The interesting and fun game of Mathdialog proofs

The objective of this guide is to offer the basics notions to prove theorems in Mathdialog, the computer implementation of CZFC (Contextual ZFC) set theory. This is not an exhaustive treatise but a starting point to the first source to learn CZFC and Mathdialog: **experimentation!** To understand and enjoy this guide, participants must have the mathematical maturity that give undergraduate courses of topology, abstract algebra or set theory. If you have learn that all set theories are getting by specifying some undefined predicates and formulating a few axioms in the language of first order logic, you have to forget it for a while, this is a practical guide.

FIGURE 1. View of the mathdialog.com web site.

In a human-to-machine Mathdialog proof [Figure 1] there are two windows, the command window, and the blackboard. In the first one users write CZFC sentences and send them to be processed one by one by Mathdialog which will respond in the second one. A CZFC sentence is a theorem, a definition or a

sort of inference rule named logic-command. Outside of a proof you can write definitions and theorems but not logic-commands. A CZFC theorem is not a logical formula but a name, a hypotheses list or assumptions and a logical formula. The informal syntax for theorem is:

THEOR[LABEL;LFA;FORMULA]

LABEL is a string with no blank spaces, the theorem name.
LFA is a List of Atomic Formulas, the theorem hypotheses for humans and the **theorem local context** for Mathdialog.
FORMULA is the theorem thesis.

We will call natural language to the mathematical language in which mathematical books are written or in which mathematical classes are given. Writing Mathdialog theorems, definitions and proofs is a translation from theorems, definitions and proofs you already know in natural language. Mathdialog is not to write or develop mathematics but to translate existent undergraduate level mathematics, in a more precise way. Translating theorems in natural language to Mathdialog is already an art by itself. In the Appendix there are some theorems in the Mathdialog language, so you can focus in the proofs.

The suggested strategy is:

1) Translate the theorem you want to prove into natural language.
2) Prove it in the way you know.
3) Translate that proof into Mathdialog using the web site.

Of course, after your first one, some proofs are so easy that those steps will be redundant for them.

In Mathdialog, the collection of all theorems and definitions available is named **general context**. A human-to-machine proof starts when you write a theorem in the command window and send it. If its thesis makes sense respect to the theorem local context and the general context, Mathdialog responds displaying two lines in the backboard; the first one has the theorem thesis labeled GL1, the second one has the theorem LFA with their commas replaced with the logical connective AND and labeled H1. For example, if you send the theorem:

THEOR[SUBSET_TRANSITIVITY;SUBSET(A,B),SUBSET(B,C);SUBSET(A,C)]

Mathdialog will responds with the following lines in the blackboard

GL1 - SUBSET(A,C)
H1 - SUBSET(A,B) AND SUBSET(B,C)

The formulas labeled with GL are goals, and the ones with H are hypotheses. The purpose of a human-to-machine proof is to produce, using logic-commands, a hypothesis that matches the formula in GL1. Each one of the 25 available logic-commands in Mathdialog, maps each kind of proof steps found on mathematics textbooks. All of them are described in the Logic commands Reference Guide, and have in common that each one displays a new sub goal and/or new hypotheses on the blackboard.

There are logic-commands to reduce goal formulas to the components of its logic operators, to introduce subproof, to display as hypothesis a formula that is a propositional consequence of the **proof context** (the collection of all hypotheses available in a given point of the proof), to assume as hypothesis the definition of a atomic formula already present in the proof context, etc. The main guideline shared by all logic-commands is to left to the user the hardest choices in a proof and to the

system determining if those choices have 'sense' respect to all available kind of contexts (general, proof and local contexts). Next, we will describe the GL and H syntax and a few logic-commands.

The line labels in the blackboard such as GL1 and H1 have a general structure, you need learn to read them but not to write them, Mathdialog will; the first part indicates the type of line: GL for goal formulas, H for hypothesis formulas and U for user sent logic-commands. The second part deals with the proof structure and relates to GL; its syntax is dsdsd ... sd (DS), where s is '.' (a point) or ',' (a comma) and d is a sequence of digits. Examples of GL labels are

GL2.3,2 , GL2,3 , GL3.2 and GL3

The second part of a H line indicates to which GL that hypothesis can be used to prove the goal in that GL. For example, H3.2 means that its hypothesis can only be used to prove the goal formula in the upper closest GL3.2 line. The figure 2 shows a commented human-to-machine proof; a simply but good exercise would be formulate and prove the theorem in natural language.

In CZFC:

BG(x,A) means x belongs to A
SET(x) means that x is a set
EQ(A,B) means that A is equal to B
FA means For All
TE means There Exists
TE! means There Exists one and only one

Now we will show how the DSs are related to the proof structure. When a new goal line is introduced which does not necessarily imply the upper closest GL line, the DS ".1" (as in GL1.1 in the figure 2 and as in GL3.1 of the next example) is added to the label of the new goal line. For example, suppose the formula F is in GL3 and we want use the theorem T to get an important new hypothesis G. But we realize we don't have in the proof context one of the hypothesis W in the LFA of T. So, we want to temporarily change the actual goal formula F to the needed hypothesis W in order to try to prove it. If we are successful, we will be able to use W as hypothesis of the old goal formula F so we can use T. That is done with the logic-command IP.

