AXolotl: get to the **AX**ioms
A Manual

David M. Cerna

March 11, 2019

Abstract

In this document we outline how to play our preliminary version of **AX**olotl. We present a sequence of graphics illustrating the step by step process of playing the game.

The idea behind this game is to capture the following two proof situation:

$$\dfrac{\dfrac{\dfrac{P(t_i\sigma) \vdash P(t') \qquad P(r), \Delta \vdash P(s_i\sigma)}{P(r), P(s_i\sigma) \to P(t_i\sigma) \vdash P(t')} \to : l}{P(r), \Delta, \forall x_1, \cdots x_n (P(s_i) \to P(t_i)) \vdash P(t')} \forall : l}{P(r), \Delta \vdash P(t')}$$

$$\dfrac{\dfrac{\dfrac{P(t_i\theta), \Delta \vdash P(t') \qquad P(r) \vdash P(s_i\theta)}{P(r), P(s_i\theta) \to P(t_i\theta) \vdash P(t')} \to : l}{P(r), \Delta, \forall x_1, \cdots x_n (P(s_i) \to P(t_i)) \vdash P(t')} \forall : l}{P(r), \Delta \vdash P(t')}$$

where σ is a unifier of t_i and t', θ is a unifier of r and s_i, and Δ is a set of m rules of the form $\forall x_1, \cdots x_n (P(s_i) \to P(t_i))$ for $1 \leq i \leq m$. Note that one of the two branches is always axiomatic and thus can be ignored. If both are axiomatic the proof is complete. Essentially, all proofs of this form are linear rather then trees. Furthermore, we limit ourselves to a single predicate representing truth. This highlights the major difference between first-order and propositional logic, i.e. unification. This weak formalism is enough to encode full propositional logic as well as other theories of interest. Future work plans include relaxing some of these restrictions, especially the limitation to this particular proof situation. Below we document how to use the software.

1 For Instructors: .ax files

AXolotl Takes input from .ax files which store problems, the rules needed to solve the problems, the necessary variables and term language. The files have the following simple format.

```
Function: N 2
Function: a 0
Function: b 0
Function: c 0
Function: d 0
Variable: x
Variable: y
Variable: z
Problem: N(a,N(b,c)) N(c,N(b,a))
Problem: N(a,N(b,N(c,d))) N(d,N(c,N(b,a)))
Rule: N(z,N(x,y)) N(z,N(y,x))
Rule: N(N(x,y),z) N(N(y,x),z)
Rule: N(x,N(y,z)) N(N(x,y),z)
Rule: N(N(x,y),z)  N(x,N(y,z))
```

Note that Functions and variables must always occur before rules and problems. Concerning function symbols, the first position is the function symbol and the second is its arity. all symbols are case sensitive. Concerning problems and rules, the first position is the antecedent and the second is the succedent.

2 Source Code for Axolotl

Up to date source code is available at the following SVN repository:

svn checkout svn://svn.risc.jku.at/dcerna/AXolotl

Any question concerning the code may be directed at the author, David M. Cerna, who may be reached at one of the following E-mail addresses

dcerna@risc.jku.at
dcerna@jku.at
cernadavid1@gmail.com

Looking forward to any questions comments or suggestions concerning AXolotl. If you would like to test this system in a real classroom setting please let me know the details being that we currently need empirical data concerning the education benefits of the system.

