LOgic Semantics Tutor
(LOST)

User Guide v1.0

June 2007

Table of contents

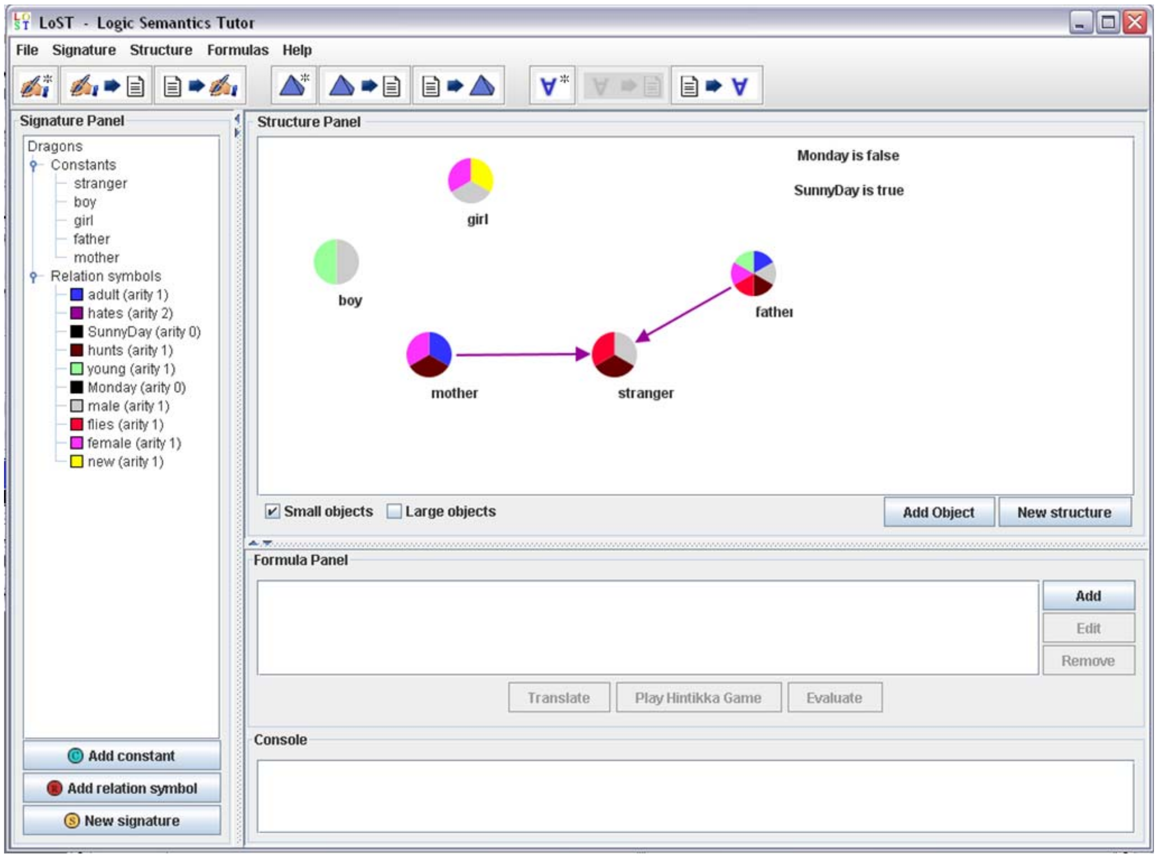
1. Introduction

Welcome! The LOgic Semantics Tutor (LOST) has been developed as an educational tool to assist students who are learning first-order logic. We assume that you are familiar with the syntax and semantics of first-order logic, namely notions such as that of a signature, a structure, and a formula. In this document we will explain the different features of the software and how to use them. For more information you can always refer to the report of this project.

Using this software you will have the opportunity to create and edit signatures and structures with total control via the user interface and 100% graphical representation. Once a structure is loaded you can write logic sentences which you can then evaluate on the structure, or play the Hintikka game (as seen in the *140 Logic* course). If you are interested you can also obtain literal translations of logic sentences in English.

2. The User Interface

This is interface of LOST with a structure and its signature loaded.



Monday is false

SunnyDay is true

The Signature Panel in the left-hand side will be used to display signatures. These can either be created via the interface or loaded from files.

The Structure Panel will display structures which can be edited using the mouse and the signature tree in the Signature Panel.

Once you have loaded (or created) a structure, you can write first-order sentences in the Formula Panel. Once a new sentence has been created, you can evaluate it on the structure. The Console is used to display comments and results.

You can also obtain a literal translation of a logic sentence in English, by providing a dictionary. We will see all this in detail shortly. Finally, the most interesting feature perhaps in the Hintikka game that is played using the formation tree of a sentence and a structure.

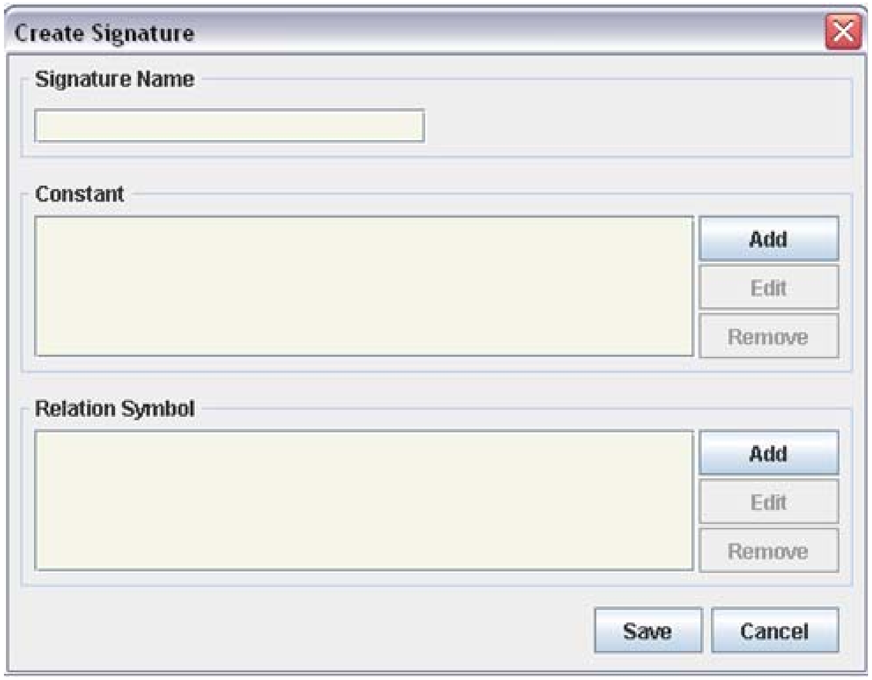
3. Working with signatures

Only one signature can be loaded in the interface at any time, to keep things simple.

