, [B|E1], T3),
            append(Phi, [A & B|E], T),
            append(Psi, [C|Theta], E),
            append(Psi, Theta, E1),
            append(Phi, [A|E], T4),
            append(Phi, [B|E], T5))),
         !,
            gensym(f,F1),
            assertz(proved(Z = [ais(Z1,F1,A & B), G => T])),
            assertz(proved(Z1 = [th(Z2,C), G => T4])),
            assertz(proved(F1 = [th(Z3,C), G => T5])),
            retract(proved(Z = [th(Z1,C), G => T])),
            retract(proved(Z1 = [ais(Z2,Z3,A & B), G => T1])),
         !.


/* (R5,R10) - (D.35.3,4) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
            proved(Z1 = [th(Z3,B), G => T1]),
            proved(Z2 = [th(Z4,A), G => T2]),
            proved(Z3 = [K1, G => T3]),
            proved(Z4 = [K2, G => T4]),
         ((append(Phi,[A],T3), append(Phi,[C],T4),
            append(Phi,[A,B],T1), append(Phi,[A,C],T2),
            append(Phi, [A,B & C], T))
         ;(append([A],Psi,T3), append([C],Psi,T4),
            append([B,A],Psi,T1), append([C,A],Psi,T2),
            append([B & C,A], Psi, T))),
         !,
            assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
            assertz(proved(Z2 = [th(Z4,C), G => T2])),
            assertz(proved(Z4 = [K1, G => T3])),
            retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
            retract(proved(Z2 = [th(Z4,A), G => T2])),
            retract(proved(Z4 = [K2, G => T4])),
         !.


/* (R5,R10) - (D.35.5) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
            proved(Z1 = [th(Z3,B), G => T1]),
            proved(Z2 = [th(Z4,C), G => T2]),
            proved(Z3 = [th(Z5,D), G => T3]),
            proved(Z4 = [th(Z6,A), G => T3]),
            proved(Z5 = [K1, G => T4]),
            proved(Z6 = [K2, G => T5]),
            append(Phi, [A|[B & C|[D|Psi]]], T),
            append(Phi, [A|[B|[D|Psi]]], T1),
            append(Phi, [A|[C|[D|Psi]]], T2),
            append(Phi, [A|[D|Psi]], T3),
            append(Phi, [A|Psi], T4),
            append(Phi, [D|Psi], T5),
         !,
            assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
            assertz(proved(Z4 = [th(Z6,D), G => T3])),
            assertz(proved(Z6 = [K1, G => T4])),
            retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
            retract(proved(Z4 = [th(Z6,A), G => T3])),
            retract(proved(Z6 = [K2, G => T5])),
         !.


/* (R5,R10) - (D.35.6) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
            proved(Z1 = [th(Z3,B), G => T1]),
            proved(Z2 = [th(Z4,A), G => T2]),
            proved(Z3 = [th(Z5,D), G => T3]),
            proved(Z4 = [th(Z6,D), G => T4]),
```

```
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [A|[D|Psi]], T3),
        append(Phi, [C|[D|Psi]], T4),
        append(Phi, [A|Psi], T5),
        append(Phi, [C|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z2 = [th(Z4,C), G => T2])),
        assertz(proved(Z4 = [th(Z6,D), G => T3])),
        assertz(proved(Z6 = [K1, G => T5])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z2 = [th(Z4,A), G => T2])),
        retract(proved(Z4 = [th(Z6,D), G => T4])),
        retract(proved(Z6 = [K2, G => T6])),
    !.


/* (R5,R10) - (D.35.7) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
        proved(Z1 = [th(Z3,B), G => T1]),
        proved(Z2 = [th(Z4,A), G => T2]),
        proved(Z3 = [th(Z5,D), G => T3]),
        proved(Z4 = [th(Z6,C), G => T4]),
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [A|[D|Psi]], T3),
        append(Phi, [C|[D|Psi]], T4),
        append(Phi, [A|Psi], T5),
        append(Phi, [D|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z1 = [th(Z3,A), G => T1])),
        append(Phi, [B|[D|Psi]], T7),
        assertz(proved(Z3 = [th(Z5,B), G => T7])),
        assertz(proved(Z5 = [K2, G => T6])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z1 = [th(Z3,B), G => T1])),
        retract(proved(Z3 = [th(Z5,D), G => T3])),
        retract(proved(Z5 = [K1, G => T5])),
    !.


/* (R5,R10) - (D.35.8) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
        proved(Z1 = [th(Z3,A), G => T1]),
        proved(Z2 = [th(Z4,C), G => T2]),
        proved(Z3 = [th(Z5,D), G => T3]),
        proved(Z4 = [th(Z6,D), G => T4]),
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [B|[D|Psi]], T3),
        append(Phi, [A|[D|Psi]], T4),
        append(Phi, [B|Psi], T5),
        append(Phi, [A|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z1 = [th(Z3,B), G => T1])),
        assertz(proved(Z3 = [th(Z5,D), G => T4])),
        assertz(proved(Z5 = [K2, G => T6])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z1 = [th(Z3,A), G => T1])),
        retract(proved(Z3 = [th(Z5,D), G => T3])),
        retract(proved(Z5 = [K1, G => T5])),
    !.

/* (R5,R10) - (D.35.9) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
        proved(Z1 = [th(Z3,A), G => T1]),
        proved(Z2 = [th(Z4,A), G => T2]),
        proved(Z3 = [th(Z5,D), G => T3]),
        proved(Z4 = [th(Z6,C), G => T4]),
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
```

```
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [B|[D|Psi]], T3),
        append(Phi, [C|[D|Psi]], T4),
        append(Phi, [B|Psi], T5),
        append(Phi, [D|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z3 = [th(Z5,B), G => T3])),
        assertz(proved(Z5 = [K2, G => T6])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z3 = [th(Z5,D), G => T3])),
        retract(proved(Z5 = [K1, G => T5])),
    !.

/* (R5,R10) - (D.35.10) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
        proved(Z1 = [th(Z3,A), G => T1]),
        proved(Z2 = [th(Z4,C), G => T2]),
        proved(Z3 = [th(Z5,B), G => T3]),
        proved(Z4 = [th(Z6,D), G => T4]),
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [B|[D|Psi]], T3),
        append(Phi, [A|[D|Psi]], T4),
        append(Phi, [D|Psi], T5),
        append(Phi, [A|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z2 = [th(Z4,A), G => T2])),
        append(Phi, [C|[D|Psi]], T7),
        assertz(proved(Z4 = [th(Z6,C), G => T7])),
        assertz(proved(Z6 = [K1, G => T5])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z2 = [th(Z4,C), G => T2])),
        retract(proved(Z4 = [th(Z6,D), G => T4])),
        retract(proved(Z6 = [K2, G => T6])),
    !.

/* (R5,R10) - (D.35.11) */
norm2(proved(Z = [ais(Z1,Z2,B & C), G => T])) :-
        proved(Z1 = [th(Z3,A), G => T1]),
        proved(Z2 = [th(Z4,A), G => T2]),
        proved(Z3 = [th(Z5,B), G => T3]),
        proved(Z4 = [th(Z6,D), G => T4]),
        proved(Z5 = [K1, G => T5]),
        proved(Z6 = [K2, G => T6]),
        append(Phi, [A|[B & C|[D|Psi]]], T),
        append(Phi, [A|[B|[D|Psi]]], T1),
        append(Phi, [A|[C|[D|Psi]]], T2),
        append(Phi, [B|[D|Psi]], T3),
        append(Phi, [C|[D|Psi]], T4),
        append(Phi, [D|Psi], T5),
        append(Phi, [C|Psi], T6),
    !,
        assertz(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        assertz(proved(Z4 = [th(Z6,C), G => T4])),
        assertz(proved(Z6 = [K1, G => T5])),
        retract(proved(Z = [ais(Z1,Z2,B & C), G => T])),
        retract(proved(Z4 = [th(Z6,D), G => T4])),
        retract(proved(Z6 = [K2, G => T6])),
    !.

/* (R5,R11) - (D.36) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
        proved(Z1 = [ala(Z2,A & B), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [C|Psi], T),
        append(Phi, Psi, T1),
    !,
        assertz(proved(Z = [ala(Z1,A & B), G => T])),
        assertz(proved(Z1 = [th(Z2,C), G1 => T])),
        retract(proved(Z = [th(Z1,C), G => T])),
        retract(proved(Z1 = [ala(Z2,A & B), G => T1])),
    !.
```

```
/* (R5,R12) - (D.37) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
        proved(Z1 = [ola(Z2,Z3,A # B), G => T1]),
        proved(Z2 = [_, G1 => T1]),
        proved(Z3 = [_, G2 => T1]),
        append(Gamma, [A # B|Delta], G),
        append(Gamma, [A|Delta], G1),
        append(Gamma, [B|Delta], G2),
        append(Phi, [C|Psi], T),
        append(Phi, Psi, T1),
      !,
        gensym(f,F1),
        assertz(proved(Z = [ola(Z1,F1,A # B), G => T])),
        assertz(proved(Z1 = [th(Z2,C), G1 => T])),
        assertz(proved(F1 = [th(Z3,C), G2 => T])),
        retract(proved(Z = [th(Z1,C), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,A # B), G => T1])),
      !.

/* (R5,R13) - (D.38.1,2) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
        proved(Z1 = [ois(Z2,A # B), G => T1]),
        proved(Z2 = [_, G => T2]),
      ((append(S1, [A # B|Theta], T1),
        append(S1, [A|[B|Theta]], T2),
        append(S, [A # B|Theta], T),
        append(Phi, [C|Psi], S),
        append(Phi, Psi, S1),
        append(S, [A|[B|Theta]], T3))
      ;(append(Phi, [A # B|E1], T1),
        append(Phi, [A|[B|E1]], T2),
        append(Phi, [A # B|E], T),
        append(Psi, [C|Theta], E),
        append(Psi, Theta, E1),
        append(Phi, [A|[B|E]], T3))),
      !,
        assertz(proved(Z = [ois(Z1,A # B), G => T])),
        assertz(proved(Z1 = [th(Z2,C), G => T3])),
        retract(proved(Z = [th(Z1,C), G => T])),
        retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
      !.


/* (R5,R14) - (D.39) */
norm2(proved(Z = [th(Z1,C), G => T])) :-
        proved(Z1 = [hia(Z2,Z3,A -> B), G => T1]),
        proved(Z2 = [_, G1 => [A]]),
        proved(Z3 = [_, G2 => T1]),
        append(S, [A -> B|Lambda], G),
        append(Delta, [B|Lambda], G2),
        append(Phi, [C|Psi], T),
        append(Phi, Psi, T1),
      !,
        assertz(proved(Z = [hia(Z2,Z1,A -> B),G => T])),
        assertz(proved(Z1 = [th(Z3,C), G2 => T])),
        retract(proved(Z = [th(Z1,C), G => T])),
        retract(proved(Z1 = [hia(Z2,Z3,A -> B),G => T1])),
      !.