⋮
GL3 - F
U - IP[W] *(The IP (Intermediate Proof) logic-command, allows to introduce any valid formula under our context, as a new goal)*
GL3.1 - W *(The new goal)*
...(proof of W)
U - Logic-command[W] *(logic-command that ends the proof of W by generating it)*
GL3 - F
HD3 - W *(Now, the formula W in HD3 is a new hypotheses that can be used to prove GL3. The D after H means that its formula comes from a proof. D comes from the Spanish word "Demostrada" that means proved)*
BY_THEOR[Thesis of T] *(The BY_THEOR logic-command by BY THEORem has as argument any valid formula under our context. If formula is the thesis of a theorem which hypotheses are available in the proof context, it assumes formula as a new hypothesis)*
H3 – *Thesis of T*
⋮

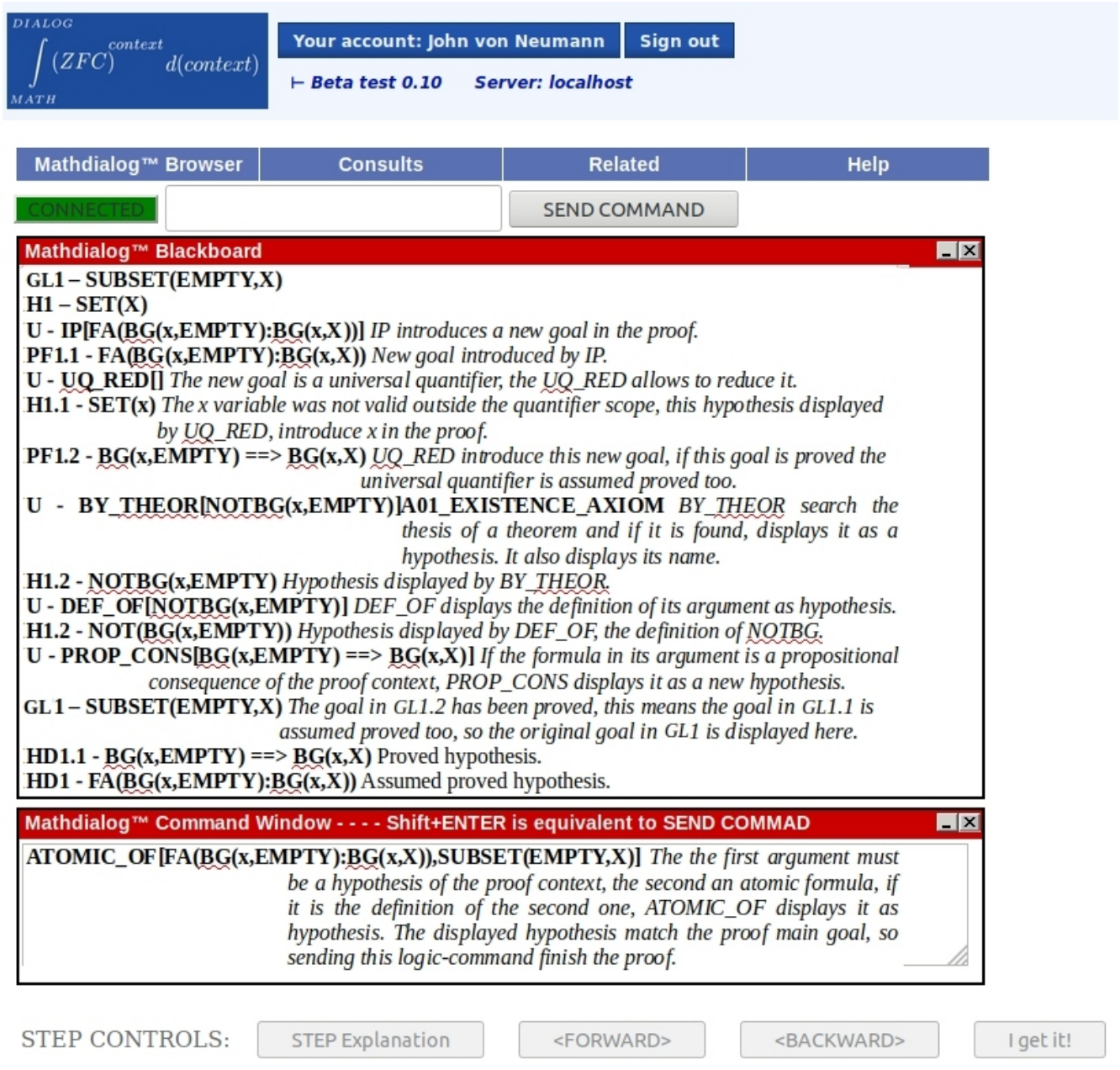

Figure 2. A commented finishing human-to-machine proof of THEOR[EMPTY_P01;SET(X);SUBSET(EMPTY,X)]