Creating a signature

To create a signature, go to the menu and select **Signature > New Signature**. Alternatively, click on the menu bar button or on the button "New Signature" in the bottom of the Signature Panel.

You will be presented with the following window:



In this window you can add constants and relation symbols, as well as specify the name for the new signature. Once you have created the new signature, click on **Save** to load it in the main window.

Saving a signature

You can save the signature that you are working with in a file. It will be stored in an XML format, as follows:

```
<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE signature SYSTEM "signature.dtd">

<signature name="People">

   <constants>
      <constant name="mary"></constant>
      <constant name="marc"></constant>
      <constant name="jane"></constant>
      <constant name="clara"></constant>
   </constants>

   <relation_symbols>
      <relation_symbol name="human" arity="1" R="204" G="0" B="204">
            </relation_symbol>
      <relation_symbol name="likes" arity="2" R="0" G="0" B="0">
            </relation_symbol>
      <relation_symbol name="hates" arity="2" R="0" G="0" B="0">
            </relation_symbol>
      <relation_symbol name="young" arity="1" R="102" G="255" B="102">
            </relation_symbol>
      <relation_symbol name="tall" arity="1" R="255" G="255" B="0">
            </relation_symbol>
   </relation_symbols>

</signature>
```

The section “constants” contains the constants of the signature, with an entry for each. The second section, “relation_symbols” contains en entry for each relation symbol.

The RGB values that appear as attributes in the relation symbols are not part of a signature in theory. The colour (as we will see later) is used to represent the relation symbol structures. For this reason it needs to be stored along with the relation symbol.

The “signature.dtd” file that is mentioned in the second line is a Document Type Definition file that defines the correct structure of a signature XML file. Ensure that any signature files you write by hand are stored in the same folder as this DTD file, as it will be required to successfully load the signature in LOST.

To save a signature go to the menu and select **Signature** > **Save signature**. Alternatively, click on the menu bar button . Choose a file name and a directory (ensure that the directory where you save the signature contains the **signature.dtd** file). Click **Save**.

Loading a signature

To load a signature, go to the menu and select **Signature** > **Load signature**. Alternatively, click on the menu bar button . Choose the file that you want to load and press **Open**.

Closing a signature

To remove a signature from the interface, go to the menu and select **Signature** > **Close signature**.

Modifying a signature

Once a signature has been successfully loaded, you will see a tree (representing the signature) in the Signature Panel. There are a number of ways in which a signature can be modified.

Renaming a signature

To rename a signature, right click on its name in the signature tree (this is the root element of the tree) and select “Rename signature”.

Adding a constant

To add a constant to the signature, either click on the button

in the bottom of the Signature Panel, or right click on the signature tree element called “Constants” and choose “Add constant”.

Renaming a constant

To rename a constant right click on its name in the signature tree and select “Rename”.

Deleting a constant

To delete a constant right click on the name on its name in the signature tree and select “Delete”.

Adding a relation symbol

To add a relation symbol to the signature, either click on the button

in the bottom of the Signature Panel, or right click on the signature tree element called “Relation Symbols” and choose “Add relation symbol”.

Renaming a relation symbol

To rename a relation symbol right click on its name in the signature tree and select "Rename".

Changing the colour of a relation symbol

To change to colour of a relation symbol, right click on its name and select "Change colour". Any colour changes you make will be reflected in the structure that is loaded, if there is one.

Changing the arity of a relation symbol

To change the arity of a relation symbol, right click on its name, go over "Change arity" with the mouse and select a new arity for the relation symbol. If there is a structure loaded in LOST, the relation symbol will be removed from it.

Deleting a relation symbol

To delete a relation symbol from the signature, right click on its name in the signature tree and select "Delete".

4. Working with structures

Creating a structure

In order to be able to create a structure, there needs to be a signature present. Structures are defined based on a signature.

Once a signature has been loaded, you can go to menu and select **Structure > New structure**. Alternatively, click on the menu bar button .

A new structure will be created that will contain one object, since structures are defined as a non-empty set of objects. This will by default have all the constants of the signature mapped to it.

Saving a structure

You can save the structure that you are working with in a file. It will be stored in an XML format, as follows:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE structure SYSTEM "structure.dtd">

<structure name="null">

   <signature name="Dragons">

      <constants>
            <constant name="boy"></constant>
            <constant name="girl"></constant>
      </constants>

      <relation_symbols>
            <relation_symbol name="playsWith" arity="2" R="255" G="204"
                  B="0"></relation_symbol>
            <relation_symbol name="Monday" arity="0" R="0" G="0"
                  G="204" B="204"></relation_symbol>
            <relation_symbol name="female" arity="1" R="255" G="51"
                  B="255"></relation_symbol>
      </relation_symbols>

   </signature>

   <objects>
      <object id="1">
            <position x="285.0" y="165.0"></position>
            <labels>
                  <label name="boy"></label>
            </labels>
            <unary_predicates>
            </unary_predicates>
      </object>

      <object id="2">
            <position x="355.0" y="247.0"></position>
            <labels>
                  <label name="girl"></label>
            </labels>g=
            <unary_predicates>
                  <unary_predicate name="female"></unary_predicate>
            </unary_predicates>
      </object>
   </objects>