/* (R10,R10) - (D.48) */
norm2(proved(Z = [ais(Z1,Z2,A & A), G => T])) :-
        proved(Z1 = [ais(Z3,Z4,A & A), G => T1]),
        proved(Z2 = [ais(Z5,Z6,A & A), G => T1]),
        proved(Z3 = [_, G => T2]),
        proved(Z4 = [_, G => T2]),
        proved(Z5 = [_, G => T2]),
        proved(Z6 = [_, G => T2]),
        append(Phi, [A & A|E1], T1),
        append(Phi, [A|E1], T2),
        append(Phi, [A & A|E], T),
        append(Psi, [A & A|Theta], E),
        append(Psi, [A|Theta], E1),
      !,
        assertz(proved(Z = [ais(Z1,Z2,A & A), G => T])),
        append(Phi, [A|Psi], S1),
        append(S1, [A & A|Theta], T3),
        assertz(proved(Z1 = [ais(Z3,Z4,A & A), G => T3])),
        assertz(proved(Z2 = [ais(Z5,Z6,A & A), G => T3])),
        retract(proved(Z = [ais(Z1,Z2,A & A), G => T])),
        retract(proved(Z1 = [ais(Z3,Z4,A & A), G => T1])),
        retract(proved(Z2 = [ais(Z5,Z6,A & A), G => T1])),
      !.
```

```
/* (R10,R11) - (D.49) */
norm2(proved(Z = [ais(Z1,Z2,C & D), G => T])) :-
        proved(Z1 = [aia(Z3,A & B), G => T1]),
        proved(Z2 = [aia(Z4,A & B), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G1 => T2]),
        append(Gamma, [A & B|Delta], G),
        append(Gamma, [A|[B|Delta]], G1),
        append(Phi, [C & D|Psi], T),
        append(Phi, [C|Psi], T1),
        append(Phi, [D|Psi], T2),
      !,
        assertz(proved(Z = [aia(Z1,A & B), G => T])),
        assertz(proved(Z1 = [ais(Z3,Z4,C & D), G1 => T])),
        retract(proved(Z = [ais(Z1,Z2,C & D), G => T])),
        retract(proved(Z1 = [aia(Z3,A & B), G => T1])),
        retract(proved(Z2 = [aia(Z4,A & B), G => T2])),
      !.

/* (R10,R12) - (D.50) */
norm2(proved(Z = [ais(Z1,Z2,C & D), G => T])) :-
        proved(Z1 = [oia(Z3,Z4,A # B), G => T1]),
        proved(Z2 = [oia(Z5,Z6,A # B), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T1]),
        proved(Z5 = [_, G1 => T2]),
        proved(Z6 = [_, G2 => T2]),
        append(Gamma, [A # B|Delta], G),
        append(Gamma, [A|Delta], G1),
        append(Gamma, [B|Delta], G2),
        append(Phi, [C & D|Psi], T),
        append(Phi, [C|Psi], T1),
        append(Phi, [D|Psi], T2),
      !,
        assertz(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        assertz(proved(Z1 = [ais(Z3,Z5,C & D), G1 => T])),
        assertz(proved(Z2 = [ais(Z4,Z6,C & D), G2 => T])),
        retract(proved(Z = [ais(Z1,Z2,C & D), G => T])),
        retract(proved(Z1 = [oia(Z3,Z4,A # B), G => T1])),
        retract(proved(Z2 = [oia(Z5,Z6,A # B), G => T2])),
      !.

/* (R10,R13) - (D.51.1,2) */
norm2(proved(Z = [ois(Z1,C # D), G => T])) :-
        proved(Z1 = [ais(Z2,Z3,A & B), G => T1]),
        proved(Z2 = [_, G => T2]),
        proved(Z3 = [_, G => T3]),
      ((append(Phi, [C # D|E], T),
        append(Phi, [C|[D|E]], T1),
        append(Phi, [C|[D|E1]], T2),
        append(Phi, [C|[D|E2]], T3),
        append(Phi, [C # D|E1], T4),
        append(Phi, [C # D|E2], T5))
      ;(append(S, [C # D|Theta], T),
        append(S, [C|[D|Theta]], T1),
        append(S1, [C|[D|Theta]], T2),
        append(S2, [C|[D|Theta]], T3),
        append(S1, [C # D|Theta], T4),
        append(S2, [C # D|Theta], T5))),
      !,
        gensym(f,F1),
        assertz(proved(Z = [ais(Z1,F1,A & B), G => T])),
        assertz(proved(Z1 = [ois(Z2,C # D), G => T4])),
        assertz(proved(F1 = [ois(Z3,C # D), G => T5])),
        retract(proved(Z = [ois(Z1,C # D), G => T])),
        retract(proved(Z1 = [ais(Z2,Z3,A & B), G => T1])),
      !.

/* (R10,R14) - (D.52) */
norm2(proved(Z = [ais(Z1,Z2,C & D), G => T])) :-
        proved(Z1 = [hia(Z3,Z4,A -> B), G => T1]),
        proved(Z2 = [hia(Z5,Z6,A -> B), G => T2]),
        proved(Z3 = [_, G1 => [A]]),
        proved(Z4 = [_, G2 => T1]),
        proved(Z5 = [K, G1 => [A]]),
        proved(Z6 = [_, G2 => T2]),
        append(S, [A -> B|Lambda], G),
        append(Delta, [B|Lambda], G2),
```

```
            append(Phi, [C & D|Psi], T),
            append(Phi, [C|Psi], T1),
            append(Phi, [D|Psi], T2),
        !,
            assertz(proved(Z = [hia(Z3,Z1,A -> B), G => T])),
            assertz(proved(Z1 = [ais(Z4,Z6,C & D), G2 => T])),
            retract(proved(Z = [ais(Z1,Z2,C & D), G => T])),
            retract(proved(Z1 = [hia(Z3,Z4,A -> B), G => T1])),
            retract(proved(Z2 = [hia(Z5,Z6,A -> B), G => T2])),
            retract(proved(Z5 = [K, G1 => [A]])),
        !.


/* (R11,R11) - (D.53) */
norm2(proved(Z = [aia(Z1,C & D), G => T])) :-
            proved(Z1 = [aia(Z2,A & B), G1 => T]),
            proved(Z2 = [_, G2 => T]),
            append(Gamma, [A & B|E1], G1),
            append(Gamma, [A|[B|E1]], G2),
            append(Gamma, [A & B|E], G),
            append(Delta, [C & D|Lambda], E),
            append(Delta, [C|[D|Lambda]], E1),
        !,
            assertz(proved(Z = [aia(Z1,A & B), G => T])),
            append(Gamma, [A|[B|Delta]], S),
            append(S, [C & D|Lambda], G3),
            assertz(proved(Z1 = [aia(Z2,C & D), G3 => T])),
            retract(proved(Z = [aia(Z1,C & D), G => T])),
            retract(proved(Z1 = [aia(Z2,A & B), G1 => T])),
        !.


/* (R11,R12) - (D.54.1,2) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
            proved(Z1 = [aia(Z3,C & D), G1 => T]),
            proved(Z2 = [aia(Z4,C & D), G2 => T]),
            proved(Z3 = [_, G3 => T]),
            proved(Z4 = [_, G4 => T]),
        ((append(Gamma, [C & D|E1], G1),
            append(Gamma, [C|[D|E1]], G3),
            append(Gamma, [C & D|E2], G2),
            append(Gamma, [C|[D|E2]], G4),
            append(Gamma, [C & D|E], G),
            append(Gamma, [C|[D|E]], G5))
        ;(append(S1, [C & D|Lambda], G1),
            append(S1, [C|[D|Lambda]], G3),
            append(S2, [C & D|Lambda], G2),
            append(S2, [C|[D|Lambda]], G4),
            append(S, [C & D|Lambda], G),
            append(S, [C|[D|Lambda]], G5))),
        !,
            assertz(proved(Z = [aia(Z1,C & D), G => T])),
            assertz(proved(Z1 = [oia(Z3,Z4,A # B), G5 => T])),
            retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
            retract(proved(Z1 = [aia(Z3,C & D), G1 => T])),
            retract(proved(Z2 = [aia(Z4,C & D), G2 => T])),
        !.


/* (R11,R13) - (D.55) */
norm2(proved(Z = [ois(Z1,C # D), G => T])) :-
            proved(Z1 = [aia(Z2,A & B), G => T1]),
            proved(Z2 = [_, G1 => T1]),
            append(Gamma, [A & B|Delta], G),
            append(Gamma, [A|[B|Delta]], G1),
            append(Phi, [C # D|Psi], T),
            append(Phi, [C|[D|Psi]], T1),
        !,
            assertz(proved(Z = [aia(Z1,A & B), G => T])),
            assertz(proved(Z1 = [ois(Z2,C # D), G1 => T])),
            retract(proved(Z = [ois(Z1,C # D), G => T])),
            retract(proved(Z1 = [aia(Z2,A & B), G => T1])),
        !.


/* (R11,R14) - (D.56.1) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
            proved(Z1 = [aia(Z3,C & D), G1 => [A]]),
            proved(Z2 = [_, G2 => T]),
            proved(Z3 = [_, G3 => [A]]),
            append(Gamma, [C & D|Delta], G1),
            append(Gamma, [C|[D|Delta]], G3),
            append(S, [A -> B|Zeta], G),
```

```
            append(Lambda, [B|Zeta], G2),
        !,
            assertz(proved(Z = [ala(Z1,C & D), G => T])),
            append(Lambda, G3, S1),
            append(S1, [A -> B|Zeta], G4),
            assertz(proved(Z1 = [hia(Z3,Z2,A -> B),G4 => T])),
            retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
            retract(proved(Z1 = [ala(Z3,C & D), G1 => [A]])),
        !.


/* (R11,R14) - (D.56.2,3) */
norm2(proved(Z = [hia(Z1,Z2,A -> B), G => T])) :-
            proved(Z1 = [_, G1 => [A]]),
            proved(Z2 = [ala(Z3,C & D), G2 => T]),
            proved(Z3 = [_, G3 => T]),
        ((append(Delta, [C & D|E1], G2),
            append(Delta, [C|[D|E1]], G3),
            append(Delta, [C & D|E], G),
            append(Delta, [C|[D|E]], G4))
        ;(append(S1, [C & D|Zeta], G2),
            append(S1, [C|[D|Zeta]], G3),
            append(S, [C & D|Zeta], G),
            append(S, [C|[D|Zeta]], G4))),
        !,
            assertz(proved(Z = [ala(Z2,C & D), G => T])),
            assertz(proved(Z2 = [hia(Z1,Z3,A -> B),G4 => T])),
            retract(proved(Z = [hia(Z1,Z2,A -> B), G => T])),
            retract(proved(Z2 = [ala(Z3,C & D), G2 => T])),
        !.


/* (R11,R15) - (D.57.1,2) */
norm2(proved(Z = [his(Z1,A -> B), G => [A -> B]])) :-
            proved(Z1 = [ala(Z2,C & D), G1 => [B]]),
            proved(Z2 = [_, G2 => [B]]),
        ((append(S1, [C & D|Lambda], G1),
            append(S1, [C|[D|Lambda]], G2),
            append(S, [C & D|Lambda], G),
            append(S, [C|[D|Lambda]], G3))
        ;(append(Gamma, [C & D|E1], G1),
            append(Gamma, [C|[D|E1]], G2),
            append(Gamma, [C & D|E], G),
            append(Gamma, [C|[D|E]], G3))),
        !,
            assertz(proved(Z = [ala(Z1,C & D),G => [A -> B]])),
            assertz(proved(Z1 = [his(Z2,A->B),G3 => [A->B]])),
            retract(proved(Z = [his(Z1,A->B),G => [A->B]])),
            retract(proved(Z1 = [ala(Z2,C & D), G1 => [B]])),
        !.