On textbook proofs, we frequently see the reduction of a formula A we want to prove as chain of implications, such that if A3 is proved, then this means that A2 can be assumed as proved, but if A2 is proved then A1 is assumed as proved and so is A. As example let be A the formula A1 ==> (A2 ==> A3). Here, we can assume A1 to prove A2 ==> A3, then under A1 as hypothesis, we can assume A2 to prove A3; if A3 is proved, then A2 ==> A3 is considered proved and consequently A. This kind of automatic inference chains is the meaning of digits greater than 1 in a DS. A segment of a proof with this kind of deduction chain would look as follow:

⋮
GL3 - F1
...
GL3.1 - A1 ==> (A2 ==> A3)
U - IF_RED[] *(The IF_RED logic-command by IF REDuction, reduce the implication in the closest goal, in our case the formula on GL3.1)*
GL3.2 - A2 ==> A3 *(new goal. If GL3.2 is proved, then GL3.1 is assumed as proved)*
H3.2 - A1 *(new hypothesis generated by IF_RED. It can be used to prove the formula in GL3.2)*
U - IF_RED[]
GL3.3 - A3 *(If GL3.3 is proved, then GL3.2 is assumed as proved)*
H3.3 - A2 *(new hypothesis generated by the second IF_RED)*
... (proof of A3)
U – Logic-command[A3] *(The last logic-command of the proof of A3, displays A3 as hypothesis)*
GL3 – F1 *(Original goal formula)*
HD3.2 - A3 *(Proved hypothesis of the goal in GL3.3 added to the GL3.2 context)*
HD3.1 - A2 ==> A3 *(Proved hypothesis of the goal in GL3.2 added to the GL3.1 context)*
HD3 - A1 ==> (A2 ==> A3) *(Proved hypothesis of the goal in GL3.1 added to the GL3 context)*
...(Now, the formula in HD3 can be used in the proof of F1. In other words, the formula in HD3 is in the proof context of the goal on GL3)
⋮

All formulas in the logic-commands' arguments must be valid respect to the proof context on the point where it is sent. The proof context is dynamic, on the last example, for instance, after the second occurrence of GL3, the hypotheses in H3.3, H3.2 are not in the proof context even though they are still displayed. Also, the ones in HD3.2 and HD3.1 are not in the proof context, however the one in HD3 is. The string ",1" (as in GL3,1 of the next example) is added when the new goal formula does not necessarily imply the old one, but the system knows what hypothesis and how many of them are needed in order to consider proved the old goal formula. In the next example, we will see this respect to the formula F1 <==> F2 in GL3. The standard way to prove this kind of formula is to prove first an implication and later the other.

⋮
GL3 - F1 <==> F2
U – IFF_RED_IF[] (By IFF REDuction, IF first)
GL3,1 - F1 ==> F2 (the system knows that after this formula has been proved, it is necessary to prove F2 ==> F1 to assume proved the formula in GL3)
...(proof of F1 ==> F2)
U - Logic-command[F1 ==> F2] (The last logic-command of the F1 ==> F2 proof generate the F1 ==> F2 formula)
GL4 - F2 ==> F1 (The system provides this new goal and knows that with its proof, the goal on GL3 F1< ==> F2, is proved and so, it can be assumed)
HD3 - F1 ==> F2 (Hypothesis proved on the first stage of the logic-command IFF_RED_IF)
⋮

On the following description of two logic-commands, we are going to pay less attention to the particular details of each one, and more on the general ideas about logic-commands implicit in each one. As we have seen, the hypotheses appear and disappear form the proof context as the proof develops.

Our next logic-command named CONTR_PROOF by CONTRadiction PROOF has two arguments, assuming that the goal formula is F, the first one must be a negation NF of it, and the second a

hypothesis H from the proof context that may facilitate finding the contradiction. The system responds by generating the new goal formula NF <==> NOT(F) with ",1" added to the old label. As we have already seen, this means that the system knows the task performed by this logic-command has not been finished yet and that it should continue with the following phase as soon as the formula in the new goal line has been proved. On the second phase, it assumes NF as hypothesis and generates another new goal formula H ==> BG(0,0) AND NOT(BG(0,0)) with 1 arithmetically added to old label (the original one). If the user proves it, the system considers the formula F as proved. The formula BG(0,0) AND NOT(BG(0,0)) is a falsehood, so if H ==> BG(0,0) AND NOT(BG(0,0)) is proved, there is a contradiction. 'BG(0,0)' or '0 ∈ 0' or 'zero is in zero' may sound weird, but it is a valid logical formula, in mathematics everything is a set, numbers (even π and e), vectors, matrices, functions are sets.