   <binary_predicates>
      <binary_predicate name="playsWith" sourceObjectID="1"
            targetObjectID="2"></binary_predicate>
   </binary_predicates>

   <nullary_predicates>
      <nullary_predicate name="Monday" value="true">
            </nullary_predicate>
      </nullary_predicates>

</structure>
```

When a structure is saved, its signature is saved in the same file. This way, when a structure is loaded in LOST its signature can be loaded at the same time. The “structure” element of the XML file contains three elements: the signature, the objects, the binary predicates and the nullary predicates. A signature will be stored in the same way that signature files are structured. The “objects” section will contain an entry for each structure object, each holding the following information:

- Id : each structure object is given a unique id, used for internal operations
- Position: the position that the object has in the structure graph. This, like the ‘id’ attribute, is not part of a structure conceptually. It is information needed when drawing structures in LOST.
- Labels : each label entry represents a constant mapped to the structure object
- Unary Predicates: one entry for each unary predicate that holds for the structure object.

The “binary predicates” section contains an entry for each occurrence of a binary relation symbol that holds in the structure. Each such entry contains the id of the two objects that are the relation symbol’s arguments.

Finally, the “nullary predicates” section contains an entry for each relation symbol of arity 0 that appears in the signature. The value of a nullary predicate can either be “true” or “false” in the structure.

To save a structure go to the menu and select **Structure > Save structure**. Alternatively, click on the menu bar button . Choose a file name and a directory (ensure that the directory where you save the structure contains the **structure.dtd** file that will be needed to load it in the future). Click **Save**.

Loading a structure

To load a structure, go to the menu and select **Structure > Load structure**. Alternatively, click on the menu bar button . Choose the file that you want to load and press **Open**.

Closing a structure

To remove a structure from the main window, go to the menu and select **Structure > Close structure**.

Modifying a structure

Adding an object to the structure

To add an object to the structure, click on the “Add Object” button which is located in the bottom of the Structure Panel. This will add a new object to the top left corner of the structure graph.

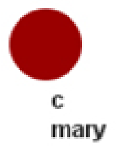
Deleting an object from the structure

To delete an object from the structure, right click on it and select “Delete object”. This action will not be permitted if the object in question is the only object in the structure. Remember that a structure is defined as a *non-empty* set of objects.

Mapping a constant to a structure object

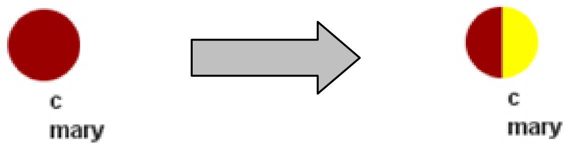