/* (R11,R15) - (D.57.3) */
norm2(proved(Z = [his(Z1,(A&B) -> L),G => [(A&B) -> L]])) :-
            proved(Z1 = [ala(Z2,A & B), G1 => [L]]),
            proved(Z2 = [ala(Z3,C & D), G2 => [L]]),
            proved(Z3 = [_, G3 => [L]]),
            append(S1, [C & D|Lambda], G2),
            append(S1, [C|[D|Lambda]], G3),
            append(S2, [C & D|Lambda], G1),
            append(S, [C & D|Lambda], G),
            append(Gamma, [A & B|Delta], S2),
            append(Gamma, Delta, S),
        !,
            assertz(proved(Z = [ala(Z1,C&D),G => [(A&B)->L]])),
            append(S, [C|[D|Lambda]], G4),
            assertz(proved(Z1=[his(Z2,(A&B)->L),G4=>[(A&B)->L]])),
            append(S2, [C|[D|Lambda]], G5),
            assertz(proved(Z2 = [ala(Z3,A & B), G5 => [L]])),
            retract(proved(Z=[his(Z1,(A&B)->L),G=>[(A&B)->L]])),
            retract(proved(Z1 = [ala(Z2,A & B), G1 => [L]])),
            retract(proved(Z2 = [ala(Z3,C & D), G2 => [L]])),
        !.


/* (R12,R12) - (D.58) */
norm2(proved(Z = [ola(Z1,Z2,A # A), G => T])) :-
            proved(Z1 = [ola(Z3,Z4,A # A), G1 => T]),
            proved(Z2 = [ola(Z5,Z6,A # A), G1 => T]),
            proved(Z3 = [_, G2 => T]),
            proved(Z4 = [_, G2 => T]),
            proved(Z5 = [_, G2 => T]),
            proved(Z6 = [_, G2 => T]),
```

```
            append(Gamma, [A # A|E1], G1),
            append(Gamma, [A|E1], G2),
            append(Gamma, [A # A|E], G),
            append(Delta, [A # A|Lambda], E),
            append(Delta, [A|Lambda], E1),
        !,
            assertz(proved(Z = [oia(Z1,Z2,A # A), G => T])),
            append(Gamma, [A|E], G3),
            assertz(proved(Z1 = [oia(Z3,Z4,A # A), G3 => T])),
            assertz(proved(Z2 = [oia(Z5,Z6,A # A), G3 => T])),
            retract(proved(Z = [oia(Z1,Z2,A # A), G => T])),
            retract(proved(Z1 = [oia(Z3,Z4,A # A), G1 => T])),
            retract(proved(Z2 = [oia(Z5,Z6,A # A), G1 => T])),
        !.

/* (R12,R13) - (D.59) */
norm2(proved(Z = [ois(Z1,C # D), G => T])) :-
            proved(Z1 = [oia(Z2,Z3,A # B), G => T1]),
            proved(Z2 = [_, G1 => T1]),
            proved(Z3 = [_, G2 => T1]),
            append(Gamma, [A # B|Delta], G),
            append(Gamma, [A|Delta], G1),
            append(Gamma, [B|Delta], G2),
            append(Phi, [C # D|Psi], T),
            append(Phi, [C|[D|Psi]], T1),
        !,
            gensym(f,F1),
            assertz(proved(Z = [oia(Z1,F1,A # B), G => T])),
            assertz(proved(Z1 = [ois(Z2,C # D), G1 => T])),
            assertz(proved(F1 = [ois(Z3,C # D), G2 => T])),
            retract(proved(Z = [ois(Z1,C # D), G => T])),
            retract(proved(Z1 = [oia(Z2,Z3,A # B), G => T1])),
        !.

/* (R12,R14) - (D.60.1,2) */
norm2(proved(Z = [oia(Z1,Z2,C # D), G => T])) :-
            proved(Z1 = [hia(Z3,Z4,A -> B), G1 => T]),
            proved(Z2 = [hia(Z5,Z6,A -> B), G2 => T]),
            proved(Z3 = [_, G3 => [A]]),
            proved(Z4 = [_, G4 => T]),
            proved(Z5 = [K, G3 => [A]]),
            proved(Z6 = [_, G5 => T]),
        ((append(S, [C # D|Zeta], G),
            append(S, [C|Zeta], G1),
            append(S, [D|Zeta], G2),
            append(S1, [C|Zeta], G4),
            append(S1, [D|Zeta], G5),
            append(S1, [C # D|Zeta], G6))
        ;(append(Delta, [C # D|E], G),
            append(Delta, [C|E], G1),
            append(Delta, [D|E], G2),
            append(Delta, [C|E1], G4),
            append(Delta, [D|E1], G5),
            append(Delta, [C # D|E1], G6))),
        !,
            assertz(proved(Z = [hia(Z3,Z1,A -> B), G => T])),
            assertz(proved(Z1 = [oia(Z4,Z6,C # D), G6 => T])),
            retract(proved(Z = [oia(Z1,Z2,C # D), G => T])),
            retract(proved(Z1 = [hia(Z3,Z4,A -> B),G1 => T])),
            retract(proved(Z2 = [hia(Z5,Z6,A -> B),G2 => T])),
            retract(proved(Z5 = [K, G3 => [A]])),
        !.

/* (R12,R15) - (D.61.1,2) */
norm2(proved(Z = [his(Z1,C -> D), G => [C -> D]])) :-
            proved(Z1 = [oia(Z2,Z3,A # B), G1 => [D]]),
            proved(Z2 = [_, G2 => [D]]),
            proved(Z3 = [_, G3 => [D]]),
        ((append(S1, [A # B|Lambda], G1),
            append(S1, [A|Lambda], G2),
            append(S1, [B|Lambda], G3),
            append(S, [A # B|Lambda], G),
            append(S, [A|Lambda], G4),
            append(S, [B|Lambda], G5))
        ;(append(Gamma, [A # B|E1], G1),
            append(Gamma, [A|E1], G2),
            append(Gamma, [B|E1], G3),
            append(Gamma, [A # B|E], G),
            append(Gamma, [A|E], G4),
            append(Gamma, [B|E], G5))),
```

```
        !,
          gensym(f,F1),
          assertz(proved(Z = [oia(Z1,F1,A#B),G => [C->D]])),
          assertz(proved(Z1 = [his(Z2,C->D),G4 => [C->D]])),
          assertz(proved(F1 = [his(Z3,C->D),G5 => [C->D]])),
          retract(proved(Z = [his(Z1,C->D),G => [C -> D]])),
          retract(proved(Z1 = [oia(Z2,Z3,A#B), G1 => [D]])),
        !.


/* (R13,R13) - (D.62) */
norm2(proved(Z = [ois(Z1,C # D), G => T])) :-
          proved(Z1 = [ois(Z2,A # B), G => T1]),
          proved(Z2 = [_, G => T2]),
          append(Phi, [A # B|E1], T1),
          append(Phi, [A|[B|E1]], T2),
          append(Phi, [A # B|E], T),
          append(Psi, [C # D|Theta], E),
          append(Psi, [C|[D|Theta]], E1),
        !,
          assertz(proved(Z = [ois(Z1,A # B), G => T])),
          append(Phi, [A|[B|E]], T3),
          assertz(proved(Z1 = [ois(Z2,C # D), G => T3])),
          retract(proved(Z = [ois(Z1,C # D), G => T])),
          retract(proved(Z1 = [ois(Z2,A # B), G => T1])),
        !.


/* (R13,R14) - (D.63) */
norm2(proved(Z = [ois(Z1,C # D), G => T])) :-
          proved(Z1 = [hia(Z2,Z3,A -> B), G => T1]),
          proved(Z2 = [_, G1 => [A]]),
          proved(Z3 = [_, G2 => T1]),
          append(S, [A -> B|Lambda], G),
          append(S1, [B|Lambda], G2),
          append(Phi, [C # D|Psi], T),
          append(Phi, [C|[D|Psi]], T1),
        !,
          assertz(proved(Z = [hia(Z2,Z1,A -> B), G => T])),
          assertz(proved(Z1 = [ois(Z3,C # D), G2 => T])),
          retract(proved(Z = [ois(Z1,C # D), G => T])),
          retract(proved(Z1 = [hia(Z2,Z3,A -> B), G => T1])),
        !.


/* (R14,R14) - (D.64) */
norm2(proved(Z = [hia(Z1,Z2,C -> D), G => T])) :-
          proved(Z1 = [_, G1 => [C]]),
          proved(Z2 = [hia(Z3,Z4,A -> B), G2 => T]),
          proved(Z3 = [_, G3 => [A]]),
          proved(Z4 = [_, G4 => T]),
          append(S, [C -> D|Bal], G),
          append(S1, [D|Bal], G2),
          append(S2, [D|Bal], G4),
          append(S3, [A -> B|Zeta], S1),
          append(Lambda, [B|Zeta], S2),
          append(S1, G1, S),
        !,
          assertz(proved(Z = [hia(Z3,Z2,A -> B), G => T])),
          append(S2, G1, E),
          append(E, [C -> D|Bal], G5),
          assertz(proved(Z2 = [hia(Z1,Z4,C -> D), G5 => T])),
          retract(proved(Z = [hia(Z1,Z2,C -> D), G => T])),
          retract(proved(Z2 = [hia(Z3,Z4,A -> B), G2 => T])),
        !.


/* (R14,R15) - (D.65.1,2) */
norm2(proved(Z = [his(Z1,C -> D), G => [C -> D]])) :-
          proved(Z1 = [hia(Z2,Z3,A -> B), G1 => [D]]),
          proved(Z2 = [_, G2 => [A]]),
          proved(Z3 = [_, G3 => [D]]),
        ((append(S1, [A -> B|Zeta], G1),
          append(S2, [B|Zeta], G3),
          append(S, [A -> B|Zeta], G),
          append(Delta_Lambda, G2, S),
          append(Delta_Lambda, [B|Zeta], G4))
        ;(append(S, [A -> B|E1], G1),
          append(Delta, [B|E1], G3),
          append(S, [A -> B|E], G),
          append(Delta, [B|E], G4))),
        !,
          assertz(proved(Z = [hia(Z2,Z1,A->B),G => [C->D]])),
```

```
        assertz(proved(Z1 = [his(Z3,C->D),G4 => [C->D]])),
        retract(proved(Z = [his(Z1,C -> D), G => [C->D]])),
        retract(proved(Z1 = [hia(Z2,Z3,A->B),G1 => [D]])),
    !.