In CZFC, the informal syntax of a existential quantifier formula and of a uniqueness existential quantifier formula are respectively:

TE(SET(x)|BG(x,A):F) and TE!(SET(x)|BG(x,A):F)

where x is a variable, A is a term and F is a formula. SET(x)|BG(x,A) is named the **quantifier local context**, it can be SET(x) or BG(x,A).

If the present goal formula is a (uniqueness) existential quantifier formula QF, the logic-command EQ_RED[C] allows the user to prove QF. C is the candidate which we want to prove satisfy the requirement of QF and must be a term. If the quantifier local context is BG(x,A), the system put BG(C,A) as the new goal, if it is proved, the system displays BG(C,A) as a new hypothesis and displays as a new goal the formula F(C), if it is proved and the quantifier is not of uniqueness, QF is assumed as proved. If the quantifier is of uniqueness, the system displays as hypothesis F(NC), where NC is a new variable and displays EQ(NC,C) as the new goal.

There are about 25 logic-commands that perform a variety of tasks to translate, with relative comfort, textbooks proof of some mathematics fields.

A way to learn to make human-to-machine proof is consulting the ones already exist. If you are logged (before the first round, you always will in the Training Camp server), when you consult a machine-to-human proof in mathdialog.com, a window will appear with the human-to-machine proof, it is what you have to study together with the Logic commands Reference Guide. Then try to prove the theorems in the Appendix I and if you got stuck, see the included proofs and send one by one their logic-commands.

TIPS AND NOTES

01) The server connection (the red or green button) is independent from login, you can be logged and being in a server that may be connected or disconnected. Also you can be connected without login, in that case you are connected to a default account such as the Mathdialog Practice Area. There you can formulate definitions, theorems and execute proof but they will not accredited to you and will be deleted. The login in needed to accredit to you, your definitions and theorems.

02) When you are making a proof it is a good idea to write each logic-command elsewhere in case you stuck for more than 15 minutes and lost them. After 15 minutes of inactivity the system will logout you. You can write logic-commands in a editor and copy one there and paste it in the command window or even drag it.

03) You can prove the theorems in any order but sometimes will be.

04) An unfinished proof cannot be saved, users can interrupt a proof with the logic-command DROP_PROOF[] but it will not be saved, in fact, send it give the following message but it would be too late, your proof is lost:

> Mathdialog™ message: The command:
> DROP_PROOF
> Have been successfully executed. The proof has
> been dropped. At this time, there is no way to save
> an unfinished proof.
> Please write your proof in a editor and paste
> each Logic Command into the CommandWindow.

05) The Mathdialog Practice Area will be available to consult definitions, theorems and consult and make proofs. However, notice that those proofs will be deleted once a week.

06) Normally, when we need define a function f, we would do let f(x)=Term(x) were Term(x) is a term that depends of x such as 3+x, 1/x or x*3+5. In those cases our function would be f(x)=1+x, f(x)=1/x or f(x)=x*3+5 respectively. In those cases the set A where we are taking x and the set B where we are taken its image f(x), are given elsewhere. This is too ambiguous for Mathdialog, to use a term to define a function f use LET(f,FUN(x ∈ C:Term(x))) in the theorem local context. The domain of f is C and its range is REM(x ∈ C:Term(x)), so f is defined in a way that we must give its domain explicitly, in our case it is C. To find its range we must characterize REM(x ∈ C:Term(x)). For example if LET(f,FUN(x ∈ REANZ:1/(x))) (where REANZ is set of non-zero real numbers), then we already know REANZ is its domain by definition, but to prove REANZ is its range, then we must prove:

EQ(REANZ,REM(x ∈ REANZ:1/(x)))

Moreover, Mathdialog checks if LET(f,FUN(x ∈ C:Term(x))) is valid, for example if you write LET(f,FUN(x ∈ REALS:1/(x))), Mathdialog will complain.

07) A technical clarification. The NUCLEUS is the minimal set of theorems and definitions that makes Mathdialog consistent. Each one is show in the credit as “Mathdialog NUCLEUS”.

APPENDIX I

Next there are a few theorems with its proof, you can use them to create those proofs in the Mathdialog Training Camp by writing, cutting and pasting or dragging each logic command to the Command Window and SEND it to experience how it feels to make real human-to-machine proofs.