To map a signature constant to an object, simply click on the constant name in the signature tree (located in the Signature Panel) and then left click the object in the structure. If the constant was previously mapped to another object of the structure, it will be *removed* from that object and added to the new one. Remember that a constant must be mapped to one object in the structure at any time.

The names of the constants mapped to an object will be written below the object, one under the other:

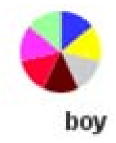


Adding a unary relation symbol to an object

To add a relation symbol to an object, click on the name of the relation symbol in the signature tree (located in the Signature Panel) and then left click on the object in the structure. If we add for example a relation symbol of yellow colour to an object, this is the result:



There is no restriction to the number of relation symbols that an object can have. The circle will be split in as many parts as necessary each time and drawn accordingly, as we can see in the picture on the right.



Removing a unary relation symbol from an object

To remove a unary relation symbol from an object, right click the object and select “Remove x” where x is the name of the relation symbol.

Adding a binary relation to the structure

To add a binary relation symbol to the structure, click on the name of the relation symbol in the signature tree (located in the Signature Panel). Then, left-click once the first object and then left-click the second object that represent the arguments of the relation symbol. An arrow (coloured in the colour of the relation symbol in the signature tree) will be drawn between the two objects.

Removing a binary relation symbol from the structure

You can remove a binary relation symbol from a structure by right clicking on the arrow that represents the relation symbol and selecting “remove x”, where x is the name of the relation symbol that the arrow represents.

Changing the value of a nullary relation symbol

Relation symbols of arity 0 are represented as text in the structure graph. Their truth value will either be true or false in the structure. You can change the truth value of nullary relation symbols in a structure by right clicking on their name and selecting “make true” or “make false”, depending on the current truth value of the relation symbol.

Changing the size of objects

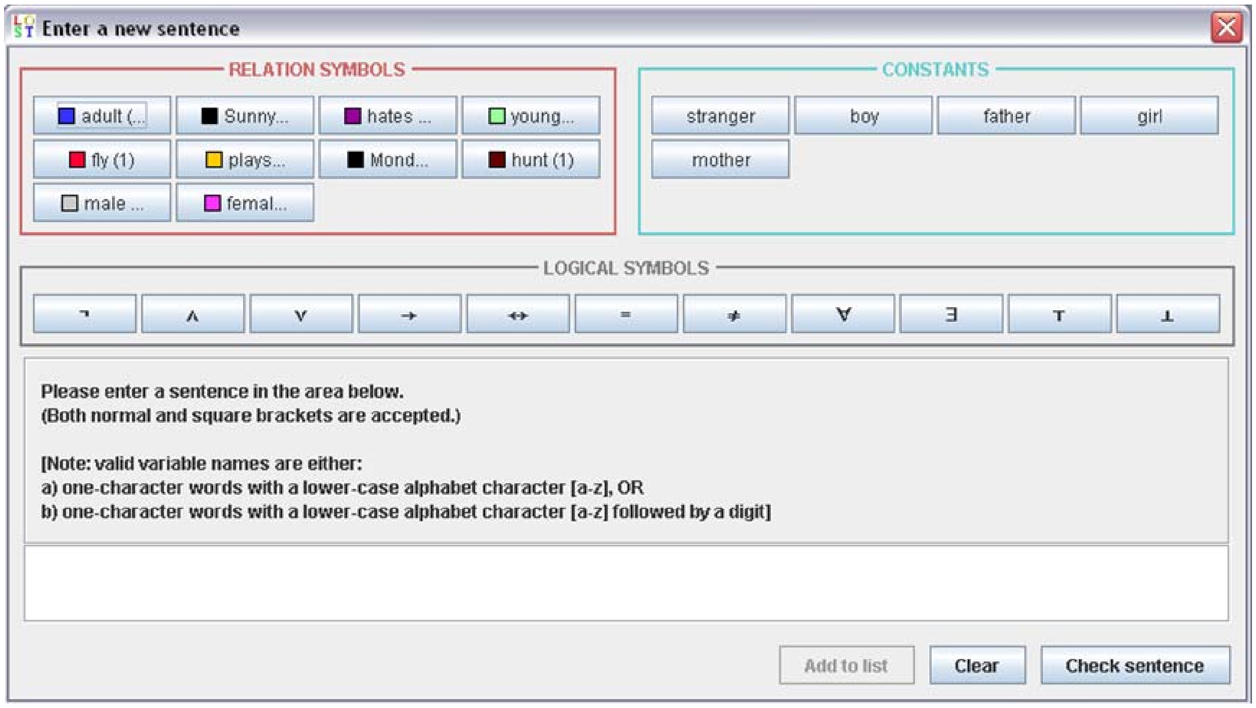
The buttons in the bottom of the Structure Panel allow you to select the size that you want the structure objects to have. This can be changed at any time and as often as you want.

5. Working with sentences

Note: a sentence is a formula with no unquantified variables.

Adding a sentence

Before you can create a new sentence there needs to be a signature present. To add a sentence, click on the “Add” button of the Formula Panel. You can also do this by going to the menu and selecting **Formula > Add sentence**, or by clicking the menu bar button ∀*. You will be presented with the following window:



Please enter a sentence in the area below.
(Both normal and square brackets are accepted.)

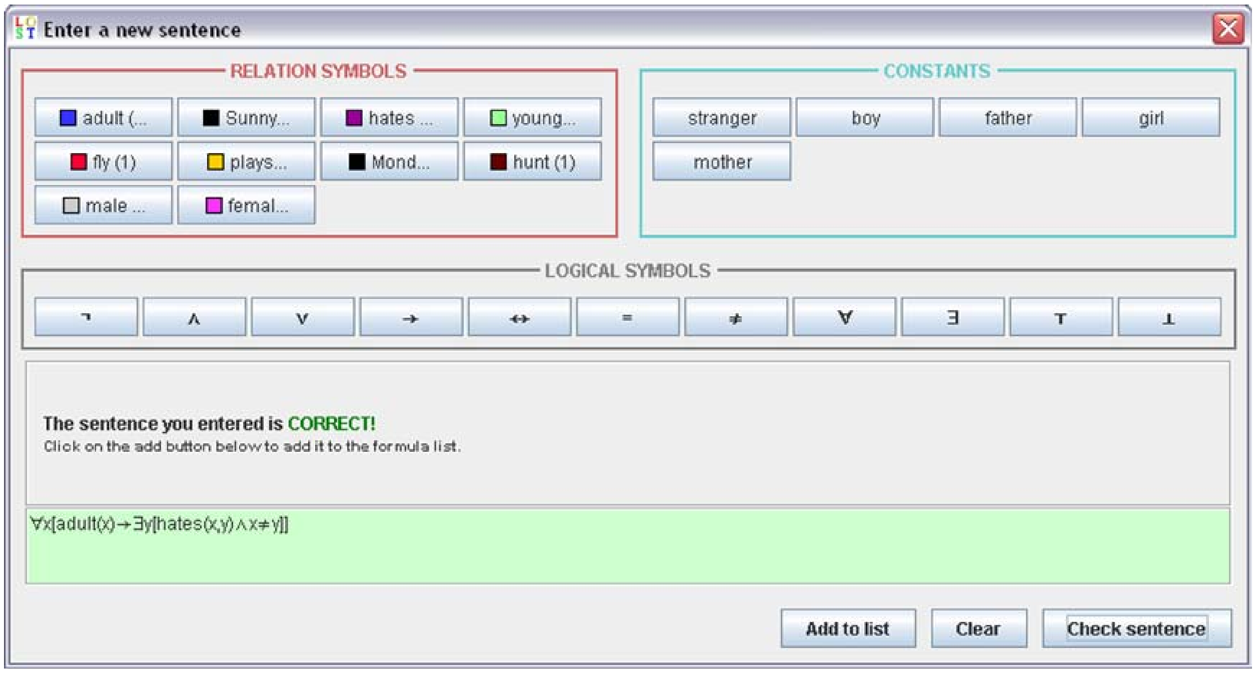
[Note: valid variable names are either:
a) one-character words with a lower-case alphabet character [a-z], OR
b) one-character words with a lower-case alphabet character [a-z] followed by a digit]