3 An Introduction to Axolotl

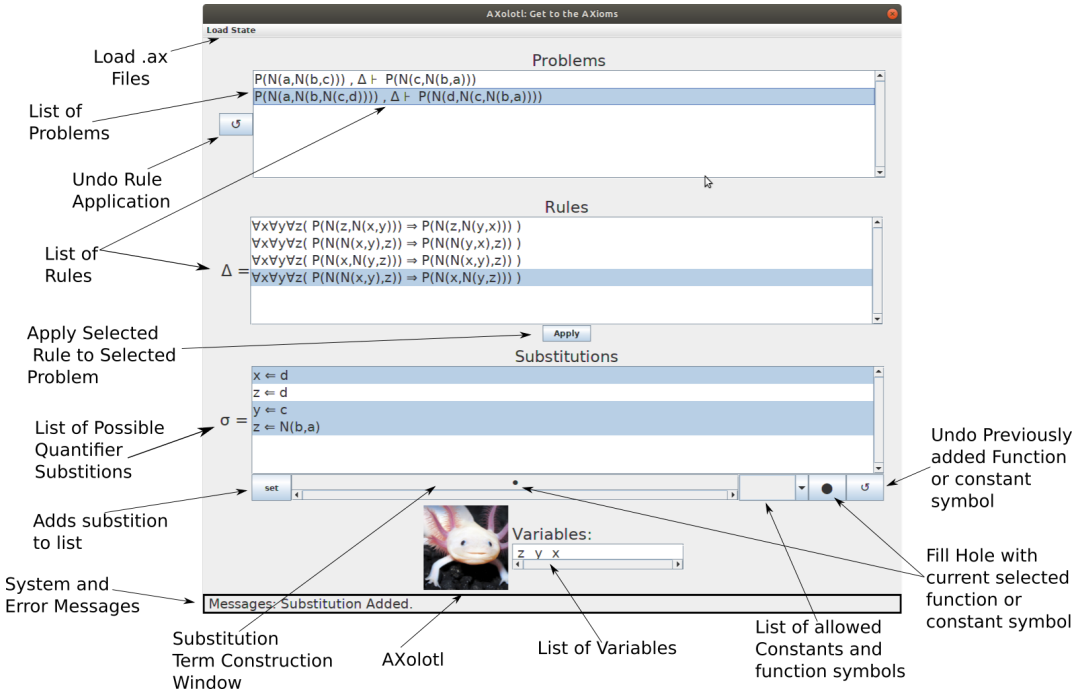

Figure 1: In this image we outline all of the components making up the Axolotl Dispplay. Through this tutorial we discuss the various features of the program and how one goes about solving the logical puzzles.

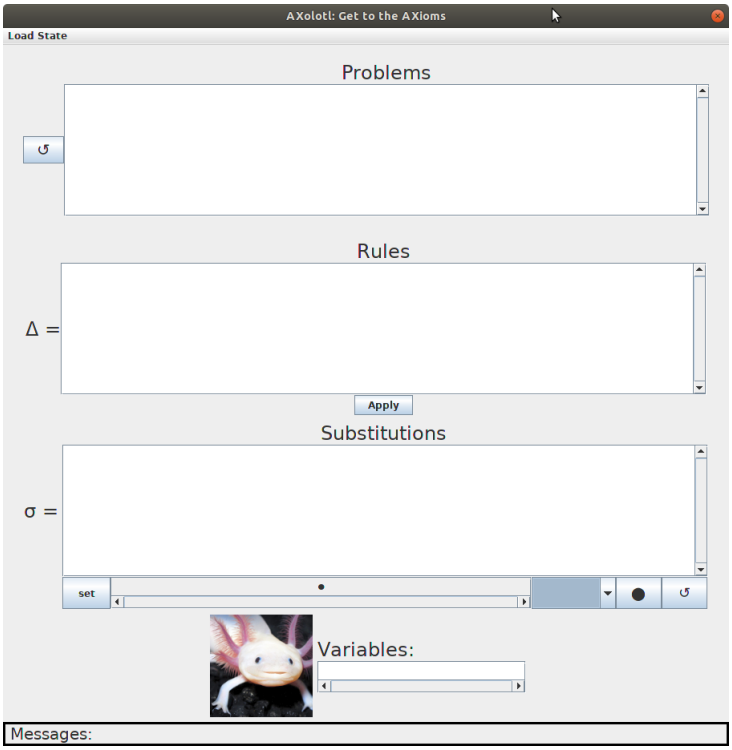

Figure 2: When AXolotl is first opened all displays are blank. One must load an AXolotl File.