/* (R14,R15) - (D.65.3) */
norm2(proved(Z = [his(Z1,(A->B)->F), G => [(A->B)->F]])) :-
        proved(Z1 = [hia(Z2,Z3,A -> B), G1 => [F]]),
        proved(Z2 = [_, G2 => [A]]),
        proved(Z3 = [hia(Z4,Z5,C -> D), G3 => [F]]),
        proved(Z4 = [_, G4 => [C]]),
        proved(Z5 = [_, G5 => [F]]),
        append(S1, [C -> D|Bai], G3),
        append(S2, [D|Bai], G5),
        append(S3, [C -> D|Bai], G1),
        append(S, [C -> D|Bai], G),
        append(Lambda_Delta, [A -> B|E1], S3),
        append(Lambda, [B|E1], S1),
        append(Lambda_Delta_Zeta, G4, S),
    !,
        assertz(proved(Z=[hia(Z4,Z1,C->D),G=>[(A->B)->F]])),
        append(Lambda_Delta_Zeta, [D|Bai], G6),
        assertz(proved(Z1=[his(Z3,(A->B)->F),G6=>[(A->B)->F]])),
        append(K, G4, S3),
        append(K, [D|Bai], G7),
        assertz(proved(Z3 = [hia(Z2,Z5,A -> B), G7 => [F]])),
        retract(proved(Z=[his(Z1,(A->B)->F),G=>[(A->B)->F]])),
        retract(proved(Z1 = [hia(Z2,Z3,A->B),G1 => [F]])),
        retract(proved(Z3 = [hia(Z4,Z5,C->D),G3 => [F]])),
    !.

/* (R14,R15) - (D.65.4,5) */
norm2(proved(Z = [his(Z1,(C&D) -> F), G => [(C&D) -> F]])):-
        proved(Z1 = [ala(Z2,C & D), G1 => [F]]),
        proved(Z2 = [hia(Z3,Z4,A -> B), G2 => [F]]),
        proved(Z3 = [_, G3 => [A]]),
        proved(Z4 = [_, G4 => [F]]),
    ((append(Delta, [C & D|E], G1),
        append(Delta, [C|[D|E]], G2),
        append(Delta, [C|[D|E1]], G4),
        append(Delta, E, G),
        append(Lambda, [B|Zeta], E1),
        append(S1, [A -> B|Zeta], E),
        append(Delta, E1, G5),
        append(Delta, [C & D|E1], G6))
    ;(append(S, [C & D|Zeta], G1),
        append(S, [C|[D|Zeta]], G2),
        append(S1, [C|[D|Zeta]], G4),
        append(S, Zeta, G),
        append(Delta, [B|Lambda], S1),
        append(S2, [A -> B|Lambda],S),
        append(S1, Zeta, G5),
        append(S1, [C & D|Zeta], G6))),
    !,
        assertz(proved(Z=[hia(Z3,Z1,A->B),G=>[(C&D)->F]])),
        assertz(proved(Z1=[his(Z2,(C&D)->F),G5=>[(C&D)->F]])),
        assertz(proved(Z2 = [ala(Z4,C & D), G6 => [F]])),
        retract(proved(Z=[his(Z1,(C&D)->F),G=>[(C&D)->F]])),
        retract(proved(Z1 = [ala(Z2,C & D), G1 => [F]])),
        retract(proved(Z2 = [hia(Z3,Z4,A->B),G2 => [F]])),
    !.


/* (R14,R15) - (D.65.6,7) */
norm2(proved(Z = [his(Z1,C -> F), G => [C -> F]])) :-
        proved(Z1 = [th(Z2,C), G1 => [F]]),
        proved(Z2 = [hia(Z3,Z4,A -> B), G => [F]]),
        proved(Z3 = [_, G2 => [A]]),
        proved(Z4 = [_, G3 => [F]]),
    ((append(S, [C|Zeta], G1),
        append(S, Zeta, G),
        append(S1, Zeta, G3),
        append(Delta, [B|Lambda], S1),
        append(Delta_Gamma, [A -> B|Lambda], S),
        append(S1, [C|Zeta],G4))
    ;(append(Delta,[C|E], G1),
        append(Delta, E, G),
        append(Delta, E1, G3),
        append(Lambda, [B|Zeta], E1),
        append(Lambda_Gamma, [A -> B|Zeta], E),
```

```
          append(Delta, [C|E1], G4))),
        !,
          assertz(proved(Z = [hia(Z3,Z1,A->B),G => [C->F]])),
          assertz(proved(Z1 = [his(Z2,C->F),G3 => [C->F]])),
          assertz(proved(Z2 = [th(Z4,C), G4 => [F]])),
          retract(proved(Z = [his(Z1,C->F), G => [C->F]])),
          retract(proved(Z1 = [th(Z2,C), G1 => [F]])),
          retract(proved(Z2 = [hia(Z3,Z4,A->B), G => [F]])),
        !.


/* (R2,R3,R10) - (D.73.1) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
          proved(Z1 = [_, G => T1]),
          proved(Z2 = [con(Z3,C), G => T2]),
          proved(Z3 = [_, G1 => T2]),
          append(Gamma, [C|[C|Delta]], G1),
          append(Gamma, [C|Delta], G),
          append(Phi, [A & B|Psi], T),
          append(Phi, [A|Psi], T1),
          append(Phi, [B|Psi], T2),
        !,
          assertz(proved(Z = [con(Z2,C), G => T])),
          gensym(f,F1),
          assertz(proved(Z2 = [ais(F1,Z3,A & B), G1 => T])),
          assertz(proved(F1 = [th(Z1,C), G1 => T1])),
          retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
          retract(proved(Z2 = [con(Z3,C), G => T2])),
        !.


/* (R2,R3,R10) - (D.73.2) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
          proved(Z1 = [con(Z3,C), G => T1]),
          proved(Z2 = [_, G => T2]),
          proved(Z3 = [_, G1 => T1]),
          append(Gamma, [C|[C|Delta]], G1),
          append(Gamma, [C|Delta], G),
          append(Phi, [A & B|Psi], T),
          append(Phi, [A|Psi], T1),
          append(Phi, [B|Psi], T2),
        !,
          assertz(proved(Z = [con(Z1,C), G => T])),
          gensym(f,F1),
          assertz(proved(Z1 = [ais(Z3,F1,A & B), G1 => T])),
          assertz(proved(F1 = [th(Z2,C), G1 => T2])),
          retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
          retract(proved(Z1 = [con(Z3,C), G => T1])),
        !.


/* (R2,R3,R12) - (D.74.1,3) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
          proved(Z1 = [_, G1 => T]),
          proved(Z2 = [con(Z3,C), G2 => T]),
          proved(Z3 = [_, G3 => T]),
        ((append(S, [A # B|Lambda], G),
          append(S, [A|Lambda], G1),
          append(S, [B|Lambda], G2),
          append(S1, [B|Lambda], G3),
          append(Gamma, [C|[C|Delta]], S1),
          append(Gamma, [C|Delta], S),
          append(S1, [A # B|Lambda], G4),
          append(S1, [A|Lambda], G5))
        ;(append(Gamma, [A # B|E], G),
          append(Gamma, [A|E], G1),
          append(Gamma, [B|E], G2),
          append(Gamma, [B|E1], G3),
          append(Delta, [C|[C|Lambda]], E1),
          append(Delta, [C|Lambda], E),
          append(Gamma, [A # B|E1], G4),
          append(Gamma, [A|E1], G5))),
        !,
          assertz(proved(Z = [con(Z2,C), G => T])),
          gensym(f,F1),
          assertz(proved(Z2 = [oia(F1,Z3,A # B), G4 => T])),
          assertz(proved(F1 = [th(Z1,C), G5 => T])),
          retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
          retract(proved(Z2 = [con(Z3,C), G2 => T])),
        !.
```

```
/* (R2,R3,R12) - (D.74.2,4) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
        proved(Z1 = [con(Z3,C), G1 => T]),
        proved(Z2 = [_, G2 => T]),
        proved(Z3 = [_, G3 => T]),
      ((append(S, [A # B|Lambda], G),
        append(S, [A|Lambda], G1),
        append(S, [B|Lambda], G2),
        append(S1, [A|Lambda], G3),
        append(Gamma, [C|[C|Delta]], S1),
        append(Gamma, [C|Delta], S),
        append(S1, [A # B|Lambda], G4),
        append(S1, [A|Lambda], G5))
      ;(append(Gamma, [A # B|E], G),
        append(Gamma, [A|E], G1),
        append(Gamma, [B|E], G2),
        append(Gamma, [A|E1], G3),
        append(Delta, [C|[C|Lambda]], E1),
        append(Delta, [C|Lambda], E),
        append(Gamma, [A # B|E1], G4),
        append(Gamma, [A|E1], G5))),
      !,
        assertz(proved(Z = [con(Z1,C), G => T])),
        gensym(f,F1),
        assertz(proved(Z1 = [oia(Z3,F1,A # B), G4 => T])),
        assertz(proved(F1 = [th(Z2,C), G5 => T])),
        retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        retract(proved(Z1 = [con(Z3,C), G1 => T])),
      !.


/* (R2,R4,R10) - (D.75.1) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
        proved(Z1 = [th(Z3,D), G => T1]),
        proved(Z2 = [inc(Z4,D,C), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [C|Delta], G1),
        append(Gamma, [C|[D|Delta]], G2),
        append(Phi, [A & B|Psi], T),
        append(Phi, [A|Psi], T1),
        append(Phi, [B|Psi], T2),
      !,
        assertz(proved(Z = [inc(Z2,D,C), G => T])),
        assertz(proved(Z2 = [ais(Z1,Z4,A & B), G2 => T])),
        assertz(proved(Z1 = [th(Z3,D), G2 => T1])),
        retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        retract(proved(Z1 = [th(Z3,D), G => T1])),
        retract(proved(Z2 = [inc(Z4,D,C), G => T2])),
      !.


/* (R2,R4,R10) - (D.75.2) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
        proved(Z1 = [th(Z3,C), G => T1]),
        proved(Z2 = [inc(Z4,D,C), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [D|Delta], G1),
        append(Gamma, [C|[D|Delta]], G2),
        append(Phi, [A & B|Psi], T),
        append(Phi, [A|Psi], T1),
        append(Phi, [B|Psi], T2),
      !,
        assertz(proved(Z = [inc(Z2,D,C), G => T])),
        assertz(proved(Z2 = [ais(Z1,Z4,A & B), G2 => T])),
        assertz(proved(Z1 = [th(Z3,C), G2 => T1])),
        retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        retract(proved(Z1 = [th(Z3,C), G => T1])),
        retract(proved(Z2 = [inc(Z4,D,C), G => T2])),
      !.


/* (R2,R4,R10) - (D.75.3) */
norm2(proved(Z = [ais(Z1,Z2,B & A), G => T])) :-
        proved(Z1 = [inc(Z3,D,C), G => T1]),
        proved(Z2 = [th(Z4,D), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [C|[D|Delta]], G1),
```

```
        append(Gamma, [C|Delta], G2),
        append(Phi, [B & A|Psi], T),
        append(Phi, [B|Psi], T1),
        append(Phi, [A|Psi], T2),
    !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        assertz(proved(Z1 = [ais(Z3,Z2,B & A), G1 => T])),
        assertz(proved(Z2 = [th(Z4,D), G1 => T2])),
        retract(proved(Z = [ais(Z1,Z2,B & A), G => T])),
        retract(proved(Z1 = [inc(Z3,D,C), G => T1])),
        retract(proved(Z2 = [th(Z4,D), G => T2])),
    !.