On top of this window there is a button for each constant and each relation symbol of the signature that is loaded in the main window. The buttons that represent the relation symbols also include the arity of the relation symbol, which can be helpful when writing sentences. By clicking these buttons text will be inserted in the white text area located in the lower part of the window. You can also write text manually if you prefer.

Below these buttons is a row of buttons that represent logical symbols. These cannot be entered via the keyboard; you will need to click on the buttons to get the corresponding symbol.

Note: when writing sentences that use quantifiers, make sure that you follow the following structure: ∀x(A) and ∃x(A). The parser needs the brackets around A in order to successfully parse the sentence.

Once you have finished writing the sentence, click on “Check formula”. This will parse it, checking for syntactic errors and errors with respect to the signature. If the sentence is correct, you will be presented with the following screen and the ”Add to list” button will become clickable:

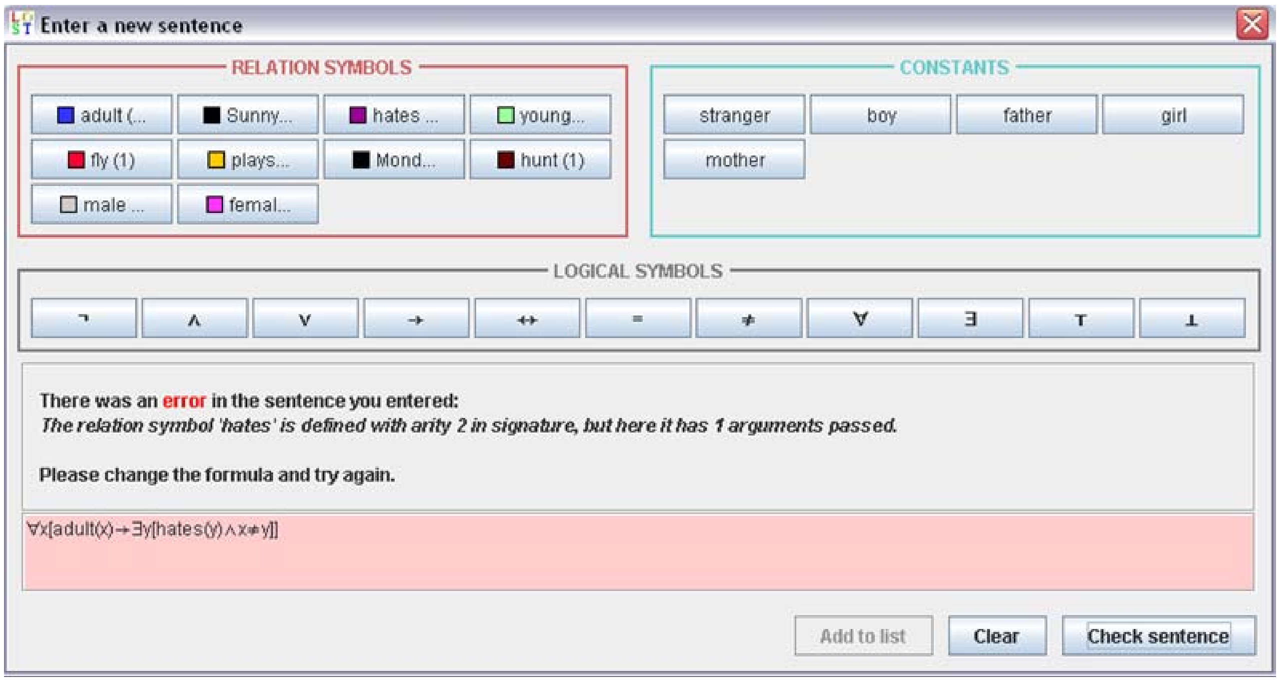


The sentence you entered is CORRECT!
Click on the add button below to add it to the formula list.

∀x[adult(x)→∃y[hates(x,y)∧x≠y]]

If you click on the “Add to list” button the window will disappear and the sentence will be added to the list in the Formula Panel.

If, on the other hand, the sentence that you entered was incorrect, you will be presented with the following screen:



There was an error in the sentence you entered:
The relation symbol 'hates' is defined with arity 2 in signature, but here it has 1 arguments passed.

Please change the formula and try again.

∀x[adult(x)→∃y[hates(y)∧x≠y]]

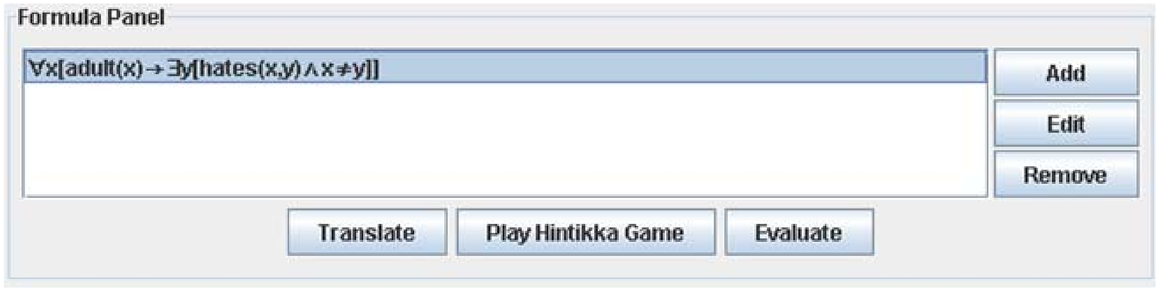
You will be prevented from adding the sentence to the Formula Panel. Most of the times when the sentence is incorrect, a meaningful message will shown to say what was wrong with it (like the example shown above). In a small number of cases the message will just read “Unmatched input”. In any case, if the sentence you wrote is incorrect you will have to edit it and retry checking it.

What can I do with a formula?

With any of the sentences that appear in the FormulaPanel, you can do one of the following things:

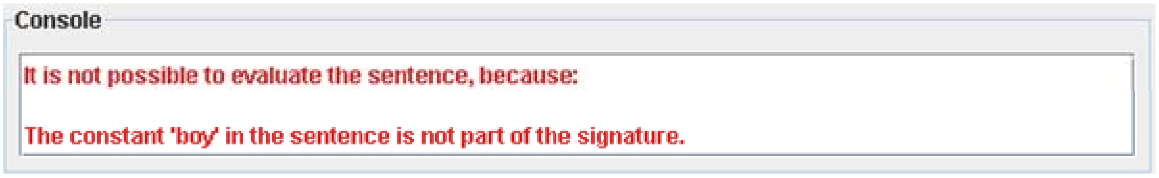
- Evaluate it on a structure
- Play the Hintikka game on a structure
- Translate it

For each of these actions there is a button in the Formula Panel:



Evaluating a sentence

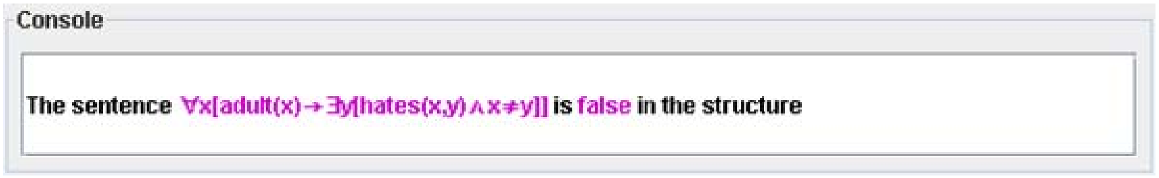
In order for a sentence to be evaluated there needs to be a structure present. Assuming a structure is loaded, clicking “Evaluate” will calculate the truth value of the sentence.
For this to be possible, all the constants and relation symbols that appear in the sentence must also be present in the signature. If, for example, after writing a sentence that contains the constant ‘boy’ you remove the constant from the signature, the evaluation of the sentence will not be possible. If you try to evaluate it the following message will appear in the console:



```
It is not possible to evaluate the sentence, because:

The constant 'boy' in the sentence is not part of the signature.
```

If no such error occurs, the truth value of the sentence will be calculated and returned in the console:



```
The sentence ∀x[adult(x)→∃y[hates(x,y)∧x≠y]] is false in the structure
```

Translating a sentence

This feature implements *literal* translation of logic sentences to English. In order to get meaningful English sentences, a dictionary needs to be defined. The dictionary includes an entry for each *relation symbol* of the signature, specifying how the relation symbol is read in English. There are six possible options:

Type	**Meaning**	
1	'is a' property (e.g. dragon)	: dragon(x) → ‘x is a dragon’
2	adjective (e.g. tall)	: tall(x) → ‘x is tall’
3	capability (e.g. fly)	: fly(x) → ‘x can fly’
4	description (e.g. smokes)	: smokes(x) → ‘x smokes’
5	transitive verb, active form (e.g. likes)	: likes(x,y) → ‘x likes y’
6	nullary predicate (e.g. Tuesday)	: Tuesday → ‘Tuesday’

We will now see the structure of dictionary files, which has an XML format:

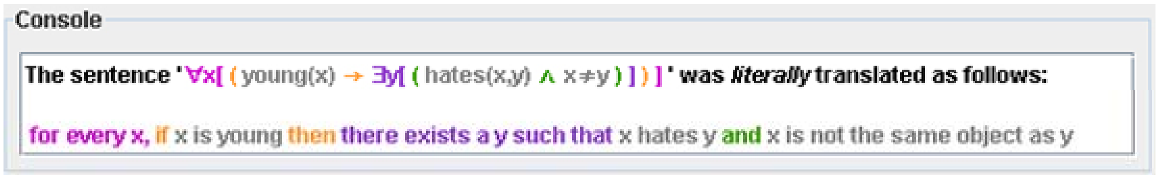
```xml
<dictionary name="dragon dictionary">
    <word name="adult" type="1"></word>
    <word name="tall" type="2"></word>
    <word name="male" type="2"></word>
    <word name="female" type="2"></word>
    <word name="hunt" type="3"></word>
    <word name="young" type="2"></word>
    <word name="fly" type="3"></word>
    <word name="likes" type="5"></word>
    <word name="hates" type="5"></word>
    <word name="Monday" type="6"></word>
</dictionary>
```

Note that if a relation symbol is not found in the dictionary, its translation will be left as the atomic formula is written in logic, e.g. tall(x)

The dictionary that we presented above can be used to translate the following sentence:

$$\forall x[young(x) \rightarrow \exists y[hates(x,y) \wedge x \neq y]]$$

And this is the result:



Console

The sentence '∀x[(young(x) → ∃y[(hates(x,y) ∧ x≠y)])] ' was *literally* translated as follows:

for every x, if x is young then there exists a y such that x hates y and x is not the same object as y

The colours hopefully help visually separate the sub-formulae that make up the sentence. They also help in suggesting how the sentence should be read. Keep in mind however that this is just a *literal* translation, not the way that we would naturally express this sentence in English. A natural English translation of the sentence would be: "everything that is young hates something else".

Saving sentences

You can save all the sentences that are shown in the Formula Panel in a file. It will be stored in an XML format, as follows:

```
<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE formulas SYSTEM "formula.dtd">