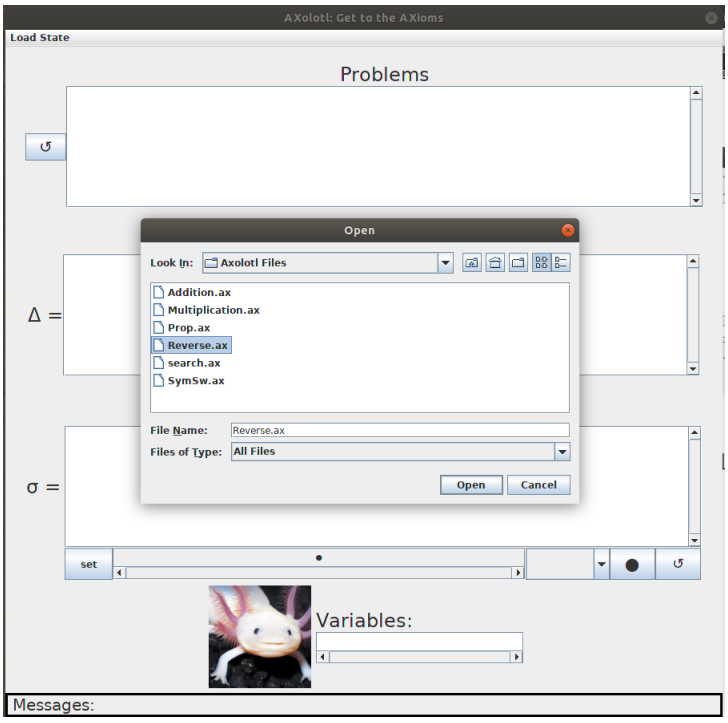

Figure 3: To open an .ax file double click on Load State and select a file.

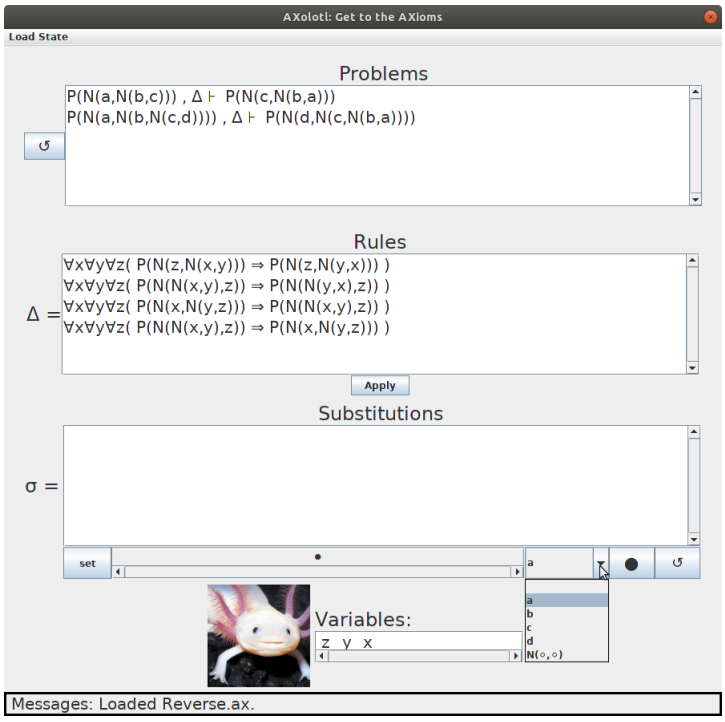

Figure 4: After loading reverse.ax both the problems and rules display contain information as well as the function and constant symbol list.

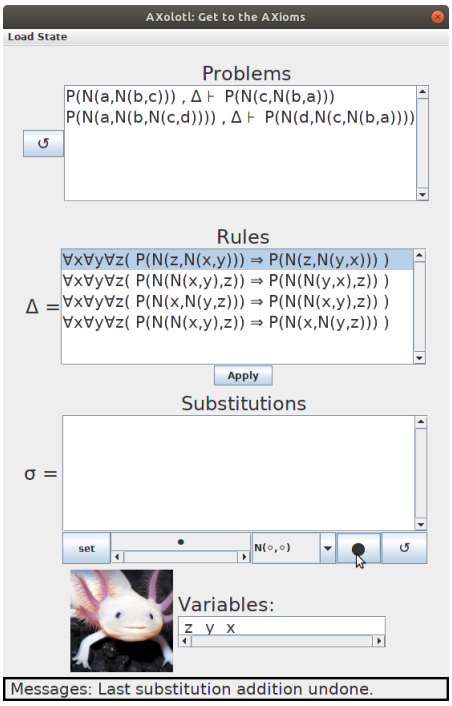

Figure 5: To add a symbol to substitution display one must first choose a symbol from the symbol list and then either hit enter or press the black circle button.