/* (R2,R4,R10) - (D.75.4) */
norm2(proved(Z = [ais(Z1,Z2,B & A), G => T])) :-
        proved(Z1 = [inc(Z3,D,C), G => T1]),
        proved(Z2 = [th(Z4,C), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Gamma, [D|[C|Delta]], G),
        append(Gamma, [C|[D|Delta]], G1),
        append(Gamma, [D|Delta], G2),
        append(Phi, [B & A|Psi], T),
        append(Phi, [B|Psi], T1),
        append(Phi, [A|Psi], T2),
    !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        assertz(proved(Z1 = [ais(Z3,Z2,B & A), G1 => T])),
        assertz(proved(Z2 = [th(Z4,C), G1 => T2])),
        retract(proved(Z = [ais(Z1,Z2,B & A), G => T])),
        retract(proved(Z1 = [inc(Z3,D,C), G => T1])),
        retract(proved(Z2 = [th(Z4,C), G => T2])),
    !.


/* (R2,R4,R12) - (D.76.1,5) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
        proved(Z1 = [th(Z3,D), G1 => T]),
        proved(Z2 = [inc(Z4,D,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        proved(Z4 = [_, G4 => T]),
    ((append(Gamma, [D|[C|E1]], G1),
        append(Gamma, [C|E1], G3),
        append(Gamma, [D|[C|E2]], G2),
        append(Gamma, [C|[D|E2]], G4),
        append(Gamma, [D|[C|E]], G),
        append(Delta, [A # B|Lambda], E),
        append(Delta, [A|Lambda], E1),
        append(Delta, [B|Lambda], E2),
        append(Gamma, [C|[D|E]], G5),
        append(Gamma, [C|[D|E1]], G6))
    ;(append(S1, [D|[C|Lambda]], G1),
        append(S1, [C|Lambda], G3),
        append(S2, [D|[C|Lambda]], G2),
        append(S2, [C|[D|Lambda]], G4),
        append(S, [D|[C|Lambda]], G),
        append(Gamma, [A # B|Delta], S),
        append(Gamma, [A|Delta], S1),
        append(Gamma, [B|Delta], S2),
        append(S, [C|[D|Lambda]], G5),
        append(S1, [C|[D|Lambda]], G6))),
    !,
        assertz(proved(Z = [inc(Z2,D,C), G => T])),
        assertz(proved(Z2 = [oia(Z1,Z4,A # B), G5 => T])),
        assertz(proved(Z1 = [th(Z3,D), G6 => T])),
        retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        retract(proved(Z1 = [th(Z3,D), G1 => T])),
        retract(proved(Z2 = [inc(Z4,D,C), G2 => T])),
    !.


/* (R2,R4,R12) - (D.76.2,6) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
        proved(Z1 = [th(Z3,C), G1 => T]),
        proved(Z2 = [inc(Z4,D,C), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        proved(Z4 = [_, G4 => T]),
    ((append(Gamma, [D|[C|E1]], G1),
        append(Gamma, [D|E1], G3),
        append(Gamma, [D|[C|E2]], G2),
        append(Gamma, [C|[D|E2]], G4),
        append(Gamma, [D|[C|E]], G),
```

```
            append(Delta, [A # B|Lambda], E),
            append(Delta, [A|Lambda], E1),
            append(Delta, [B|Lambda], E2),
            append(Gamma, [C|[D|E]], G5),
            append(Gamma, [C|[D|E1]], G6))
          ;(append(S1, [D|[C|Lambda]], G1),
            append(S1, [D|Lambda], G3),
            append(S2, [D|[C|Lambda]], G2),
            append(S2, [C|[D|Lambda]], G4),
            append(S, [D|[C|Lambda]], G),
            append(Gamma, [A # B|Delta], S),
            append(Gamma, [A|Delta], S1),
            append(Gamma, [B|Delta], S2),
            append(S, [C|[D|Lambda]], G5),
            append(S1, [C|[D|Lambda]], G6))),
          !,
            assertz(proved(Z = [inc(Z2,D,C), G => T])),
            assertz(proved(Z2 = [ola(Z1,Z4,A # B), G5 => T])),
            assertz(proved(Z1 = [th(Z3,C), G6 => T])),
            retract(proved(Z = [ola(Z1,Z2,A # B), G => T])),
            retract(proved(Z1 = [th(Z3,C), G1 => T])),
            retract(proved(Z2 = [inc(Z4,D,C), G2 => T])),
          !.


/* (R2,R4,R12) - (D.76.3,7) */
norm2(proved(Z = [ola(Z1,Z2,B # A), G => T])) :-
            proved(Z1 = [inc(Z3,D,C), G1 => T]),
            proved(Z2 = [th(Z4,D), G2 => T]),
            proved(Z3 = [_, G3 => T]),
            proved(Z4 = [_, G4 => T]),
          ((append(Gamma, [D|[C|E1]], G1),
            append(Gamma, [C|[D|E1]], G3),
            append(Gamma, [D|[C|E2]], G2),
            append(Gamma, [C|E2], G4),
            append(Gamma, [D|[C|E]], G),
            append(Delta, [B # A|Lambda], E),
            append(Delta, [B|Lambda], E1),
            append(Delta, [A|Lambda], E2),
            append(Gamma, [C|[D|E]], G5),
            append(Gamma, [C|[D|E2]], G6))
          ;(append(S1, [D|[C|Lambda]], G1),
            append(S1, [C|[D|Lambda]], G3),
            append(S2, [D|[C|Lambda]], G2),
            append(S2, [C|Lambda], G4),
            append(S, [D|[C|Lambda]], G),
            append(Gamma, [B # A|Delta], S),
            append(Gamma, [B|Delta], S1),
            append(Gamma, [A|Delta], S2),
            append(S, [C|[D|Lambda]], G5),
            append(S2, [C|[D|Lambda]], G6))),
          !,
            assertz(proved(Z = [inc(Z1,D,C), G => T])),
            assertz(proved(Z1 = [ola(Z3,Z2,B # A), G5 => T])),
            assertz(proved(Z2 = [th(Z4,D), G6 => T])),
            retract(proved(Z = [ola(Z1,Z2,B # A), G => T])),
            retract(proved(Z1 = [inc(Z3,D,C), G1 => T])),
            retract(proved(Z2 = [th(Z4,D), G2 => T])),
          !.


/* (R2,R4,R12) - (D.76.4,8) */
norm2(proved(Z = [ola(Z1,Z2,B # A), G => T])) :-
            proved(Z1 = [inc(Z3,D,C), G1 => T]),
            proved(Z2 = [th(Z4,C), G2 => T]),
            proved(Z3 = [_, G3 => T]),
            proved(Z4 = [_, G4 => T]),
          ((append(Gamma, [D|[C|E1]], G1),
            append(Gamma, [C|[D|E1]], G3),
            append(Gamma, [D|[C|E2]], G2),
            append(Gamma, [D|E2], G4),
            append(Gamma, [D|[C|E]], G),
            append(Delta, [B # A|Lambda], E),
            append(Delta, [B|Lambda], E1),
            append(Delta, [A|Lambda], E2),
            append(Gamma, [C|[D|E]], G5),
            append(Gamma, [C|[D|E2]], G6))
          ;(append(S1, [D|[C|Lambda]], G1),
            append(S1, [C|[D|Lambda]], G3),
            append(S2, [D|[C|Lambda]], G2),
            append(S2, [D|Lambda], G4),
            append(S, [D|[C|Lambda]], G),
            append(Gamma, [B # A|Delta], S),
```

```
        append(Gamma, [B|Delta], S1),
        append(Gamma, [A|Delta], S2),
        append(S, [C|[D|Lambda]], G5),
        append(S2, [C|[D|Lambda]], G6))),
      !,
        assertz(proved(Z = [inc(Z1,D,C), G => T])),
        assertz(proved(Z1 = [ola(Z3,Z2,B # A), G5 => T])),
        assertz(proved(Z2 = [th(Z4,C), G6 => T])),
        retract(proved(Z = [ola(Z1,Z2,B # A), G => T])),
        retract(proved(Z1 = [inc(Z3,D,C), G1 => T])),
        retract(proved(Z2 = [th(Z4,C), G2 => T])),
      !.


/* (R2,R4,R12) - (D.76.9,13) */
norm2(proved(Z = [inc(Z1,L1,L2), G => T])) :-
        ((L1 = A, L2 = B # C) ; (L1 = B # C, L2 = A)),
        proved(Z1 = [ola(Z2,Z3,B # C), G1 => T]),
        proved(Z2 = [th(Z4,B), G2 => T]),
        proved(Z3 = [th(Z5,C), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        proved(Z5 = [_, G4 => T]),
      ((append(Gamma, [A|[B # C|Delta]], G),
        append(Gamma, [B # C|[A|Delta]], G1),
        append(Gamma, [B|[A|Delta]], G2),
        append(Gamma, [C|[A|Delta]], G3),
        append(Gamma, [A|Delta], G4),
        append(Gamma, [A|[B|Delta]], G5),
        append(Gamma, [A|[C|Delta]], G6))
      ;(append(Gamma, [B # C|[A|Delta]], G),
        append(Gamma, [A|[B # C|Delta]], G1),
        append(Gamma, [A|[B|Delta]], G2),
        append(Gamma, [A|[C|Delta]], G3),
        append(Gamma, [A|Delta], G4),
        append(Gamma, [B|[A|Delta]], G5),
        append(Gamma, [C|[A|Delta]], G6))),
      !,
        assertz(proved(Z = [ola(Z2,Z3,B # C), G => T])),
        assertz(proved(Z2 = [th(Z4,B), G5 => T])),
        assertz(proved(Z3 = [th(Z5,C), G6 => T])),
        retract(proved(Z = [inc(Z1,L1,L2), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,B # C), G1 => T])),
        retract(proved(Z2 = [th(Z4,B), G2 => T])),
        retract(proved(Z3 = [th(Z5,C), G3 => T])),
      !.