((*If A is a subset of B and B is a subset of A, then A and B are equals*)

```
THEOR[SUBSET_P04;SUBSET(A,B),SUBSET(B,A);EQ(A,B)]
BY_THEOR[FA(BG(z,A):BG(z,B) ) AND FA(BG(z,B):BG(z,A) ) <==> EQ(A,B)]
DEF_OF[SUBSET(A,B)]
DEF_OF[SUBSET(B,A)]
PROP_CONS[EQ(A,B)]
```

(*If R and T are relations form A to B and x is subset of A, then*
if R is subset of T then the image of x by R is subset of the image of x by T)

```
THEOR[IMAG_P01;RELAT(R,A,B),RELAT(T,A,B),SUBSET(x,A);
SUBSET(R,T) ==> SUBSET(IMAG(x,R,A,B),IMAG(x,T,A,B))]
IF_RED[]
IP[FA(BG(z,IMAG(x,R,A,B)):BG(z,IMAG(x,T,A,B)) )]
UQ_RED[]
IF_RED[]
BY_DEF_OBC[BG(z,IMAG(x,R,A,B))]
BY_DEF_OBC[BG(z,IMAG(x,T,A,B))]
PROP_CONS[BG(z,B) AND TE(BG(a,x):BG((a,z),R))]
PROP_CONS[TE(BG(a,x):BG((a,z),R))]
EQ_RED_H[TE(BG(a,x):BG((a,z),R)),a]
IP[TE(BG(s,x):BG((s,z),T))]
EQ_RED[a]
PROP_CONS[BG(a,x)]
BY_THEOR[BG((a,z),T);(a,z);]
PROP_CONS[BG(z,IMAG(x,T,A,B))]
ATOMIC_OF[FA(BG(z,IMAG(x,R,A,B)):BG(z,IMAG(x,T,A,B))),
SUBSET(IMAG(x,R,A,B),IMAG(x,T,A,B))]
```

(*If R is a relation in A and s is a subset of A, then the restriction of R to s is a subset of R*)

```
THEOR[RELAT_IN_P01;RELAT_IN(R,A),SUBSET(s,A);SUBSET(REST_IN(R,A,s),R)]
IP[FA(BG(x,REST_IN(R,A,s)):BG(x,R))]
UQ_RED[]
IF_RED[]
BY_DEF_OBC[BG(x,REST_IN(R,A,s))]
DEF_OF[RELAT_IN(R,A)]
PROP_CONS[TE(BG(a,s):TE(BG(y,s):EQ(x,(a,y)) AND BG((a,y),R)))]
EQ_RED_H[TE(BG(a,s):TE(BG(y,s):EQ(x,(a,y)) AND BG((a,y),R))),a]
PROP_CONS[TE(BG(y,s):EQ(x,(a,y)) AND BG((a,y),R))]
EQ_RED_H[TE(BG(y,s):EQ(x,(a,y)) AND BG((a,y),R)),y]
EQUAL_EQUIV[BG(x,R)]
ATOMIC_OF[FA(BG(x,REST_IN(R,A,s)):BG(x,R)),SUBSET(REST_IN(R,A,s),R)]
```

(*If R is a relation from A to B, and R is also relation from C to B, then the domain of R is a subset of C*)

```
THEOR[DOMAIN_P03;RELAT(R,A,B),RELAT(R,C,B);SUBSET(DOMAIN(R,A,B),C)]
DEF_OF[RELAT(R,C,B)]
DEF_OF[RELAT(R,A,B)]
IP[FA(BG(x,DOMAIN(R,A,B)):BG(x,C) )]
UQ_RED[]
IF_RED[]
BY_DEF_OBC[BG(x,DOMAIN(R,A,B))]
PROP_CONS[TE(BG(y,B):BG((x,y),R) )]
EQ_RED_H[TE(BG(y,B):BG((x,y),R) ),y]
BY_THEOR[BG((x,y),CART_PROD(C,B));(x,y),CART_PROD(C,B);]
BY_THEOR[BG((x,y),CART_PROD(C,B)) <==> BG(x,C) AND BG(y,B)]
PROP_CONS[BG(x,C)]
ATOMIC_OF[FA(BG(x,DOMAIN(R,A,B)):BG(x,C) ),SUBSET(DOMAIN(R,A,B),C)]
```

(*If R is a relation from A to B, then R is a relation from its domain to B*)

```
THEOR[DOMAIN_P02;RELAT(R,A,B);RELAT(R,DOMAIN(R,A,B),B)]
DEF_OF[RELAT(R,A,B)]
IP[SUBSET(R,CART_PROD(DOMAIN(R,A,B),B))]
IP[FA(BG(z,R):BG(z,CART_PROD(DOMAIN(R,A,B),B)) )]
UQ_RED[]
IF_RED[]
BY_THEOR[BG(z,CART_PROD(A,B));CART_PROD(A,B);]
BY_THEOR[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)) ))]
EQ_RED_H[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)) )),x]
PROP_CONS[TE(BG(y,B):EQ(z,(x,y)) )]
EQ_RED_H[TE(BG(y,B):EQ(z,(x,y)) ),y]
BY_DEF_OBC[BG(x,DOMAIN(R,A,B))]
IP[TE(BG(b,B):BG((x,b),R) )]
EQ_RED[y]
PROP_CONS[BG(y,B)]
EQUAL_EQUIV[BG((x,y),R)]
PROP_CONS[BG(x,DOMAIN(R,A,B))]
BY_THEOR[BG((x,y),CART_PROD(DOMAIN(R,A,B),B)) <==>
BG(x,DOMAIN(R,A,B)) AND BG(y,B);DOMAIN(R,A,B);]
PROP_CONS[BG((x,y),CART_PROD(DOMAIN(R,A,B),B))]
EQUAL_EQUIV[BG(z,CART_PROD(DOMAIN(R,A,B),B))]
ATOMIC_OF[FA(BG(z,R):BG(z,CART_PROD(DOMAIN(R,A,B),B)) ),
SUBSET(R,CART_PROD(DOMAIN(R,A,B),B))]
ATOMIC_OF[SUBSET(R,CART_PROD(DOMAIN(R,A,B),B)),
RELAT(R,DOMAIN(R,A,B),B)]
```