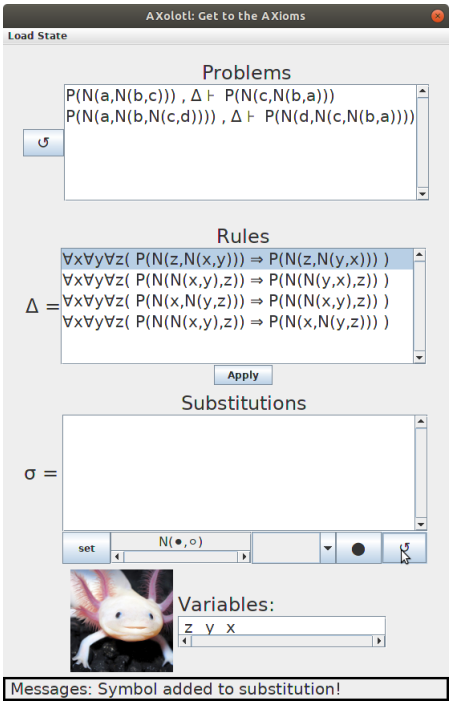

Figure 6: One will see the symbol within the substitution display. Notice that the symbol contains a black dot and a white dot. The black dot is the next position to be filled and the white dot is a position which will eventually be filled. The mouse point is over the undo point, pressing the undo button or pressing ctrl-z while the undo list is selected will undo the previous symbol addition.

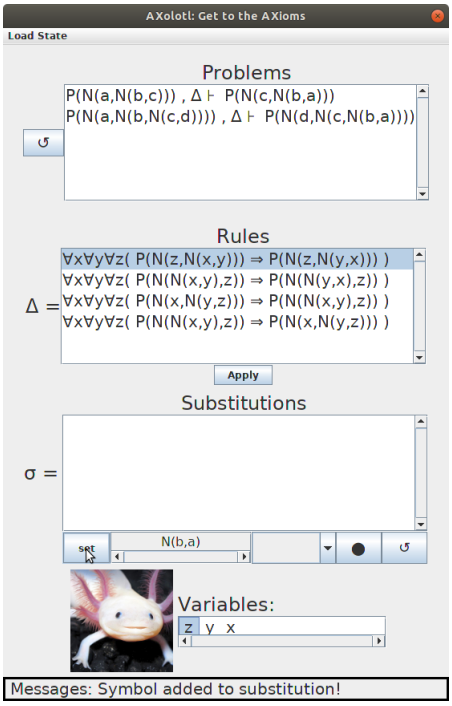

Figure 7: Once all white and black dots are filled in in the substitution display once ought to select a variable which will be replaced by the term displayed in the substitution display. In this case we select z. The next step is to either press the select button or hit enter twice after selecting the variable.

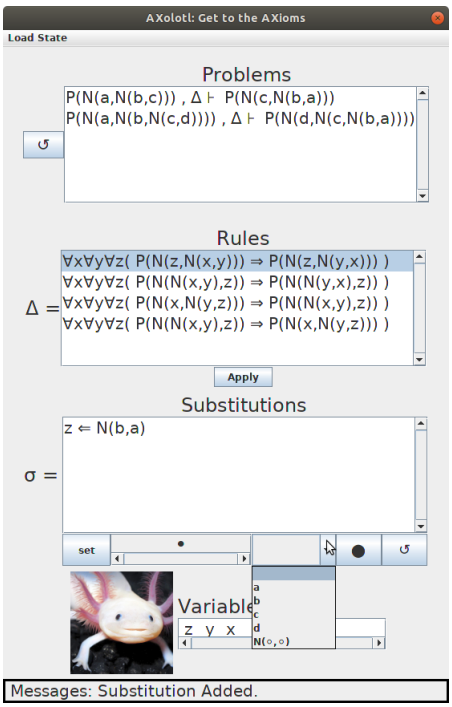

Figure 8: Notice that the substitution display now contains a substitution for the variable z. We need to construct substitutions for the other variables as well to apply one of the rules to one of the problems.

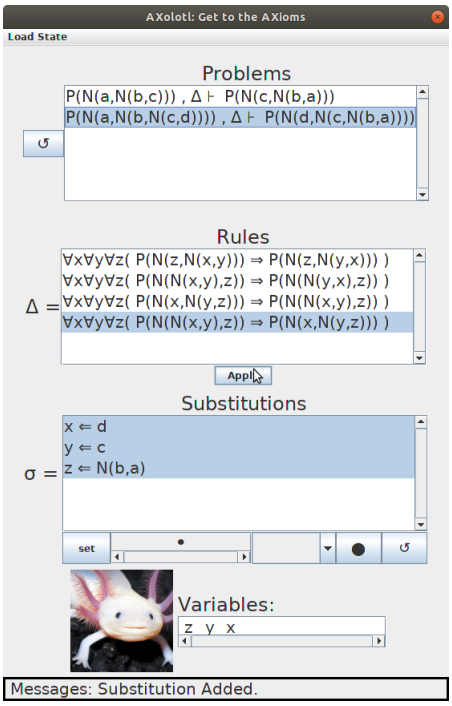

Figure 9: Once we have all the needed substitutions we may select a rules from the rule list, all the substitutions from the substitution list and one of the problems from the problem list and hit apply.