/* (R2,R4,R12) - (D.76.10,14) */
norm2(proved(Z = [inc(Z1,L1,L2), G => T])) :-
        ((L1 = A, L2 = B # C) ; (L1 = B # C, L2 = A)),
        proved(Z1 = [ola(Z2,Z3,B # C), G1 => T]),
        proved(Z2 = [th(Z4,A), G2 => T]),
        proved(Z3 = [th(Z5,C), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        proved(Z5 = [_, G5 => T]),
      ((append(Gamma, [A|[B # C|Delta]], G),
        append(Gamma, [B # C|[A|Delta]], G1),
        append(Gamma, [B|[A|Delta]], G2),
        append(Gamma, [C|[A|Delta]], G3),
        append(Gamma, [B|Delta], G4),
        append(Gamma, [A|Delta], G5),
        append(Gamma, [A|[B|Delta]], G6),
        append(Gamma, [A|[C|Delta]], G7))
      ;(append(Gamma, [B # C|[A|Delta]], G),
        append(Gamma, [A|[B # C|Delta]], G1),
        append(Gamma, [A|[B|Delta]], G2),
        append(Gamma, [A|[C|Delta]], G3),
        append(Gamma, [B|Delta], G4),
        append(Gamma, [A|Delta], G5),
        append(Gamma, [B|[A|Delta]], G6),
        append(Gamma, [C|[A|Delta]], G7))),
      !,
        assertz(proved(Z = [ola(Z2,Z3,B # C), G => T])),
        assertz(proved(Z2 = [th(Z4,A), G6 => T])),
        assertz(proved(Z3 = [th(Z5,C), G7 => T])),
        retract(proved(Z = [inc(Z1,L1,L2), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,B # C), G1 => T])),
        retract(proved(Z2 = [th(Z4,A), G2 => T])),
        retract(proved(Z3 = [th(Z5,C), G3 => T])),
      !.
```

```
/* (R2,R4,R12) - (D.76.11,15) */
norm2(proved(Z = [inc(Z1,L1,L2), G => T])) :-
        ((L1 = A, L2 = B # C) ; (L1 = B # C, L2 = A)),
        proved(Z1 = [ola(Z2,Z3,B # C), G1 => T]),
        proved(Z2 = [th(Z4,B), G2 => T]),
        proved(Z3 = [th(Z5,A), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        proved(Z5 = [_, G5 => T]),
      ((append(Gamma, [A|[B # C|Delta]], G),
        append(Gamma, [B # C|[A|Delta]], G1),
        append(Gamma, [B|[A|Delta]], G2),
        append(Gamma, [C|[A|Delta]], G3),
        append(Gamma, [A|Delta], G4),
        append(Gamma, [C|Delta], G5),
        append(Gamma, [A|[B|Delta]], G6),
        append(Gamma, [A|[C|Delta]], G7))
      ;(append(Gamma, [B # C|[A|Delta]], G),
        append(Gamma, [A|[B # C|Delta]], G1),
        append(Gamma, [A|[B|Delta]], G2),
        append(Gamma, [A|[C|Delta]], G3),
        append(Gamma, [A|Delta], G4),
        append(Gamma, [C|Delta], G5),
        append(Gamma, [B|[A|Delta]], G6),
        append(Gamma, [C|[A|Delta]], G7))),
      !,
        assertz(proved(Z = [ola(Z2,Z3,B # C), G => T])),
        assertz(proved(Z2 = [th(Z4,B), G6 => T])),
        assertz(proved(Z3 = [th(Z5,A), G7 => T])),
        retract(proved(Z = [inc(Z1,L1,L2), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,B # C), G1 => T])),
        retract(proved(Z2 = [th(Z4,B), G2 => T])),
        retract(proved(Z3 = [th(Z5,A), G3 => T])),
      !.


/* (R2,R4,R12) - (D.76.12,16) */
norm2(proved(Z = [inc(Z1,L1,L2), G => T])) :-
        ((L1 = A, L2 = B # C) ; (L1 = B # C, L2 = A)),
        proved(Z1 = [ola(Z2,Z3,B # C), G1 => T]),
        proved(Z2 = [th(Z4,A), G2 => T]),
        proved(Z3 = [th(Z5,A), G3 => T]),
        proved(Z4 = [_, G4 => T]),
        proved(Z5 = [_, G5 => T]),
      ((append(Gamma, [A|[B # C|Delta]], G),
        append(Gamma, [B # C|[A|Delta]], G1),
        append(Gamma, [B|[A|Delta]], G2),
        append(Gamma, [C|[A|Delta]], G3),
        append(Gamma, [B|Delta], G4),
        append(Gamma, [C|Delta], G5),
        append(Gamma, [A|[B|Delta]], G6),
        append(Gamma, [A|[C|Delta]], G7))
      ;(append(Gamma, [B # C|[A|Delta]], G),
        append(Gamma, [A|[B # C|Delta]], G1),
        append(Gamma, [A|[B|Delta]], G2),
        append(Gamma, [A|[C|Delta]], G3),
        append(Gamma, [B|Delta], G4),
        append(Gamma, [C|Delta], G5),
        append(Gamma, [B|[A|Delta]], G6),
        append(Gamma, [C|[A|Delta]], G7))),
      !,
        assertz(proved(Z = [ola(Z2,Z3,B # C), G => T])),
        assertz(proved(Z2 = [th(Z4,A), G6 => T])),
        assertz(proved(Z3 = [th(Z5,A), G7 => T])),
        retract(proved(Z = [inc(Z1,L1,L2), G => T])),
        retract(proved(Z1 = [ola(Z2,Z3,B # C), G1 => T])),
        retract(proved(Z2 = [th(Z4,A), G2 => T])),
        retract(proved(Z3 = [th(Z5,A), G3 => T])),
      !.


/* (R3,R3,R10) - (D.77.1) */
norm2(proved(Z = [als(Z1,Z2,A & B), G => T])) :-
        proved(Z1 = [con(Z3,C), G => T1]),
        proved(Z2 = [con(Z4,D), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Phi, [A & B|Psi], T),
        append(Phi, [A|Psi], T1),
        append(Phi, [B|Psi], T2),
        append(Gamma, [C|[D|Delta]], G),
        append(Gamma, [C|[C|[D|Delta]]], G1),
        append(Gamma, [C|[D|[D|Delta]]], G2),
      !,
```

```
        gensym(f,F1),
        assertz(proved(Z = [con(F1,C), G => T])),
        gensym(f,F2),
        assertz(proved(F1 = [con(F2,D), G1 => T])),
        append(Gamma, [C|[C|[D|[D|Delta]]]], G3),
        assertz(proved(F2 = [ais(Z1,Z2,A & B), G3 => T])),
        assertz(proved(Z1 = [th(Z3,D), G3 => T1])),
        assertz(proved(Z2 = [th(Z4,C), G3 => T2])),
        retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        retract(proved(Z1 = [con(Z3,C), G => T1])),
        retract(proved(Z2 = [con(Z4,D), G => T2])),
    !.


/* (R3,R3,R10) - (D.77.2) */
norm2(proved(Z = [ais(Z1,Z2,A & B), G => T])) :-
        proved(Z1 = [con(Z3,D), G => T1]),
        proved(Z2 = [con(Z4,C), G => T2]),
        proved(Z3 = [_, G1 => T1]),
        proved(Z4 = [_, G2 => T2]),
        append(Phi, [A & B|Psi], T),
        append(Phi, [A|Psi], T1),
        append(Phi, [B|Psi], T2),
        append(Gamma, [C|[D|Delta]], G),
        append(Gamma, [C|[D|[D|Delta]]], G1),
        append(Gamma, [C|[C|[D|Delta]]], G2),
    !,
        gensym(f,F1),
        assertz(proved(Z = [con(F1,C), G => T])),
        gensym(f,F2),
        assertz(proved(F1 = [con(F2,D), G2 => T])),
        append(Gamma, [C|[C|[D|[D|Delta]]]], G3),
        assertz(proved(F2 = [ais(Z1,Z2,A & B), G3 => T])),
        assertz(proved(Z1 = [th(Z3,C), G3 => T1])),
        assertz(proved(Z2 = [th(Z4,D), G3 => T2])),
        retract(proved(Z = [ais(Z1,Z2,A & B), G => T])),
        retract(proved(Z1 = [con(Z3,D), G => T1])),
        retract(proved(Z2 = [con(Z4,C), G => T2])),
    !.


/* (R3,R3,R12) - (D.78.1,3) */
norm2(proved(Z = [oia(Z1,Z2,A # B), G => T])) :-
        proved(Z1 = [con(Z3,C), G1 => T]),
        proved(Z2 = [con(Z4,D), G2 => T]),
        proved(Z3 = [_, G3 => T]),
        proved(Z4 = [_, G4 => T]),
    ((append(Gamma, [C|[D|E1]], G1),
        append(Gamma, [C|[C|[D|E1]]], G3),
        append(Gamma, [C|[D|E2]], G2),
        append(Gamma, [C|[D|[D|E2]]], G4),
        append(Gamma, [C|[D|E]], G),
        append(Delta, [A # B|Lambda], E),
        append(Delta, [A|Lambda], E1),
        append(Delta, [B|Lambda], E2),
        append(Gamma, [C|[C|[D|E]]], G5),
        append(Gamma, [C|[C|[D|[D|E]]]], G6),
        append(Gamma, [C|[C|[D|[D|E1]]]], G7),
        append(Gamma, [C|[C|[D|[D|E2]]]], G8))
    ;(append(S1, [C|[D|Lambda]], G1),
        append(S1, [C|[C|[D|Lambda]]], G3),
        append(S2, [C|[D|Lambda]], G2),
        append(S2, [C|[D|[D|Lambda]]], G4),
        append(S, [C|[D|Lambda]], G),
        append(Gamma, [A # B|Delta], S),
        append(Gamma, [A|Delta], S1),
        append(Gamma, [B|Delta], S2),
        append(S, [C|[C|[D|Lambda]]], G5),
        append(S, [C|[C|[D|[D|Lambda]]]], G6),
        append(S1, [C|[C|[D|[D|Lambda]]]], G7),
        append(S2, [C|[C|[D|[D|Lambda]]]], G8))),
    !,
        gensym(f,F1),
        assertz(proved(Z = [con(F1,C), G => T])),
        gensym(f,F2),
        assertz(proved(F1 = [con(F2,D), G5 => T])),
        assertz(proved(F2 = [oia(Z1,Z2,A # B), G6 => T])),
        assertz(proved(Z1 = [th(Z3,D), G7 => T])),
        assertz(proved(Z2 = [th(Z4,C), G8 => T])),
        retract(proved(Z = [oia(Z1,Z2,A # B), G => T])),
        retract(proved(Z1 = [con(Z3,C), G1 => T])),
        retract(proved(Z2 = [con(Z4,D), G2 => T])),
    !.
```

```
/* (R3,R3,R12) - (D.78.2.4) */
norm2(proved(Z = [ola(Z1,Z2,A # B), G => T])) :-
          proved(Z1 = [con(Z3,D), G1 => T]),
          proved(Z2 = [con(Z4,C), G2 => T]),
          proved(Z3 = [_, G3 => T]),
          proved(Z4 = [_, G4 => T]),
       ((append(Gamma, [C|[D|E1]], G1),
          append(Gamma, [C|[D|[D|E1]]], G3),
          append(Gamma, [C|[D|E2]], G2),
          append(Gamma, [C|[C|[D|E2]]], G4),
          append(Gamma, [C|[D|E]], G),
          append(Delta, [A # B|Lambda], E),
          append(Delta, [A|Lambda], E1),
          append(Delta, [B|Lambda], E2),
          append(Gamma, [C|[C|[D|E]]], G5),
          append(Gamma, [C|[C|[D|[D|E]]]], G6),
          append(Gamma, [C|[C|[D|[D|E1]]]], G7),
          append(Gamma, [C|[C|[D|[D|E2]]]], G8))
       ;(append(S1, [C|[D|Lambda]], G1),
          append(S1, [C|[D|[D|Lambda]]], G3),
          append(S2, [C|[D|Lambda]], G2),
          append(S2, [C|[C|[D|Lambda]]], G4),
          append(S, [C|[D|Lambda]], G),
          append(Gamma, [A # B|Delta], S),
          append(Gamma, [A|Delta], S1),
          append(Gamma, [B|Delta], S2),
          append(S, [C|[C|[D|Lambda]]], G5),
          append(S, [C|[C|[D|[D|Lambda]]]], G6),
          append(S1, [C|[C|[D|[D|Lambda]]]], G7),
          append(S2, [C|[C|[D|[D|Lambda]]]], G8))),
       !,
          gensym(f,F1),
          assertz(proved(Z = [con(F1,C), G => T])),
          gensym(f,F2),
          assertz(proved(F1 = [con(F2,D), G5 => T])),
          assertz(proved(F2 = [ola(Z1,Z2,A # B), G6 => T])),
          assertz(proved(Z1 = [th(Z3,C), G7 => T])),
          assertz(proved(Z2 = [th(Z4,D), G8 => T])),
          retract(proved(Z = [ola(Z1,Z2,A # B), G => T])),
          retract(proved(Z1 = [con(Z3,D), G1 => T])),
          retract(proved(Z2 = [con(Z4,C), G2 => T])),
       !.