<!-- Please use the following symbol mappings when you manually write
formulae in XML files
            NOT => !
            AND => +
            OR => |
            IMPLIES => #
            IF AND ONLY IF => %
            THERE EXISTS => *
            FORALL => @
            NOT EQUALS => ~
            TRUTH => TRUTH
            FALSITY => FALSITY
            [ => {
            ] => }
-->
<formulas>
      <formula>@x{adult(x)#*y{hates(x,y)+x~y}}</formula>
      <formula>@y{young(y)|adult(y)}</formula>
      <formula>@x{@y{playsWith(x,y)#x=boy+y=girl}}</formula>
      <formula>hates(mother,stranger)+young(girl)</formula>
      <formula>@x{female(x)#x=mother|x=girl}</formula>
</formulas>
```

Unfortunately, formulae could not be stored in files using logical symbol characters. These are in Unicode and are not understood by the parser that is used to load them. As a result, formulae need to be transformed before being stored in files. The comments in the XML file above show you the characters that each logical symbol is replaced with (e.g. $\wedge$ becomes '+' etc). Please ensure that if you ever write formula files manually you follow these rules, otherwise your formulae will not be understood by the software.

To save the sentences that are currently loaded in the Formula Panel, go to menu and select **Formula** > **Save Sentence(s)**. Alternatively, click on the menu bar button ∀ ➡ 🗎. Choose a file name and a directory (ensure that the directory where you save the formula file contains the **formula.dtd** file that will be needed to load the formula file in the future). Click **Save**.

Loading sentences

To load a set of sentences, go to the menu and select **Formula** > **Load sentence(s)**. Alternatively, click on the menu bar button 🗎 ➡ ∀ . Choose the file that you want to load and press **Open**.

6. The Hintikka game

The Hintikka game is a special game for sentence evaluation and is very useful in understanding first-order logic semantics. In order to play the game there needs to be a structure and a sentence. The game unfolds on the formation tree of the sentence.

The rules

The player involves two players. Before the game begins a label needs to be assigned to each player. The two labels are ∃ and ∀. The player who is believes that the sentence is true in the structure starts with the label ∃ and the other with the label ∀.

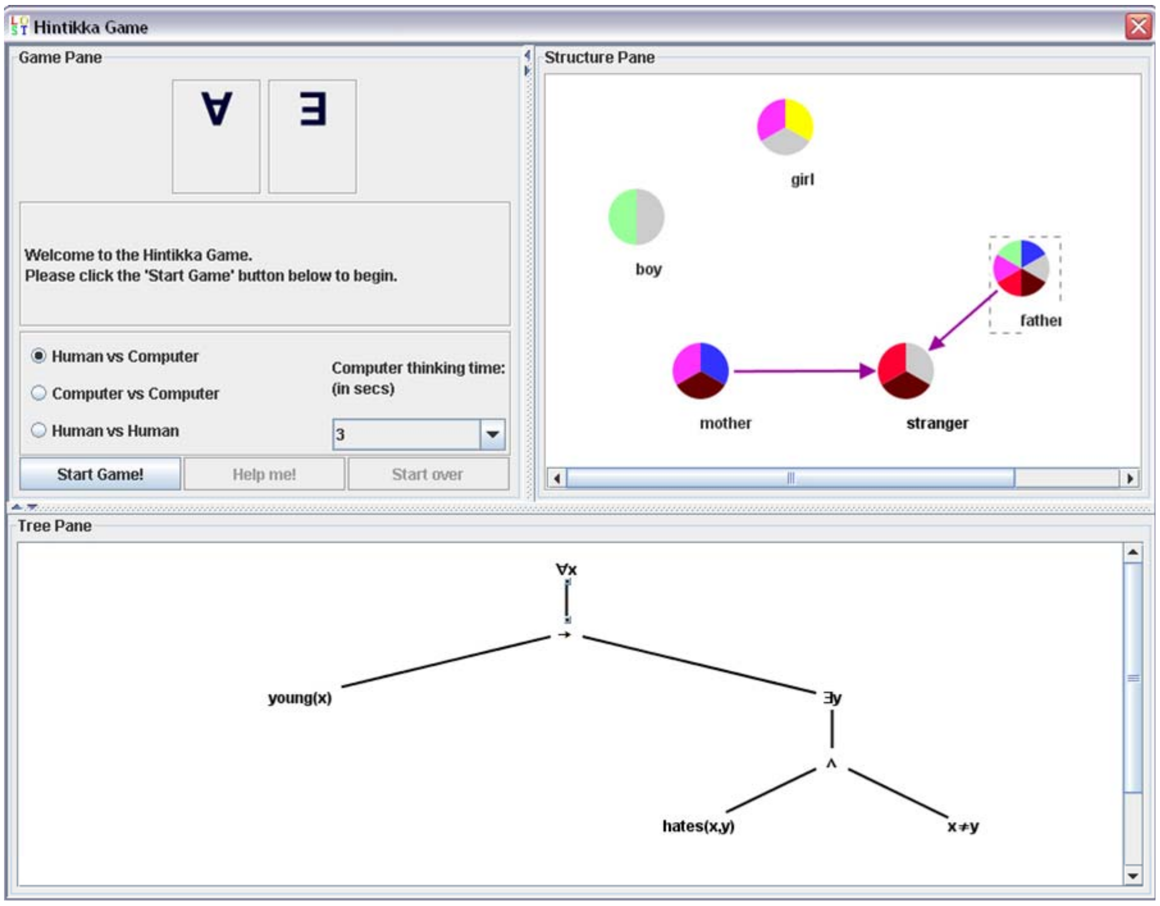
The game starts at the root of the formation tree. At each point in the game, we look at the node in the tree where the game is currently at. If that node is of the form:

- **∀x**, then the ∀ player has to choose a value for x, i.e. an object in the structure
- **∃x**, then the ∃ player has to choose a value for x, i.e. an object in the structure
- ∧, then the ∀ player chooses the next branch to follow down the tree
- ∨, then the ∃ player chooses the next branch to follow down the tree
- **↔**, regard A↔B as (A∧B) ∨ (¬A∧¬B)
- **→**, regard A→B as (¬A∧B)
- ¬, then swap labels "∃" and "∀". The player who was ∃ becomes ∀ and vice versa.
- an **atomic** (or quantifier-free) **formula**, then evaluate the formula in the given structure with the current values for the variables:
    - **If the formula is true, the ∃ player wins**
    - **If it is false, the ∀ player wins**

The game ends when a leaf (atomic formula) is reached. Therefore, during the game each player should try to *lead* the game towards a leaf that will make him/her win. This will be a leaf which evaluates to true if the player is ∃, and a leaf which evaluates to false if the player is ∀.