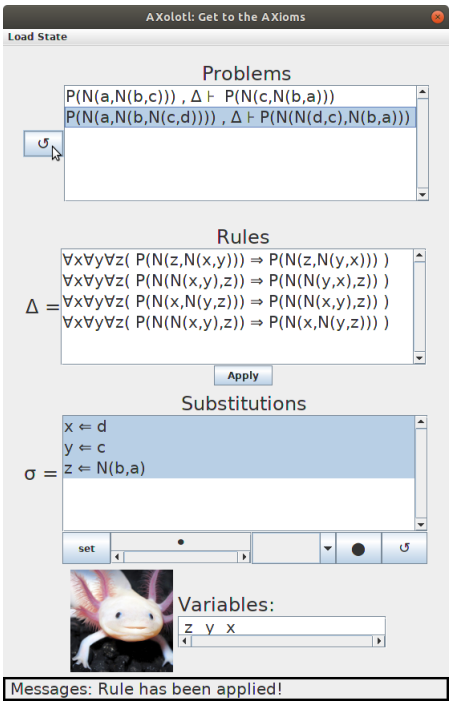

Figure 10: If one is not happy with the application of the rule to the chosen problem than one may hit the undo button after selecting the particular problem you want to undo.

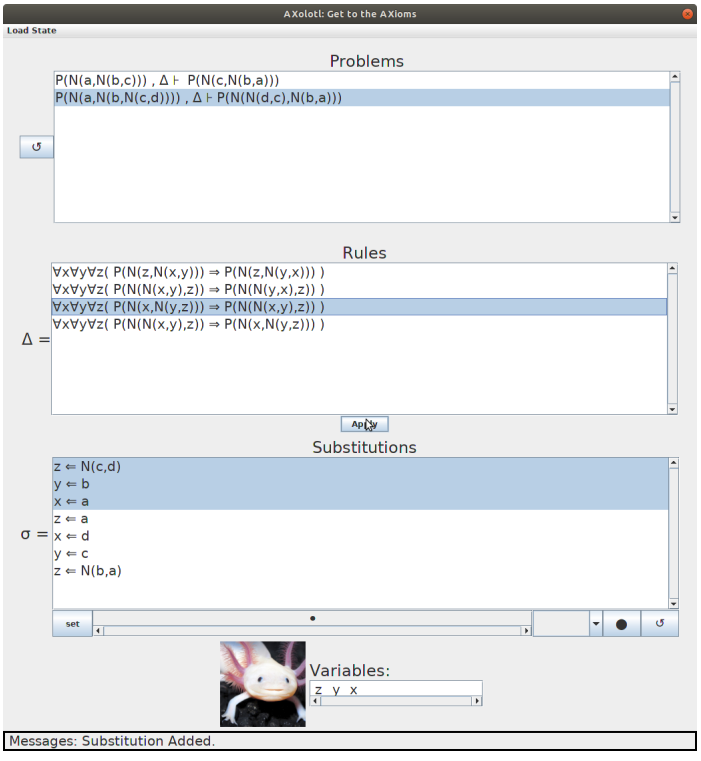

Figure 11: Rules can be used to transform either the antecedent or the succedent of the problem. Here the rule application matches the antecedent of the rule with the antecedent of the problem.