/* (R4,R4,R10) - (D.79) */
norm2(proved(Z = [als(Z1,Z2,A & B), G => T])) :-
          proved(Z1 = [inc(Z3,D,C), G => T1]),
          proved(Z2 = [_, G => T2]),
          proved(Z3 = [_, G1 => T1]),
          append(Gamma, [D|[C|Delta]], G),
          append(Gamma, [C|[D|Delta]], G1),
          append(Phi, [A & B|Psi], T),
          append(Phi, [A|Psi], T1),
          append(Phi, [B|Psi], T2),
       !,
          assertz(proved(Z = [inc(Z1,D,C), G => T])),
          gensym(f,F1),
          assertz(proved(Z1 = [als(Z3,F1,A & B), G1 => T])),
          assertz(proved(F1 = [inc(Z2,C,D), G1 => T2])),
          retract(proved(Z = [als(Z1,Z2,A & B), G => T])),
          retract(proved(Z1 = [inc(Z3,D,C), G => T1])),
       !.


/* (R2) - (D.81.3) */
norm2(proved(Z = [th(Z1,A -> B), G => []])) :-
          proved(Z1 = [K, G1 => []]),
          append(Gamma, [A -> B|Delta], G),
          append(Gamma, Delta, G1),
       !,
          gensym(f,F1),
          assertz(proved(Z = [con(F1,Gamma), G => []])),
          gensym(f,F2),
          append(Gamma, G, G2),
          assertz(proved(F1 = [con(F2,Delta), G2 => []])),
          append(G2, Delta, G3),
          gensym(Step,F3),
          assertz(proved(F2 = [inc(F3,A->B,Delta),G3=>[]])),
          gensym(f,F4),
```

```
            gensym(f,F5),
            append(Gamma, G1, S),
            append(S, [A -> B|Delta], G4),
            assertz(proved(F3 = [his(F4,F5,A->B),G4 => []])),
            assertz(proved(F4 = [th(Z1,A), G1 => [A]])),
            gensym(f,F6),
            append(Gamma, [B|Delta], G5),
            assertz(proved(F5 = [th(F6,B), G5 => []])),
            assertz(proved(F6 = [K, G1 => []])),
            retract(proved(Z = [th(Z1,A -> B), G => []])),
        !.


/* (R5) - (D.82.3) */
norm2(proved(Z = [th(Z1,A -> B), G => [A -> B]])) :-
            proved(Z1 = [_, G => []]),
        !,
            gensym(f,F1),
            assertz(proved(Z = [his(F1,A->B), G => [A->B]])),
            gensym(f,F2),
            append([A], G, G1),
            assertz(proved(F1 = [th(F2,A), G1 => [B]])),
            assertz(proved(F2 = [th(Z1,B), G => [B]])),
            retract(proved(Z = [th(Z1,A -> B), G => [A->B]])),
        !.


/* (R5) - (D.82.4) */
norm2(proved(Z = [his(Z1,A -> B), G => [A -> B]])) :-
            proved(Z1 = [th(Z2,A), G1 => [B]]),
            proved(Z2 = [th(Z3,B), G => [B]]),
            proved(Z3 = [_, G => []]),
            append(Gamma, [A|Delta], G1),
            append(Gamma, Delta, G),
        !,
            assertz(proved(Z = [his(Z1,A->B), G => [A->B]])),
            append([A], G , G2),
            assertz(proved(Z1 = [th(Z2,A), G2 => [B]])),
            retract(proved(Z = [his(Z1,A->B), G => [A->B]])),
            retract(proved(Z1 = [th(Z2,A), G1 => [B]])),
        !.


norm2(proved(Z)) :-
                  retract(proved(Z)),
                  assertz(proved(Z)), !.
```

```
                        /* expinc */

/* This part is to deal with proof-steps with interchange
 * formulae : either empty or sequences of formulae */

expinc :-
      proved(Z = [inc(Z1,A,B),K]),
      (A == []
      ;B == []
      ;A == [B]
      ;B == [A]
      ;B = A
      ;(A == [D],B == [D])),
      !,
      ((proved(Z2 = [F,M]),F =.. H,member(Z,H),sub(Z,Z1,H,H1),
        F1 =.. H1, asserta(proved(Z2 = [F1,M])),
        retract(proved(Z2 = [F,M])),
        retract(proved(Z = [inc(Z1,A,B),K])))
      ;(retract(proved(Z = [inc(Z1,A,B), K])), proved(Z1 = X),
        asserta(proved(Z = X)), retract(proved(Z1 = X)))),
      expinc.

expinc :-
          (proved(Z = [inc(Z1,[A],[B]), K])
          ;(proved(Z = [inc(Z1,[A],B),K]), not(islist(B)))
          ;(proved(Z = [inc(Z1,A,[B]),K]), not(islist(A)))),
          !,
          asserta(proved(Z = [inc(Z1,A,B), K])),
          (retract(proved(Z = [inc(Z1,[A],[B]),K]))
          ;retract(proved(Z = [inc(Z1,[A],B), K]))
          ;retract(proved(Z = [inc(Z1,A,[B]), K]))),
          expinc.

expinc :-
          proved(Z = [inc(Z1,L,A), K]),
          not(islist(A)), L = [_|_],
          !,
          asserta(proved(Z = [inc(Z1,L,[A]), K])),
          retract(proved(Z = [inc(Z1,L,A), K])),
          expinc.

expinc :-
          proved(Z = [inc(Z1,A,L), K]),
          not(islist(A)), L = [_|_],
          !,
          asserta(proved(Z = [inc(Z1,[A],L), K])),
          retract(proved(Z = [inc(Z1,A,L), K])),
          expinc.

expinc :- expinca, expincs.


expinca :-
          proved(Z = [inc(Z1,L2,L1), G => T]),
          proved(Z1 = [_, G1 => T]),
          L1 = [A|Rest1], L2 = [B|Rest2],
          (Rest1 \== [] ; Rest2 \== []),
          !,
          break_up(G1,L1,L2,G,S,F),
          !,
          move_overa(L1,L2,S,F,Z1,T),
          retract(proved(Z = [inc(Z1,L2,L1), G => T])),
          proved(Y = [K, G => T]),
          !,
          asserta(proved(Z = [K, G => T])),
          retract(proved(Y = [K, G => T])),
          expinca.

expinca.


move_overa(L1,[],S,F,Z1,T) :-
                               !.
move_overa(N,M,S,F,Z1,T)       :-
                 M = [A|M1],
                 singlemovea(N,[A|M1],S,F,Z1,T),
                 append(S,[A],S1),
                 append(N,M1,N1),
                 append(S1,N1,S2),
                 append(S2,F,G),
                 proved(F1 =[K,G => T]),
                 move_overa(N,M1,S1,F,F1,T).
```

```
singlemovea([],M,S,F,Z1,T) :-
                        !.
singlemovea(N,[A|M1],S,F,Z1,T) :-
                        append(N1,[B],N),
                        append([B],M1,M2),
                        append(N1,[A|M2],M),
                        append(S,M,S1),
                        append(S1,F,G),
                        gensym(f,F1),
                        asserta(proved(F1 = [inc(Z1,A,B), G => T])),
                        singlemovea(N1,[A|M2],S,F,F1,T).


expincs :-
              proved(Z = [inc(Z1,L2,L1), G => T]),
              proved(Z1 = [_, G => T1]),
              L1 = [A|Rest1], L2 = [B|Rest2],
              (Rest1 \== [] ; Rest2 \== []),
              !,
              break_up(T1,L1,L2,T,S,F),
              !,
              move_overs(L1,L2,S,F,Z1,G),
              retract(proved(Z = [inc(Z1,L2,L1), G => T])),
              proved(Y = [K, G => T]),
              !,
              asserta(proved(Z = [K, G => T])),
              retract(proved(Y = [K, G => T])),
              expincs.
expincs.


break_up(E1,L1,L2,E,S,F) :-
                              append(S,L1,S1),
                              append(S1,L2,F1),
                              append(F1,F,E1),
                              append(S,L2,S2),
                              append(S2,L1,F2),
                              append(F2,F,E).


move_overs(L1,[],S,F,K,E)   :-
                              !.
move_overs(N,M,S,F,K,E)     :-
                        M = [A|M1],
                        singlemoves(N,[A|M1],S,F,K,E),
                        append(S,[A],S1),
                        append(N,M1,N1),
                        append(S1,N1,S2),
                        append(S2,F,T),
                        proved(F1 =[H,E => T]),
                        move_overs(N,M1,S1,F,F1,E).


singlemoves([],M,S,F,K,E)   :-
                              !.
singlemoves(N,[A|M1],S,F,K,E) :-
                        append(N1,[B],N),
                        append([B],M1,M2),
                        append(N1,[A|M2],M),
                        append(S,M,S1),
                        append(S1,F,T),
                        gensym(f,F1),
                        asserta(proved(F1 = [inc(K,A,B), E => T])),
                        singlemoves(N1,[A|M2],S,F,F1,E).