(*If R is a relation from A to B, and R is also a relation from K to B, then*
if for all set C, if R is a relation from C to B implies K is a subset of C, then K is the domain of R from A to B)

```
THEOR[DOMAIN_P04;RELAT(R,A,B),RELAT(R,K,B);
FA(SET(C):RELAT(R,C,B) ==> SUBSET(K,C) ) ==> EQ(K,DOMAIN(R,A,B))]
IF_RED[]
SUBST_UQV[FA(SET(C):RELAT(R,C,B) ==> SUBSET(K,C) ),DOMAIN(R,A,B)]
BY_THEOR[RELAT(R,DOMAIN(R,A,B),B)]
PROP_CONS[SUBSET(K,DOMAIN(R,A,B))]
IP[FA(BG(x,DOMAIN(R,A,B)):BG(x,K) )]
UQ_RED[]
IF_RED[]
DEF_OF[RELAT(R,K,B)]
BY_DEF_OBC[BG(x,DOMAIN(R,A,B))]
```

```
PROP_CONS[TE(BG(y,B):BG((x,y),R) )]
EQ_RED_H[TE(BG(y,B):BG((x,y),R) ),y]
BY_THEOR[BG((x,y),CART_PROD(K,B));(x,y),CART_PROD(K,B);]
BY_THEOR[BG((x,y),CART_PROD(K,B)) <==> BG(x,K) AND BG(y,B)]
PROP_CONS[BG(x,K)]
ATOMIC_OF[FA(BG(x,DOMAIN(R,A,B)):BG(x,K) ),SUBSET(DOMAIN(R,A,B),K)]
BY_THEOR[EQ(K,DOMAIN(R,A,B));DOMAIN(R,A,B);]
```

(*If R is a relation from A to B, then R is a relation from A to its domain*)

```
THEOR[RANGE_P02;RELAT(R,A,B);RELAT(R,A,RANGE(R,A,B))]
DEF_OF[RELAT(R,A,B)]
IP[SUBSET(R,CART_PROD(A,RANGE(R,A,B)))]
IP[FA(BG(z,R):BG(z,CART_PROD(A,RANGE(R,A,B))) )]
UQ_RED[]
IF_RED[]
BY_THEOR[BG(z,CART_PROD(A,B));CART_PROD(A,B);]
BY_THEOR[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)) ))]
EQ_RED_H[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)) )),x]
PROP_CONS[TE(BG(y,B):EQ(z,(x,y)) )]
EQ_RED_H[TE(BG(y,B):EQ(z,(x,y)) ),y]
BY_DEF_OBC[BG(y,RANGE(R,A,B))]
IP[TE(BG(b,A):BG((b,y),R) )]
EQ_RED[x]
PROP_CONS[BG(x,A)]
EQUAL_EQUIV[BG((x,y),R)]
PROP_CONS[BG(y,RANGE(R,A,B))]
BY_THEOR[BG((x,y),CART_PROD(A,RANGE(R,A,B))) <==>
BG(x,A) AND BG(y,RANGE(R,A,B));RANGE(R,A,B);]
PROP_CONS[BG((x,y),CART_PROD(A,RANGE(R,A,B)))]
EQUAL_EQUIV[BG(z,CART_PROD(A,RANGE(R,A,B)))]
ATOMIC_OF[FA(BG(z,R):BG(z,CART_PROD(A,RANGE(R,A,B))) ),
SUBSET(R,CART_PROD(A,RANGE(R,A,B)))]
ATOMIC_OF[SUBSET(R,CART_PROD(A,RANGE(R,A,B))),RELAT(R,A,RANGE(R,A,B))]
```