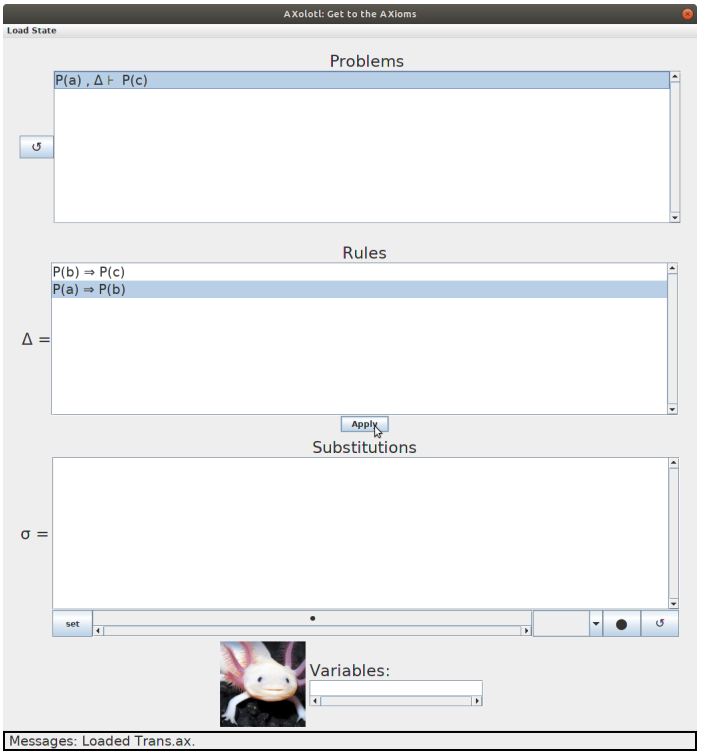

Figure 12: Rules may also be quantifier free, in which case one does not need to define substitutions to apply the rules.

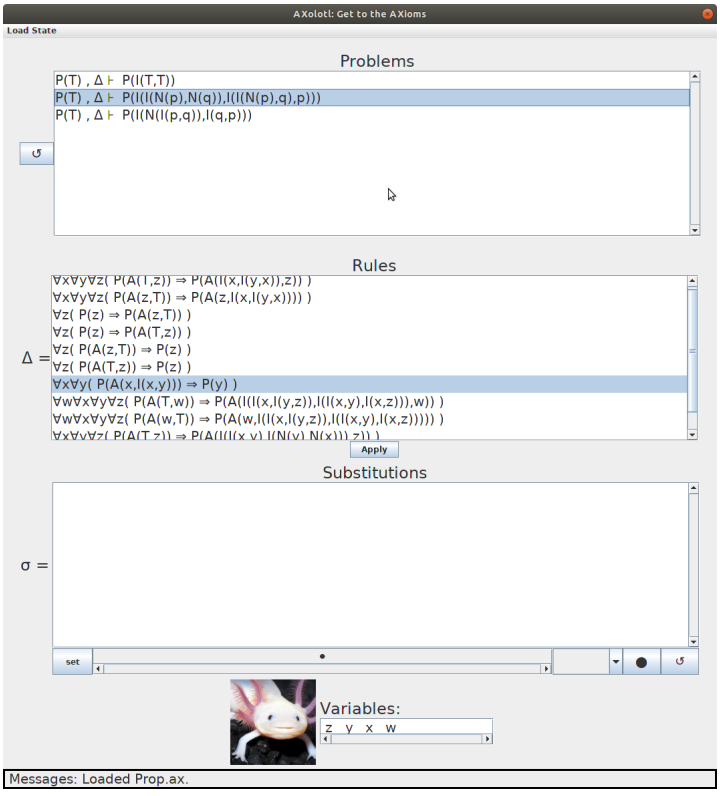

Figure 13: This formalism is quiet expressive. Here we provide a encoding of propositional logic together with derivations trees. The selected rule is an encoding of Modus Ponens. Below we provide the .ax file input.

Propositional.ax

```
Function: A 2
Function: I 2
Function: N 1
Function: p 0
Function: q 0
Function: T 0
Variable: x
Variable: y
Variable: z
Variable: w
Problem: T I(T,T)
Problem: T I(I(N(p),N(q)),I(I(N(p),q),p))
Problem: T I(N(I(p,q)),I(q,p))
Rule: A(T,z) A(I(x,I(y,x)),z)
Rule: A(z,T) A(z,I(x,I(y,x)))
Rule: z A(z,T)
Rule: z A(T,z)
Rule: A(z,T) z
Rule: A(T,z) z
Rule: A(x,I(x,y)) y
Rule: A(T,w) A(I(I(x,I(y,z)),I(I(x,y),I(x,z))),w)
Rule: A(w,T) A(w,I(I(x,I(y,z)),I(I(x,y),I(x,z))))
Rule: A(T,z) A(I(I(x,y),I(N(y),N(x))),z)
Rule: A(z,T) A(z,I(I(x,y),I(N(y),N(x))))
```