                              /* expcon */

/* This part deals with proof-steps which have contraction
   formula either empty or a sequence of formulae */

expcon :-
      proved(Z = [con(Z1,[]),K]),
      !,
      ((proved(Z2 = [F,M]),F =.. H,member(Z,H),sub(Z,Z1,H,H1),
        F1 =.. H1, asserta(proved(Z2 = [F1,M])),
        retract(proved(Z2 = [F,M])),
        retract(proved(Z = [con(Z1,[]), K])))
       ;(retract(proved(Z = [con(Z1,[]), K])), proved(Z1 = X),
         asserta(proved(Z = X)), retract(proved(Z1 = X)))),
      expcon.
```

```
expcon :-
          proved(Z = [con(Z1,[A]), K]),
          !,
          asserta(proved(Z = [con(Z1,A), K])),
          retract(proved(Z = [con(Z1,[A]), K])),
          expcon.
expcon :- expcona, expcons.


expcona :-
          proved(Z = [con(Z1,L), G => T]),
          proved(Z1 = [_, G1 => T]),
          L = [A|Rest], [Rest] \== [],
          !,
          break_up_con(G1,L,G,S,F),
          !,
          move_over_alla(L,L,S,F,Z1,T),
          retract(proved(Z = [con(Z1,L), G => T])),
          proved(Y = [K, G => T]),
          !,
          asserta(proved(Z = [K, G => T])),
          retract(proved(Y = [K,G => T])),
          expcona.
expcona.


move_over_alla(L,[],S,F,Z1,T)  :-
                                 !.
move_over_alla(L,N,S,F,Z1,T)   :-
              N = [A|M],
              append(L1,[A|M],L),
              singlemovea(L1,A,M,[A|M],S,F,Z1,T),
              proved(F1 = [con(_,A),_]),
              move_over_alla(L,M,S,F,F1,T).


singlemovea(L1,A,[],[A|M],S,F,K,T) :-
                !,
                  append(L1,[A|M],L),
                  append(S,L,S1),
                  append(S1,F,G),
                  gensym(f,F1),
                  asserta(proved(F1 = [con(K,A), G => T])).
singlemovea(L1,A,L2,[A|M],S,F,Z1,T) :-
                  append(L3,[B],L2),
                  append([B],M,N),
                  append(L1,[A],L4),
                  append(L4,L3,L5),
                  append(L5,[A|N],L6),
                  append(S,L6,S1),
                  append(S1,F,G),
                  gensym(f,F1),
                  asserta(proved(F1 = [inc(Z1,A,B), G => T])),
                  singlemovea(L1,A,L3,[A|N],S,F,F1,T).


expcons :-
          proved(Z = [con(Z1,L), G => T]),
          proved(Z1 = [_, G => T1]),
          L = [A|Rest], [Rest] \== [],
          !,
          break_up_con(T1,L,T,S,F),
          !,
          move_over_alls(L,L,S,F,Z1,G),
          retract(proved(Z = [con(Z1,L), G => T])),
          proved(Y = [K, G => T]),
          !,
          asserta(proved(Z = [K, G => T])),
          retract(proved(Y = [K,G => T])),
          expcons.
expcons.


break_up_con(E1,L,E,S,F) :-
             /* E is S then L then F          */
             /* E1 is S then L then L then F  */
             /* E1,L,E are imported           */
             /* S, F are exported             */
             append(S,L,S1),
             append(S1,F,E),
             append(S1,L,S2),
             append(S2,F,E1).
```

```
move_over_alls(L,[],S,F,Z1,E) :-
                                 !.
move_over_alls(L,N,S,F,Z1,E)  :-
                N = [A|M],
                append(L1,[A|M],L),
                singlemoves(L1,A,M,[A|M],S,F,Z1,E),
                proved(F1 = [con(_,A),_]),
                move_over_alls(L,M,S,F,F1,E).


singlemoves(L1,A,[],[A|M],S,F,K,E) :-
              !,
                append(L1,[A|M],L),
                append(S,L,S1),
                append(S1,F,T),
                gensym(f,F1),
                asserta(proved(F1 = [con(K,A), E => T])).
singlemoves(L1,A,L2,[A|M],S,F,K,E) :-
                append(L3,[B],L2),
                append([B],M,N),
                append(L1,[A],L4),
                append(L4,L3,L5),
                append(L5,[A|N],L6),
                append(S,L6,S1),
                append(S1,F,T),
                gensym(f,F1),
                asserta(proved(F1 = [inc(K,A,B), E => T])),
                singlemoves(L1,A,L3,[A|N],S,F,F1,E).


                          /* expth */


/* This part deals with proof-steps which have thinning
   formula either empty or a sequence of formulae */

expth :-
     proved(Z = [th(Z1,[]),K]),
     !,
     ((proved(Z2 = [F,M]),F =.. H,member(Z,H),sub(Z,Z1,H,H1),
       F1 =.. H1, asserta(proved(Z2 = [F1,M])),
       retract(proved(Z2 = [F,M])),
       retract(proved(Z = [th(Z1,[]),K])))
     ;(retract(proved(Z = [th(Z1,[]), K])), proved(Z1 = X),
       asserta(proved(Z = X)), retract(proved(Z1 = X)))),
     expth.

expth :-
          proved(Z = [th(Z1,[A]), K]),
          !,
          asserta(proved(Z = [th(Z1,A), K])),
          retract(proved(Z = [th(Z1,[A]), K])),
          expth.

expth :- expths, expths.


expths :-
          proved(Z = [th(Z1,L), G => T]),
          proved(Z1 = [_, G1 => T]),
          L = [A|Rest], Rest \== [],
          !,
          break_up_th(G,L,G1,S,F),
          !,
          add_alls(L,S,F,Z1,T),
          retract(proved(Z = [th(Z1,L), G => T])),
          proved(Y = [K, G => T]),
          !,
          asserta(proved(Z = [K, G => T])),
          retract(proved(Y = [K, G => T])),
          expths.
expths.


add_alls([],S,F,Z1,T) :-
                       !.
add_alls(N,S,F,Z1,T) :-
               N = [A|M],
               append(S,[A],S1),
               append(S1,F,G),
               gensym(f,F1),
               asserta(proved(F1 = [th(Z1,A), G => T])),
               add_alls(M,S1,F,F1,T).
```

```
expths :-
               proved(Z = [th(Z1,L), G => T]),
               proved(Z1 = [_, G => T1]),
               L = [A|Rest], Rest \== [],
               !,
               break_up_th(T,L,T1,S,F),
               !,
               add_alls(L,S,F,Z1,G),
               retract(proved(Z = [th(Z1,L), G => T])),
               proved(Y = [K, G => T]),
               !,
               asserta(proved(Z = [K, G => T])),
               retract(proved(Y = [K, G => T])),
               expths.
expths.


break_up_th(E,L,E1,S,F):-
                      append(S,L,S1),
                      append(S1,F,E),
                      append(S,F,E1).


add_alls([],S,F,K,E) :-
                      !.
add_alls(N,S,F,K,E) :-
                      N = [A|M],
                      append(S,[A],S1),
                      append(S1,F,T),
                      gensym(f,F1),
                      asserta(proved(F1 = [th(K,A), E => T])),
                      add_alls(M,S1,F,F1,E).


                                        /* printall */

/* This part is used to put all the proof in a list */

put_in_list(X,L) :-  proved(X = _),
                          prinlist(X,L).

prinlist(X,L) :-
                 prinlist(X,[],L).


prinlist(X, Sofar, K) :-
                 proved(X = Y), append([X = Y],Sofar,L2),
               ((nullary(X), K = L2)
               ;(unary(X,L), prinlist(L,L2,K))
               ;(binary(X,L,M), prinlist(L,L2,L4), prinlist(M,L4,K))
               ).

/* gives a print on the scren for the proof in the database */

printall(X) :- proved(X = Y),
                 (nullary(X)
                 ;(unary(X,L), printall(L))
                 ;(binary(X,L,M), printall(L), printall(M))),
                 nl, write(X = Y).


nullary(X)  :-
                         proved(X = [Z,_]),
                       (Z =.. [A] ; Z =.. [id,C]).


unary(X,L)  :-
                              proved(X = [Z,_]),
                              Z =.. [A|[L|_]],
                              (A = th
                              ;A = inc
                              ;A = con
                              ;A = ois
                              ;A = ala
                              ;A = his).
```

```
binary(X, L, M)      :-
                                proved(X = [Z,_]),
                                   Z =.. [A,L,M,B],
                                   (A = cut
                                   ;A = ais
                                   ;A = oia
                                   ;A = hia).


member(H,[H|_]).
member(H,[_|L]) :- member(H,L).


sub(X,C,[],[]).

sub(X,C,[X|L],[C|L1]) :- !,
                                sub(X,C,L,L1).

sub(X,C,[Y|M],[Y|M1]) :-
                                sub(X,C,M,M1).


gensym(Root,Atom) :-     /* gensym means generate symbols,
                            gensym(f,N). it will give you N=f1,
                            and continue giving you f2,f3,...
                            if you repote this question many
                            times */
    get_num(Root,Num),
    name(Root,Name1),
    integer_name(Num,Name2),
    append(Name1,Name2,Name),
    name(Atom,Name).

get_num(Root,Num) :-
    retract(current_num(Root,Num1)), !,
    Num is Num1 + 1,
    asserta(current_num(Root,Num)).
get_num(Root,1) :- asserta(current_num(Root,1)).

integer_name(Int,List) :- integer_name(Int,[],List).
integer_name(I,Sofar,[C|Sofar]) :-
    I<10, !, C is I+48.
integer_name(I,Sofar,List) :-
    Tophalf is floor(I/10), /* floor(I/10) gives the integer
                               part of the division */
    Bothalf is I mod 10,
    C is Bothalf+48,
    integer_name(Tophalf,[C|Sofar],List).



append([],L,L).
append([M|L1],L2,[M|L3]) :-
                                append(L1,L2,L3).


append3(X,Y,Z,U) :-
                       append(X,Y,Z1),
                       append(Z1,Z,U).


append4(X,Y,Z,V,U) :-
                         append3(X,Y,Z,Z1),
                         append(Z1,V,U).


append5(X,Y,Z,U,V,W) :-
                           append4(X,Y,Z,U,U1),
                           append(U1,V,W).


islist([]).
islist([_|_]).
islist([A|B]) :- islist(B).


cop :-
      proved(Z1 = [K1,H1]),
      K1 =.. [L1|Arg1],member(X,Arg1),
      proved(Z2 = [K2,H2]), Z2 \== Z1,
      K2 =.. [L2|Arg2],member(X,Arg2),
      proved(X = [F,G]),
```

```
          gensym(f,F1),
          assert(proved(F1 = [F,G])),
          sub(X,F1,Arg1,Arg11), K11 =.. [L1|Arg11],
          assert(proved(Z1 = [K11,H1])),
          retract(proved(Z1 = [K1,H1])),
          gensym(f,F2),
          assert(proved(F2 = [F,G])),
          sub(X,F2,Arg2,Arg22), K22 =.. [L2|Arg22],
          assert(proved(Z2 = [K22,H2])),
          retract(proved(Z2 = [K2,H2])),
          cop.
cop.


branch(Z) :-
              proved(Z = Y),
            ( bra_nul(Z)
            ;(bra_unar(Z,Z1), branch(Z1))
            ;(bra_binar(Z,Z1,Z2), branch(Z1), branch(Z2))).


bra_nul(Z) :-
              proved(Z = [X,Y]),
             (X =.. [_] ; X =.. [id,_]).


bra_unar(Z,Z1) :-
              proved(Z = [X,Y]),
              gensym(f,Z1),
            ((X =.. [H, L, T],
              X1 =.. [H, Z1, T])
            ;(X =.. [inc, L, T1, T2],
              X1 =.. [inc, Z1, T1, T2])),
              proved(L = K),
              assertz(proved(Z = [X1,Y])),
              assertz(proved(Z1 = K)),
              retract(proved(Z = [X,Y])).


bra_binar(Z,Z1,Z2) :-
              proved(Z = [X,Y]),
              X =.. [H, L1, L2, T],
              proved(L1 = K1),
              proved(L2 = K2),
              gensym(f,Z1),
              gensym(f,Z2),
              X1 =.. [H, Z1, Z2, T],
              assertz(proved(Z = [X1,Y])),
              assertz(proved(Z1 = K1)),
              assertz(proved(Z2 = K2)),
              retract(proved(Z = [X,Y])).
```