(*If R is a relation from A to B, and R is also relation from C to B, then the range of R is a subset of C*)

```
THEOR[RANGE_P03;RELAT(R,A,B),RELAT(R,A,C);SUBSET(RANGE(R,A,B),C)]
DEF_OF[RELAT(R,A,B)]
DEF_OF[RELAT(R,A,C)]
IP[FA(BG(y,RANGE(R,A,B)):BG(y,C) )]
UQ_RED[]
IF_RED[]
BY_DEF_OBC[BG(y,RANGE(R,A,B))]
PROP_CONS[TE(BG(x,A):BG((x,y),R) )]
EQ_RED_H[TE(BG(x,A):BG((x,y),R) ),x]
BY_THEOR[BG((x,y),CART_PROD(A,C));(x,y),CART_PROD(A,C);]
BY_THEOR[BG((x,y),CART_PROD(A,C)) <==> BG(x,A) AND BG(y,C)]
PROP_CONS[BG(y,C)]
ATOMIC_OF[FA(BG(y,RANGE(R,A,B)):BG(y,C) ),SUBSET(RANGE(R,A,B),C)]
```

(*If A is not empty, then A is a not empty subset of itself*)
```
THEOR[NOEMP_SUBSET_P02;NOEMP(A);
NOEMP_SUBSET(A,A)]
BY_THEOR[SUBSET(A,A)]
ATOMIC_OF[NOEMP(A),NOEMP_SUBSET(A,A)]
```

(*If A and B are a sets, then A is equal to B if and only if for all set x, x belongs to A if and only if x belongs to B*)
```
THEOR[EQUAL_P01;SET(A),SET(B);EQ(A,B) <==> FA(SET(x):BG(x,A) <==> BG(x,B))]
IFF_RED_IF[]
IF_RED[]
UQ_RED[]
PROP_CONS[BG(x,A) ==> BG(x,A)]
IFF_RED_IF[]
EQUAL_EQUIV[BG(x,A) ==> BG(x,B)]
EQUAL_EQUIV[BG(x,B) ==> BG(x,A)]
IF_RED[]
IP[FA(BG(x,A):BG(x,B) )]
UQ_RED[]
IF_RED[]
SUBST_UQV[FA(SET(a):BG(a,A) <==> BG(a,B) ),x]
PROP_CONS[BG(x,B)]
IP[FA(BG(x,B):BG(x,A) )]
UQ_RED[]
IF_RED[]
SUBST_UQV[FA(SET(a):BG(a,A) <==> BG(a,B) ),x]
PROP_CONS[BG(x,A)]
BY_THEOR[FA(BG(z,A):BG(z,B) ) AND FA(BG(z,B):BG(z,A) ) <==> EQ(A,B)]
PROP_CONS[EQ(A,B)]
```

(*Try to translate this theorem into natural language, that is the suggested first step for each theorem.*)
```
THEOR[RELAT_P02;RELAT(R,A,B),BG(z,R);
TE!(BG(x,DOMAIN(R,A,B)):TE!(BG(y,RANGE(R,A,B)):EQ(z,(x,y)) ))]
DEF_OF[RELAT(R,A,B)]
BY_THEOR[BG(z,CART_PROD(A,B))]
BY_THEOR[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)) ))]
EQ_RED_H[TE(BG(x,A):TE(BG(y,B):EQ(z,(x,y)))),x]
PROP_CONS[TE(BG(y,B):EQ(z,(x,y)))]
EQ_RED_H[TE(BG(y,B):EQ(z,(x,y))),y]
BY_DEF_OBC[BG(y,RANGE(R,A,B))]
IP[TE(BG(b,A):BG((b,y),R) )]
EQ_RED[x]
PROP_CONS[BG(x,A)]
EQUAL_EQUIV[BG((x,y),R)]
PROP_CONS[BG(y,RANGE(R,A,B))]
EQ_RED[x]
BY_DEF_OBC[BG(x,DOMAIN(R,A,B))]
IP[TE(BG(b,B):BG((x,b),R) )]
EQ_RED[y]
PROP_CONS[BG(y,B)]
EQUAL_EQUIV[BG((x,y),R)]
PROP_CONS[BG(x,DOMAIN(R,A,B))]
EQ_RED[y]
PROP_CONS[BG(y,RANGE(R,A,B))]
PROP_CONS[EQ(z,(x,y))]
EQUAL_EQUIV[EQ((x,a),(x,y))]
```

```
BY_THEOR[EQ((x,a),(x,y)) <==> EQ(x,x) AND EQ(a,y)]
PROP_CONS[EQ(a,y)]
EQUAL_EQUIV[EQ(y,a)]
PROP_CONS[TE!(BG(k,RANGE(R,A,B)):EQ(z,(a,k)))]
EQ_RED_H[TE!(BG(k,RANGE(R,A,B)):EQ(z,(a,k))),s]
PROP_CONS[FA(BG(k,RANGE(R,A,B)):EQ(z,(a,k)) ==> EQ(s,k) )]
SUBST_UQV[FA(BG(k,RANGE(R,A,B)):EQ(z,(a,k)) ==> EQ(s,k) ),y]
PROP_CONS[BG(y,RANGE(R,A,B))]
PROP_CONS[EQ(z,(x,y))]
EQUAL_EQUIV[EQ((a,s),(x,y))]
BY_THEOR[EQ((a,s),(x,y)) <==> EQ(a,x) AND EQ(s,y)]
PROP_CONS[EQ(a,x)]
EQUAL_EQUIV[EQ(x,a)]
```