The game interface

Once you have created a sentence, you can select it and click on the “Play Hintikka Game” button. A new window will be loaded, looking like this:



Welcome to the Hintikka Game.
Please click the 'Start Game' button below to begin.

(●) Human vs Computer
( ) Computer vs Computer
( ) Human vs Human

Computer thinking time: (in secs) 3

The Game Panel

The top left of the screen holds instructions about the game and will be updated as the game unfolds, showing which player's turn it is. Instructions are of the form "It is ∃ player's turn to select the next node down", or "It is ∃ player's turn to select an object in the structure for the variable 'y' ".

Before the game starts, you can choose which players will play. The possible options are:

- Human against the computer
- Computer against itself
- Human against another human

If a computer player is going to be involved, you can define how much time the computer will be given to "think". The options are 1, 3, 5, 10 and 15 seconds.

The Structure Panel

On the top right of the screen is displayed the structure on which the game is played. No possible actions are possible on the structure in this window. Objects cannot be added, altered or deleted, it is not allowed in the game.
The only thing that is possible is assigning variables to objects. This will only be allowed when the player who has turn is asked to select an object in the structure for a variable. This will be the case when the current node in the formation tree is either ∃ or ∀.

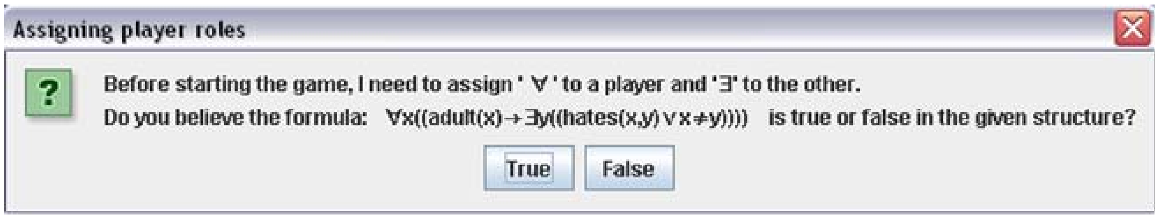
The Formula Tree Panel

The bottom half of the Hintikka game screen is filled with the sentence formation tree, on which the game will be played. When a player is required to select the next branch to go along in the tree, he/she can click one of the two nodes immediately below the last node that has been highlighted in blue. The path that the game has unfolded on is highlighted in blue in the formation tree.

The signature is not present in the window, since no modifications to it are possible during the game. If the user needs to look at the signature - to see which colours correspond to which predicates for example - he can move the Hintikka window and look at the signature panel in main window of the application.

Playing the game

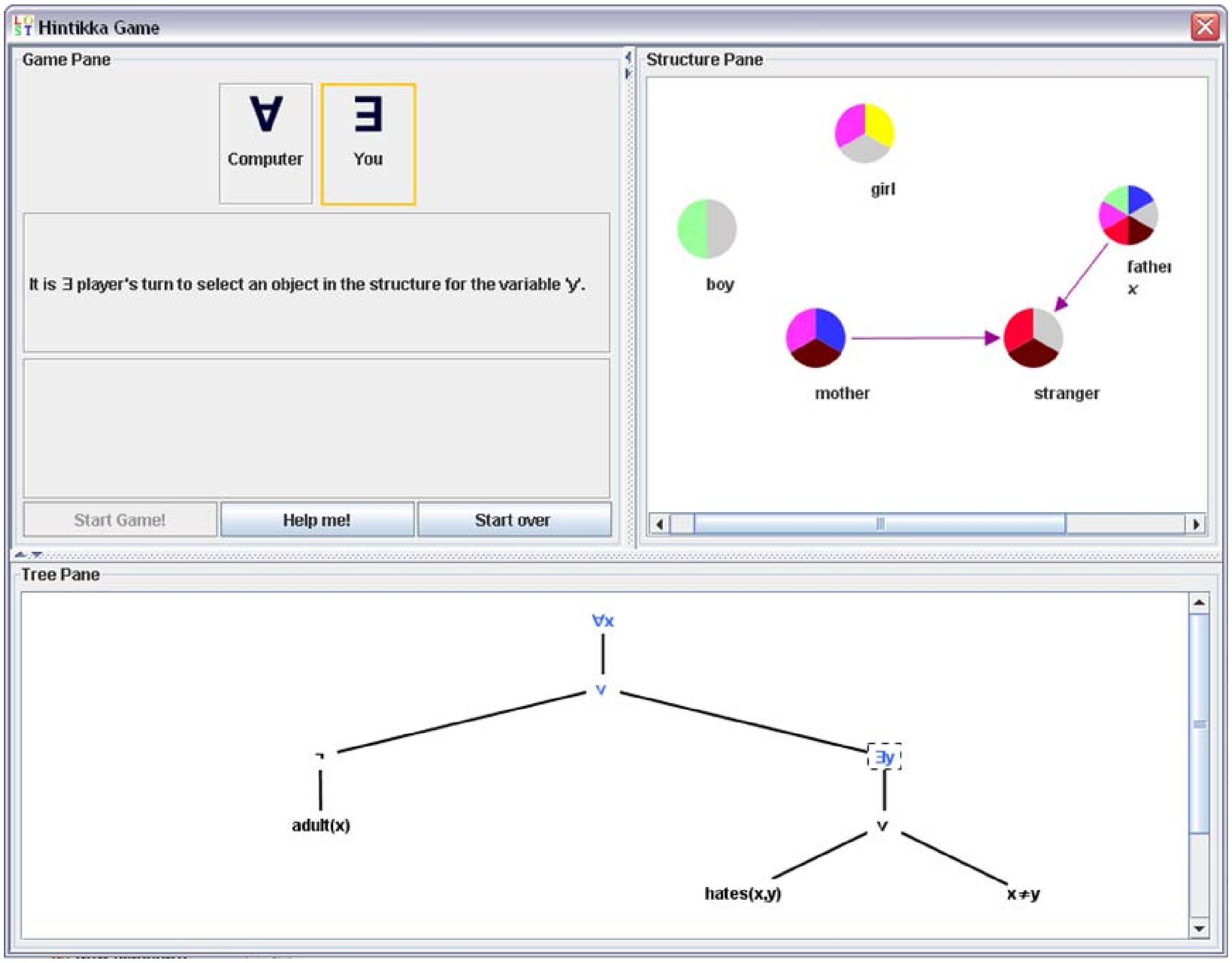
When you click the “Start button”, labels need to be assigned for each player. Unless the computer plays against itself (in which case the assignment of labels is done internally), the following window is presented:



Before starting the game, I need to assign ' ∀ ' to a player and '∃' to the other.
Do you believe the formula: ∀x((adult(x)→∃y((hates(x,y)∨x≠y))) is true or false in the given structure?

If you click “True” you will be given the ∃ label and the computer the ∀ label, otherwise it will be the other way round.

Now the game can begin. Let’s have a look at a screenshot in the middle of a Hintikka game of a human player against the computer:



It is ∃ player's turn to select an object in the structure for the variable 'y'.

The label of the player who has to make a move is in an orange border (which will become green when a player wins the game). An instruction tells the player what to do at this stage of the game.

In the screenshot, it is the ∃ player's turn to select an object in the structure to assign the variable 'y' to.