(*Try to translate this theorem into natural language, that is the suggested first step for each theorem.*)

```
THEOR[RELAT_P03;RELAT(R,A,B),BG(z,R);
TE!(BG(y,RANGE(R,A,B)):TE!(BG(x,DOMAIN(R,A,B)):EQ(z,(x,y)) ))]
BY_THEOR[TE!(BG(x,DOMAIN(R,A,B)):TE!(BG(y,RANGE(R,A,B)):EQ(z,(x,y)) ))]
EQ_RED_H[TE!(BG(s,DOMAIN(R,A,B)):TE!(BG(y,RANGE(R,A,B)):EQ(z,(s,y)) )),x]
PROP_CONS[TE!(BG(s,RANGE(R,A,B)):EQ(z,(x,s)) )]
EQ_RED_H[TE!(BG(s,RANGE(R,A,B)):EQ(z,(x,s)) ),y]
EQ_RED[y]
PROP_CONS[BG(y,RANGE(R,A,B))]
EQ_RED[x]
PROP_CONS[BG(x,DOMAIN(R,A,B))]
PROP_CONS[EQ(z,(x,y))]
EQUAL_EQUIV[EQ((a,y),(x,y))]
BY_THEOR[EQ((a,y),(x,y)) <==> EQ(a,x) AND EQ(y,y)]
PROP_CONS[EQ(a,x)]
EQUAL_EQUIV[EQ(x,a)]
PROP_CONS[TE!(BG(s,DOMAIN(R,A,B)):EQ(z,(s,a)) )]
EQ_RED_H[TE!(BG(s,DOMAIN(R,A,B)):EQ(z,(s,a)) ),k]
EQUAL_EQUIV[EQ((x,y),(k,a))]
BY_THEOR[EQ((x,y),(k,a)) <==> EQ(x,k) AND EQ(y,a)]
PROP_CONS[EQ(y,a)]
```

Incentives:
Importance: foundational and philosophical
Importance: educational (machine-to-human proof)
Importance: prestige (your name in each theorem you translate)
Importance: fun and interesting
Importance: pioneering in a new form of mathematics

Explain the machine-to-human proof as a incentive using the proof of EMPTY_P01 and the name in the theorem.

This formal proof was achieved by sending only six logic-commands. To be precise, the CZFC formal proof of EMPTY_P01 is

IP[FA(BG(x,EMPTY):BG(x,X))], UQ_RED[], BY_THEOR[NOTBG(x,EMPTY)],
DEF_OF[NOTBG(x,EMPTY)],
PROP_CONS[BG(x,EMPTY) ==> BG(x,X)], ATOMIC_OF[FA(BG(x,EMPTY):BG(x,X)),
SUBSET(EMPTY,X)]

EXAMPLE TAKEN FROM proofwiki.com

Let n be an integer such that $n>1$, then n can be expressed as the product of one or more primes.

01) Let n be an integer such that n>1, then exist a no empty set of primes P such that n = ProdP(P)
02) For all n>1, exist a no empty set of primes P such that n = ProdP(P)

Proof
Aiming for a contradiction, suppose this supposition is false.
03) There exists k>1, for all no empty set of primes Pn k != ProdP(Pn).

Let m
be the smallest integer which can not be expressed as the product of primes.
04) Theorem: Let be n in Z, if S is a no empty set of integer such that for all x in S k>n, then there exists m in S such that for all x in S x >= m.
05) Let be X = {x in Z: for all no empty set of primes Pn x != ProdP(Pn)}
06) By 03 X is not empty so by 04 m exists

As a prime number is trivially a product of primes, m can not itself be prime.
07) It is no so trivial, we must include in the ProdP definition is prime or is product of primes.
If 1 were prime, a prime P would be product of primes P = 1xP but 1 is not prime. May be argued the the enunciate state “the product of one or more primes.” but “ the product of one prime” is too vague or even a nonsense. This is the kind of glitch translating to Mathdialog helps to spot being interesting instead of painful.
08) Proof by contradiction m can not be prime.

Hence:
$$\exists r,s \in \mathbb{Z} : 1<r<m, 1<s<m : m=rs$$

As m is our least counterexample, both r and s can be expressed as the product of primes.
09)

Say $r=p_1 p_2 \cdots p_k$ and $s=q_1 q_2 \cdots q_l$, where all of $p_1,\ldots,p_k,q_1,\ldots,q_l$

are prime.

Hence $m=rs=p_1 p_2 \cdots p_k q_1 q_2 \cdots q_l$, which is a product of primes.

Hence there is no such counterexample.

VERY IMPORTANT: Explain how LET(G,FUN(BG(x,A):F(x))) works